MMTE-001
GRAPH THEORY

Indira Gandhi National Open University
School of Sciences

Block

3

MATCHINGS, CONNECTIVITY AND FLOWS

UNIT 8

Matching and Covers **213**

UNIT 9

Connectivity **249**

UNIT 10

Network Flows **275**

Appendix **295**

Curriculum Design Committee

Dr. B.D. Acharya
Dept. of Science & Technology, New Delhi

Prof. Adimurthi
School of Mathematics
TIFR, Bangalore

Prof. Archana Aggarwal
CESP, School of Social Sciences
JNU, New Delhi

Prof. R.B. Bapat
Indian Statistical Institute
New Delhi

Prof. M.C. Bhandari
Dept. of Mathematics
IIT, Kanpur

Prof R. Bhatia
Indian Statistical Institute, New Delhi

Prof. A.D. Dharmadhikari
Dept. of Statistics
University of Pune

Prof. O.P. Gupta
Dept. of Financial Studies
University of Delhi

Prof. S.D. Joshi
Dept. of Electrical Engineering
IIT, Delhi

Dr. R.K. Khanna
Scientific Analysis Group, DRDO, Delhi

Prof. Susheel Kumar
Dept. of Management Studies
IIT, Delhi

Prof. Veni Madhavan
Scientific Analysis Group
DRDO, Delhi

Prof. J.C. Misra
Dept. of Mathematics
IIT, Kharagpur

Prof. C. Musili
Dept. of Mathematics and Statistics
University of Hyderabad

Prof. Sankar Pal
ISI, Kolkata

Prof. A.P. Singh
PG Dept. of Mathematics
University of Jammu

Faculty Members
School of Sciences, IGNOU
Prof. Poornima Mital
Dr. Atul Razdan
Prof. Parvin Sinclair
Prof. Sujatha Varma
Dr. S. Venkataraman

Course Design Committee

Prof. C. A. Murthy
ISI, Kolkatta

Prof. S. B. Pal
IIT, Kharagpur

Prof. C. E. Veni Madhavan
IISC, Bangalore

Dr. B. S. Panda
IIT, Delhi

Faculty Members
School of Sciences, IGNOU
Prof. Parvin Sinclair
Prof. Poornima Mital
Dr. S. Venkataraman
Dr. Deepika
Dr. Atul Razdan

Block Preparation Team

Prof. B.S. Panda (Editor)
Dept. of Mathematics
IIT Delhi

Dr. Pawan Kumar
School of Sciences
IGNOU

Course Coordinator: Dr. Pawan Kumar

Acknowledgements: To Mr. Prashant Kumar for typesetting the manuscript, and Mr. Surender Singh Chauhan for preparing the CRC.

BLOCK INTRODUCTION

In this block, we shall introduce some more concepts. In Unit 8, we shall start with the definition of a matching, and see when a matching is maximal, maximum or perfect. We shall discuss some characterisations of a maximum matching and some algorithms for detecting such a matching in bipartite graphs. The concept of vertex-cover is also introduced here and a relation is established between the maximum size of a matching and the minimum size of a vertex-cover in bipartite graphs.

In U**nit 9,** we discuss the connectivity of a graph, namely vertex-connectivity and edge-connectivity which are helpful in network analysis. We also discuss the concepts of blocks, k-connected graphs and k-edge connected graphs, and present some charaterisations of these concepts.

Unit 10, which is the last unit of this block, introduces the concept of network flows. We shall discuss here the Ford-Fulkerson Algorithm to find maximum flow in a network. Lastly, we discuss the Max-Flow Min-Cut Theorem.

As we mentioned in the course introduction, this course has a practical component. The objective of the practical component is to give you some practical experience of some of the algorithms learnt in this course. There are 8 practical sessions of 3 hours duration each. The details are given in the Lab Manual given at the end of this block. All these sessions are compulsory.

NOTATIONS AND SYMBOLS (Used in Block 3)

$N(S)$	the neighbourhood of a vertex subset S
$\beta(G)$	the vertex-covering number of G
$\alpha'(G)$	the matching number of G
GxH	the cartesian product of G and H
$\kappa(G)$	the connectivity of G
$\kappa'(G)$	the edge-connectivity of G
$[S_1T]$	the set of edges having one endpoint in S and the other in T
$BC(G)$	the block-cutpoint graph of G
$Odd(G)$	the number of odd components of G

UNIT 8

MATCHINGS AND COVERS

Structure **Page Nos.**

8.1 Introduction 213
Objectives
8.2 Matching 214
8.3 Maximal and Maximum Matchings 219
8.4 Hall's Theorem 223
8.5 Independent Sets and Vertex-Covers 227
8.6 A Min-Max Theorem 230
8.7 Augmenting Path Algorithm 232
8.8 Hungarian Algorithm 234
8.9 Summary 241
8.10 Solutions / Answers 241

8.1 INTRODUCTION

Suppose there is a set A of girls and a set B of boys such that each girl in A knows (or likes, or wants to marry) some boy in B. The problem is: Is it possible to pair every girl with some boy in such a way that no two girls are paired with the same boy? This problem has many names such as "matching problem", "assignment problem" or the traditional name "marriage problem". To address this problem we shall introduce the concept of "matching" in graphs and discuss the necessary and sufficient conditions for the existence of a solution. We shall also see how to find such a solution.

In Section 8.2, we give the definitions and illustrations of matching and perfect matching in a graph. Section 8.3 deals with maximal, and maximum matchings. We shall see here how to obtain a matching larger than a given matching, if such a matching exists, using the concept of augmenting paths. The main theorem – Hall's Theorem – is discussed in Section 8.4. In Section 8.5, we discuss the relationship between independent sets and vertex-covers. Section 8.6 introduces an algorithm to find a maximum matching in a bipartite graph. The subject of Section 8.7 is a min-max theorem which gives the relationship between the solutions to a maximisation problem and a minimisation problem.

Finally, in Section 8.8, we shall discuss an algorithm to find a matching with maximum weight in a weighted complete bipartite graph, $K_{n,n}$.

Objectives

After studying this unit, you should be able to

- understand what a matching is, and how a maximum matching differs from a maximal matching;
- check whether a given graph possesses a perfect matching or not;
- decide whether a bipartite graph contains a matching that saturates any one of its partite sets, using Hall's Theorem;
- check whether a given matching is maximum or not;
- find vertex-covers, and independent sets in a graph;
- understand the relationships among the parameters $\alpha(G)$, $\alpha'(G)$ and $\beta(G)$ for a graph G;
- find a maximum matching in a bipartite graph using the Augmenting Path Algorithm;
- find a maximum-weight matching in a weighted complete bipartite graph, $K_{n,n}$, using the Hungarian Algorithm.

8.2 MATCHING

In this section we shall introduce to you the concept of matching and some related concepts.

Let us look at the **Assignment Problem** discussed in Unit 1. It is a classical problem that can be solved using graph models. The problem is stated as follows:

Consider a set A of jobs a company would like to fill and B a set of candidates applied for. The jobs may be of different nature and some of the candidates might have qualified for some of the jobs only, and not for some other. The questions to be considered are:

Whether all the positions can be filled with suitably qualified candidates from among the applicants? And if so, how?

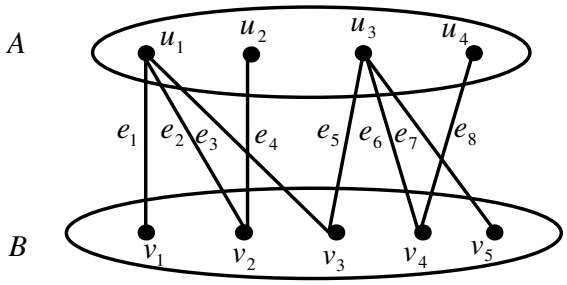
Let us model this problem using graphs:

Consider a graph G with vertex set $A \cup B$ where, A is the set of jobs, and B is the set of candidates, and there is an edge ab, $a \in A$, $b \in B$ if and only if the candidate corresponding to $b \in B$ is qualified for the job corresponding to $a \in A$. So there are no edges between any two vertices in A, and no edges between any two vertices in B.

Now the problem is to find a set M of pairwise nonadjacent edges in G of maximum cardinality. Such a set of edges is a "matching" of G.

Nonadjacent edges means the edges which share no endpoint.

For example in the following graph G the sets $M_1 = \{e_1, e_4, e_5, e_8\}$ and $M_2 = \{e_3, e_4, e_7, e_8\}$ are matchings of maximum cardinality.



G

Definition: A **matching** in a graph G is a subset M of the edge set $E(G)$ such that no pair of edges in M shares a common endpoint in G.

Definition: Let M be a matching in a graph G. A vertex u of G is said to be **saturated by** M in G **(**in short, u is called M **-saturated)** if u is an endpoint of some edge in M. If u is not M -saturated then we say u is M -**unsaturated**. If each vertex of a set $S \subseteq V(G)$, is saturated by M, then we say M **saturates** S.

For instance, in the graph G given above both M_1 and M_2 are matchings. The vertex u_1 is M_1-saturated, but v_5 is not M_1-saturated. It is interesting to know whether a graph has a matching that saturates all its vertices. This leads to the following definition.

Definition: A matching that saturates all the vertices of a graph G is called a **perfect matching** of G

Let us consider some examples.

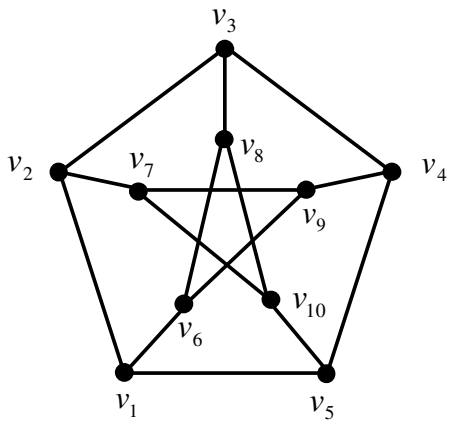
Example 1: Look at the Petersen graph given below. Which of the following sets are matchings? Which of the matchings are perfect?

i) $M_1 = \{v_1v_6, v_2v_7, v_3v_8, v_4v_9, v_5v_{10}\}$

ii) $M_2 = \{v_1v_2, v_3v_4, v_5v_{10}, v_6v_8, v_7v_9\}$

iii) $M_3 = \{v_2v_3, v_1v_5, v_4v_9\}$

iv) $M_4 = \{v_1v_2, v_2v_3, v_3v_4, v_8v_{10}\}$



Solution: i) Look at the edge v_1v_6. There is no other edge in M_1 which has v_1 or v_6 as its endpoints. Now consider the edge v_2v_7. No other edge in M_1 has v_2 or v_7 as its endpoints. In fact, any two edges of M_1 have no common endpoints. Therefore, M_1 is a matching. Note that every vertex of the graph is an endpoint of some edge of M_1. Therefore, M_1 is a perfect matching.

ii) Similar to part (i). Verify that M_2 is a perfect matching.

iii) M_3 is a matching, but it is not perfect as v_6 is not an endpoint of any edge in M_3.

iv) M_4 is not a matching as it contains the edges v_1v_2 and v_2v_3 which have v_2 as a common endpoint.

***

The problem of pairing roommates: Consider a set of people to be paired to accommodate two in each room. Here the constraint is not on the number of rooms or number of people but the individuals' willingness to share rooms with others. If everybody is ready to share a room with every other there is no difficulty, provided the number of people is even. But that may not be the real situation.

Consider the graph model of this problem. The vertex set of the graph corresponds to the set of people and an edge joins two vertices if and only if the respective persons are ready to share a room. This graph need not be bipartite, and need not be a complete graph. Then the question reduces to finding a perfect matching in the graph.

An obvious necessary condition is that **the order of the graph must be even**, i.e., the number of people must be even.

Of course, this condition is, in no way, sufficient. Because, for instance $K_{1,3}$ has no perfect matching. We shall see some sufficient conditions later. First, let us look at the following examples which have theoretical importance.

Example 2: Find the number of perfect matchings in $K_{n,n}$.

Solution: Recall that $K_{n,n}$ has n vertices in each partite set, and every vertex in a partite set is adjacent to all the vertices of the other partite set. So it is easy to find a perfect matching in it. For example $M = \{x_i\ y_i : i = 1, 2, 3, \ldots, n\}$ is a perfect matching, where $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ are the partite sets.

For finding all perfect matchings in $K_{n,n}$, consider any permutation P of the set Y. For each $i = 1, 2, \ldots, n$ pick the edge with one endpoint the i^{th} vertex in P and the other endpoint x_i. This gives us a perfect matching.

Since the order of arrangement of elements of Y is different in distinct permutations of it, we get distinct matchings. These are the only perfect matchings possible since there are only n edges incident with each vertex in $K_{n,n}$. Thus, $K_{n,n}$ has $n!$ perfect matchings.

***

An alternate computation of the number of perfect matchings in $K_{n,n}$ is given below:

There are n edges incident at x_1. So we have n choices to select an edge incident at x_1. Let it be x_1y_i. After selecting it, for an edge incident at x_2 we have exactly $n-1$ choices. Similarly, we have $n-2, n-3, \ldots,$ choices for successive selection of edges incident at $x_3, x_4, \ldots$ etc. Hence, by the principle of counting the total choices for perfect matching in $K_{n,n}$ is $n!$

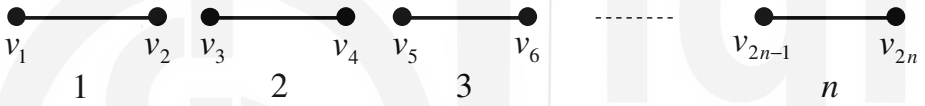
***

Observe that a complete graph has a perfect matching if and only if it is of even order. To find a perfect matching in K_{2n} a complete graph of even order, consider a spanning cycle and delete edges, from it alternately. Computation of the number of perfect matchings in K_{2n} is given below.

Example 3: Find the number of perfect matchings in K_{2n}.

Solution: Arrange the $2n$ vertices on a line, and pair the first two, the next two, and so on till the last two. This gives us a perfect matching.

See the illustration below.



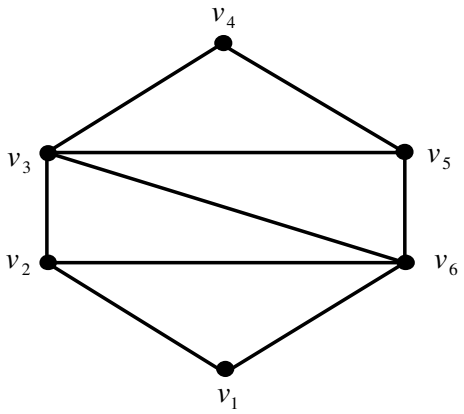
There are n pairs in this matching. If we interchange the vertices in a pair, we get the same matching. One can interchange the vertices in all the pairs in 2^n ways. Moreover, there are $n!$ ways to arrange these n pairs in a line. Thus, a perfect matching of K_{2n} gives $2^n.\, n!$ permutations of its vertices.

If there are f_n perfect matchings, of K_{2n}, we have $f_n \cdot 2^n.\, n! = (2n)!$, which implies

$$f_n = \frac{(2n)!}{2^n.n!}.$$

***

Example 4: Let us consider the following graph G.



G

i) Find a matching in G that saturates $S = \{v_1, v_2, v_3, v_4\}$.

ii) Does G contain a matching that saturates $V(G)$? If yes, find one such matching.

iii) Is every matching that saturates S a perfect matching?

Solution: i) Let $M_1 = \{v_1v_2, v_3v_6, v_4v_5\}$. Then M_1 is a matching as all the edges in M_1 are pairwise nonadjacent. Each vertex in S is an endpoint of some edge of M_1. Therefore, M_1 saturates S.

ii) Yes. M_1 saturates $V(G)$. Another matching that saturates $V(G)$ is $\{v_1v_2, v_3v_4, v_5v_6\}$.

iii) No. The matching $M_2 = \{v_1v_2, v_3v_4\}$ saturates S, but M_2 is not perfect.

***

Example 5: An n-vertex star has no perfect matching for $n \geq 3$. True or false?

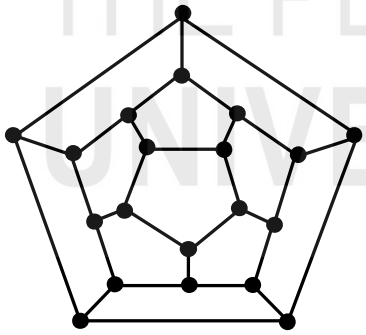
Solution: For $n \geq 3$, an n-vertex star has a vertex which is an endpoint of all the edges. So, any matching in it can have at most one edge. But a perfect matching contains exactly $\frac{n}{2}$ edges. Thus, an n-vertex star has no perfect matching if $n \geq 3$.

***

Try these exercises now.

E1) Give an example of a connected graph with order at least 6 such that every matching in it contains just one edge.

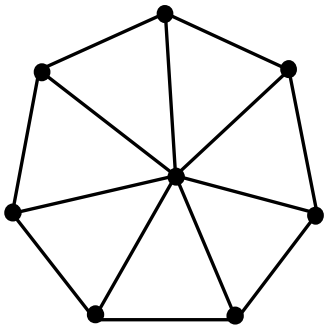
E2) Find a perfect matching in the graph given below.



E3) Let M be a matching in a graph G. Is every subset of M also a matching? What about $\overline{M}$, where $\overline{M} = E(G) \setminus M$?

E4) If M and M' are two matchings, then so is $M \cup M'$. True or false?

E5) Does the following wheel graph W_8 have a perfect matching? When does W_n have a perfect matching? Justify.



E6) If a graph has a perfect matching then it has a spanning bipartite subgraph. True or false? Justify.

E7) If a graph G has a perfect matching, and H is an induced subgraph of G, then H also has a perfect matching. True or false? Justify.

Given a matching M we can get a larger matching by adding an edge to M that is not adjacent to any edge in M. Sometimes no such edge exists, and in that case M is non-extendible or "maximal". Another type of matching occurs by maximising the cardinality. In the next section we shall explore the two types of matchings.

8.3 MAXIMAL AND MAXIMUM MATCHINGS

Intuitively, a maximal matching is one which cannot be extended further, and a maximum matching is the one which has the largest cardinality among all the matchings in a graph.

Let us start with the formal definitions.

Definition: A matching M in a graph G is called a **maximal matching** if it is not a proper subset of any other matching in G.

Definition: A matching M in a graph G is called a **maximum matching** if there is no matching in G of larger cardinality.

Now look at the following definitions.

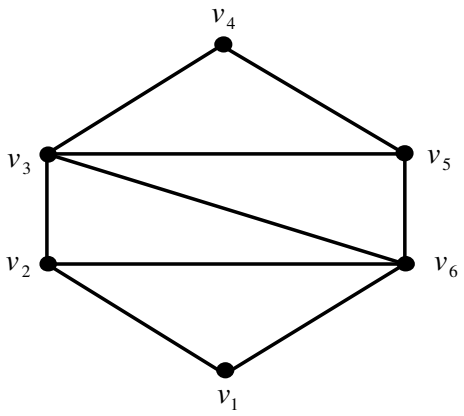
Definition: Let M be a matching in a graph G. Then a path P in G is an M **-alternating** path if the edges of P are alternately in M and $\overline{M}$.

Note that $\overline{M} = E(G) \setminus M$.

Definition: A path P is M **-augmenting** if P is M -alternating and both the endpoints of P are M -unsaturated.

Note that a maximum matching is a maximal matching also. This is because we cannot add edges any more to a maximum matching. However, a maximal matching need not be maximum. See the following example for more illustration.

Example 6: Look at the following graph G.



G

Let us consider the matchings given by
$M_1 = \{v_1v_2, v_3v_4\}$, $M_2 = \{v_1v_2, v_3v_6, v_4v_5\}$, $M_3 = \{v_1v_6, v_2v_3, v_4v_5\}$ and

$M_4 = \{v_2v_6, v_3v_5\}$. Which of these matchings are maximum, and which are maximal?

Solution: Note that the order of the graph is 6, and hence no matching can have cardinality greater than 3. Here both M_2 and M_3 are maximum as well as maximal. M_4 is a maximal matching since it is impossible to find another matching in G properly containing M_4, but it is not a maximum matching. M_1 is not maximal since $M_1 \cup \{v_5v_6\}$ is a matching in G properly containing M_1.

***

Remark 1: In an M-augmenting path P both the endpoints are M-unsaturated. So the pendant edges of P are from $\overline{M}$. Thus if P contains k edges from M, then P contains $k+1$ edges from $\overline{M}$. Therefore, the length of P is $2k+1$, which is odd. This proves that, given a matching M, every M-augmenting path has odd length.

A pendant edge is an edge incident on a left vertex.

The concept of augmenting path is helpful in checking whether a given matching is maximum or not. First, let us look at the following result.

Recall that the **symmetric difference** of two sets A and B is given by $A \Delta B = (A \setminus B) \cup (B \setminus A)$.

Lemma 1: Let G be a graph, and M_1 and M_2 be two matchings of G. Let G' be the graph induced by $M_1 \Delta M_2$. Then every component of G' is a path or cycle of even length.

Proof: Consider a component H of G'. Since G' is induced by $M_1 \Delta M_2$, every vertex of H is an endpoint of some edge in M_1, or in M_2. Thus $\delta(H) \geq 1$. Now think what could be the maximum degree of any vertex in H. Let $v \in V(H)$. Then every edge incident on v either lies in M_1 or in M_2. If there are more than two incident edges, then at least two of them would be in M_1 or in M_2. This is not possible as M_1 and M_2 are matchings. Hence $d(v) \leq 2$. Since v is arbitrary, $\max\{d(v) \mid v \in V(H)\} \leq 2$, i.e., $\Delta(H) \leq 2$. Thus H is a connected graph where every vertex has degree 1 or 2. This implies either H is a path or a cycle. (Why?) In case, H is a cycle any two consecutive edges cannot be from the same matching. Hence the edges in H alternate between M_1 and M_2, and therefore H has even length. ■

Now we are ready to see a characterisation of a maximum matching in a graph, due to C. Berge.

Theorem 1: Let G be a graph. A matching M in G is maximum if and only if G has no M-augmenting path.

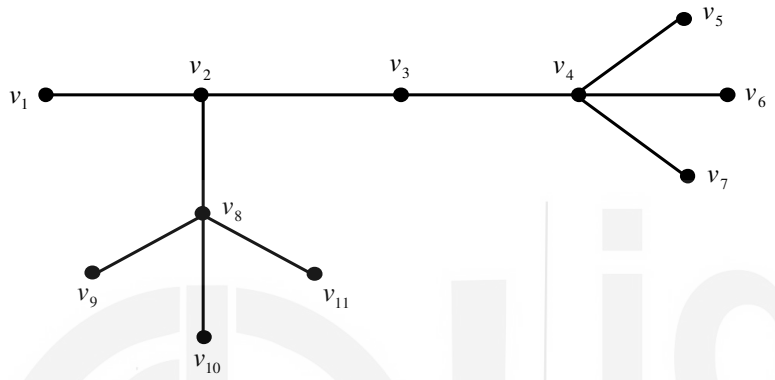
Proof: Assume that M is a maximum matching in G. We have to show that G has no M-augmenting path. Instead, let us assume that $P = (x_0, x_1, x_2, \ldots, x_{k-1}, x_k)$ is an M-augmenting path. Then, by Remark 1, P has odd length, i.e., k is odd. Hence the set $M_1 = \{x_0x_1, x_2x_3, x_4x_5, \ldots, x_{k-1}x_k\}$ is bigger than $M_2 = \{x_1x_2, x_3x_4, x_5x_6, \ldots, x_{k-2}x_{k-1}\}$. This means $(M \setminus M_2) \cup M_1$ is a matching larger than M, a contradiction. Therefore, G has no M-augmenting path.

Conversely, let us assume that G has no M-augmenting path for a matching M. Now we have to show that M is maximum. Again, we assume the

contrary. That is, we assume that M is not maximum. So, there is some matching, say M' larger than M. Let G' be the subgraph of G induced by $M \Delta M'$. By Lemma 1, every component of G' is a path or cycle of even length. If all the components of G' are cycles of even length then $|M| = |M'|$, which is not possible. Therefore, some component of G' is a path. Since $|M'| > |M|$, some path component P of G' contains more edges of M' than of M. This is possible only when the endpoints of P are M-unsaturated. This implies P is an M-augmenting path, which contradicts the assumption. ■

Let us consider more examples.

Example 7: Consider the following graph G.



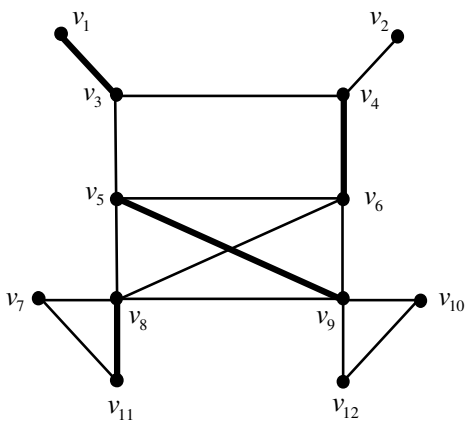
G

Let $M_1 = \{v_2v_8, v_3v_4\}$ and $M_2 = \{v_1v_2, v_3v_4, v_8v_{10}\}$. Does G have an M_1-augmenting path? Is there any M_2- augmenting path in G? Hence, conclude whether M_1 and M_2 are maximum or not.

Solution: The path $P_1 = (v_{10}, v_8, v_2, v_3, v_4, v_5)$ is an M_1-augmenting path. Therefore, by Theorem 1, M_1 is not maximum. Note that the vertices v_1, v_2, v_3, v_4, v_8 and v_{10} are M_2- saturated. Therefore, any M_2-augmenting path must have its endpoints in the set $S = \{v_5, v_6, v_7, v_9, v_{11}\}$. But there is no M_2-alternating path that begins and ends at a vertex in S. That is, G has no M_2-augmenting path. Consequently, M_2 is maximum.

***

Example 8: Let us consider the matching $M = \{v_1v_3, v_4v_6, v_5v_9, v_8v_{11}\}$ in the following graph. (See bold edges.)



Is M maximum? If not, find a matching larger than M.

Solution: The path $P = (v_2, v_4, v_6, v_9, v_5, v_8, v_{11}, v_7)$ is an M -augmenting path as both v_2 and v_7 are M -unsaturated. Therefore, M is not maximum. Let $M_1 = \{v_2v_4, v_6v_9, v_5v_8, v_7v_{11}\}$, and $M_2 = \{v_1v_3\}$. Then $M' = (M \setminus M_2) \cup M_1 = \{v_1v_3, v_2v_4, v_6v_9, v_5v_8, v_7v_{11}\}$ is a matching larger than M.

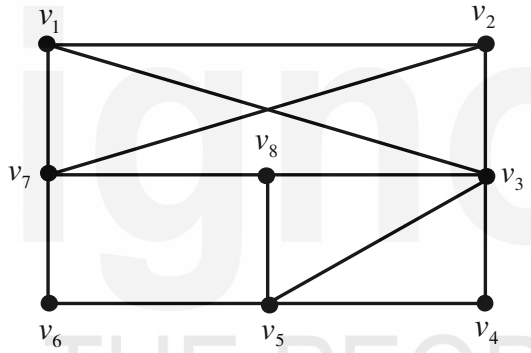
***

Try to do the following exercises, now.

E8) Every perfect matching is a maximal matching. True or false?

E9) If M is a perfect matching in a graph G, then is it possible to have an M -augmenting path in the graph? Justify your answer.

E10) Consider the following graph G.



G

i) Does G have a perfect matching? If so, find one such.

ii) Find a maximal matching that contains the edge v_3v_5. Is this matching maximum?

E11) Determine the minimum size of a maximal matching and the size of a maximum matching in the cycle C_n.

E12) If M is a matching and P is an M -augmenting path, then is it always the case that P has more edges than M ? Why?

E13) If M_1 and M_2 are two maximal matchings, then $|M_1| = |M_2|$. True or false?

E14) Check whether the Petersen graph has any perfect matching.

Recall from Section 8.2 that the Assignment Problem is to find a matching in the bipartite graph model with vertex set $A \cup B$, that saturates the partite set A. For this to happen, there are some necessary conditions as given below.

i) The number of applicants must not be less than the number of jobs, i.e., $|B| \geq |A|$,

ii) There must be at least one candidate qualified for each job, i.e., every vertex in A has non-zero degree.

More generally, we have the following condition:

iii) For every collection of k jobs there must be at least k candidates qualified for those k jobs.

Note that the first two conditions obviously follow from the third. We shall now translate this fact into graph theoretic terminology. For a graph G, let S be any subset of $V(G)$. We define the **neighbourhood** of S to be the set $N(S)$ which contains all those vertices in $V(G) \setminus S$, which are the neighbours of the vertices in S. Thus condition (iii) can be rewritten as:

For any subset S of A with $|S| = k \geq 1$, we must have $|N(S)| \geq k$.

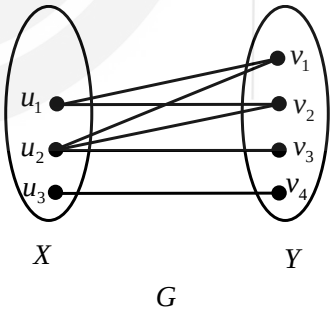
Equivalently, $|N(S)| \geq |S| \; \forall \; S \subseteq A$. In fact, in 1935 an English Mathematician Phillip Hall [1904-1982] proved that the condition "$|N(S)| \geq |S| \; \forall \; S \subseteq A$" is sufficient for any bigraph G with vertex set $A \cup B$ to have a matching that saturates A. Thereafter the condition is called **Hall's Matching condition** in honour of him.

This is discussed in the next section.

8.4 HALL'S THEOREM

Here we shall prove that the Hall's Matching Condition is not just necessary but sufficient as well for the existence of a matching that saturates one of the partite sets of a bipartite graph.

We begin with an example. Consider the following (X, Y)-bigraph G.



G

Observe that

$$N(S) = \left[\bigcup_{x \in S} N(x) \right] \setminus S.$$

One can easily see that G has a matching that saturates X. Now we show that G satisfies Hall's Matching Condition. Take any subset S of X. We always have $|N(S)| \geq |S|$. For instance, if $S = \{u_1, u_3\}$, then $N(S) = \{v_1, v_2, v_4\}$. Similarly, consider the rest of the possibilities for S. Thus, $|N(S)| \geq |S|$, for every subset S of X.

Now let us come to the result.

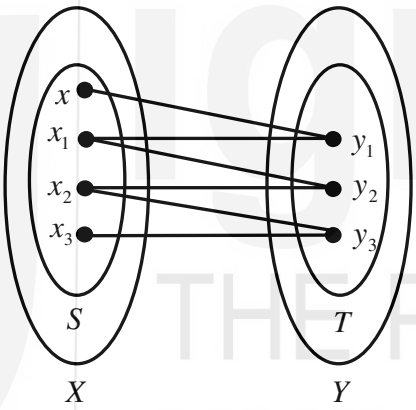
Theorem 2 (Hall's Theorem): Let G be a bipartite graph with bipartition (X, Y). Then G has a matching that saturates X if and only if $|N(S)| \geq |S| \; \forall \; S \subseteq X$.

Proof: Let M be a matching in G that saturates X. This means for every $x \in X$ there is some $y \in Y$ such that $xy \in M$. Hence for every $S \subseteq X$, there are at least $|S|$ elements in $N(S)$, i.e., $|N(S)| \geq |S|$.

Conversely, assume that G satisfies the Hall's Matching Condition, i.e., $|N(S)| \geq |S| \; \forall S \subseteq X$. We shall show that G has a matching that saturates X. If possible, assume that G has no such matching. This means every matching, and hence every maximum matching, leaves some vertex of X unsaturated. So, let M be a maximum matching of G such that some $x \in X$ is M -unsaturated. Let P be a longest M - alternating path starting at x. Since x is M -unsaturated, the last vertex of P must be M -saturated, otherwise P would be an M -augmenting path which is not possible (by Theorem 2). Thus the last vertex of P lies in X. Hence, we can take

$$P = (x = x_0, y_1, x_1, y_2, x_2, \ldots, y_k, x_k),$$

where $x_i \in X$, $y_i \in Y$ for each i. Let $S = \{x_0, x_1, x_2, \ldots, x_k\}$ and $T = \{y_1, y_2, \ldots, y_k\}$. Clearly, $|S| - 1 = |T|$. Also $T \subseteq N(S)$. Now we shall show that $N(S) \subseteq T$. Take $y \in N(S)$. Then there is some $x_i \in S$ such that $x_i y \in E(G)$. Since $N(S) \subseteq Y$, it is clear that $y \in Y$.



If $y \notin T$, then let $M_1 = \{x_1 y_1, x_2 y_2, \ldots, x_i y_i\}$ and $M_2 = \{x_0 y_1, x_1 y_2, \ldots, x_{i-1} y_i, x_i y\}$. Now $M' = (M \setminus M_1) \cup M_2$ is a matching of G such that $|M'| > |M|$, which contradicts the definition of M. Hence $y \in T$. Therefore, $N(S) \subseteq T$. Thus we have $N(S) = T$, and hence $|N(S)| = |T|$. But $|T| = |S| - 1$ implies that $|S| > |N(S)|$, which contradicts the Hall's Matching Condition. Therefore, M saturates X. ■

Look at the following real life problem.

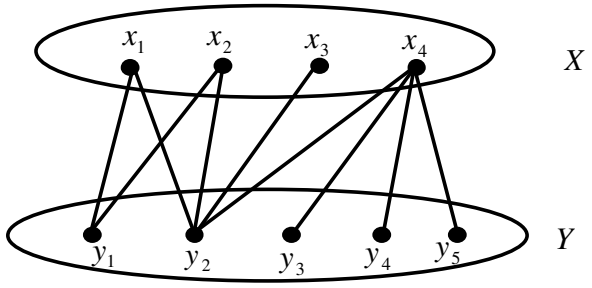
Marriage Problem

Consider a set of n girls and a set of n boys. Some of the girls love some of the boys and every girl has at least one boy whom she loves. Is it possible to find a one-to-one alliance among these boys and girls?

In the light of Hall's Matching Condition such an alliance is possible only if there is set of k boys to match with every set of k girls. A graph model of this problem is an (X, Y) -bigraph G, where X is with the set of girls and Y is the set of boys, and an edge exists between two vertices in these two sets if and only if the respective girl loves the boy represented by the other vertex. Thus, the problem reduces to find a perfect matching in G.

The following examples illustrate how to apply Hall's Theorem.

Example 9: Consider the following graph G. Check whether G has any matching that saturates X.

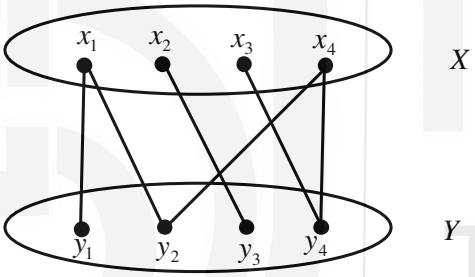


G

Solution: For $S = \{x_1, x_2, x_3\}$, we have $N(S) = \{y_1, y_2\}$. So $|N(S)| < |S|$ and hence by Hall's Theorem this graph has no matching that saturates X.

***

Example 10: Show that the following bipartite graph G with bipartition (X, Y) satisfies Hall's Matching condition. Hence conclude that G has a matching that saturates X. Also find one such matching.



G

Solution: The neighbourhoods of all the nonempty subsets of X are listed in the table given below.

S	$N(S)$
$\{x_1\}$	$\{y_1, y_2\}$
$\{x_2\}$	$\{y_3\}$
$\{x_3\}$	$\{y_4\}$
$\{x_4\}$	$\{y_2, y_4\}$
$\{x_1, x_2\}$	$\{y_1, y_2, y_3\}$
$\{x_1, x_3\}$	$\{y_1, y_2, y_4\}$
$\{x_1, x_4\}$	$\{y_1, y_2, y_4\}$
$\{x_2, x_3\}$	$\{y_3, y_4\}$
$\{x_2, x_4\}$	$\{y_2, y_3, y_4\}$
$\{x_3, x_4\}$	$\{y_2, y_4\}$
$\{x_1, x_2, x_3\}$	Y
$\{x_1, x_2, x_4\}$	Y
$\{x_1, x_3, x_4\}$	$\{y_1, y_2, y_4\}$
$\{x_2, x_3, x_4\}$	$\{y_2, y_3, y_4\}$
X	Y

From the table it is clear that for all $S \subseteq X$, $|N(S)| \geq |S|$. Thus G satisfies the Hall's Matching Condition. Therefore, by the Hall's Theorem G has a matching that saturates X. One such matching is

$$M = \{x_1y_1, x_2y_3, x_3y_4, x_4y_2\}.$$

***

Here are some exercises for you.

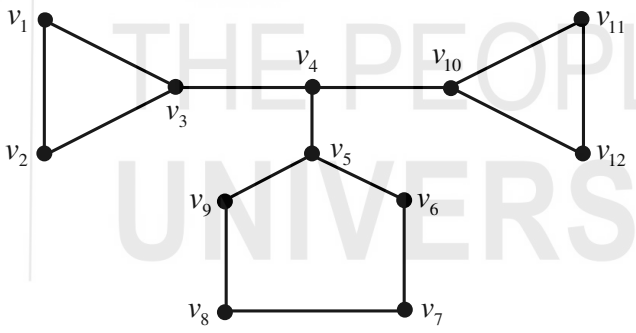
E15) In a graph G, show that for any nonempty subsets $A, B \subseteq V(G)$, $N(A \cup B) = N(A) \cup N(B)$.

E16) Let M and N be matchings in an (X,Y)-bigraph G. Suppose M saturates a subset S of X and N saturates a subset T of Y. Prove that G contains a matching that saturates $S \cup T$.

E17) Show that every k-regular bipartite graph has a perfect matching, where $k \geq 1$.

E18) For any graph G, let odd (G) denote the number of **odd components** of G, i.e., the number of those components of G which have an odd number of vertices. If G has a perfect matching then, prove that odd$(G-S) \leq |S| \forall\ S \subset V(G)$.

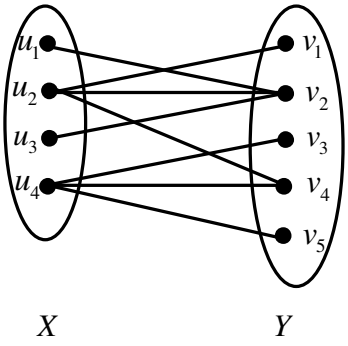
E19) Check whether the following graph has a perfect matching.



G

[**Hint:** Use E18.]

E20) Check whether or not the following (X,Y)-bigraph has any matching that saturates X.



X Y

E21) A school wants to fill six vacancies of teachers with one teacher each for Mathematics, Physics, Chemistry, Biology, Computer Science and Yoga. In order for a teacher to be selected for a particular subject, he or she must have majored or minored in that subject. The school received six applications, one each, from the candidates A, B, C, D, E and F with the following qualifications.

A: major Physics, minor Mathematics
B: major Biology, minor Chemistry and Computer Science
C: major Yoga
D: major Mathematics, minor Yoga
E: major Physics, minor Mathematics
F: major Physics, minor Mathematics and Chemistry

Is it possible for the school to hire all the applicants? Draw a graph model and use Hall's Theorem.

Hall's Theorem is an important discovery of graph theory and has wide applications. We shall see some more applications later.

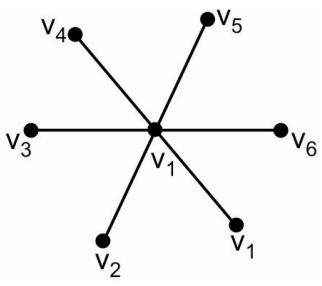
8.5 INDEPENDENT SETS AND VERTEX-COVERS

Recall that an independent set consists of pairwise nonadjacent vertices. The maximum cardinality of an independent set of a graph G is called the **independence number** of G, and we denote it by $\alpha(G)$. In this section we shall see some relationships between the graph parameters associated with independent sets and vertex-covers.
Let us first see what exactly is a vertex-cover.

Definition: A **vertex-cover** of a graph G is a subset of $V(G)$ such that it contains at least one endpoint of every edge of G.

Note that any superset of a vertex-cover is also a vertex-cover, and $V(G)$ itself is a vertex-cover of G.

Look at the following graph G, for instance.



G

Here $A = \{v_1\}$ and $B = \{v_2, v_3, v_4, v_5, v_6, v_7\}$ are two vertex-covers of G. Any set of vertices containing v_1 is a vertex-cover. However no proper subset of B is a vertex-cover of G.

The partite sets of $K_{m,n}$ are its vertex-covers. Every set of $n-1$ vertices of K_n is a vertex-cover. However, K_n has no vertex-cover of cardinality smaller than $n-1$. (Why?)
From the practical point of view, one is often interested in the minimum cardinality of a vertex-cover. This leads to the following definition.

Definition: The minimum cardinality of a vertex-cover of a graph G is called the **vertex-covering number** of G, and is denoted by $\beta(G)$.

Now let us look at the following result.

Theorem 3: Let G any graph, and $S \subseteq V(G)$. Then S is an independent set iff $\overline{S}$ is a vertex-cover. Consequently, $\alpha(G) + \beta(G) = n(G)$.

Proof: Let S be an independent set of G. Then no edge of G has both endpoints in S. This means, every edge has at least one endpoint in $\overline{S}$, and hence $\overline{S}$ is a vertex-cover.
Conversely, assume that $\overline{S}$ is a vertex-cover of G. Then every edge of G has at least one endpoint in $\overline{S}$. Hence, no edge of G has both endpoints in S. This means S is an independent set.

Let S be an independent set of maximum cardinality. Then $\overline{S}$ is a vertex-cover. Moreover, $\overline{S}$ has minimum cardinality. (Why?) Therefore, $\alpha(G) = |S|$ and $\beta(G) = |\overline{S}| = n(G) - |S|$. Consequently, $\alpha(G) + \beta(G) = n(G)$. ■

Let us consider some examples.

Example 11: Find $\beta(C_n)$ for all $n \geq 3$.

Solution: In E11 of Unit 6, we have seen that

$$\alpha(C_n) = \begin{cases} \frac{n}{2}, \text{ if } n \text{ is even} \\ \frac{n-1}{2}, \text{ if } n \text{ is odd.} \end{cases}$$

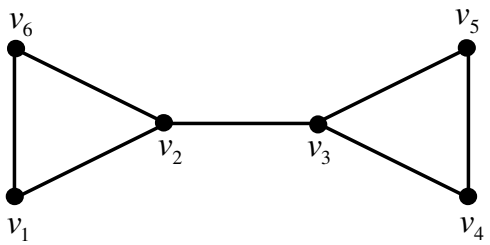
Therefore, by Theorem 3, we get

$$\beta(C_n) = n - \alpha(C_n) = \begin{cases} \frac{n}{2}, \text{ if } n \text{ is even} \\ \frac{n+1}{2}, \text{ if } n \text{ is odd.} \end{cases}$$

***

Observe that if H is an induced subgraph of a graph G, then $\beta(G) \geq \beta(H)$. Moreover, if H_1 and H_2 are two vertex-disjoint induced subgraphs of a graph G, then $\beta(G) \geq \beta(H_1) + \beta(H_2)$.

Example 12: Find $\beta(G)$ for the graph G given below.



G

Solution: It is easy to see that in a triangle at least two vertices are required to cover every edge. Since G has two vertex-disjoint triangles, $\beta(G) \geq 4$. A vertex-cover of size 4 is $S = \{v_1, v_2, v_4, v_5\}$. Therefore, $\beta(G) = 4$.

***

The parameters $\alpha(G)$ and $\beta(G)$ have one more relation as stated below.

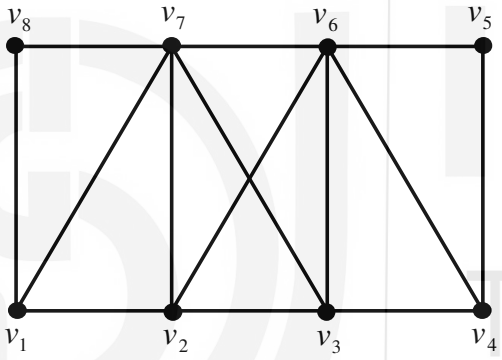
Theorem 4: For any graph G, $\beta(G) \leq \Delta(G)\,\alpha(G)$.

A **minimum vertex-cover** is just a vertex-cover of minimum cardinality.

Proof: Consider a minimum vertex-cover S of G. Then by Theorem 3, $\overline{S}$ is an independent set. Moreover, since S has minimum cardinality, $|\overline{S}| = \alpha(G)$. So, there are $\alpha(G)$ vertices in $\overline{S}$, and each of these vertices has at most $\Delta(G)$ neighbours, all in S. Therefore, S can have at most $\alpha(G)\Delta(G)$ vertices. In other words, $\beta(G) \leq \alpha(G)\Delta(G)$. ■

Now consider the following example.

Example 13: Find $\alpha(G)$ and $\beta(G)$ for the graph G given below.



G

Solution: Note that G has a clique $S = \{v_2, v_3, v_6, v_7\}$ of size 4. Therefore, $\beta(G) \geq \beta(K_4) = 3$. However, 3 vertices are not sufficient to cover all the edges of G. For instance, the edges v_1v_8 and v_4v_5 share no endpoint, and neither of these is incident on a vertex of S. Thus, any minimum vertex-cover of G must have at least 5 vertices, i.e., $\beta(G) \geq 5$. A vertex-cover with 5 vertices is $\{v_1, v_2, v_4, v_6, v_7\}$. Hence, $\beta(G) = 5$. Now by Theorem 3, $\alpha(G) = 8 - \beta(G) = 3$.

***

Now, you can try the following exercises.

E22) If S is a vertex-cover of a graph G, then show that $N(S)$ is also a vertex-cover of G.

E23) If S is a vertex-cover of a graph G, then $N(S) = \overline{S}$. True or false? Justify.

E24) Find the independence number and the vertex-covering number of the Petersen graph.

The relationship between $\alpha(G)$ and $\beta(G)$ is quite useful. The parameter $\beta(G)$ is also related to the maximum cardinality of a matching in a bipartite graph G. This is the topic of our discussion in the next section.

8.6 A MIN-MAX THEOREM

In this section we shall discuss the min-max theorem due to König and Egerváry, which establishes the equality of the size of a maximum matching and a minimum vertex-cover in any bipartite graph. There are other min-max theorems elsewhere.

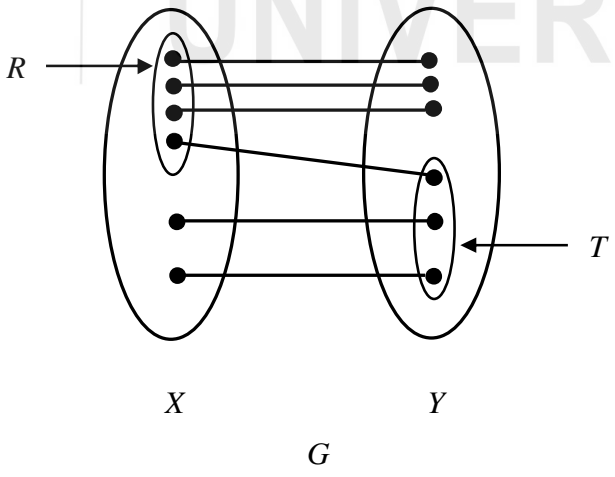
Just like the parameter $\alpha(G)$, we define $\alpha'(G)$ to be the maximum cardinality of a matching in G, and call it is called the **matching number** of G. In other words,

$$\alpha'(G) = \max\{|M|: M \text{ is a matching in } G.\}.$$

Theorem 5 (König-Egerváry Theorem): If G is a bipartite graph, then $\alpha'(G) = \beta(G)$. In other words, the maximum cardinality of a matching in a bipartite graph equals the minimum cardinality of a vertex-cover of the graph.

Proof: Assume that G is an (X,Y)-bigraph. Consider a matching M of G. Any vertex-cover S of G must have at least as many vertices as there are edges in M. (Why?) This means, $|M| \leq |S|$. Since this inequality holds for any S, we have $|M| \leq \beta(G)$. Since M is an arbitrary matching, $\alpha'(G) \leq \beta(G)$. Now it remains to show that the equality indeed holds for some matching M and some vertex-cover S.

So, consider a minimum vertex-cover S of G. Let $R = S \cap X$ and $T = S \cap Y$. Define H to be the graph induced by $R \cup (Y \setminus T)$, and H' to be the graph induced by $T \cup (X \setminus R)$. An illustration is given below.



Then both H and H' are bipartite. By the definition of S there is no edge from $X \setminus R$ to $Y \setminus T$.

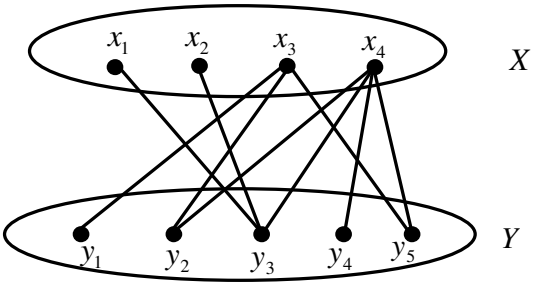
Consider any $A \subseteq R$. Then $|N_H(A)| \geq |A|$, because otherwise $(S \setminus A) \cup N_H(A)$ is a vertex-cover of G with cardinality smaller than S, which is impossible. Therefore, by Hall's Theorem, H contains a matching, say M_1, that saturates R.

Similarly, show that H' contains a matching, say M_2, that saturates T. Since H and H' are disjoint, no edge of M_1 is adjacent to an edge of M_2. Thus $M = M_1 \cup M_2$ is a matching of cardinality $|S|$. This completes the proof. ■

Now we see an example.

Example 14: Verify the König-Egerváry Theorem for the following graph.

Note that the König-Egerváry Theorem may not be true for nonbipartite graphs. For instance, consider K_3.



G

Solution: First, let us find $\alpha'(G)$. Since $|X| < |Y|$, any matching in G can have cardinality at most $|X|$, i.e., $\alpha'(G) \leq 4$. If G has a matching of size 4, then it must saturate X. However, for $S = \{x_1, x_2\}$ we have $N(S) = \{y_3\}$. This implies $|N(S)| \not\geq |S|$. Hence by Hall's Theorem, G has no matching that saturates X. Consequently, $\alpha'(G) \leq 3$. A matching of size 3 is $M = \{x_1y_3, x_3y_1, x_4y_4\}$. Thus $\alpha'(G) = 3$.

Now we find $\beta(G)$. It is easy to see that $\beta(G) \neq 1$ as G has no vertex which is an endpoint of every edge. If $\beta(G) = 2$, then we must have two vertices u and v in G such that every edge is incident on u or v. Since G has no such pair, $\beta(G) \neq 2$. This means, $\beta(G) \geq 3$. A vertex-cover of size 3 is $S = \{x_3, x_4, y_3\}$. Therefore, $\beta(G) = 3$. Thus we have established $\alpha'(G) = \beta(G)$. Hence, the König-Egerváry Theorem is verified.

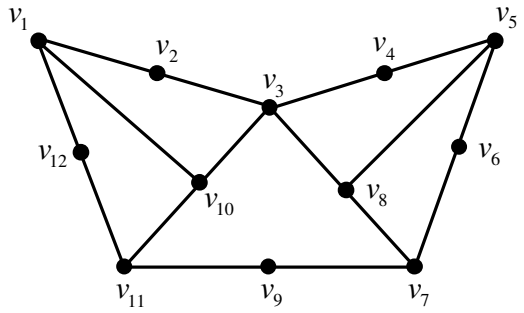
***

Now you can try the following exercises.

E25) Show that if G is a bipartite graph, then $\alpha(G) + \alpha'(G) = n(G)$.

E26) If T is an n-vertex tree where no independent set has more than k vertices, then show that T has a matching of size at least $n - k$.

E27) Verify the König-Egerváry Theorem for the following graph G.



G

The results we have seen thus far are useful for theoretical understanding of matchings, and their relationship with independent sets and vertex-covers. In the next sections we shall concentrate on how to find a maximum matching in a bipartite graph.

8.7 AUGMENTING PATH ALGORITHM

In this section we shall discuss an algorithm to find an M -augmenting path given a matching M in a graph. This algorithm is known as the Augmenting Path Algorithm.

Recall from Section 8.2, that a matching M is a maximum matching iff there is no M -augmenting path. You also know how to get a matching of cardinality $|M|+1$ from M if an M -augmenting path is given. If there is no M -augmenting path then M must be a maximum matching. So, to find a maximum matching it is enough to find an M -augmenting path, given a matching M .

Now we describe the Augmenting Path Algorithm.

Procedure (Augmenting Path Algorithm)

Input: A bigraph $G=(X,Y)$, and a matching M of G.

Output: An M -augmenting path in G , if it exists.

Step 1. Let S be the set of all M -unsaturated vertices of X. Assume that there is no M -alternating path in the beginning.

Step 2. If $S \neq \emptyset$, pick a vertex u from S.

Step 3. For each neighbour v of u such that $uv \notin M$, perform the following:

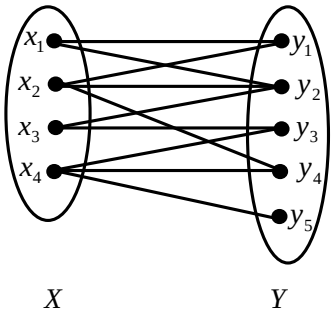
3 i) If u is the last vertex of some M -alternating path P, then add v to P. Otherwise, start a new M -alternating path P through the edge uv.

3 ii) If v is M -unsaturated, return P and stop. Otherwise, let $x \in X$ be such that $xv \in M$. Add x to the M -alternating path that ends at v. Then add x to S.

Step 4. Remove u from S, and go to Step 2.

To understand how the Augmenting Path Algorithm works, let us consider the following example.

Example 15: Let G be the bipartite graph as drawn below.



G

Find a maximum matching in G using the Augmenting Path Algorithm.

Solution: Let $M = \{x_1y_1\}$. We perform the iterations of the Augmenting Path Algorithm as follows.

Iteration 1: Here $S = \{x_2, x_3, x_4\}$. So, pick x_2. We have $x_2y_1 \notin M$. Since there is no M -alternating path as yet, we start an M -alternating path $P_1 = (x_2, y_1)$. Since y_1 is M -saturated, we find $x_1 \in X$ such that $x_1y_1 \in M$. So, we add x_1 to P_1. Then $P_1 = (x_2, y_1, x_1)$, and $S = \{x_1, x_2, x_3, x_4\}$.

Now consider the neighbour y_2 of x_2. We have $x_2y_2 \notin M$. Since there is no M -alternating path that ends at x_2, we start a new M -alternating path $P_2 = (x_2, y_2)$. Since y_2 is M -unsaturated, we return P_2 and stop.

Now we take $M \Delta E(P_2)$ as the new M. So, we have $M = \{x_1y_1, x_2y_2\}$.

Iteration 2: The M -unsaturated vertices of X are $S = \{x_3, x_4\}$. Pick x_3. The neighbours of x_3 are y_2 and y_3. We first consider y_2. We have $x_3y_2 \notin M$. Since there is no M -alternating path as yet, we start a new M -alternating path $P_1 = (x_3, y_2)$. Since y_2 is M -saturated, we find $x_2 \in X$ such that $x_2y_2 \in M$. So, we add x_2 to P_1 and to S. Then $P_1 = (x_3, y_2, x_2)$ and $S = \{x_2, x_3, x_4\}$.

Now consider the neighbour y_3 of x_3. We have $x_3y_3 \notin M$. Since x_3 is not the last vertex of any M -alternating path we start a new M -alternatives path $P_2 = (x_3, y_3)$. Since y_3 is M -unsaturated, we return P_2 and stop.

Now, we take $M \Delta E(P_2)$ as the new M. So, we have $M = \{x_1y_1, x_2y_2, x_3y_3\}$.

Iteration 3: The M -unsaturated vertices are $S = \{x_4\}$. Pick x_4. The neighbours of x_4 are y_3, y_4 and y_5.

First consider y_3. There is no M -alternating path as yet, we start a new M -alternating path $P_1 = (x_4, y_3)$. Since y_3 is M -saturated, we find $x_3 \in X$ such that $x_3y_3 \in M$. So, we add x_3 to P_1 and to S. Then $P_1 = (x_4, y_3, x_3)$ and $S = \{x_3, x_4\}$.

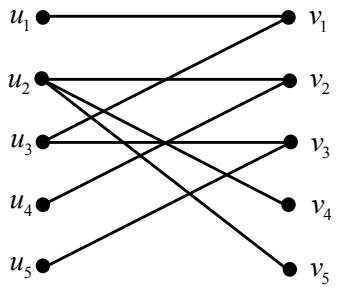
Now consider y_4. There is no M -alternating path that ends at x_4. So, we start a new M -alternating path $P_2 = (x_4, y_4)$. Since y_4 is M -unsaturated, we return P_2 and stop.

Now, we take $M \Delta E(P_2)$ as the new M. So, we have $M = \{x_1y_1, x_2y_2, x_3y_3, x_4y_4\}$.
Since $|M| = 4 = |X|$, M is a maximum matching.

***

Now try the following exercise.

E28) Using the Augmenting Path Algorithm, find a maximum matching in the graph G given below.



G

Thus far we have seen matchings in unweighted graphs. In weighted graphs one is often interested in matchings that have "maximum weight". The motivation again comes from the Job Assignment Problem. An applicant may be qualified for more than one jobs, but not equally well for all. In such situations, edges of the bipartite graph representing the problem may be assigned weights, where higher weights indicate more worthiness.

8.8 HUNGARIAN ALGORITHM

In this section we shall present a very beautiful algorithm for finding a maximum-weight matching in a weighted bipartite graph. This algorithm is known as Hungarian Algorithm in the honour of the Hungarian mathematicians König and Egerváry on whose work it is based.

Before we describe this algorithm let us understand what we mean by a maximum-weight matching. Consider a weighted (X,Y)-bigraph G, with a nonnegative weight function W. Note that if M is a matching in G, then $W(M)$ denotes the sum of the weights of the edges in M, i.e., $W(M) = \sum_{e \in M} W(e)$. Then a **maximum-weight matching** in G is a matching M such that $W(M) \geq W(M')$ for every matching M' in G. You know that if $|X| < |Y|$, then any matching in G cannot have more than $|X|$ edges. Therefore, we shall assume $|X| = |Y|$. Further, if G is not a complete bipartite graph, then the missing edges can be added with zero weight. Thus, we shall consider complete bipartite graphs only, with equal number of vertices in both partite sets.

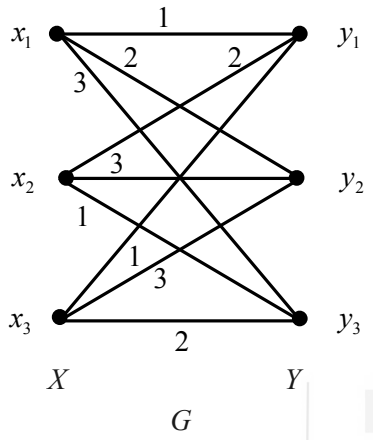
There are some more notions we need to discuss. Note that a **vertex-labelling** of a graph G is a one-one mapping $f : V(G) \to \mathbb{R}$. Given a weighted complete bipartite graph $G = (X,Y,W)$ a **feasible vertex-labelling** is a one-one mapping $f : V(G) \to \mathbb{R}$ such that $f(x) + f(y) \geq W(xy)$ for all $x \in X$ and $y \in Y$.

Definition: Let $G = (X,Y,W)$ be a weighted complete bipartite graph with $|X| = |Y|$. Given a feasible vertex-labelling f of G, the **equality subgraph** G_f is a subgraph of G with

i) $V(G_f) = V(G)$

ii) $E(G_f) = \{xy \in E(G) \mid f(x) + f(y) = W(xy)\}$.

Thus the equality subgraph G_f is a spanning subgraph of G with precisely those edges whose weights are equal to the sum of the vertex-labels of their endpoints. For instance, consider the graph G given below.



G

Define a vertex-labelling $f : V(G) \to \mathbb{R}$ as follows:

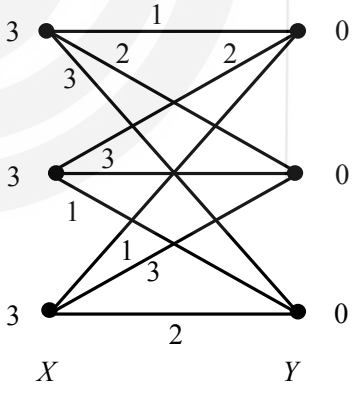
$f(x_i) = \max_{1 \le j \le 3} W(x_i y_i)$ for all $i = 1, 2, 3,$ and

$f(y_i) = 0$ for all $i = 1, 2, 3.$

Note that for all i and $j,$ we have

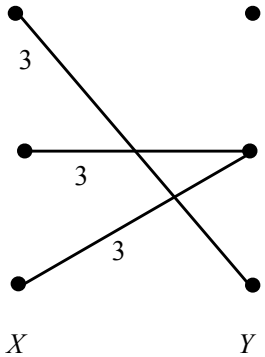
$$f(x_i) + f(y_j) = \max_{1 \le j \le 3} W(x_i y_j) \ge W(x_i y_j).$$

Thus, f is a feasible vertex-labelling. The graph G can be redrawn as follows, where the vertex labels are the feasible vertex-labels.



G

Corresponding to $f,$ the equality subgraph G_f is drawn below.



G_f

We have almost reached to the algorithm. The main idea of the algorithm lies in the following lemma.

Lemma 2: Let $G = (X, Y, W)$ be a weighted complete bipartite graph with $|X| = |Y|$. Let f be a feasible vertex-labelling of G. Then G has a maximum-weight matching if G_f has a perfect matching.

Proof: Let M^* be a perfect matching in G_f. Then M^* saturates every vertex of G_f, and hence of G. Thus M^* is a perfect matching of G. It remains to show that M^* has the maximum weight among all the matchings of G.
First note that for each $uv \in M^*$ with $u \in X$ and $v \in Y$, we have $f(u) + f(v) = W(uv)$. Therefore, $\sum_{uv \in M^*} (f(u) + f(v)) = \sum_{uv \in M^*} W(uv)$.
While computing the sum on the left hand side of the above equality, each vertex label is counted exactly once. (Why?) Therefore,

$$\sum_{uv \in M^*} (f(u) + f(v)) = \sum_{u \in X} f(u) + \sum_{v \in Y} f(v).$$

This implies, $W(M^*) = \sum_{u \in X} f(u) + \sum_{v \in Y} f(v)$. However for any matching M of G, we have

$$\sum_{u \in X} f(u) + \sum_{v \in Y} f(v) \geq \sum_{uv \in M} W(uv) = W(M). \text{ (Why?)}$$

This gives us $W(M^*) \geq W(M)$. Since M is arbitrary, M^* is a maximum-weight matching. ■

The idea of the Hungarian algorithm is now as follows. Let $G = (X, Y, W)$ be a weighted complete bipartite graph with $|X| = |Y|$. Then the existence of a maximum-weight matching in G depends on the existence of a perfect matching in its equality subgraph G_f, for a suitably chosen feasible vertex-labelling f.

If G_f has a perfect matching we are done. Otherwise, we change f and obtain a new equality subgraph G_f, and repeat the process until we get a maximum-weight matching in G.

The detailed procedure is given below.

Procedure (Hungarian Algorithm):

Input: A weighted complete bipartite graph $G = (X, Y, W)$ with $|X| = |Y|$.

Output: A maximum-weight matching in G.

Step 1: Define a feasible vertex-labelling $f : V(G) \to \mathbb{R}$ as follows:

$f(x) = \max \{W(xy) : y \in Y\}$ for all $x \in X$,

$f(y) = 0$ for all $y \in Y$.

Step 2: Construct the equality subgraph G_f corresponding to the feasible vertex-labelling f.

Step 3: Find a maximum matching M in G_f, using the Augmenting Path Algorithm.

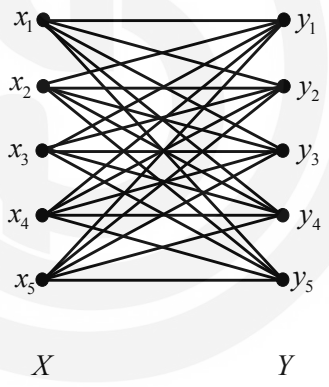
Step 4: If M is a perfect matching, return M and stop. Otherwise, let S be a minimum vertex-cover corresponding to M. Let $R = S \cap X$ and $T = S \cap Y$.

Step 5: Compute $\varepsilon = \min\{f(u) + f(v) - W(uv) : u \in X \setminus R, v \in Y \setminus T\}$.

Step 6: Replace $f(u)$ by $f(u) - \varepsilon$ for all $u \in X \setminus R$, and $f(v)$ by $f(v) + \varepsilon$ for all $v \in T$. (This gives us a new feasible vertex-labelling f.) Go to Step 2.

Now, we shall take an example. For convenience we shall use a matrix called the **excess matrix** corresponding to each feasible vertex-labelling f. The $(i,j)^{th}$ entry of the excess matrix is defined as $f(x_i) + f(y_j) - W(x_i y_j)$. A zero entry at the $(i,j)^{th}$ position indicates that $x_i y_j$ is an edge in G_f.

Example 16: Consider the following weighted $K_{5,5}$ whose weights are given in the matrix alongside.



$G = K_{5,5}$

	y_1	y_2	y_3	y_4	y_5
x_1	2	3	6	2	1
x_2	4	5	0	2	3
x_3	1	6	8	2	3
x_4	4	5	6	7	2
x_5	1	3	4	5	6

Weight matrix W

For instance, $W(x_1y_1) = 2$, $W(x_1y_2) = 3$, $W(x_3y_3) = 8$, etc.

Find a maximum-weight matching in this graph.

Solution: We show step by step how the Hungarian Algorithm works.

Step 1: Let $f : V(G) \to \mathbb{R}$ be defined as follows:

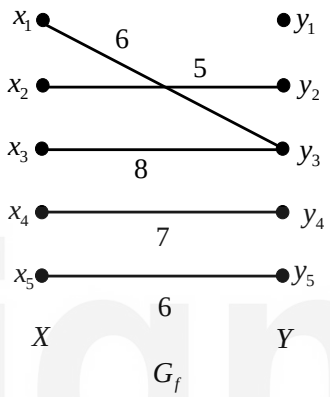
$$f(x_1) = 6, f(x_2) = 5, f(x_3) = 8, f(x_4) = 7, f(x_5) = 6$$

$$f(y_i) = 0 \text{ for } i = 1,2,3,4,5.$$

We now construct the excess matrix whose $(i,j)^{th}$ entry is $f(x_i) + f(y_i) - W(x_i y_i)$. We can represent $f(x_i)$ values along the rows and the $f(y_i)$ values along the columns of the excess matrix, as shown below.

$$\begin{array}{c} \\ 6 \\ 5 \\ 8 \\ 7 \\ 6 \end{array}\begin{array}{c} \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array} \\ \begin{bmatrix} 4 & \boxed{3} & 0 & 4 & 5 \\ \boxed{1} & 0 & 5 & 3 & 2 \\ 7 & \boxed{2} & 0 & 6 & 5 \\ 3 & 2 & \boxed{1} & 0 & 5 \\ 5 & 3 & 2 & \boxed{1} & 0 \end{bmatrix} \end{array}$$

Step 2: The entries marked 0 in the excess matrix correspond to the edges of the equality subgraph, G_f. So we construct G_f as follows.



G_f

Step 3: Using the Augmenting Path Algorithm, we find a maximum matching M in G_f as defined below.

$$M = \{x_2y_2, x_3y_3, x_4y_4, x_5y_5\}.$$

Note that $|M| = |S|$, by the Köning-Egerváry Theorem.

Step 4: Since M is not a perfect matching, we find a minimum vertex-cover S of G_f as $S = \{y_2, y_3, y_4, y_5\}$.
Now $R = S \cap X = \emptyset$, and $T = S \cap Y = S$.

Step 5: We compute

$$\begin{aligned} \varepsilon &= \min\{f(u) + f(v) - W(uv) : u \in X, v \in Y \setminus S\} \\ &= \min\{f(x_1) + f(y_1) - W(x_1y_1),\ f(x_2) + f(y_1) - W(x_2y_1),\ f(x_3) + f(y_1) \\ &\quad - W(x_3y_1),\ f(x_4) + f(y_1) - W(x_4y_1),\ f(x_5) + f(y_1) - W(x_5y_1)\} \\ &= \min\{6+0-2,\ 5+0-4,\ 8+0-1,\ 7+0-4,\ 6+0-1\} \\ &= \min\{4,1,7,3,5\} = 1. \end{aligned}$$

Step 6: Now we change the labelling of the vertices of $X \setminus R$ and of T as follows:

$$f(x_1) = 6-1 = 5,\ f(x_2) = 5-1 = 4,\ f(x_3) = 8-1 = 7,$$

$$f(x_4) = 7-1 = 6,\ f(x_5) = 6-1 = 5,$$

$$f(y_2) = 0+1 = 1,\ f(y_3) = 0+1 = 1,\ f(y_4) = 0+1 = 1,$$

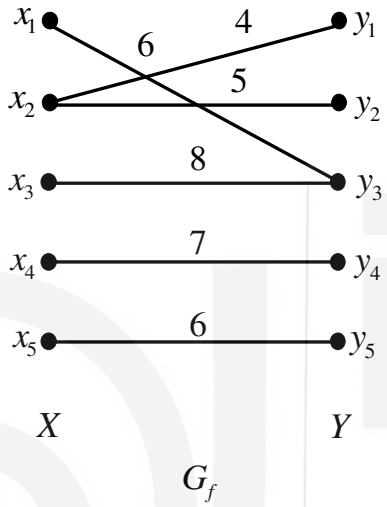
$$f(y_5) = 0+1 = 1.$$

So, the excess matrix is updated as follows.

$$\begin{array}{c} \\ 5 \\ 4 \\ 7 \\ 6 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 0 & 1 & 1 & 1 & 1 \end{array} \\ \begin{bmatrix} 3 & 3 & \boxed{0} & 4 & 5 \\ \boxed{0} & \boxed{0} & 5 & 3 & 2 \\ 6 & 2 & \boxed{0} & 6 & 5 \\ 2 & 2 & 1 & \boxed{0} & 5 \\ 4 & 3 & 2 & 1 & \boxed{0} \end{bmatrix} \end{array}$$

Now, we repeat the steps 2 to 6.

Step 2: The equality subgraph G_f corresponding to new f is given below.



G_f

Step 3: Using the Augmenting Path Algorithm, we find a maximum matching $M = \{x_2y_2, x_3y_3, x_4y_4, x_5y_5\}$ in G_f.

Remember that G_f is bipartite. So, $|S| = |M|$.

Step 4: Since M is not a perfect matching we find a minimum vertex-cover $S = \{x_2, y_3, y_4, y_5\}$ in G_f. Then $R = S \cap X = \{x_2\}$ and $T = S \cap Y = \{y_3, y_4, y_5\}$.

Step 5: We compute

$$\varepsilon = \min\{f(u) + f(v) - W(uv) : u \in X \setminus R, v \in Y \setminus T\}$$

$$= \min\{f(u) + f(v) - W(uv) : u \in \{x_1, x_3, x_4, x_5\}, v \in \{y_1, y_2\}\}$$

$$= \min\{3,3,6,2,2,2,4,3\} = 2.$$

Step 6: We change the labelling of the vertices of $X \setminus R$ and of T as follows:

$$f(x_1) = 5 - 2 = 3, f(x_3) = 7 - 2 = 5, f(x_4) = 6 - 2 = 4,$$

$$f(x_5) = 5 - 2 = 3,$$

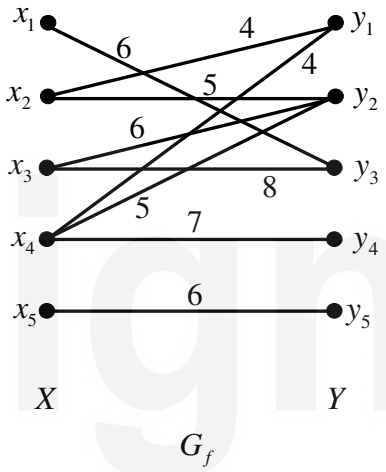
$$f(y_3) = 1 + 2 = 3, f(y_4) = 1 + 2 = 3, f(y_5) = 1 + 2 = 3$$

The excess matrix is now updated as follows.

$$\begin{array}{c} \\ 3 \\ 4 \\ 5 \\ 4 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccccc} 0 & 1 & 3 & 3 & 3 \end{array} \\ \begin{bmatrix} 1 & 1 & \boxed{0} & 4 & 5 \\ \boxed{0} & \boxed{0} & 7 & 5 & 4 \\ 4 & \boxed{0} & \boxed{0} & 6 & 5 \\ \boxed{0} & \boxed{0} & 1 & \boxed{0} & 5 \\ 2 & 1 & 2 & 1 & \boxed{0} \end{bmatrix} \end{array}$$

We again repeat the steps 2 to 6.

Step 2: The equality subgraph G_f corresponding to the new f is given below.



G_f

Step 3: Using the Augmenting Path Algorithm, we find a maximum matching $M = \{x_1y_3, x_2y_1, x_3y_2, x_4y_4, x_5y_5\}$ in G_f.

Step 4: Since M is a perfect matching in G_f, we return M and stop.
Thus, M is a maximum weight-matching with

$$W(M) = W(x_1y_3) + W(x_2y_1) + W(x_3y_2) + W(x_4y_4) + W(x_5y_5)$$
$$= 6 + 4 + 6 + 7 + 6 = 29.$$

***

Now you can try the following exercise.

E29) Find a maximum-weight matching in a weighted copy of $K_{4,4}$, whose weights are given by the following matrix.

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\begin{array}{c} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \end{array} \\ \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 0 & 2 \\ 4 & 1 & 5 & 6 \\ 6 & 2 & 3 & 4 \end{bmatrix} \end{array}$$

Let us now summaries the points discussed in this unit.

8.9 SUMMARY

In this unit, we have covered the following:

1. We have defined the concept of matching, and its different types such as perfect matching, maximal matching, maximum matching.

2. We have seen that the number of perfect matchings of K_n is $\frac{(2n)!}{2^n \cdot n!}$, and that of $K_{n,n}$ is $n!$.

3. We have seen that a perfect matching is maximum, and a maximum matching is maximal.

4. Every component of the symmetric difference of two matchings is a path or cycle of even length.

5. We have defined the concept of an augmenting path. It is proved that a matching M in a graph G is a maximum matching in G if and only if G has no M -augmenting path.

6. We have described the Hall's Matching Condition, and proved that it is necessary as well as sufficient for a bigraph to have a matching that saturates one of its partite sets.

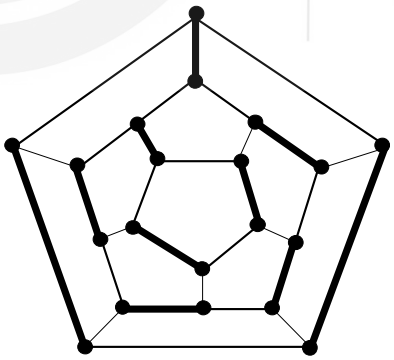
7. In a bipartite graph G, the maximum size of a matching equals the minimum size of a vertex-cover, i.e., $\alpha'(G) = \beta(G)$.

8. We have seen that $\alpha(G) + \beta(G) = n(G)$ in any graph G.

8.10 SOLUTIONS / ANSWERS

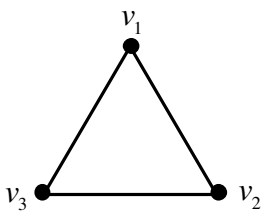
E1) An n -vertex star is one such graph, where $n \geq 6$.

E2) A perfect matching is shown in bold edges.

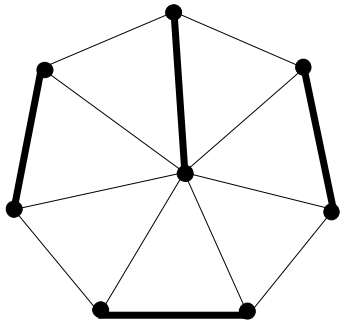


E3) Yes. However, $\overline{M}$ need not be a matching. (Think of some example.)

E4) False. For instance in the graph given below $\{v_1v_2\}$ and $\{v_2v_3\}$ are two matchings, however $\{v_1v_2, v_2v_3\}$ is not.



E5) Yes. A perfect matching in the given wheel graph is shown below in bold edges.



In general, W_n has a perfect matching iff n is even.

E6) True. Let M be a perfect matching in a graph G. Let H be the subgraph of G induced by M. Then $V(H) = V(G)$, and $E(H) = M$. Thus H is a spanning bipartite subgraph of G.

E7) False. Just find a counterexample.

E8) True. Every perfect matching is a maximum matching, and hence a maximal matching.

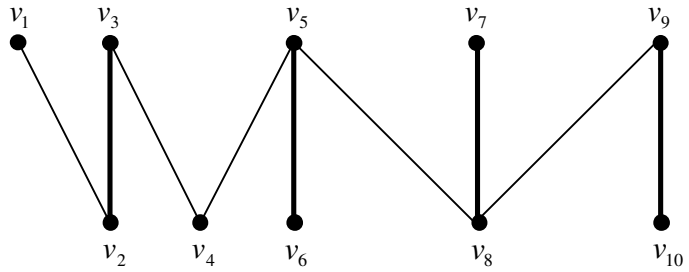
E9) No. If M is a perfect matching of G, then M is a maximum matching. Then by Theorem 1, G has no M -augmenting path.

E10) i) Yes. A perfect matching in G is $M = \{v_1v_2, v_3v_4, v_5v_8, v_6v_7\}$.

ii) The matching $M' = \{v_3v_5, v_2v_7\}$ is a maximal matching of G, because it saturates an endpoint of every edge not in it. Clearly, M' is not maximum as the matching M given in part (i) is bigger than M'.

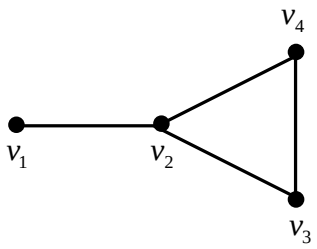
E11) Let $C = (v_1, v_2, v_3, \ldots, v_n, v_1)$ be a cycle on n vertices. First assume that n is odd. Then $M_1 = \{v_1v_2, v_3v_4, \ldots, v_{n-2}v_{n-1}\}$ and $M_2 = \{v_2v_3, v_4v_5, \ldots, v_{n-1}v_n\}$ are two maximal matchings in C. Note that $|M| = |M_2| = \frac{n-1}{2}$, which is the maximum possible size of any matching in C. Thus M_1 and M_2 are maximum matchings. Now assume that n is even. Then $M_1' = \{v_1v_2, v_3v_4, \ldots, v_nv_1\}$ and $M_2' = \{v_2v_3, v_4v_5, \ldots, v_nv_1\}$ are two maximal matchings in C, and each of them have size $\frac{n}{2}$. Hence M_1' and M_2' are maximum matching of C. Thus, in a cycle every maximal matching is a maximum matching.

E12) No. For instance, consider the following graph.



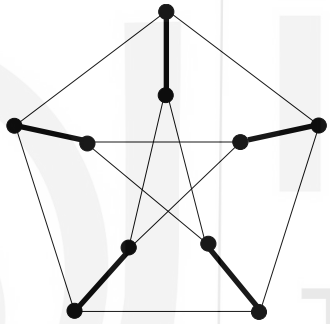
Take $M = \{v_2v_3, v_5v_6, v_7v_8, v_9v_{10}\}$. Then $P = (v_1, v_2, v_3, v_4)$ is an M-augmenting path. Clearly, P has fewer edges than M.

E13) False. For instance look at the graph given below.



Then $M_1 = \{v_1v_2, v_3v_4\}$ and $M_2 = \{v_2v_3\}$ are two maximal matchings in this graph. But $|M_1| \neq |M_2|$.

E14) A perfect matching in the Petersen graph is shown below in bold edges.



E15) Let $v \in N(A \cup B)$. Then

$\exists u \in A \cup B$ s.t. $u \leftrightarrow v \Rightarrow u \in A$ or $u \in B$ s.t. $u \leftrightarrow v$

$\Rightarrow v \in N(A)$ or $v \in N(B)$

$\Rightarrow v \in N(A) \cup N(B)$

This gives us $N(A \cup B) \subseteq N(A) \cup N(B)$. Similarly, show that $N(A) \cup N(B) \subseteq N(A \cup B)$. Hence, $N(A \cup B) = N(A) \cup N(B)$.

E16) Since M saturates S, by the Hall's Theorem $|N(A)| \geq |A|$ for every nonempty subset A of S. Similarly, $|N(B)| \geq |B|$ for every nonempty subset B of T. Now let A and B be two nonempty subsets such that $A \subseteq S$ and $B \subseteq T$. Then

$$\begin{aligned}
|A \cup B| &= |A| + |B| - |A \cap B| \\
&= |A| + |B| \qquad (\because A \cap B = \emptyset) \\
&\leq |N(A)| + |N(B)| \\
&= |N(A) \cup N(B)| + |N(A) \cap N(B)| \\
&= |N(A) \cup N(B)| \qquad (\because N(A) \cap N(B) = \emptyset) \\
&= |N(A \cup B)| \qquad \text{(by E15)}
\end{aligned}$$

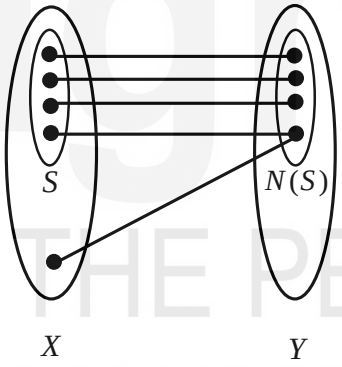
Every nonempty subset of $S \cup T$ is of the form $A \cup B$ for some $A \subseteq S$ and $B \subseteq T$. This proves that G has a matching that saturates $S \cup T$.

E17) Let G be a k-regular (X, Y)-bigraph. First, we shall prove that $|X| = |Y|$. Since G is bipartite every edge has one endpoint in X and the other endpoint in Y. Thus, we can count the number of edges of G in two ways.

Since each vertex of X has degree k, it follows that $m(G) = k|X|$. Similarly, each vertex of Y also has degree k, which implies $m(G) = k|Y|$. Therefore, $k|X| = k|Y|$, i.e., $|X| = |Y|$.

Now we show that G has a perfect matching. Observe that a perfect matching saturates X as well as Y. Indeed, any matching that saturates X also saturates Y. Thus it is enough to find a matching that saturates X.

So, let $\emptyset \neq S \subseteq X$. Since each vertex of S has degree k, there are at least $k|S|$ edges with one endpoint in S and the other endpoint in $|N(S)|$. (See the picture below.)



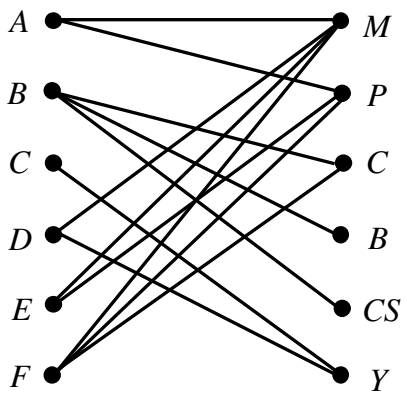
However, every edge incident on a vertex of $N(S)$ need not have the other endpoint in S. Thus, the number of edges incident on a vertex in $N(S)$ is $k|N(S)|$, which is greater than or equal to $k|S|$. This gives us $|S| \leq |N(S)|$. Since S is arbitrary, by Hall's Theorem G has a matching that saturates X, and hence G has perfect matching.

E18) Let $H_1, H_2, \ldots, H_k$ be all the odd components of $G - S$, for some $\emptyset \neq S \subset V(G)$. Take any $i, 1 \leq i \leq k$. Since G has a perfect matching and H_i has odd number of vertices, there is some edge that joins a vertex of H_i to a vertex of S. Since i is arbitrary, we find that odd $(G - S) \leq |S|$.

E19) Just take $S = \{v_4\}$, and count the odd components of $G - S$.

E20) Take $S = \{u_1, u_3\}$. Then $N(S) = \{v_2\}$. Clearly, $|N(S)| \not\geq |S|$. Therefore, by Hall's Theorem, G has no matching that saturates X.

E21) A graph model of the problem is as given below.

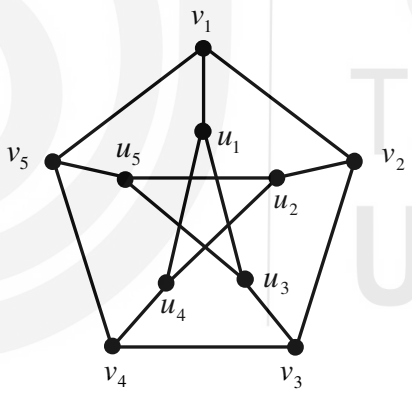


Now take $S = \{A, C, D, E\}$. Then $N(S) = \{M, P, Y\}$. Clearly, $|N(S)| \ngeq |S|$. Therefore, by Hall's Theorem, there is no matching in this graph that saturates the set $\{A, B, C, D, E, F\}$. That is, the school cannot hire all the applicants.

E22) S is a vertex-cover of G means that every edge of G is incident on a vertex of S, and every such edge is also incident on a vertex of $N(S)$. Therefore, $N(S)$ is also a vertex-cover of G.

E23) True. Try yourself.

E24) Let us consider the following drawing of the Petersen graph G.



G

Observe that G has two vertex-disjoint copies of C_5. Therefore, $\beta(G) \geq \beta(C_5) + \beta(C_5) = 3 + 3 = 6$. A vertex-cover of size 6 is $S = \{u_1, u_2, u_3, v_2, v_4, v_5\}$. This means $\beta(G) = 6$. Hence, by Theorem 3, $\alpha(G) = 10 - 6 = 4$.

E25) Use Theorem 5.

E26) Use the previous exercise.

E27) First observe that G is an (X, Y)-bigraph, where $X = \{v_1, v_3, v_5, v_7, v_{11}\}$ and $Y = \{v_2, v_4, v_6, v_8, v_9, v_{10}, v_{12}\}$.
Since $|X| < |Y|$, we find that $\alpha'(G) \leq |X| = 5$. A matching of size 5 that saturates X is $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_9, v_{11}v_{12}\}$. Thus $\alpha'(G) = 5$.

Now we compute $\beta(G)$. Note that G has an induced cycle, namely $C = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9, v_{11}, v_{12}, v_1)$. Therefore, $\beta(G) \geq \beta(C) = 5$. A vertex-cover of size 5 is X. Therefore, $\beta(G) = 5$.
Thus, we have verified the Köning-Egerváry Theorem.

E28) For each iteration of the Augmenting Path Algorithm, we list the steps in a tabular from.

Iteration 1: Take $M = \{u_1v_1\}$.

S	u	v	M -alternating paths	P (path returned)
$\{u_2, u_3, u_4, u_5\}$	u_2	v_2	(u_2, v_2)	(u_2, v_2)

Iteration 2: Take $M = M \Delta E(P)$, where P is the path returned from the previous iteration. So, we get $M = \{u_1v_1, u_2v_2\}$.

S	u	v	M -alternating paths	P
$\{u_3, u_4, u_5\}$	u_3	v_1	$(u_3, v_1) \rightarrow (u_3, v_1, u_1)$	
$\{u_3, u_4, u_5, u_1\}$		v_3	(u_3, v_3)	(u_3, v_3)

Iteration 3: Now take $M = M \Delta E(P)$, where P is the path returned in the previous iteration. So, $M = \{u_1v_1, u_2v_2, u_3v_3\}$.

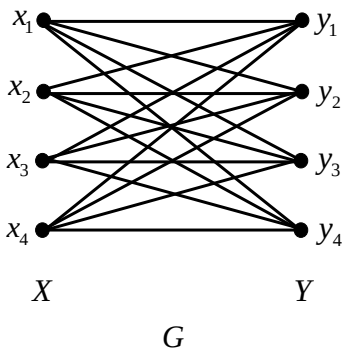
S	u	v	M -alternating paths	P
$\{u_4, u_5\}$ $\{u_4, u_5, u_2\}$	u_4	v_2	$(u_4, v_2) \rightarrow (u_4, v_2, u_2)$	
$\{u_5, u_2\}$ $\{u_5, u_2, u_3\}$	u_5	v_3	$(u_5, v_3) \rightarrow (u_5, v_3, u_3)$	
$\{u_2, u_3\}$	u_2	v_4	(u_4, v_2, u_2, v_4)	(u_4, v_2, u_2, v_4)

Iteration 4: Now take $M = M \Delta E(P)$, where P is the path returned in the previous iteration. So, $M = \{u_1v_1, u_2v_4, u_3v_3, u_4v_2\}$.

S	u	v	M -alternating paths	P
$\{u_5\}$ $\{u_5, u_3\}$	u_5	v_3	$(u_5, v_3) \rightarrow (u_5, v_3, u_3)$	
$\{u_3\}$ $\{u_3, u_1\}$	u_3	v_1	$(u_5, v_3, u_3, v_1) \rightarrow (u_5, v_3, u_3, v_1, u_1)$	
$\{u_1\}$	u_1			
$\emptyset$				

The last iteration does not return any path. Therefore, $M = \{u_1v_1, u_2v_4, u_3v_3, v_4v_2\}$ is a maximum matching.

E29) We represent the given graph and its weight matrix as below.



G

	y_1	y_2	y_3	y_4
x_1	1	2	3	4
x_2	2	3	0	2
x_3	4	1	5	6
x_4	6	2	3	4

Weight matrix W

Step 1: We begin with the feasible vertex-labellling $f : V(G) \to \mathbb{R}$ as defined below.

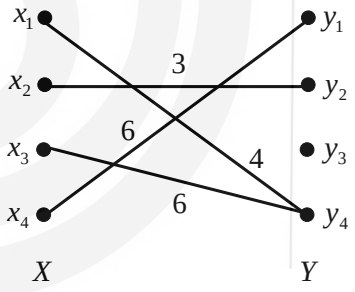
$$f(x_1) = 4,\ f(x_2) = 3,\ f(x_3) = 6,\ f(x_4) = 6,$$

$$f(y_i) = 0\ \forall\, i = 1,2,3,4$$

Then the excess matrix is

	0	0	0	0
4	3	2	1	[0]
3	2	[0]	3	1
6	2	5	1	[0]
6	[0]	4	3	2

Step 2: The equality subgraph G_f is as drawn below.



G_f

Step 3: We find a maximum matching $M = \{x_1y_4, x_2y_2, x_4y_1\}$ in G_f using the Augmenting Path Algorithm.

Step 4: Since M is not perfect, we find a minimum vertex-cover $S = \{y_1, y_2, y_4\}$ in G_f. Now, let $R = S \cap X = \emptyset,$ and $T = S \cap Y = S.$

Step 5: We now compute

$$\begin{aligned}\varepsilon &= \min\{f(u) + f(v) - W(uv) : u \in X \setminus R, v \in Y \setminus T\}\\ &= \min\{f(u) + f(v) - W(uv) : u \in X, v = y_3\}\\ &= \min\{f(x_1) + f(y_3) - W(x_1y_3),\\ &\quad f(x_2) + f(y_3) - W(x_2y_3),\ f(x_3) + f(y_3) - W(x_3y_3),\\ &\quad f(x_4) + f(y_3) - W(x_4y_3)\}\\ &= \min\{1,3,1,3\} = 1.\end{aligned}$$

Step 6: We now change the labelling of the vertices of $X \setminus R$ and of T as follows:

$$f(x_1) = 4-1 = 3, f(x_2) = 3-1 = 2, f(x_3) = 6-1 = 5,$$

$$f(x_4) = 6-1 = 5,$$

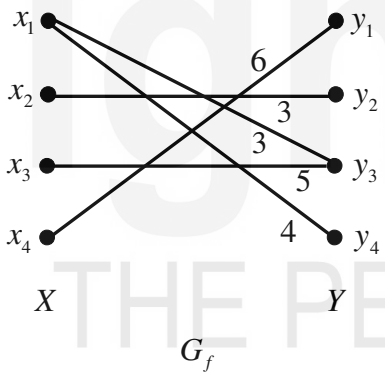
$$f(y_1) = 0+1 = 1, f(y_2) = 0+1 = 1, f(y_4) = 0+1 = 1.$$

So, the now excess matrix is

$$\begin{array}{c} \\ 3 \\ 2 \\ 5 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 1 & 0 & 1 \end{array} \\ \begin{bmatrix} 3 & 2 & \boxed{0} & \boxed{0} \\ 2 & \boxed{0} & 2 & 1 \\ 2 & 5 & \boxed{0} & \boxed{0} \\ \boxed{0} & 4 & 2 & 2 \end{bmatrix} \end{array}$$

We repeat the steps 2 to 6.

Step 2: The equality subgraph G_f is as given below.



Step 3: Now we find a maximum matching $M = \{x_1y_4, x_2y_2, x_3y_3, x_4y_1\}$ in G_f using the Augmenting Path Algorithm.

Since M is a perfect matching of G_f, we return M and stop.

Thus M is a maximum-weight matching of G, and the weight of M is

$$W(M) = W(x_1y_4) + W(x_2y_2) + W(x_3y_3) + W(x_4y_1)$$

$$= 4+3+5+6 = 18.$$

UNIT 9

CONNECTIVITY

Structure **Page Nos.**

9.1 Introduction 249
Objectives
9.2 Connectivity 250
9.3 Edge Connectivity 255
9.4 Blocks 260
9.5 k-Connected Graphs 263
9.6 Summary 268
9.7 Solutions / Answers 269

9.1 INTRODUCTION

In this unit, we shall define the concept of connectivity of a graph which has wide-spread applications in network theory. Recall, from Unit 1, that connectedness tells us whether a vertex is reachable from another vertex in a graph. Thus connectedness is a structural property. On the other hand, connectivity is a measure of the strength of connectedness. Using connectivity of graphs it is possible to analyse the parameters of a network related to its strength, capacity, reliability, efficiency, etc. To minimise the cost, the number of links must be minimised and for efficiency number of transit points are to be minimised. But the system should be so designed that even if one or two vertices (transit points) fail, the communication is not disrupted. Similarly, even if one or two edges (cable/links) fail the communication should remain unaffected.

We begin with the definition of connectivity in Section 9.2, and compute the connectivity of some standard graphs. We shall see what we mean by a k-connected graph. Section 9.3 introduces the concept of edge-connectivity, and k-edge-connected graphs. Here we shall also see a fundamental inequality between the connectivity and edge-connectivity of a graph. Section 9.4 talks about blocks of a graph, and block-cutpoint graphs. Finally, in Section 9.5, we discuss a characterisation of 2-connected graphs, and of k-connected graphs in general.

Objectives

After studying this unit, you should be able to

- identify vertex-cuts, and edge-cuts in a graph;

- compute the connectivity and edge-connectivity of small graphs;
- find the Cartesian product of two graphs;
- describe and give examples of k-connected graphs, and k-edge connected graphs;
- find the blocks in a graph;
- understand the properties of 2-connected graphs, and k-connected graphs in general.

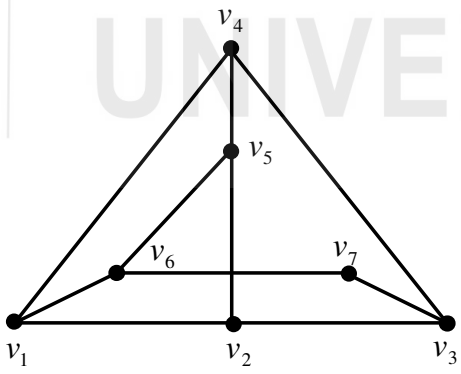
9.2 CONNECTIVITY

You are already familiar with the concept of cut-vertex from Unit 1. A vertex v of a connected graph G is a cut-vertex of G if $G-v$ is disconnected. Thus the cut-vertices of a graph are essentially the locations where one can attack on the graph to disconnect it. This is particularly important when the graph represents a communications network. Note that a communications network must be robust from the point of view of "connectivity". That is, removing a small chunk of vertices from anywhere should not result in network disconnection. Connectivity is often defined in two ways–through vertex deletion and edge deletion.

In this section we shall discuss the vertex-connectivity which is simply called connectivity.

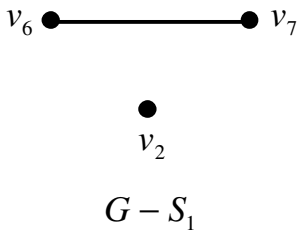
Definition: Let G be a graph. A proper subset S of $V(G)$ is called a **vertex-cut** of G if its removal from G increases the number of components or results in a trivial graph.

Note that if S is a vertex-cut of a connected n-vertex graph, then $1 \leq |S| \leq n-1$. Let us look at the following graph G.



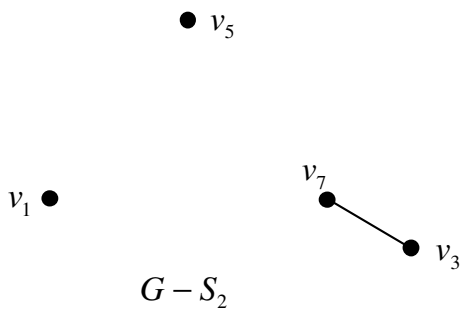
G

Take $S_1 = \{v_1, v_3, v_4, v_5\}$. Then the graph $G - S_1$, has two components as shown below.



$G - S_1$

Thus, S_1 is a vertex-cut of G. Note that $|S_1| = 4$. Does G have a smaller vertex-cut? Take $S_2 = \{v_2, v_4, v_6\}$. Then $G - S_2$ has 3 components as shown below.



That is, $G - S_2$ is also disconnected, and hence S_1 is also a vertex-cut of G. We have $|S_2| = 3$. Does G have any vertex-cut of size 2? The answer is yes, again. Just remove the neighbours of v_7 from G, and you will get a disconnected graph. What we want to emphasise here is that one is practically inclined to find the vertex-cuts of minimum size, which are also called **minimum vertex-cuts**.

This leads to the following definition.

Definition: The **connectivity** $\kappa(G)$ of a graph G is defined as the minimum cardinality of a vertex-cut of G. Symbolically, it can be stated as follows:

$$\kappa(G) = \min\{|S| : S \text{ is a vertex-cut of } G.\}$$

A graph G is said to be k-**connected** if $\kappa(G) \geq k$.

Thus, from a k-connected graph one has to remove at least k vertices to get a disconnected subgraph. Note that a k-connected graph has at least $k+1$ vertices. In other words, if a graph G has n vertices, then $\kappa(G) \leq n-1$.

The following example lists some basic facts about connectivity.

Example 1: i) If a connected graph G has a cut-vertex, then $\kappa(G) = 1$.

ii) Every k-connected graph is also $(k-1)$-connected. Thus every k-connected graph is 1-connected. Moreover, a nontrivial graph is 1-connected iff it is connected.

iii) Note that any vertex-cut in $K_n, n \geq 2$ must have at least $n-1$ vertices, because removing fewer than $n-1$ vertices from K_n leaves a connected graph. Therefore, $\kappa(K_n) = n-1$ for all $n \geq 2$.

iv) The connectivity of K_1 is zero by definition, because an empty set is a vertex-cut of K_1. Thus $\kappa(K_1) = 0$.

***

Let us look at some more examples.

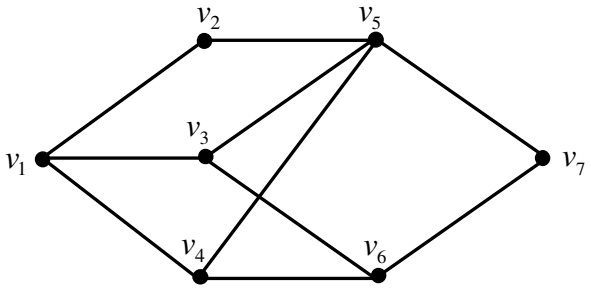
Example 2: Find the connectivity of a complete bipartite graph.

Solution: To obtain a disconnected graph from $K_{m,n}$ all vertices in one of the

partite sets must be removed. So, the connectivity of $K_{m,n}$ is the minimum of the cardinality of the partite sets. That is, $\kappa(K_{m,n}) = \min\{m,n\}$.

***

Example 3: Show that the following graph is 2-connected. Also find its connectivity.



G

Solution: It is easy to see that G has no cut-vertices. (Just remove any vertex of G. You will find that the resulting graph is connected.) Hence $\kappa(G) \geq 2$. This means, G is 2-connected.

Since $\{v_5, v_6\}$ is a vertex-cut of size 2, it follows that $\kappa(G) = 2$.

***

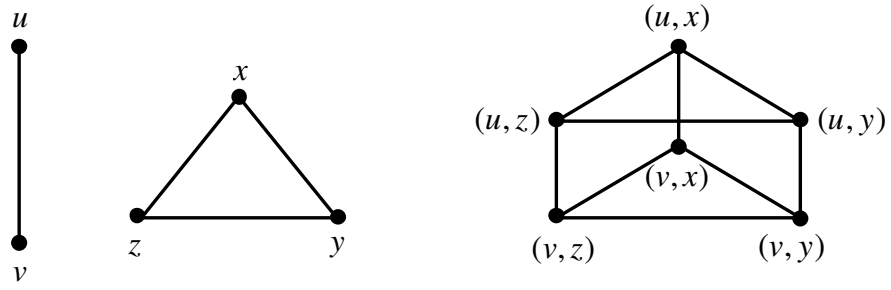
Now we define an important operation on graphs, namely the Cartesian product of two graphs.

Definition: Let G and H be two graphs. Then the **Cartesian product** of G and H is the graph $G \times H$, where $V(G \times H) = V(G) \times V(H)$, and any two vertices $(u,x),(v,y)$ in $G \times H$ are adjacent if any one of the following holds:

i) $u = v, x$ is adjacent to y in H

ii) $x = y, u$ is adjacent to v in G.

For instance, look at the graphs G and H and their Cartesian product as drawn below.

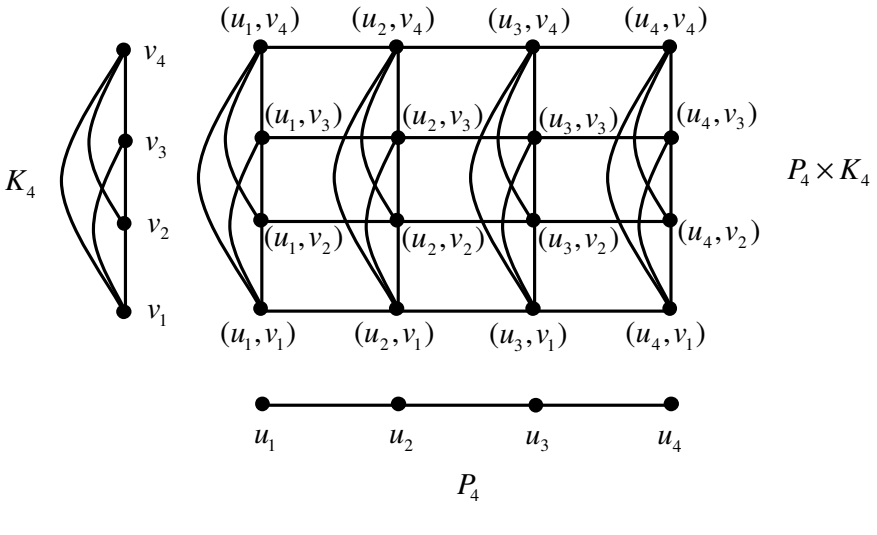


G H $G \times H$

Given two graphs G and H, we can draw $G \times H$ on the Cartesian plane by putting the vertices of one graph along the x-axis and the vertices of another graph along the y-axis. Then join the vertices according to the conditions (i) and (ii) of the definition. The following example illustrates this.

Example 4: Draw $P_4 \times K_4$.

Solution: To draw $P_4 \times K_4$ we put the vertices of P_4 along the x-axis and those of K_4 along the y-axis. Then $P_4 \times K_4$ is as shown below.



***

It is easy to show that $G \times H \cong H \times G$ for any graphs G and H. (See E2.)
Now we describe how $G \times K_2$ looks like for any graph G. You have seen above that $C_3 \times K_2$ contains two copies of C_3, say H_1 and H_2, and a matching in which each edge joins a vertex of H_1 to the corresponding vertex of H_2. To find the two copies of G in $G \times K_2$, let $V(G) = \{u_1, u_2, \ldots, u_n\}$ and $V(K_2) = \{v_1, v_2\}$. Then

$$V(G \times K_2) = \{(u_1, v_1), (u_2, v_1), \ldots, (u_n, v_1), (u_1, v_2), (u_2, v_2), \ldots, (u_n, v_2)\}.$$

$$= V_1 \cup V_2,$$

where $V_1 = \{(u_i, v_1) : u_i \in V(G)\}$ and $V_2 = \{(u_i, v_2) : u_i \in V(G)\}$.
Also,

$E(G \times K_2) = E_1 \cup E_2 \cup M$ where

$$E_1 = \{(u_i, v_1)(u_j, v_1) : u_i u_j \in E(G)\},$$

$$E_2 = \{(u_i, v_2)(u_j, v_2) : u_i u_j \in E(G)\}$$

$$M = \{(u_i, v_1)(u_i, v_2) : u_i \in V(G)\}.$$

The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two copies of G in $G \times K_2$. The set M is a matching where each edge joins a vertex of G_1 to the corresponding vertex of G_2. Thus $G \times K_2 = G_1 \cup G_2 \cup M$.

Now consider the following example.

Example 5: Show that $Q_n \cong Q_{n-1} \times K_2$ for all $n \geq 2$.

Solution: We know that $Q_2 \cong Q_1 \times K_2$ as $Q_1 = K_2$ and $Q_2 = C_4$.
Let $V(Q_n) = \{(a_1, a_2, \ldots, a_n) : a_i \in \{0,1\}, i = 1, 2, \ldots, n\}$
$V(Q_{n-1}) = \{(a_1, a_2, \ldots, a_{n-1}) : a_i \in \{0,1\}, i = 1, 2, \ldots, n-1\}$ and $V(K_2) = \{0,1\}$.
Then $V(Q_{n-1} \times K_2) = \{((a_1, a_2, \ldots, a_{n-1}), a_n) : a_i \in \{0,1\}, i = 1, 2, \ldots, n\}$.

Define a map $f : V(Q_n) \to V(Q_{n-1} \times K_2)$ by

$$f(a_1, a_2, \ldots, a_n) = ((a_1, a_2, \ldots, a_{n-1}), a_n).$$

Clearly f is one-one and onto.

We know that $Q_{n-1} \times K_2 = H_1 \cup H_2 \cup M$, where H_1 and H_2 are two disjoint copies of Q_{n-1}, and M is a perfect matching in $Q_{n-1} \times K_2$ such that each edge in M joins a vertex of H_1 to the corresponding vertex in H_2.

Take any edge uv in Q_n. Then $f(u)$ and $f(v)$ differ exactly at the same position where u and v differ. If u and v differ at a position from 1 to $n-1$, then $f(u)f(v)$ is an edge in either H_1 or H_2. If u and v differ at the nth position, then $f(u)f(v) \in M$. Similarly, if $f(u)f(v)$ is an edge in $Q_{n-1} \times K_2$, then $uv \in E(Q_n)$. Thus f is an isomorphism. Hence $Q_n \cong Q_{n-1} \times K_2$.

***

Example 6: Show the connectivity of the hypercube Q_n is n.

Solution: We shall use the Principle of Mathematical Induction to prove it. We first note that $Q_1 = K_2, Q_2 = K_2 \times K_2 = C_4$. So $\kappa(Q_1) = 1, \kappa(Q_2) = 2$.

Take any $m \geq 2$. Let $\kappa(Q_k) = k$ for all $k \leq m-1$. We shall show that $\kappa(Q_m) = m$. Pick any vertex in Q_m, and remove all its neighbours. The resulting graph is disconnected. Since each vertex of Q_m is of degree $m, \kappa(Q_m) \leq m$.

To prove $\kappa(Q_m) = m$, it is enough to show that if S is any vertex-cut of Q_m, then $|S| \geq m$.

We have $Q_m = Q_{m-1} \times K_2$. So, Q_m is obtained by taking two copies of Q_{m-1}, say Q and Q', and by joining the corresponding vertices of Q and Q'. Let S be a vertex-cut of Q_m. Then $Q_m - S$ is disconnected. Two cases arise.

Case 1: $Q - S$ and $Q' - S$ are connected. Since $Q - S$ and $Q' - S$ are connected but $Q_m - S$ is disconnected, we can see that while deleting S, all the edges joining a vertex of Q to a vertex of Q' are deleted. That is, S should contain at least one endpoint of each of such edges, i.e., $|S| \geq 2^{m-1}$. Since $2^{m-1} \geq m$, we get $|S| \geq m$.

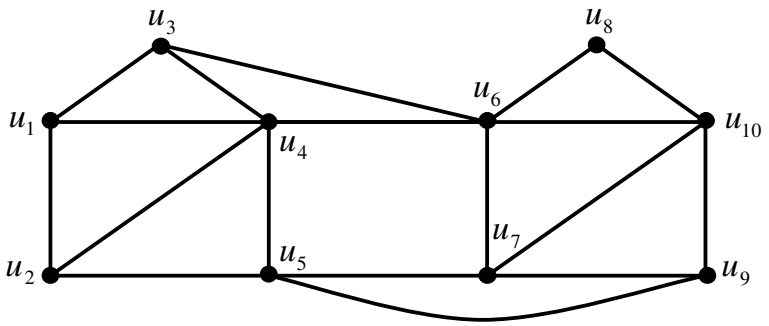
Case 2: $Q - S$ or $Q'-S$ is disconnected.

Suppose $Q - S$ is disconnected. (Similar is the case when $Q'-S$ is disconnected.) By induction hypothesis, $|S \cap Q| \geq m-1$. If S contains no vertex of Q', then Q' is connected, and each vertex in $Q - S$ is connected to the corresponding vertex in Q'. This means, $Q_m - S$ is connected, which is a contradiction. Hence S contains at least one vertex of Q'. Therefore $|S| \geq m$. So we have $\kappa(Q_m) = m$. Hence for any $n, \kappa(Q_n) = n$.

***

Try these exercises now.

E1) Find the connectivity of the following graph G.

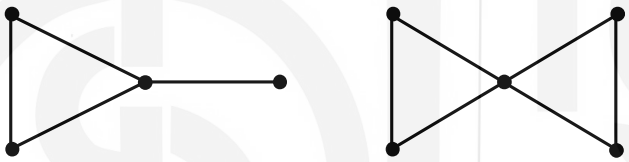


G

E2) Show that $G \times H \cong H \times G$ for any graphs G and H.

E3) What is the connectivity of a wheel W_n for $n \geq 4$?

E4) Find $G \times H$, where G and H are as given below.



G H

E5) Every nontrivial tree has connectivity 1. True or false?

E6) Draw any regular graph with at least 5 vertices with connectivity 1.

E7) Show that $\kappa(C_n) = 2$ for all $n \geq 3$.

As in the case of vertices, the removal of some edges from a connected graph may also result in a disconnected graph. This leads to the concept of edge-connectivity. We shall discuss this in the next section.

9.3 EDGE-CONNECTIVITY

The concept of edge-connectivity is similar to connectivity. Here we study how many edges one should remove to disconnect a given graph.

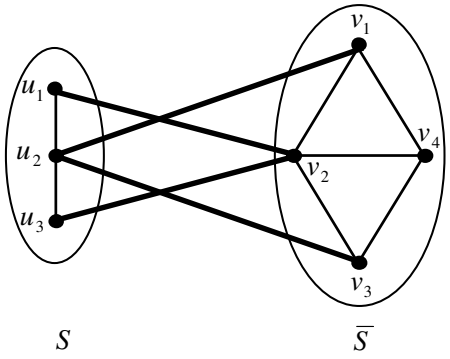
We start with the following notation.

For a graph G, consider any nonempty proper subsets S and T of $V(G)$. We shall use the notation $[S,T]$ to denote the set of all edges of G which have one endpoint in S and the other endpoint in T. Symbolically, it means

$$[S,T] = \{uv \in E(G) : u \in S, v \in T\}.$$

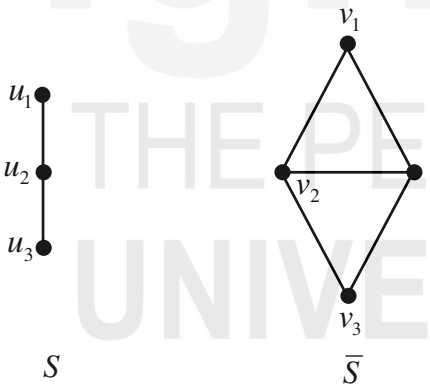
Note that if G is a connected graph, then $G-[S,T]$ need not be disconnected. However, $G-[S,\overline{S}]$ is always disconnected. This is because the set $[S,\overline{S}]$ contains all those edges which have one endpoint in S, and the other endpoint in $\overline{S}$. So if $u\in S$ and $v\in\overline{S}$, then there is no (u,v) -path in $G-[S,\overline{S}]$.

For instance, look at the graph G given below.



G

Here we take $S=\{u_1,u_2,u_3\}$. Then $\overline{S}=\{v_1,v_2,v_3,v_4\}$. So, $[S,\overline{S}]=\{u_1v_2,u_2v_1,u_2v_3,u_3v_2\}$. (See the bold edges in the graph.) The graph $G-[S,\overline{S}]$ is drawn below, which is clearly disconnected.



$G-[S,\overline{S}]$

Thus to disconnect a graph by means of edge deletion, one essentially seeks a set of the from $[S,\overline{S}]$. This leads to the following definition.

Definition: Let G be a graph. Then an **edge-cut** of G is a set of the form $[S,\overline{S}]$ for some nonempty proper subset S of $V(G)$.

The set $[S,\overline{S}]=\{u_1v_2,u_2v_1,u_2v_3,u_3v_2\}$ is an edge-cut for the graph G given above. Another edge-cut of G is $[T,\overline{T}]$, where $T=\{u_1\}$, and $\overline{T}=\{u_2,u_3,v_1,v_2,v_3,v_4\}$. So $[T,\overline{T}]=\{u_1u_2,u_1v_2\}$. Clearly, $[T,\overline{T}]$ has fewer edges than $[S,\overline{S}]$.

In fact, $[T,\overline{T}]$ has the fewest possible edges, and such an edge-cut is important to compute the “edge-connectivity” of a graph.

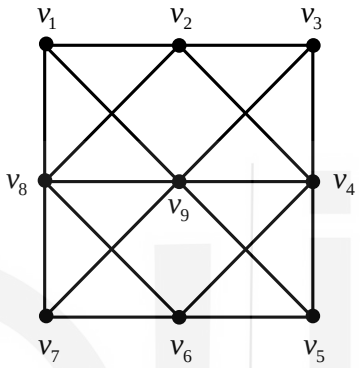
See the following definition.

Definition: The **edge-connectivity** $\kappa'(G)$ of a graph G is the minimum cardinality of an edge-cut of G. A graph G with $\kappa'(G) \geq k$ is called k-**edge-connected**.

For instance, the graph G which we were discussing above is 2-edge-connected as $\kappa'(G) = 2$. Of course, it is 1-edge-connected also. In general, if G is k-edge-connected, then G is $(k-1)$-edge-connected as well, whenever $k \geq 1$.

Let us look at some more examples.

Example 7: Find the edge-connectivity of the graph G given below.



G

Solution: Note that G is connected and hence $\kappa'(G) \geq 1$. Each edge of G lies on a cycle. This means G has no cut-edges, and hence $G - e$ is connected for every edge e in G. This implies $\kappa'(G) \geq 2$. Now we shall show that $\kappa'(G) \geq 3$. For this, we shall show that for every nonempty proper subset S of $V(G)$, $[S, \bar{S}]$ contains at least 3 edges.

Case 1: $|S| = 1$. Then there are at least 3 edges incident on a vertex in S because $\delta(G) = 3$. This means $|[S, \bar{S}]| \geq 3$.

Case 2: $|S| = 2$. Then every vertex in S has at least 2 neighbours in $\bar{S}$. This means $|[S, \bar{S}]| \geq 4$.

Case 3: $|S| = 3$. Then each vertex in S has at least one neighbour in $\bar{S}$. So, in this case also, $|[S, \bar{S}]| \geq 3$.

Case 4: $|S| = 4$. If $v_9 \in S$, then v_9 has at most 3 neighbours in S, and hence at least 3 neighbours in $\bar{S}$. So, $|[S, \bar{S}]| \geq 3$. On the other hand, if $v_9 \notin S$, then $v_9 \in \bar{S}$. Note that all vertices of degree 3 have v_9 as a neighbour.
So, if a vertex in S has degree 3, then it has v_9 as a neighbour in $\bar{S}$. If a vertex in S has degree 4 or more, then also it has a neighbour in $\bar{S}$. Thus, $|[S, \bar{S}]| \geq 3$.
The remaining cases are repetitive. Thus, we have shown for all $\emptyset \neq S \subset V(G)$ that $|[S, \bar{S}]| \geq 3$. Therefore, $\kappa'(G) \geq 3$. Now, let $S = \{v_1\}$. Then

$[S, \bar{S}] = \{v_1v_2, v_1v_8, v_1v_9\}$ is an edge-cut of G. This means $\kappa'(G) \leq 3$. Thus, it follows that $\kappa'(G) = 3$.

***

You must have observed that if we remove all the edges incident on a vertex, we get a disconnected graph. This means the edge-connectivity of a graph cannot be larger than its minimum degree. That is,

$$\kappa'(G) \leq \delta(G)$$

holds always. The parameters $\kappa(G)$ and $\kappa'(G)$ also satisfy an inequality as stated in the following theorem due to Whitney.

Theorem 1: For any graph G, $\kappa(G) \leq \kappa'(G)$.

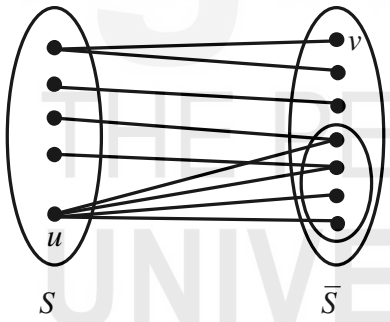
Proof: Consider a smallest edge-cut $[S, \bar{S}]$ of G. If every vertex of S is adjacent to every vertex of $\bar{S}$, then

$$\kappa'(G) = |[S, \bar{S}]| = |S||\bar{S}|.$$

If $|S| = k$, and G has n vertices, then $|\bar{S}| = n - k$. So,

$$|S||\bar{S}| = k(n-k) \geq n-1 \geq \kappa(G).$$ (Why?)

Thus the inequality follows in this case.
Now assume that there is some $u \in S$ and $v \in \bar{S}$ such that $uv \notin E(G)$. (See the picture below.)



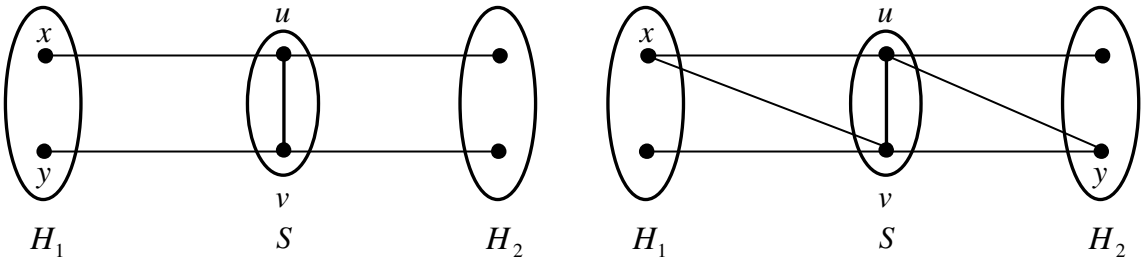
Let T be the set consisting of all the neighbours of u in $\bar{S}$, and all those vertices in $S \setminus \{u\}$ that have a neighbour in $\bar{S}$. Then every (u,v)-path passes through a vertex of T. This means T is a vertex-cut of G, and hence $\kappa(G) \leq |T|$. By the definition of T, we also have $T \subseteq [S, \bar{S}]$, which implies $|T| \leq |[S, \bar{S}]|$.
This proves that $\kappa(G) \leq \kappa'(G)$. ■

The following corollary is an important consequence of Theorem 1.

Corollary 1: If G is a 3-regular graph, then $\kappa(G) = \kappa'(G)$.

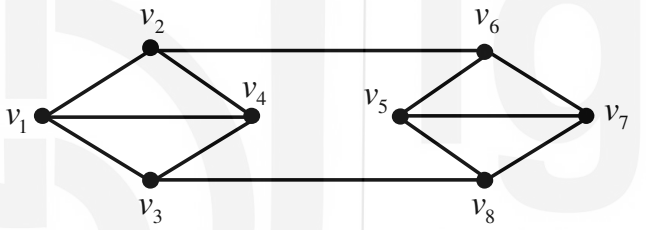
Proof: Since G is 3-regular, $\delta(G) = 3$. Therefore, by Theorem 1, $\kappa(G) \leq \kappa'(G) \leq 3$. Thus $\kappa(G)$ can be $0, 1, 2$ or 3. We consider the case $\kappa(G) = 2$, leaving other cases for you to try.
Let $S = \{u, v\}$ be a vertex-cut of G. Let H_1 and H_2 be two components of $G - S$. By definition of S, u has at least one neighbour in H_1 and at least one

neighbour in H_2. Since G is 3-regular, $\Delta(G) = 3$, and hence u cannot have two neighbours in each of H_1 and H_2. So, let x be the only neighbour of u in H_i. Working a similar argument for v, let y be the only neighbour of v in H_j. Note that i and j could be the same or distinct. However, when u and v are adjacent, we necessarily take $i = j$. Otherwise, $G - \{ux, vy\}$ may not be disconnected. (See the pictures given below.)



With the choice of the vertices x and y, we are sure that $G - \{ux, vy\}$ is disconnected. Therefore, $\kappa'(G) \leq 2$. But $2 = \kappa(G) \leq \kappa'(G)$. So, $\kappa'(G) = 2$. ■

Example 8: Consider the graph G given below. Find $\kappa(G)$ and $\kappa'(G)$.



G

Solution: Observe that each vertex of G has degree 3. That is, G is 3-regular. Therefore, by Corollary 1, $\kappa(G) = \kappa'(G)$. We can see that G has no cut-vertices. Therefore, $\kappa(G) \geq 2$. A vertex-cut of size 2 is $\{v_2, v_3\}$. Thus $\kappa(G) = 2$, and hence $\kappa'(G) = 2$.

***

Now you try the following exercises.

E8) Finish the proof of Corollary 1 by considering the cases $\kappa(G) = 0, 1$ and 3.

E9) Prove that if $\kappa(G) < \kappa'(G)$, then $\Delta(G) \geq 4$.

E10) Give an example of a graph with $\kappa(G) = 1, \kappa'(G) = 2$ and $\delta(G) = 4$.

E11) Draw a graph $G \neq K_5$ with $\kappa(G) = \kappa'(G) = \delta(G) = 4$.

We have seen that 2-connected graphs are graphs without cut-vertices. Absence of a cut-vertex in a network increases its reliability. Thus, the class of graphs without cut-vertices play prominent role in many contexts. We will study this class of graphs in the next section.

9.4 BLOCKS

You can recall that a maximal connected subgraph of a graph is called its component. However, a component may not be well-connected, in the sense that it may have cut-vertices. Therefore, in this section we shall discuss a stronger concept, namely, that of a "block".

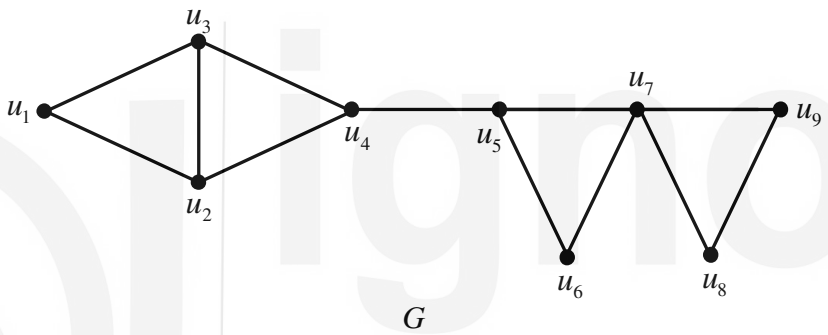
Definition: A **block** of a graph G is defined as a maximal connected subgraph of G without any cut-vertex of its own.

The definition says that any connected proper supergraph of a block must contain a cut-vertex of its own. Here the demand of cut-vertex of its own is necessary since a block in a graph G may contain a cut-vertex of G.

Note that a supergraph of a block is not a block.

See the following example.

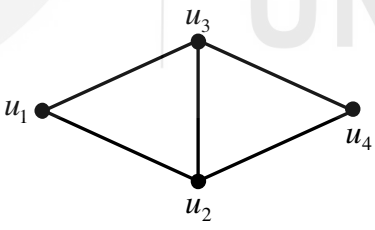
Example 9: Consider the graph given below.



G

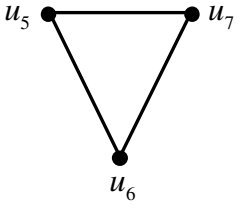
Find all the blocks of G.

Solution: Note that u_4, u_5, u_7 are all the cut-vertices of G. The blocks of G are B_1, B_2, B_3, and B_4 as given below.

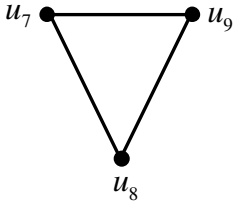


B_1

B_2



B_3



B_4

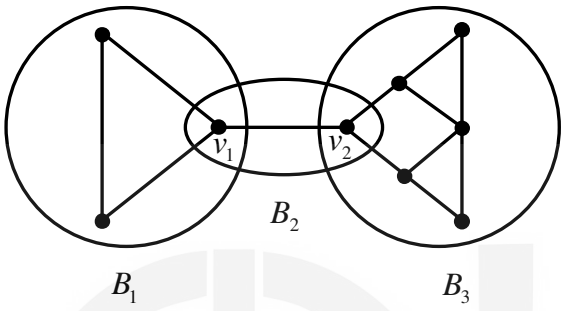
***

Remark 2: You may note that if B_1 and B_2 are two blocks in a graph G which share a vertex u, then u is a cut-vertex in G. Moreover, every graph has more blocks than the cut-vertices in it.

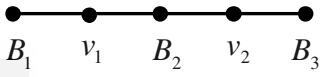
The information about the blocks of a graph is useful in many applications. For instance, one can study the "block-cutpoint graph" of a graph to analyse the structure of the graph. Below we define what we mean by the block-cutpoint graph of a graph.

Definition: The **block-cutpoint graph** $BC(G)$ of a graph G is the bipartite graph whose one partite set consists of all the cut-vertices of G, and the other partite set consists of all the blocks of G, such that two vertices in $BC(G)$ are adjacent iff one of them is contained in the other.

Of course, for the definition of $BC(G)$ to make sense, G must have at least one cut-vertex. For instance, look at the graph G, and its block-cutpoint graph given below.



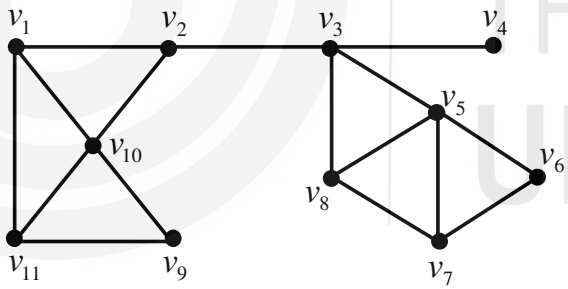
G



$BC(G)$

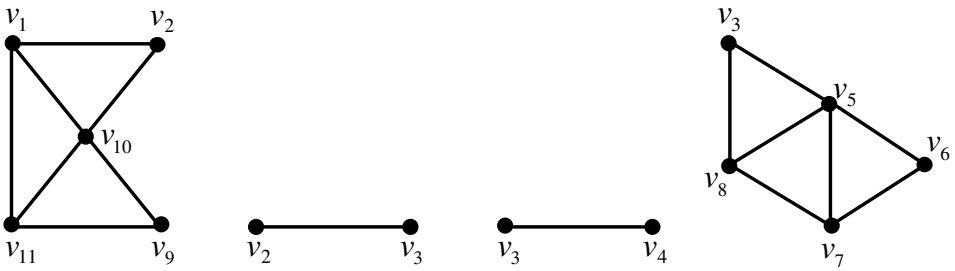
Let us look at one more example.

Example 10: Construct $BC(G)$ for the graph given G below.



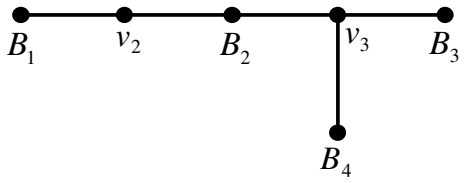
G

Solution: Observe that all the cut-vertices of G are $X = \{v_2, v_3\}$. The set of all the blocks of G is $Y = \{B_1, B_2, B_3, B_4\}$. The blocks are drawn as below.



B_1 B_2 B_3 B_4

Thus $BC(G)$ can be drawn as follows.



$BC(G)$

One thing you might have observed is that $BC(G)$ is a tree when G is connected. Indeed, this is true as stated below.

Proposition 1: If G is a connected graph with at least one cut-vertex, then $BC(G)$ is a tree. Moreover, every leaf of $BC(G)$ is a block of G.

However, every block of a graph G need not be a leaf of $BC(G)$, as we have seen in the examples above. We shall call a block of a graph G a **leaf block** of G if it is a leaf in $BC(G)$. Since $BC(G)$ has at least two leaves, and every leaf is a block of G, it follows that G has at least two leaf blocks.

Now, you try the following exercises.

E12) The blocks of a tree are precisely its edges. True or false? Justify.

E13) Give an example of a graph G with 5 blocks and

(i) 1 cut-vertex

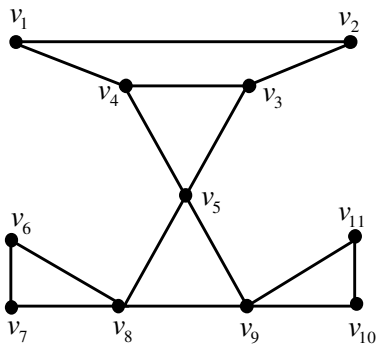
(ii) 2 cut-vertices

(iii) 3 cut-vertices

(iv) 4 cut-vertices.

E14) Let G be a 3-regular graph with a cut-vertex. Show that G has at least three blocks.

E15) Find the block-cutpoint graph of the graph G given below.



G

E16) Let G be a connected graph with blocks $B_1, B_2, \ldots, B_k$. Show that

$$n(G) = \sum_{i=1}^{k} n(B_i) - k + 1.$$

After having a basic understanding of the connectivity and blocks, we now move towards the characterisation of k-connected graphs.

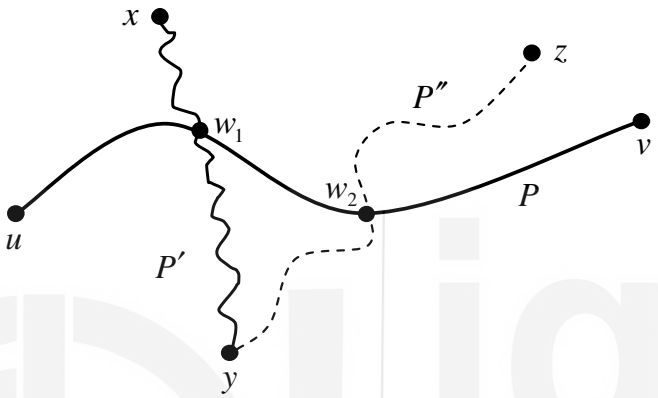
9.5 *k*-CONNECTED GRAPHS

We begin with the characterisation of 2-connected graphs, discovered by Whitney in 1932. But, first let us have a definition.

An **internal** vertex of a path is a vertex different from its endpoints.

Definition: Two paths P and P' are said to be **internally disjoint** if no internal vertex of P lies on P', and no internal vertex of P' lies on P.

For instance, look at the paths P, P' and P'' drawn below.

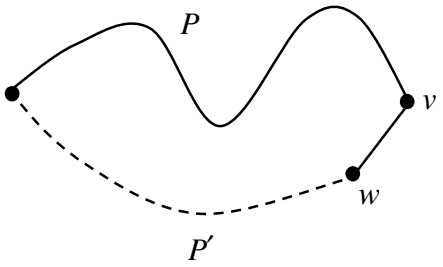


P joins the vertices u and v, P' joins x and y, and P'' joins y and z. Clearly, P and P' are not internally disjoint as both P and P' have w_1 as an internal vertex. Similarly, P and P'' are also not internally disjoint. However, P' and P'' are internally disjoint because their only common vertex is y, which is neither an internal vertex of P' nor of P''.
It is straight forward to see that if P and P' are two internally disjoint (u,v)-paths, then $P \cup P'$ is a cycle. Let us now look at the Whitney's characterisation of 2-connected graphs.

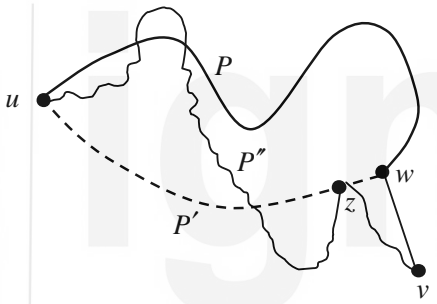
Theorem 2 (Whitney's Theorem): Let G be a graph with at least 3 vertices. Then G is 2-connected iff for every pair of vertices $u, v \in V(G)$, there are two internally disjoint (u,v)-paths in G.

Proof: First let us assume that G is 2-connected. We shall use the Principle of Mathematical Induction on $d(u,v)$ for any $u, v \in V(G)$ to show that G has two internally disjoint (u,v)-paths.
Pick any $u, v \in V(G)$, and assume that $d(u,v) = 1$. Then $uv \in E(G)$. Since $\kappa'(G) \geq \kappa(G)$ and $\kappa(G) = 2$, it follows that G has no cut-edges. This means $G - uv$ is connected. Therefore, G has a (u,v)-path say P, which is internally disjoint from the path formed by the edge uv and its endpoints.
Now assume that G has two internally disjoint (x,y)-paths when $d(x,y) = k$ for some $k \geq 2$. Consider a pair $u, v \in V(G)$ with $d(u,v) = k+1$. Let w be a vertex preceding v on a shortest (u,v)-path. Clearly, $d(u,v) = d(u,w) + d(w,v)$. Since $d(w,v) = 1$, we get $d(u,w) = k$. Now by the induction hypothesis, there are two internally disjoint (u,w)-paths, say P and P' in G.

Case 1: $v \in V(P) \cup V(P')$. Then v lies on exactly one of P and P'. Say v lies on P. Then $P - w$ and $P' + wv$ are two internally disjoint (u,v)-paths. (See the picture below.)



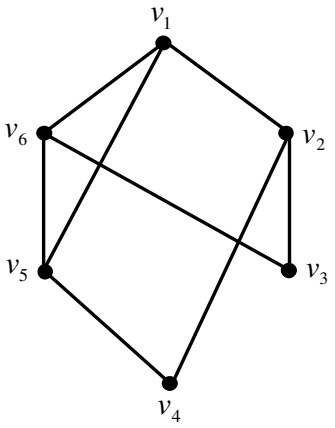
Case 2: $v \notin V(P) \cup V(P')$. Then since G is 2- connected, $G - w$ is connected. Hence there exists a (u,v) - path, say P'', which does not pass through w. Note that if P'' is internally disjoint from P or P' we are done. Otherwise, let z be the last vertex of P'' which lies on P or P'. Assume that z lies on P'. (See the picture below.)



In this case, the required internally disjoint (u,v)-paths are $P + wv$ and the path formed by joining the (u,z)-subpath of P' to (z,v)-subpath of P''. Thus we have shown G has two internally disjoint (u,v)-paths when $d(u,v) = k+1$. Therefore, by the Principle of Mathematical Induction, G has two internally disjoint (u,v)-paths for every pair $u,v \in V(G)$.
We are leaving the converse part as an exercise for you. (See E17) ■

Let us consider the following example.

Example 11: Using Whitney's Theorem show that the graph G given below is 2-connected.



G

Solution: The following table lists two internally disjoint (u,v)-paths for every pair of vertices u and v of G.

Pair (u,v)	Internally disjoint (u,v)-paths	
(v_1,v_2)	(v_1,v_2),	(v_1,v_5,v_4,v_2)
(v_1,v_3)	(v_1,v_2,v_3),	(v_1,v_6,v_3)
(v_1,v_4)	(v_1,v_2,v_4),	(v_1,v_5,v_4)
(v_1,v_5)	(v_1,v_5),	(v_1,v_6,v_5)
(v_1,v_6)	(v_1,v_6),	(v_1,v_5,v_6)
(v_2,v_3)	(v_2,v_3),	(v_2,v_1,v_6,v_3)
(v_2,v_4)	(v_2,v_4),	(v_2,v_1,v_5,v_4)
(v_2,v_5)	(v_2,v_4,v_5),	(v_2,v_1,v_6,v_5)
(v_2,v_6)	(v_2,v_1,v_6),	(v_2,v_3,v_6)
(v_3,v_4)	(v_3,v_2,v_4),	(v_3,v_6,v_5,v_4)
(v_3,v_5)	(v_3,v_6,v_5),	(v_3,v_2,v_4,v_5)
(v_3,v_6)	(v_3,v_6),	(v_3,v_2,v_1,v_6)
(v_4,v_5)	(v_4,v_5),	(v_4,v_2,v_1,v_5)
(v_4,v_6)	(v_4,v_5,v_6),	(v_4,v_2,v_1,v_6)
(v_5,v_6)	(v_5,v_6),	(v_5,v_1,v_6)

Consequently, G is 2-connected.

***

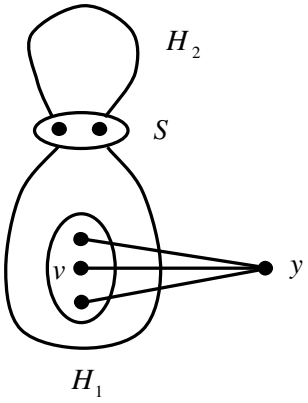
Now let us move towards the study of k-connected graphs. The following lemma tells us how to construct a larger k-connected graph from a given k-connected graph.

Lemma 1 (Expansion Lemma): Let G be a k-connected graph. Let G' be obtained by adding a new vertex, say y, to G and joining y to at least k vertices in G. Then G' is also k-connected.

Proof: Let us take an arbitrary vertex-cut S in G'. We will show that $|S| \geq k$.
Case 1: $y \in S$. Then $G-(S\setminus\{y\})$ is disconnected. (Why?) This means, $S\setminus\{y\}$ is a vertex-cut of G, and hence $|S\setminus\{y\}| \geq k$ as G is k-connected. So $|S| \geq k+1 > k$.

Case 2: $y \notin S$ and $N(y) \subseteq S$. Since $d_{G'}(y) \geq k$, we have $|S| \geq k$.

Case 3: $y \notin S$ and $N(y) \not\subseteq S$. Then $N(y)$ contains an element v which is not in S. Then y and v are adjacent in $G'-S$ and hence lie in some component say H_1, of $G'-S$. Let H_2 be another component of $G'-S$. Then $H_1\setminus\{y\}$ and H_2 are two components of $G-S$. Therefore, S is a vertex-cut of G. (See the following illustration.)

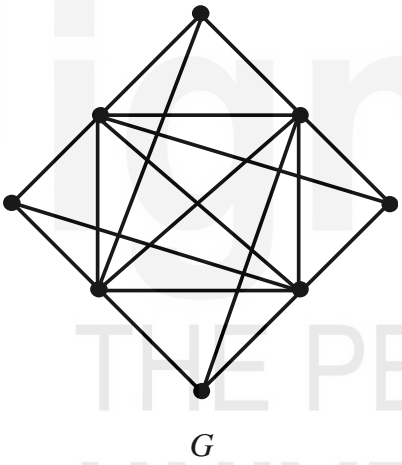


So, $|S| \geq k$, as G is k-connected.

Thus a vertex-cut of G' always contains at least k vertices. This proves the Expansion Lemma. ■

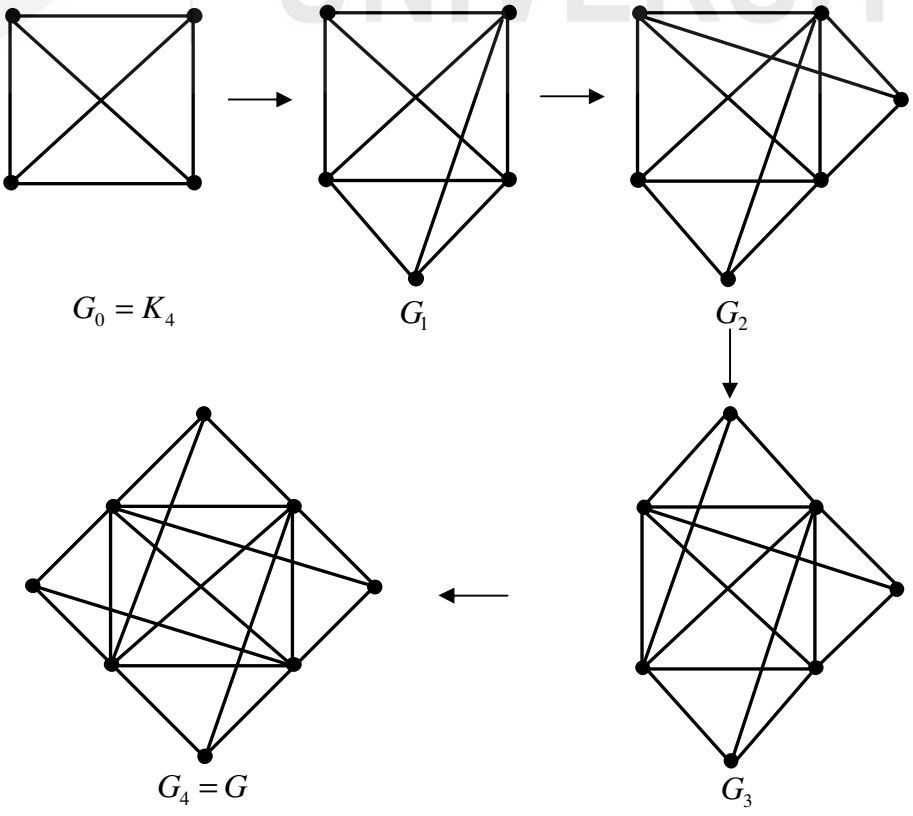
Let us see some examples.

Example 12: Using the Explanation Lemma, show that the graph G given below is 3-connected.



G

Solution: The graph G can be obtained from K_4 as follows.



At each stage of the construction, we have added a new vertex and joined it to 3 existing vertices. We know that K_4 is 3-connected. Thus, by the Expansion Lemma,

G_0 is 3-connected. $\Rightarrow G_1$ is 3-connected.

$\Rightarrow G_2$ is 3-connected.

$\Rightarrow G_3$ is 3-connected.

$\Rightarrow G_4 = G$ is 3-connected.

***

Now we state, without proof, a characterisation of k-connected graphs.

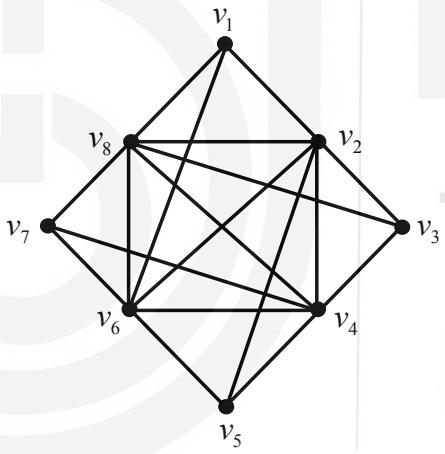
Theorem 3: A graph G is k-connected iff G has least $k+1$ vertices and for every pair of vertices u and v of G, there are at least k pairwise internally disjoint (u,v)-paths in G.

You can see that Whitney's Theorem is a special case of Theorem 3.

Let us now look at an example.

Example 13: Using Theorem 3, show that the graph given in Example 12 is 3-conneced.

Solution: Let us label the vertices of G as follows.



G

Since G has 8 vertices, there are $\binom{8}{2} = 28$ pairs of vertices of G. The following table lists 3 pairwise internally disjoint paths for the pairs (v_1, v_j) for $2 \le j \le 8$.

Pair (u,v)	3 pairwise internally disjoint (u,v)-paths		
(v_1,v_2)	(v_1,v_2),	(v_1,v_8,v_2),	(v_1,v_6,v_2)
(v_1,v_3)	(v_1,v_2,v_3)	(v_1,v_8,v_3),	(v_1,v_6,v_4,v_3)
(v_1,v_4)	(v_1,v_2,v_3,v_4),	(v_1,v_8,v_4),	(v_1,v_6,v_4)
(v_1,v_5)	(v_1,v_2,v_5),	(v_1,v_6,v_5),	(v_1,v_8,v_4,v_5)
(v_1,v_6)	(v_1,v_6),	(v_1,v_8,v_6),	(v_1,v_2,v_6)
(v_1,v_7)	(v_1,v_8,v_7),	(v_1,v_6,v_7),	(v_1,v_2,v_4,v_7)
(v_1,v_8)	(v_1,v_8),	(v_1,v_2,v_8),	(v_1,v_6,v_8)

The paths for the remaining 20 pairs can be constructed in the same way. Thus, we conclude that G is 3-connected.

***

Now try the following exercises.

E17) Prove the converse statement of Theorem 2.

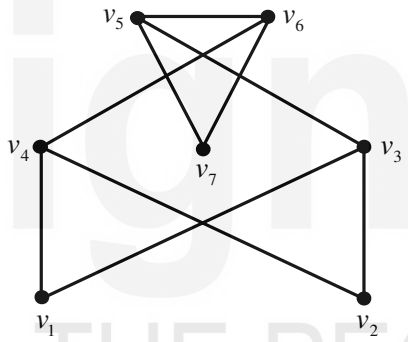
E18) If P is a (u,v)-path in a 2-connected graph G then G has another (u,v)-path that is internally disjoint from P. True or false? Justify.

E19) Show that every Hamiltonian graph is 2-connected. Is every Eulerian graph 2-conneced? Justify

E20) Show that the following graph is 2-connected, using

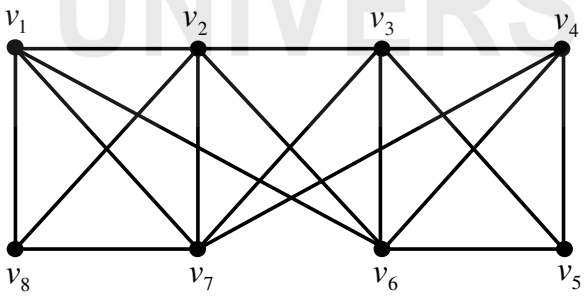
i) Whitney's Theorem,

ii) Expansion Lemma.



G

E21) Is the graph G given below 4-connected? Is G 3-connected? Justify your answers.



G

E22) Show that every 2-connected graph is 2-edge connected.

With this we come to the end of this unit.

9.6 SUMMARY

In this unit we have covered the following points:

1. The concepts of vertex-cuts, and edge-cuts, are defined for a graph.

2. The parameters such as vertex-connectivity and edge-connectivity of a graph are studied.

3. The concepts of block and block-cutpoint graph are introduced.

4. An inequality between the parameters $\kappa(G), \kappa'(G)$ and $\delta(G)$ is established.

5. Some properties and characterisation of k-connected graphs are studied.

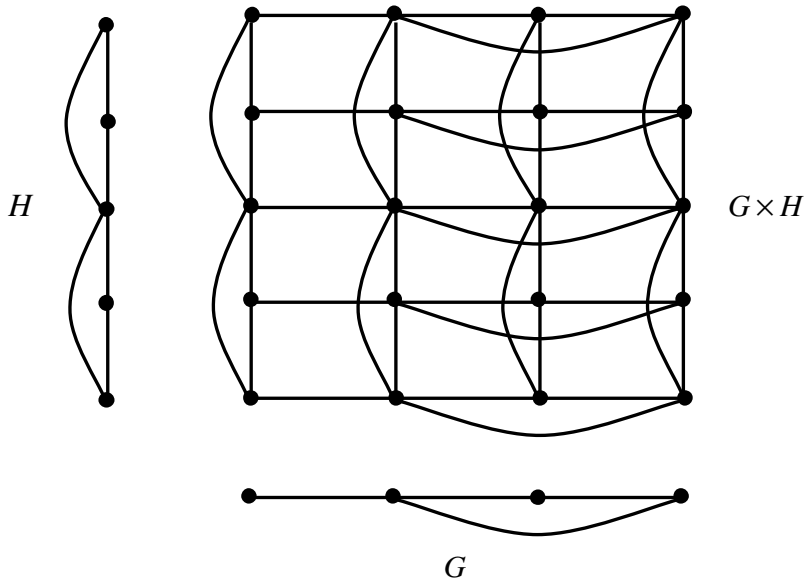
9.7 SOLUTIONS / ANSWERS

E1) First you check that no vertex of G is a cut-vertex. Therefore, $\kappa(G) \geq 2$. A vertex-cut of size 2 is $S = \{u_5, u_6\}$. Therefore, $\kappa(G) = 2$.

E2) Define $f : V(G \times H) \to V(H \times G)$ by $f(u,v) = (v,u)$. Then f is one-one and onto. Now, let $(u_1, v_1), (u_2, v_2) \in V(G \times H)$ be two adjacent vertices. Then either ($u_1 = u_2$ and $v_1 \leftrightarrow v_2$) or ($u_1 \leftrightarrow u_2$ and $v_1 \leftrightarrow v_2$). This is equivalent to either ($v_1 = v_2$ and $u_1 \leftrightarrow u_2$) or ($v_1 \leftrightarrow v_2$ and $u_1 = u_2$). This means $(v_1, u_1), (v_2, u_2)$ are adjacent in $H \times G$. Conversely, consider any two adjacent vertices in $H \times G$ and show that their inverse images under f are adjacent in $G \times H$. This proves that $G \times H \cong H \times G$.

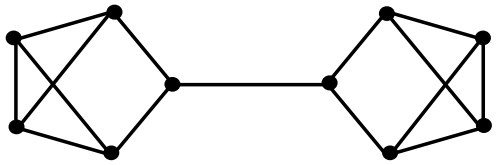
E3) The wheel graph W_n consists of a cycle C on $n-1$ vertices and a vertex v that is adjacent to every vertex of C. Clearly removal of a single vertex from W_n leaves a connected graph. If we remove any two vertices from W_n, then also we get a connected graph. Thus $\kappa(W_n) \geq 3$. However, if we remove v and any two nonadjacent vertices on C, then the resulting graph is disconnected. This means $\kappa(W_n) = 3$, for all $n \geq 4$.

E4) We draw G along the x-axis and H along the y-axis, and $G \times H$ on the xy-plane as follows.



E5) True. Why?

E6) One such graph is drawn below.

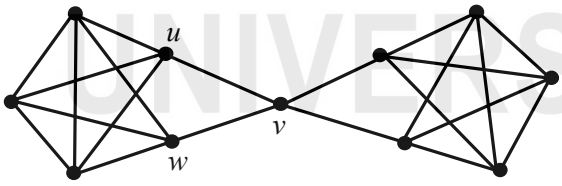


E7) Removal of any vertex from C_n leaves a path on $n-1$ vertices, which is connected. Therefore, $\kappa(C_n) \geq 2$. If we remove any two nonadjacent vertices from C_n we get a disconnected graph. Therefore, $\kappa(C_n) = 2$, for all $n \geq 3$.

E8) If $\kappa(G) = 0$, then G is disconnected, and hence $\kappa'(G) = 0$. Now assume $\kappa(G) = 1$. Then G has a cut-vertex, say u. let H be a component of $G-u$ that has exactly one neighbour of u, say v, (Why does such a component exist?) Then uv is a cut-edge of G. Therefore, $\kappa'(G) \leq 3$.
The case $\kappa(G) = 3$ follows trivially from the inequality $\kappa(G) \leq \kappa'(G) \leq 3$.

E9) On the contrary assume that $\Delta(G) \leq 3$. Then $\delta(G) \leq 3$. Hence $\kappa(G) < \kappa'(G) \leq 3$ implies that $\kappa(G) \leq 2$. The cases $\kappa(G) = 0$ and $\kappa(G) = 1$ easily give contradictions. The case $\kappa(G) = 2$ is similar to the proof of Corollary 1.

E10) One such graph is given below.



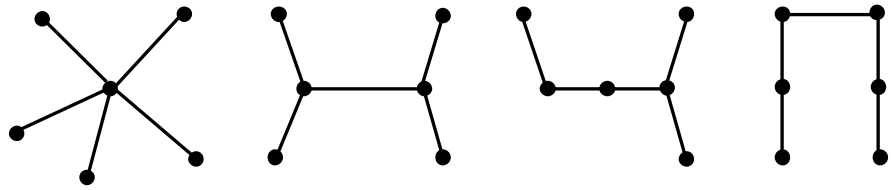
G

Note that $\delta(G) = 4$. (In fact, G is 4-regular as $\Delta(G) = 4$.) If we remove v from G, we get a disconnected subgraph. Therefore, $\kappa(G) = 1$. Note that G has no cut-edge. If we remove the edges uv and vw from G, we get a disconnected graph. Therefore, $\kappa'(G) = 2$.

E11) $K_{4,4}$ is one such graph.

E12) True. Every edge belongs to exactly one block. If any block of a tree T contains more than one edge, then it would have a cut-vertex of its own, which is a contradiction. Therefore, every block of T is an edge of T.

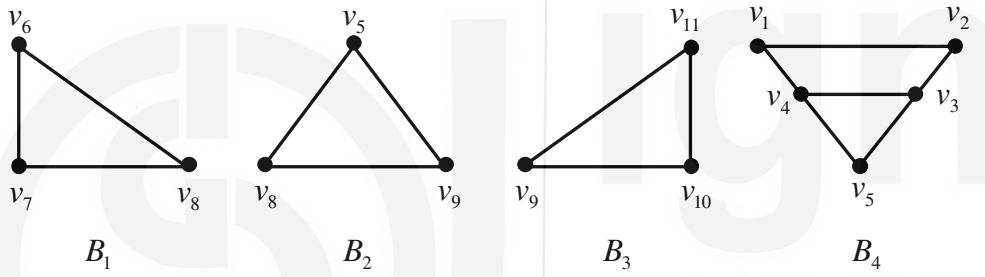
E13) The required graphs are drawn below.



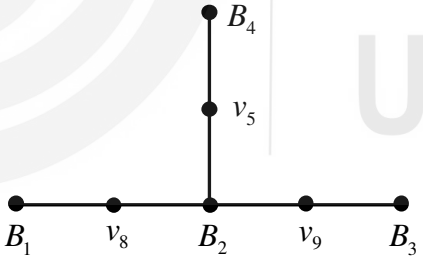
(i) (ii) (iii) (iv)

E14) Let v be a cut-vertex of G. Since G is 3-regular, there are three neighbours of v, say x, y and z. Since $G-v$ has at least 2 components, each components cannot have more than one neighbour. That is some component, say H_1 has exactly one neighbour, say x. Since $d(x)=3$, there are two more neighbours of x, say u and w. Note that every path from u to v or from w to v passes through x. That means, x is also a cut-vertex of G. Therefore, G has at least 3 blocks.

E15) The cut-vertices of G are $X=\{v_5, v_8, v_9\}$. The blocks of G are as drawn below.



Thus the block-cutpoint graph is



$BC(G)$

E16) We shall prove it by applying the Principle of Mathematical Induction on the number of blocks of G. Assume that G has only one block, say B. Then $n(G)=n(B)-1+1$. So, the induction step is true.
Now assume that the result is true for all connected graphs with k blocks for some $k \geq 2$. Consider a connected graph G with $k+1$ blocks $B_1, B_2, \ldots, B_{k+1}$. We know that G has at least one leaf block, and let it be B_{k+1}. Remove all the vertices of B_{k+1} except the cut-vertex, and let the resulting graph be H. Clearly H is connected, and has k blocks. Therefore, by the induction hypothesis, we have

$$n(H)=\sum_{i=1}^{k}(B_i)-k+1.$$

Note that

$$n(G) = n(H) + n(B_{k+1}) - 1$$

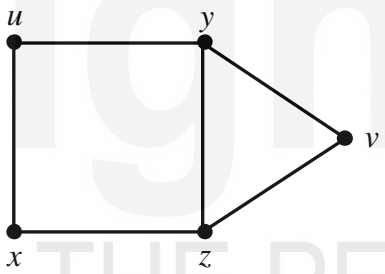
$$= \left(\sum_{i=1}^{k} n(B_i) - k + 1 \right) + n(B_{k+1}) - 1$$

$$= \sum_{i=1}^{k+1} n(B_i) - (k+1) + 1.$$

This proves that the result holds for G. Therefore, by the Principle of Mathematical Induction, the result holds for all connected graphs.

E17) Let G be a graph with at least 3 vertices such that for every pair of vertices u and v of G, there are 2 internally disjoint (u,v)-paths in G. We shall show that $\kappa(G) \geq 2$. Assume, if possible, that G has a cut-vertex x. Then there exists a pair $u, v \in V(G) \setminus \{x\}$ such that every (u,v)-path passes through x. This means, G does not have 2 internally disjoint (u,v)-paths. This is a contradiction. Therefore, $\kappa(G) \geq 2$.

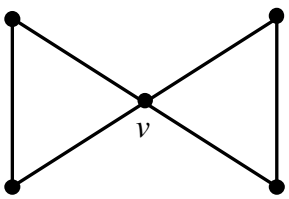
E18) False. For instance, look at the graph G given below.



G

Note that G is 2-connected. (Just prove it.)
Consider a (u,v)-path $P = (u, x, y, v)$ in G. Then every (u,v)-path different from P, has either x or y.

E19) Let G be a Hamiltonian graph. Then G has a Hamiltonian cycle C. Let $u, v \in V(G)$. Then $u, v \in V(C)$. Therefore, G has two internally disjoint (u,v)-paths. Since u and v are arbitrary, by Theorem 2, G is 2-connected.
Every Eulerian graph is not 2-connected. For instance, look at the graph G given below.

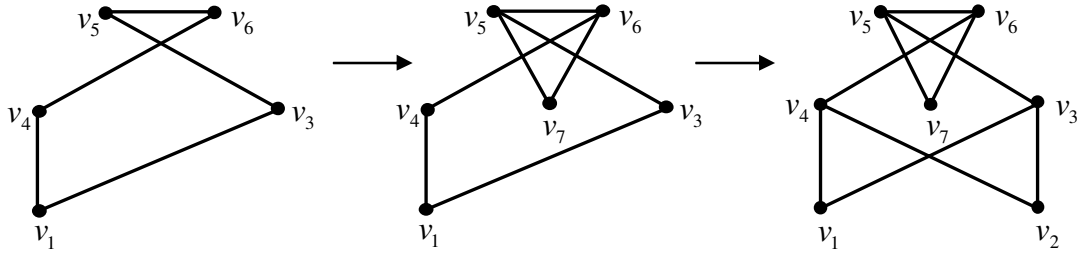


G

Justify yourself why G is Eulerian but not 2-connected.

E20) i) Try yourself. (Similar to Example 11.)

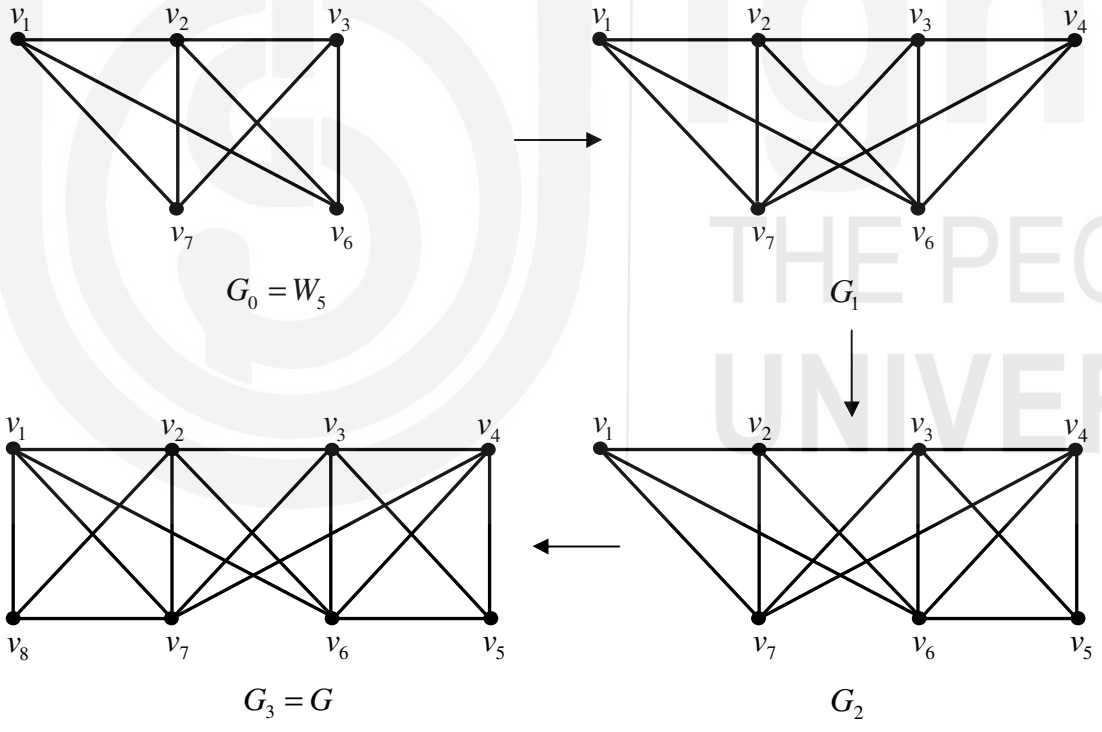
ii) We construct G from C_5 as follows.



$G_0 = C_5$ G_1 $G_2 = G$

At each stage, we have added a new vertex and joined it with two existing vertices. Since C_5 is 2-connected, G_1 is 2-connected, which implies G_2 is 2-connected. Hence, G is 2-connected.

E21) We know that $\kappa'(G) \leq \delta(G)$, and from Theorem 1, $\kappa(G) \leq \kappa(G)$. Therefore, if G is 4-connected, then $\kappa(G) \geq 4$, which implies $\delta(G) \geq 4$. However, G has a vertex of degree 3. Therefore, G is not 4-connected. Now, we check whether G is 3-connected. We construct G from W_5 as follows.



$G_0 = W_5$ G_1 $G_3 = G$ G_2

We know, from E3, that W_5 is 3-connected. Therefore, by the Expansion Lemma.

G_0 is 3-connected $\Rightarrow$ G_1 is 3-connected.

$\Rightarrow$ G_2 is 3-connected.

$\Rightarrow$ $G_3 = G$ is 3-connected.

E22) Assume, on the contrary, that G is not 2-edge-connected. Then G has a cut-edge say $e = uv$. To reach at a contradiction we shall show that v is a cut-vertex of G. Since G is 2-connected, $\delta(G) \geq 2$.

Therefore, let w be a neighbour of v different from u. Since e is a cut-edge, by Theorem 5 of Unit 1, e does not lie in any cycle. Therefore, every (u, w)-path in G passes through v. Then, by Theorem 4 of Unit 1, v is a cut-vertex. This is a contradiction. Hence, G is 2-connected.

UNIT 10

NETWORK FLOWS

Structure **Page Nos.**

10.1	Introduction Objectives	275
10.2	Network Flows	275
10.3	Ford-Fulkerson Algorithm	283
10.4	Max-Flow Min-Cut Theorem	284
10.5	Summary	292
10.6	Solutions / Answers	292

10.1 INTRODUCTION

This is the last unit of this course. We shall introduce here the concept of network flows and discuss how to solve the maximum flow problem. We shall formally define the concept of network and network flow in Section 10.2. We shall also introduce here the augmenting chains in a network, and see how they can be used to produce larger flows. Then in Section 10.3, we talk about the Ford-Fulkerson Algorithm. Finally, in Section 10.4, we discuss the Max-Flow Min-Cut Theorem.

Objectives

After studying this unit, you should be able to

- describe what is a network, and what is a flow on a network;
- describe maximum flow problem;
- identify whether a chain in a network with a flow f, is an f-augmenting chain or not;
- describe, and apply the Ford-Fulkerson Algorithm;
- state and apply the Max-Flow Min-Cut Theorem.

10.2 NETWORK FLOWS

Many real life systems such as water or power distribution systems, transportation or communication systems are often analysed through network models. Such networks are equipped with some flow of information or material which usually takes place in one direction–from source to sink. In

communications system, however, the flow takes place in both directions (source to sink and sink to source), and some systems may even have multiple sources and multiple sinks. Here we confine to one direction flow namely from source to sink, with one source and one sink only.

To represent a network we shall use the concept of weighted directed graphs. Intuitively, a network is a directed graph with two distinguished vertices called "source" and "sink", and where each edge is assigned a "capacity" which tells us how much amount of material or information can pass through the edge, in a given time.

Note that a directed graph $G = (V, E)$ is **asymmetric** if $uv \in E$ implies that $vu \notin E$. In this unit we shall assume that all the directed graphs are asymmetric. Let us, now, start with the formal definition of network.

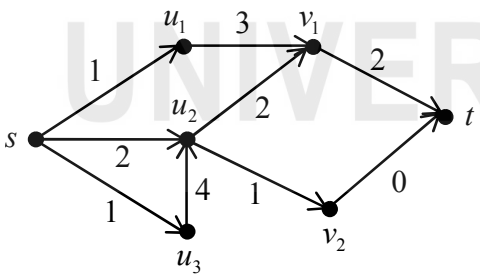
Definition: A **network** is a 4-tuple $N = (G, c, s, t)$, where:

i) G is a directed graph, with node set V and edge set E.

ii) $c : E \to \mathbb{R}$ is a **capacity** function, which assigns a nonnegative value $c(e)$ to each edge e. We call $c(e)$ the **capacity** of the edge e.

iii) s and t are two distinct nodes of G, where s is called the **source,** and t is called the **sink**.

A network can also be shortened as N. For instance, the following graph represents a network, where s and t are its source and sink vertices, respectively.



A network N

Here, the capacity of the edge su_1 is $c(su_1) = 1$. Likewise, $c(v_2t) = 0$. The maximum capacity is possessed by the edge u_3u_2, which is $c(u_3u_2) = 4$. If you treat an edge just as a pipeline, then its capacity would represent the maximum amount of liquid it can pass through at a given time.

Now we define the flow in a network.

Definition: Given a network $N = (G, c, s, t)$ a **flow** on N is a function $f : E \to \mathbb{R}$ such that

i) **(Capacity Constraints)** $0 \le f(e) \le c(e) \qquad \forall e \in E,$

ii) **(Conservation Constraints)** $f^-(v) = f^+(v)\ \forall v \in V \setminus \{s,t\}$, where $f^-(v) = \sum_{uv \in E} f(uv)$ and $f^+(v) = \sum_{vw \in E} f(vw)$, for any $v \in V$.

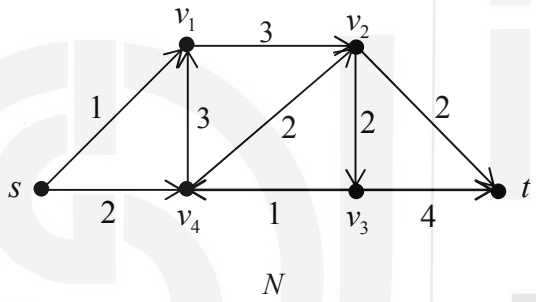
The first condition tells us that the flow on any edge cannot exceed the edge capacity. In the second condition, we can call $f^-(v)$ the **flow entering into** v, and $f^+(v)$ the **flow leaving** v. Thus the flow leaving a vertex is always equal to the flow entering into the vertex, for all vertices except s and t. This is equivalent to Kirchhoff's law for electrical networks.

A flow with value zero is called a **zero flow**.

We define the **value** of a flow f on N as $\text{val}(f) = f^-(t) - f^+(t)$, which is the net flow entering into the sink t. A flow f on a network N is said to be **maximum** if $\text{val}(f) \geq \text{val}(f')$ for all flows f' on N.

Let us consider an example.

Example 1: Look at the following network N.



Note that for any flow f on a network N with source s and sink t, we shall always assume that $f^-(s) = 0$ and $f^+(t) = 0$.

Define a function $f : E \to \mathbb{R}$ as follows:

$f(sv_1) = 0,\ f(sv_4) = 2,\ f(v_1v_2) = 3, f(v_2v_3) = 1,$

$f(v_3v_4) = 1, f(v_4v_1) = 3,\ f(v_4v_2) = 0, f(v_2t) = 2,\ f(v_3t) = 0.$

Show that f is a flow on N.

Solution: First we check the capacity constraints.

$0 \leq f(sv_1) \leq c(sv_1) = 1,\ 0 \leq f(sv_4) = 2 = c(sv_4),$

$0 \leq f(v_1v_2) = 3 = c(v_1v_2), 0 \leq f(v_2v_3) = 1 \leq 2 = c(v_2v_3),$

$0 \leq f(v_3v_4) = 1 = c(v_3v_4), 0 \leq f(v_4v_1) = 3 = c(v_4v_1),$

$0 \leq f(v_4v_2) = 0 \leq 2 = c(v_4v_2), 0 \leq f(v_2t) = 2 = c(v_2t),$

$0 \leq f(v_3t) = 0 \leq 4 = c(v_4t).$

Thus $0 \leq f(e) \leq c(e)$ holds for all $e \in E$. Now we check the conservation constraints.

$f^-(v_1) = f(sv_1) + f(v_4v_1) = 0 + 3 = 3$
$f^+(v_1) = f(v_1v_2) = 3$
$\therefore f^-(v_1) = f^+(v_1)$

$f^-(v_2) = f(v_1v_2) + f(v_4v_2) = 3 + 0 = 3$
$f^+(v_2) = f(v_2v_3) + f(v_2t) = 1 + 2 = 3$
$\therefore f^-(v_2) = f^+(v_2)$

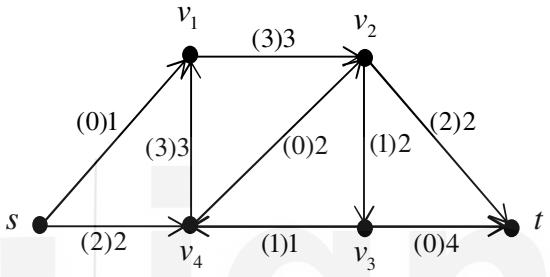
$$f^-(v_3) = f(v_2v_3) = 1$$
$$f^+(v_3) = f(v_3v_4) + f(v_3t) = 1 + 0 = 1 \quad \Big| \quad \therefore f^-(v_3) = f^+(v_3)$$

$$f^-(v_4) = f(sv_4) + f(v_3v_4) = 2 + 1 = 3$$
$$f^+(v_4) = f(v_4v_1) + f(v_4v_2) = 3 + 0 = 3 \quad \Big| \quad \therefore f^-(v_4) = f^+(v_4)$$

Thus $f^-(v) = f^+(v)$ holds for all $v \in V \setminus \{s,t\}$. Therefore, f is a flow on N.

***

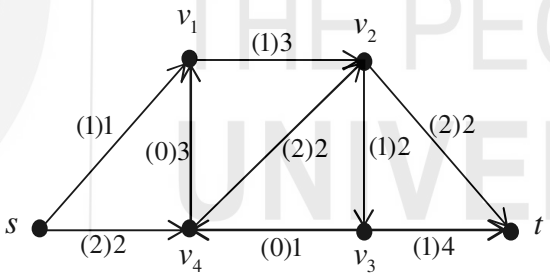
It is often convenient to represent the flow amount (in parentheses) on the edges alongside the edge capacities. For instance, the flow of Example 1 can be represented as follows:



The value of flow f on the network above is

$$\text{val}(f) = f^-(t) - f^+(t) = 2 - 0 = 2.$$

We can obtain a flow g with value larger than $\text{val}(f)$ as follows.



A flow with value 3

You can verify yourself that g is a flow. We have

$$\text{val}(g) = g^-(t) - g^+(t) = 2 + 1 - 0 = 3.$$

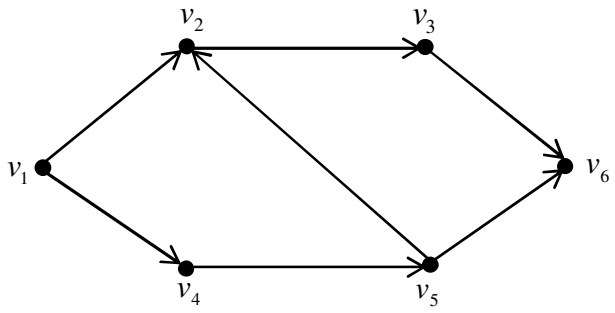
Thus g is a larger flow than f. In fact, g is a maximum flow because the value of g equals the upper bound $g^+(s) = g(sv_1) + g(sv_4) = 1 + 2 = 3.$

Finding a maximum flow in a network is a well known problem called the **Maximum Flow Problem**. In the next section we shall discuss an algorithm to find a solution to this problem. This algorithm uses the concept of f-augmenting chain, which we shall define shortly.

A **chain** in a directed graph G is an alternating sequence of vertices and edges of the form

$$(v_0, e_1, v_1, e_2, \ldots, e_k, v_k),$$

where $v_0, v_1, v_2, \ldots, v_k$ are the vertices, and $e_1, e_2, \ldots, e_k$ are the edges in G such that $e_i = v_{i-1}v_i$ for each i. Since between any two vertices of a directed graph there is at most one edge. We can simply write a chain as a sequence of vertices. A chain that starts at u and ends at v is called a (u,v)-**chain**. For instance, look at the following directed graph G.

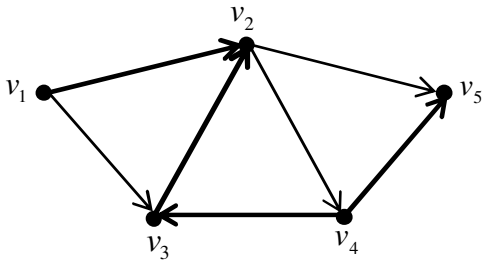


G

Let $P = (v_1, v_2, v_3, v_6)$. We can see that the successive vertices in P are adjacent. So, P is a (v_1, v_6)-chain of G. But, (v_1, v_3, v_6) is not a chain. Observe that $P' = (v_1, v_2, v_5, v_6)$ is also a (v_1, v_6)-chain. How does P' differ from P? All the edges of P have the same direction, which is the forward direction. In other words, P follows every edge in the forward direction. However, this is not the case with P'. The edge v_1v_2 of P' is followed in the forward direction, v_5v_2 in the backward direction, and then v_5v_6 in the forward direction. This leads to the following definition.

Definition: Let $P = (v_0, v_1, v_2, \ldots, v_k)$ be a chain in a directed graph G. An edge e of P is said to be **forward** if $e = v_iv_{i+1}$ for some i, otherwise e is called a **backward** edge.

For instance, $P = (v_1, v_2, v_3, v_4, v_5)$ is a chain in the following directed graph G.



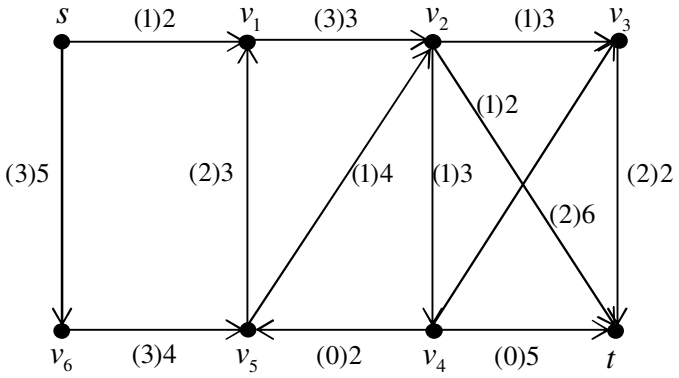
G

P is shown in G through bold edges. The forward edges of P are v_1v_2 and v_4v_5, whereas the backward edges are v_3v_2 and v_4v_3.

Definition: Given a flow f on a network $N = (G, c, s, t)$ an f-**augmenting chain** is an (s,t)-chain P in N such that for each edge e of P, $f(e) < c(e)$ if e is forward, and $f(e) > 0$ if e is backward.

Let us consider an example.

Example 2: Let N be the network with a flow f as given below.



A network N with a flow f

Which of the following chains in N, are f-augmenting chains?

i) $P = (s, v_1, v_5, v_4, t)$

ii) $P' = (s, v_6, v_5, v_2, v_3, v_4, t)$

iii) $P'' = (s, v_1, v_2, v_3, t)$

Solution: i) The forward edges of P are $s v_1, v_4 t$ and the backward edges are $v_1 v_5, v_5 v_4$. Since $f(v_5 v_4) = 0$, P is not an f-augmenting chain.

ii) The forward edges of P' are $s v_6, v_6 v_5, v_5 v_2, v_2 v_3, v_4 t$ and the backward edge is $v_4 v_3$. So, we have
$f(s v_6) = 3 < c(s v_6), f(v_6 v_5) = 3 < c(v_6 v_5), f(v_5 v_2) = 1 < c(v_5 v_2),$
$f(v_2 v_3) = 1 < c(v_2 v_3), f(v_4 t) = 0 < c(v_4 t).$
Also, $f(v_4 v_3) = 1 > 0$. Thus, P' is an f-augmenting chain.

iii) Note that $v_1 v_2$ is a forward edge of P'' but $f(v_1 v_2) = 3 = c(v_2 v_3)$.
Therefore, P'' is not an f-augmenting chain.

***

If you have understood the concept of an f-augmenting chain, we can proceed to see how the existence of an f-augmenting chain in a network can lead to a flow with larger value.

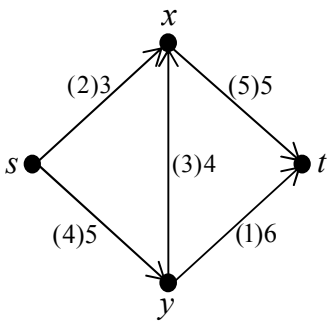
But before that we have one more definition.

Definition: Let $N = (G, c, s, t)$ be a network with a flow f. Let P be an f-augmenting chain in N. Then for each $e \in E(P)$, define the **residual capacity** of e as

$$c_f(e) = \begin{cases} c(e) - f(e), & \text{if } e \text{ is forward} \\ f(e), & \text{if } e \text{ is backward} \end{cases}$$

The **residual capacity** of P is the quantity $c_f(P) = \min\{c_f(e) : e \in E(P)\}$.

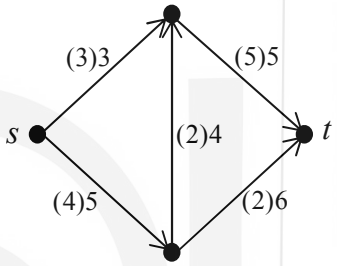
For instance, look at the following network N with a flow f.



Note that $\text{val}(f) = 6$. Here, $P = (s, x, y, t)$ is an f-augmenting chain in N. (Just verify.) Now the residual capacity of P is

$$c_f(P) = \min\{c_f(sx), c_f(xy), c_f(yt)\}$$
$$= \min\{3-2, 3, 6-1\} = 1.$$

Now increase the flow by 1 on the forward edges of P and decrease the flow by 1, on the backward edges of P. This gives us a new flow g as shown below.



Clearly, $\text{val}(g) = 7 > \text{val}(f)$.

Thus, the existence of an f-augmenting chain leads to a larger flow. This fact is recorded in the following lemma.

Lemma 1: Let $N = (G, c, s, t)$ be a network with a flow f, and let P be an f-augmenting chain in N. Define $g : E \to \mathbb{R}$ as follows:

$g(e) = f(e)$ if $e \notin E(P)$, and for each $e \in E(P)$

$$g(e) = \begin{cases} f(e) + c_f(P), & \text{if } e \text{ is forward} \\ f(e) - c_f(P), & \text{if } e \text{ is backward} \end{cases}$$

Then g is a flow on N with $\text{val}(g) = \text{val}(f) + c_f(P)$.

Proof: Since $c_f(e) > 0$ for all $e \in E(P)$, $c_f(P) > 0$. Consider any $e \in E(P)$.

When e is forward $c_f(P) \le c(e) - f(e)$, which gives

$g(e) = f(e) + c_f(P) \le c(e)$.

On the other hand, when e is backward then $c_f(P) \le f(e)$, which gives

$$g(e) = f(e) - c_f(P) \le c(e) - c_f(P) < c(e).$$

Thus, $g(e) \le c(e)$ for all $e \in E$. The inequality $g(e) \ge 0$ holds trivially. Thus, g satisfies the capacity constraints. Now we check the conservation constraints.

Let $v \in V \setminus \{s, t\}$. If $v \notin V(P)$, then the flow on the edges incident on v remain unaffected. Therefore, $g^-(v) = g^+(v)$ holds. Now assume that

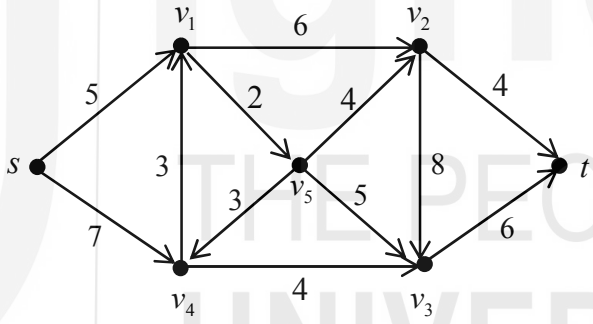
$v \in V(P) \setminus \{s,t\}$. Then there are two edges incident on v. If both these edges have v as the head, then the flow increases on one edge and decreases on the other by the same amount. The same is the case when v is the tail of both these edges. When v is the tail of one edge and the head of the other, then also the flow remains unaffected. Thus $g^-(v) = g^+(v)$ for all $v \in V \setminus \{s,t\}$. This proves that g is a flow on N. Now

$$\begin{aligned}
\text{val}(g) &= g^-(t) \\
&= \sum_{u \in V} g(ut) \\
&= \sum_{u \in V \setminus V(P)} f(ut) + g(xt), \text{ where } x \in V(P) \\
&= \sum_{u \in V \setminus V(P)} f(ut) + f(xt) + c_f(P) \\
&= \sum_{u \in V} f(ut) + c_f(P) \\
&= f^-(t) + c_f(P) \\
&= \text{val}(f) + c_f(P).
\end{aligned}$$

■

Now, try the following exercises.

E1) Consider the network $N = (G,c,s,t)$ as drawn below.



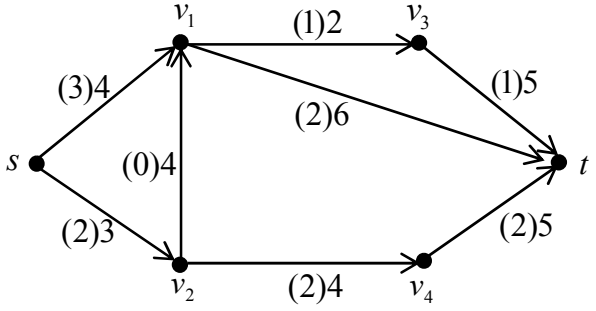
Define a function $f : E \to \mathbb{R}$ as follows:

$f(sv_1) = 3, f(sv_4) = 5, f(v_4v_1) = 3, f(v_1v_2) = 5, f(v_1v_5) = 2, f(v_5v_4) = 1,$
$f(v_2v_3) = 5, f(v_4v_3) = 2, f(v_2t) = 3, f(v_3t) = 5, f(v_5v_2) = 2, f(v_5v_3) = 2$

Check whether f is a flow or not.

E2) Given two flows f and g on a network N, is $\max\{f,g\}$ a flow on N? Justify your answer.

E3) Consider a flow f on a network N as given below.



Using Lemma 1, find a flow on N larger than f.

Thus far we have seen basic definitions and results concerning network flows. Next, we shall see how to find a maximum flow on a network.

9.3 FORD-FULKERSON ALGORITHM

From Lemma 1, of the previous section we have understood that to get a flow larger than a given flow f on a network, we need to find an f-augmenting chain in the network. However, finding an f-augmenting chain can be a difficult task if the network at hand is large. The mathematicians L.R. Ford and D.R. Fulkerson devised a beautiful algorithm for finding an f-augmenting chain, in a network with a given flow f. If there exists no such chain, then the algorithm returns a set S which can be used to show that f is a maximum flow. This algorithm is known as the **Ford-Fulkerson Algorithm**, and its procedure is given below.

Procedure (Ford-Fulkerson Algorithm)

Input: A network $N=(V,E,c,s,t)$ with a flow f.

Output: An f-augmenting chain, if it exists, or a set S.

Step 1 Let $R=\{s\}$ and $S=\emptyset$.

Step 2 If $R\setminus S\neq\emptyset$, pick a vertex u from $R\setminus S$. Otherwise, return S and stop.

Step 3 For each edge e incident on u, let v be the other endpoint of e. If $v\notin R$ and any one of the following is true

i) $v=$ head (e) and $f(e)<c(e)$

ii) $v=$ tail (e) and $f(e)>0$

then

i) add v to R

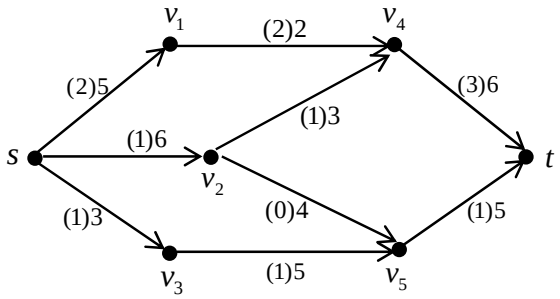
ii) mark u as the parent of v.

iii) if $v=t$, construct an f-augmenting chain P by going backward from t to s, then return P and stop.

Step 4 Add u to S, and go to Step 2.

We illustrate the algorithm with the following example.

Example 3: Consider a network N with a flow f as given below.



Find an f-augmenting chain in N using the Ford-Fulkerson Algorithm.

Solution: The steps involved in the algorithm are shown in the table given below.

R	S	Vertex-chosen	(vertex, parent) pairs
$\{s\}$	$\emptyset$	s	$(v_1,s),(v_2,s),(v_3,s)$
$\{s,v_1,v_2,v_3\}$	$\{s\}$	v_1	
$\{s,v_1,v_2,v_3\}$	$\{s,v_1\}$	v_2	$(v_4,v_2),(v_5,v_2)$
$\{s,v_1,v_2,v_3,v_4,v_5\}$	$\{s,v_1,v_2\}$	v_3	
$\{s,v_1,v_2,v_3,v_4,v_5\}$	$\{s,v_1,v_2,v_3\}$	v_4	(t,v_4)
$\{s,v_1,v_2,v_3,v_4,v_5,t\}$			

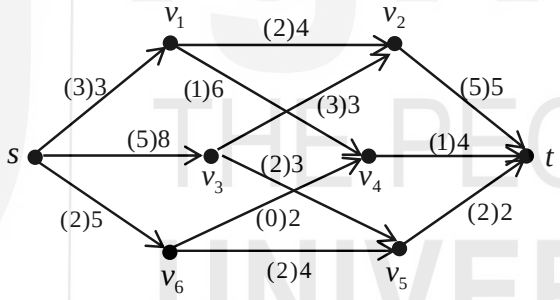
Since $t \in R,$ we stop. The f -augmenting chain P is constructed from the last column, by going backward from $t \to v_4 \to v_2 \to s.$ That is,

$P = (s, v_2, v_4, t).$

***

Now try the following exercise.

E4) Using the Ford-Fulkerson Algorithm, find an f -augmenting chain in the network N with a flow f as given below.



Hence, construct a flow on N larger than $f.$

The Ford-Fulkerson Algorithm is a very useful method for finding a maximum flow in a network, as you will learn in the next section.

10.4 MAX-FLOW MIN-CUT THEOREM

In this section, we shall discuss an important theorem that establishes the equality of a maximum flow and the minimum capacity of a source-to-sink cut in a network. Of course, we shall first define what is a "source-to-sink" cut and what is its capacity.

Definition: Given a network $N = (V, E, c, s, t),$ a **source-to-sink cut** or an (s,t)**-cut** is a set of the form $[S,T]$ which consists of the edges from S to T such that

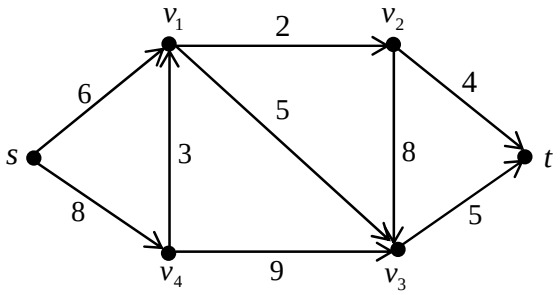
i) $s \in S,\ t \in T$

ii) $S \cup T = V$

Symbolically,
$[S,T] = \{uv \in E \mid u \in S, v \in T\}.$

iii) $S \cap T = \emptyset$.

Let us look at the following network N with $V = \{s, v_1, v_2, v_3, v_4, t\}$.



N

Take $S = \{s, v_1, v_4\}$ and $T = \{v_2, v_3, t\}$. Then $[S,T] = \{v_1v_2, v_1v_3, v_4v_3\}$. Clearly $s \in S$, $t \in T$. Also $S \cup T = V$ and $S \cap T = \emptyset$. Thus $[S,T]$ is an (s,t)-cut in N. Another (s,t)-cut, is the set $[S, \overline{S}]$, where $S = \{s\}, \overline{S} = V \setminus S = \{v_1, v_2, v_3, v_4, t\}$.

Next, we define the capacity of an (s,t)-cut.

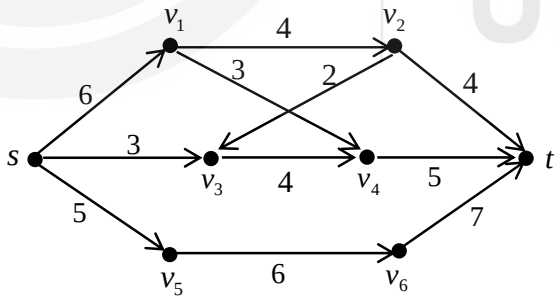
Definition: The **capacity** of an (s,t)-cut $[S,T]$ in a network $N = (G, c, s, t)$ is the quantity $\text{cap}(S,T)$ defined as:

$$\text{cap}(S,T) = \sum_{e \in [S,T]} c(e).$$

Observe that $[S, \overline{S}]$ is always an (s,t)-cut, whenever $s \in S, t \notin S$.

Thus the capacity of an (s,t)-cut is simply the sum of the capacities of its edges. One is often interested in (s,t)-cuts with smaller capacities. Let us look at the following example.

Example 4: Consider the network $N = (G, c, s, t)$ as given below.



N

Show that $[S,T]$ is an (s,t)-cut in N, where $S = \{s, v_1, v_3, v_5\}$ and $T = \{v_2, v_4, v_6, t\}$. Also find $\text{cap}(S,T)$. Does N have any other (s,t)-cut with capacity smaller than $\text{cap}(S,T)$?

Why $v_2v_3 \notin [S,T]$?

Solution: We can see that $S \cup T = V$ and $S \cap T = \emptyset$. Also $s \in S$ and $t \in T$. Therefore, $[S,T]$ is an (s,t)-cut in N. We have $[S,T] = \{v_1v_2, v_1v_4, v_3v_4, v_5v_6\}$. So, the capacity of $[S,T]$ is

$$\text{cap}(S,T) = c(v_1v_2) + c(v_1v_4) + c(v_3v_4) + c(v_5v_6)$$

$= 4 + 3 + 4 + 6 = 17.$

Now, we construct another (s,t)-cut in N. Let $S' = \{s, v_1, v_2, v_3, v_4, v_5, v_6\}$ and $T' = \{t\}$. Then $S' \cup T' = V$ and $S' \cap T' = \emptyset$. Also $s \in S'$ and $t \in T'$. Thus $[S', T']$ is an (s,t)-cut. Here $[S', T'] = \{v_2t, v_4t, v_6t\}$. Thus

$$\text{cap}(S', T') = c(v_2t) + c(v_4t) + c(v_6t) = 4 + 5 + 7 = 16.$$

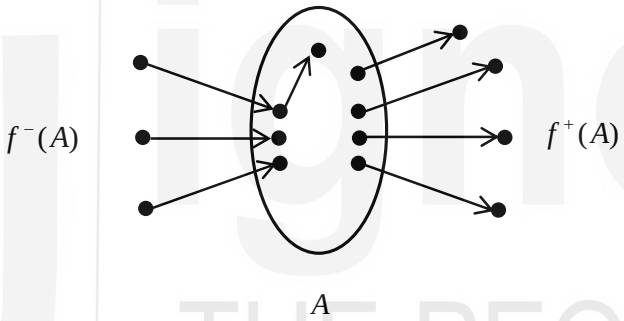
Clearly, $\text{cap}(S', T') < \text{cap}(S, T)$.

***

Before discussing the Max-Flow Min-cut Theorem, we shall need some preliminary results. Given a flow f on a network $N = (G, c, s, t)$, let $A \subseteq V$. We introduce two quantities $f^+(A)$ and $f^-(A)$ which are defined as follows:

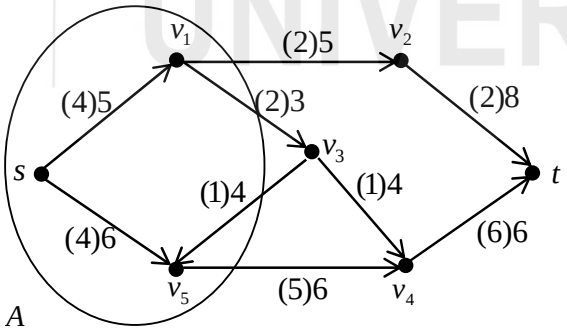
$$f^+(A) = \sum_{e \in [A, \bar{A}]} f(e), \qquad f^-(A) \sum_{e \in [\bar{A}, A]} f(e)$$

We call $f^+(A)$ the **total flow on the edges leaving** A, and $f^-(A)$ the **total flow on the edges entering** A. This is depicted in the picture given below.



Note that the edges with both endpoints in A, or no endpoint in A, contribute zero towards $f^-(A)$ as well as $f^+(A)$.

Then the net flow out of A is $f^+(A) - f^-(A)$. For instance, consider the network N with a flow f as given below.



N

Let $A = \{s, v_1, v_5\}$. Then the flow leaving A is

$$f^+(A) = f(v_1v_2) + f(v_1v_3) + f(v_5v_4) = 2 + 2 + 5 = 9.$$

The flow entering A is $f^-(A) = f(v_3v_5) = 1$.

Thus the net flow out of A is $f^+(A) - f^-(A) = 9 - 1 = 8$. Now let us compute the sum $\sum_{v \in A} (f^+(v) - f^-(v))$, which is the total net flow out of the vertices of A.

We get

$$\sum_{v\in A}(f^+(v)-f^-(v)) = (f^+(s)-f^-(s))+(f^+(v_1)-f^-(v_1))+(f^+(v_5)-f^-(v_5))$$

$$= (8-0)+(4-4)+(5-5) = 8.$$

Thus, $f^+(A)-f^-(A) = \sum_{v\in A}(f^+(v)-f^-(v))$. This is indeed true for any subset as stated below.

Lemma 2: Let $N=(G,c,s,t)$ be any network with a flow f. Then for any $A\subseteq V$, the net flow out of A is equal to the sum of the net flows out of the vertices of A. That is,

$$f^+(A)-f^-(A) = \sum_{v\in A}(f^+(v)-f^-(v)) \quad (1)$$

holds for any $A\subseteq V$. Moreover, if $[S,T]$ is any (s,t)-cut in N, then

$$f^+(S)-f^-(S) = \text{val}(f) = f^-(T)-f^+(T). \quad (2)$$

Proof: Pick any $xy\in E$ and let $f(xy)=\alpha$. Let $x,y\in A$. Then xy contributes zero to the LHS of Eqn. (1). Towards the RHS of Eqn. (1), xy contributes $+\alpha$ for $(f^+(x)-f^-(x))$ and $-\alpha$ for $(f^+(y)-f^-(y))$. Thus, total contribution of xy to the RHS of Eqn. (1) is zero. If $x,y\notin A$, then xy contributes zero to both the sides of Eqn. (1). Now assume $x\in A, y\notin A$. Then xy leaves A, and hence its contribution to the LHS of Eqn. (1) is $+\alpha$. Towards the RHS of Eqn. (1), the contribution of xy is $+\alpha$ for the term $f^+(x)-f^-(x)$.
Finally, assume that $x\notin A, y\in A$. In this case, xy contributes $-\alpha$ to both the sides of Eqn. (1). Thus in all cases, xy contributes the same amount of flow on both the sides of Eqn. (1). Since xy is arbitrary, Eqn. (1) holds.

Now we establish Eqn. (2). Since f is a flow, we know that $f^+(v)=f^-(v)$ for every $v\in V\setminus\{s,t\}$. Thus

$$f^+(S)-f^-(S) = \sum_{v\in S}(f^+(v)-f^-(v))$$

$$= f^+(s)-f^-(s) = \text{val}(f)$$

Also $f^-(T)-f^+(T) = \sum_{v\in T}(f^-(t)-f^+(v))$

$$= f^-(t)-f^+(t) = \text{val}(f).$$ ■

The following corollary is an immediate consequence of Lemma 2.

Corollary 1: If $[S,T]$ is an (s,t)-cut in a network $N=(G,c,s,t)$ with a flow f, then $\text{val}(f)\le \text{cap}(S,T)$.

Proof: From Lemma 2, we know that

$$\text{val}(f) = f^+(S)-f^-(S)$$

$$\le f^+(S) \qquad [\because f^-(S) = \sum_{e\in[\bar{S},S]} f(e)\ge 0]$$

$$= \sum_{e\in[S,\bar{S}]} f(e)$$

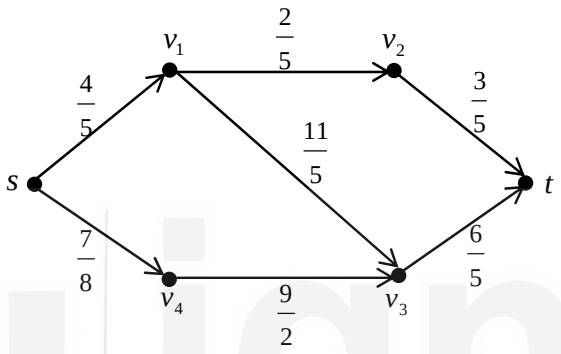
$$= \sum_{e\in[S,T]} f(e) \qquad [\because T = V \setminus S]$$

$$\leq \sum_{e\in[S,T]} c(e) \qquad [\because f(e) \leq c(e)]$$

$$= \text{cap}(S,T)$$ ■

Let us look at the following example.

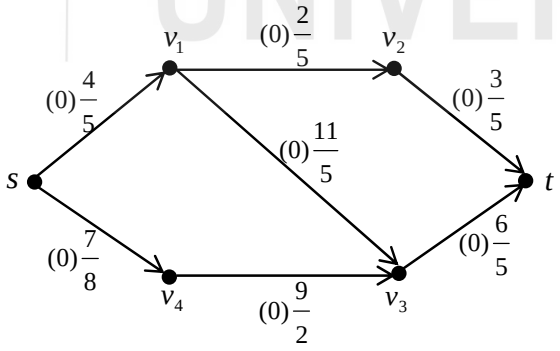
Example 5: Consider the network N given below with rational capacities.



Find a flow in N with largest possible value, using the repeated application of the Ford-Fulkerson Algorithm. Then show that this is a maximum flow, using Corollary 1.

Solution: Beginning with the zero flow we shall iterate the Ford-Fulkerson Algorithm until we get a maximum flow. The f-augmenting chain obtained from one iteration will be used to construct a new flow for the next iteration.

Iteration 1: We begin with the zero flow as shown below.



The successive steps are summarised in the following table.

R	S	vertex-chosen	(vertex, parent) pairs
$\{s\}$	$\emptyset$	s	$(v_1,s),(v_4,s)$
$\{s,v_1,v_4\}$	$\{s\}$	v_1	$(v_2,v_1),(v_3,v_1)$
$\{s,v_1,v_2,v_3,v_4\}$	$\{s,v_1\}$	v_2	(t,v_2)
$\{s,v_1,v_2,v_3,v_4,t\}$			

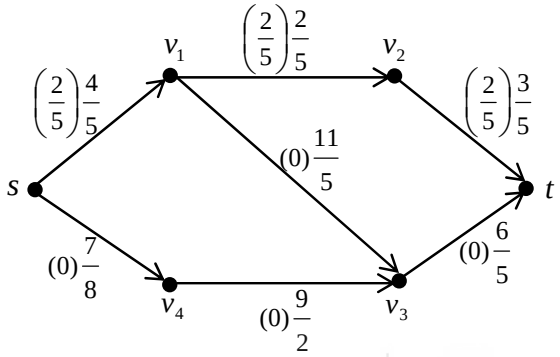
So, an f-augmenting chain P is constructed from the last column by going backward from $t \to v_2 \to v_1 \to s$. Thus, $P = (s, v_1, v_2, t)$.

Now

$$c_f(P) = \min\{c_f(sv_1), c_f(v_1v_2), c_f(v_2t)\} = \min\left\{\frac{4}{5}, \frac{2}{5}, \frac{3}{5}\right\} = \frac{2}{5}.$$

Note that the f-augmenting chains constructed by the algorithm depend on the vertices chosen at intermediate steps. For instance, if at the third row you chose the vertex v_3, then you would get (s, v_1, v_3, t) as the f-augmenting chain.

Iteration 2: New flow is shown below.



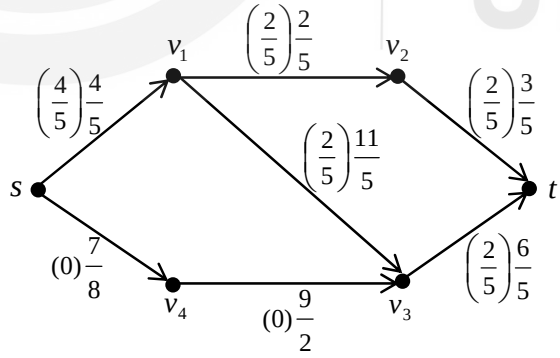
The successive steps are as follows.

R	S	vertex-chosen	(vertex, parent) pairs
$\{s\}$	$\emptyset$	s	$(v_1, s), (v_4, s)$
$\{s, v_1, v_4\}$	$\{s\}$	v_1	(v_3, v_1)
$\{s, v_1, v_3, v_4\}$	$\{s, v_1\}$	v_3	(t, v_3)
$\{s, v_1, v_3, v_4 t\}$			

Hence, the f-augmenting chain is $P = (s, v_1, v_3, t)$. So,

$$c_f(P) = \min\{c_f(sv_1), c_f(v_1v_3), c_f(v_3t)\} = \min\left\{\frac{2}{5}, \frac{11}{5}, \frac{6}{5}\right\} = \frac{2}{5}.$$

Iteration 3: New flow is as follows.

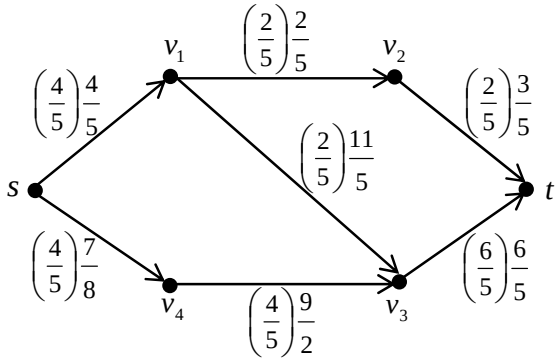


The successive steps are as follows.

R	S	vertex-chosen	(vertex, parent) pairs
$\{s\}$	$\emptyset$	s	(v_4, s)
$\{s, v_4\}$	$\{s\}$	v_4	(v_3, v_4)
$\{s, v_3, v_4\}$	$\{s, v_4\}$	v_3	(t, v_3)
$\{s, v_3, v_4, t\}$			

Hence, the f -augmenting chain is $P=(s,v_4,v_3,t)$, with $c_f(P)=\frac{4}{5}$.

Iteration 4: New flow is given below.



R	S	vertex-chosen	(vertex, parent) pairs
$\{s\}$	$\emptyset$	s	(v_4,s)
$\{s,v_4\}$	$\{s\}$	v_4	(v_3,v_4)
$\{s,v_3,v_4\}$	$\{s,v_4\}$	v_3	(v_1,v_3)
$\{s,v_1,v_3,v_4\}$	$\{s,v_3,v_4\}$	v_1	
$\{s,v_1,v_3,v_4\}$	$\{s,v_1,v_3,v_4\}$		

Now $R\setminus S=\emptyset$. So, we return $S=\{s,v_1,v_3,v_4\}$ and stop. Now, let us check whether the present flow f on N is maximum or not. We have

$$\text{val}(f)=f^{-}(t)=\frac{2}{5}+\frac{6}{5}=\frac{8}{5}$$

Take $T=V\setminus S=\{v_2,t\}$. Then it can easily be seen that $[S,T]$ is an (s,t)-cut.

Here $[S,T]=\{v_1v_2,v_3t\}$. Therefore, $\text{cap}(S,T)=c(v_1v_2)+c(v_3t)=\frac{2}{5}+\frac{6}{5}=\frac{8}{5}$.

Thus, $\text{val}(f)=\text{cap}(S,T)$. Hence, by Corollary 1, f is a maximum flow.

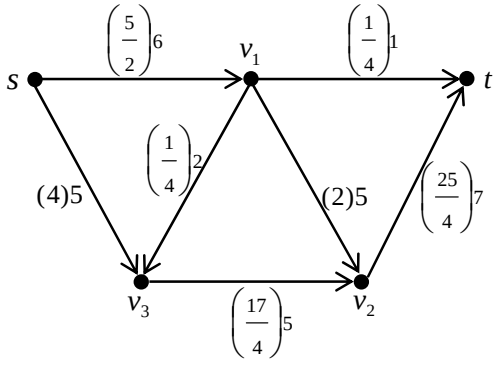
***

Now we come to the Max-Flow Min-Cut Theorem, which tells us when the equality occur in the inequality $\text{val}(f)\le\text{cap}(S,T)$.

Theorem 1 (Max-Flow Min-Cut Theorem): In every network, the maximum value of a flow equals the minimum capacity of a source-to-sink cut.

We shall not prove it, Instead, we shall see how it is useful.

Example 6: Consider the network N with a flow f as given below.



Using the Max-Flow Min-Cut Theorem, show that f is not a maximum flow.

Solution: We shall show that every (s,t)-cut $[S,\bar{S}]$ has capacity more than $\text{val}(f) = \frac{25}{4} + \frac{1}{4} = \frac{13}{2}$.

Case 1: $|S| = 1$. Then $S = \{s\}$, and hence $\text{cap}(S,\bar{S}) = 6 + 5 = 11$.

Case 2: $|S| = 2$. Then there are three possibilities $S = \{s, v_1\}, S = \{s, v_2\}$, and $S = \{s, v_3\}$. The corresponding capacities are 13, 18, and 11.

Case 3: $|S| = 3$. In this case also S has three possibilities, which are $\{s, v_1, v_2\}, \{s, v_1, v_3\}$ and $\{s, v_2, v_3\}$. The corresponding capacities are 15,11 and 13.

Case 4: $|S| = 4$. In this case there is only possibility, that is, $S = \{s, v_1, v_2, v_3\}$.

So, $\text{cap}(S,\bar{S}) = 1 + 7 = 8$.

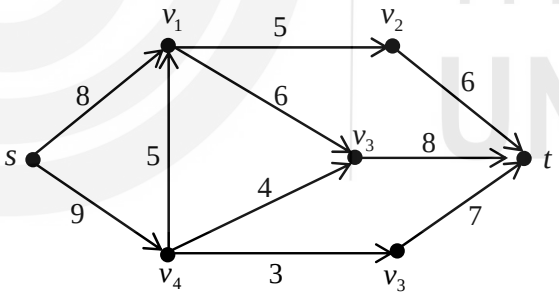
It is now clear that $\text{val}(f) < \text{cap}(S,\bar{S})$ for every (s,t)-cut $[S,\bar{S}]$ of N.

Therefore, by the Min-Cut Max-Flow Theorem, f is not a maximum flow.

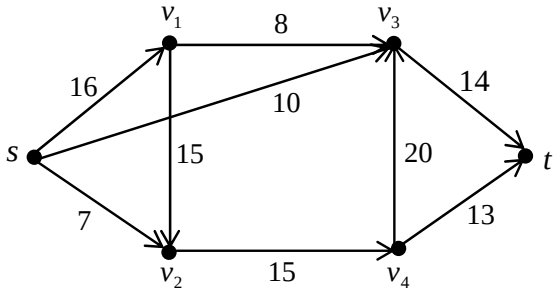
***

Now try the following exercises.

E5) Does there exist a flow of value 17 in the network given below? Justify.



E6) What is the maximum possible value of a flow in the following network?



With this we come to an end to this unit. We now summarises our discussion.

10.5 SUMMARY

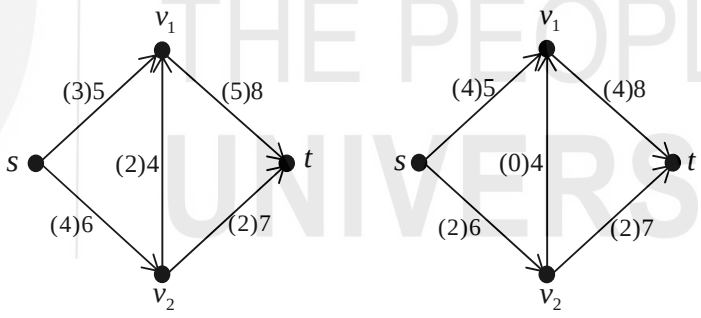
In this unit we have covered the following points:

1. The concept of network, and its characteristics are explained.
2. We have defined the concept of flow in a network, and have discussed the maximum-flow problem.
3. We have described what is an f-augmenting chain and explained how it can be used to construct a larger flow.
4. The Ford-Fulkerson Algorithm is discussed in detail.
5. We have defined the concept of an (s,t)-cut in a network, and have shown how it is related to the maximum-flow problem.
6. Finally, we have stated the Max-Flow Min-Cut Theorem, and applied it to some problems.

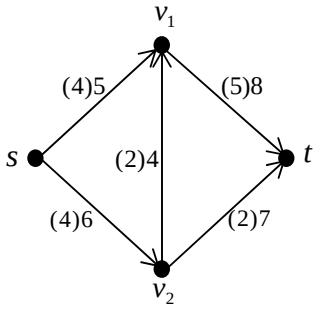
10.6 SOLUTIONS / ANSWERS

E1) Observe that $f^-(v_1) = 6$ and $f^+(v_1) = 7$. Thus, $f^-(v_1) \neq f^+(v_1)$. Hence, f is not a flow.

E2) No. To see why, consider two flows f and g on a network as shown below.



Now, let $h = \max\{f, g\}$. Then h is shown below.



$h = \max\{f, g\}$

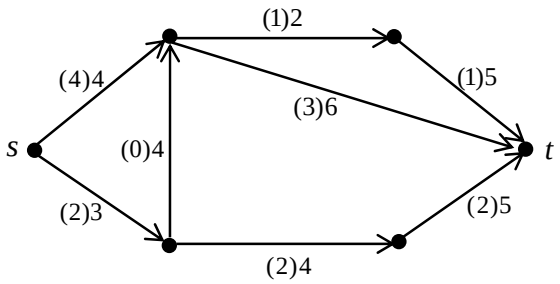
Observe that $h^-(v_1) = 4 + 2 = 6 \neq h^+(v_1) = 5$. Therefore, h is not a flow.

E3) We can see that $P=(s,v_1,t)$ is an f-augmenting chain in N, with $c_f(P)=1$. So, define $g: E \to \mathbb{R}$ as follows.

$g(e)=f(e)$ if $e \notin E(P)$, and for each $e \in E(P)$

$$g(e)=\begin{cases} f(e)+c_f(P), \text{ if } e \text{ is forward} \\ f(e)-c_f(P), \text{ if } e \text{ is backward.} \end{cases}$$

The flow g is shown as below.



By Lemma 1, $\text{val}(g)=\text{val}(f)+c_f(P)=5+1=6$. Therefore, g is a larger flow.

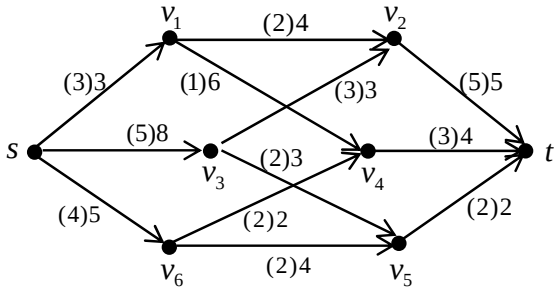
E4) The successive steps of the Ford-Fulkerson Algorithm are shown in the following table.

R	S	vertex-chosen	(vertex, parent) pairs
$\{s\}$	$\emptyset$	s	$(v_3,s),(v_6,s)$
$\{s,v_3,v_6\}$	$\{s\}$	v_3	(v_5,v_3)
$\{s,v_3,v_5,v_6\}$	$\{s,v_3\}$	v_5	
$\{s,v_3,v_5,v_6\}$	$\{s,v_3,v_5\}$	v_6	(v_4,v_6)
$\{s,v_3,v_4,v_5,v_6\}$	$\{s,v_3,v_5,v_6\}$	v_4	(t,v_4)
$\{s,v_3,v_4,v_5,v_6,t\}$			

Since $t \in R$, we step here. The f-augmenting chain P is constructed by going back from $t \to v_4 \to v_6 \to s$. That is, $P=(s,v_6,v_4,t)$. Now

$c_f(P)=\min\{c_f(sv_6),c_f(v_6v_4),c_f(v_4t)\}=\min\{3,2,3\}=2.$

So, we construct a new flow g shown as below.



By Lemma 1, $\text{val}(g)=\text{val}(f)+c_f(P)=8+2=10$. Hence, g is a larger flow.

E5) No. To see why, consider $S = \{s, v_1, v_3, v_4\}$. Then $[S, \overline{S}]$ is an (s,t)-cut with cap $(S, \overline{S}) = 5 + 8 + 3 = 16$. By The Min-Cut Max-Flow Theorem, no flow can have more value than the capacity of any (s,t)-cut. Therefore, the network has no flow of value 17.

E6) There are two ways to solve this problem. One way is to apply the Ford-Fulkerson Algorithm repeatedly as done in Example 5, until you get a set S. Then cap$(S, \overline{S})$ is the required answer.
Another way is to use the Max-Flow Min-Cut Theorem, in which case you have to compute the capacities of all the (s,t)-cuts. Then, the minimum capacity is the required answer.

We use the second approach. We consider all the sets S with $s \in S, t \notin S$, and compute cap $(S, \overline{S})$ as given below.

$\|S\|$	S	cap$(S, \overline{S})$
1	$\{s\}$	33
2	$\{s, v_1\}, \{s, v_2\}, \{s, v_3\}, \{s, v_4\}$	40,41,37,66
3	$\{s, v_1, v_2\}, \{s, v_1, v_3\}, \{s, v_1, v_4\}$ $\{s, v_2, v_3\}, \{s, v_2, v_4\}, \{s, v_3, v_4\}$	33,36,73 65,59,60
4	$\{s, v_1, v_2, v_3\}, \{s, v_1, v_2, v_4\}$ $\{s, v_1, v_3, v_4\}, \{s, v_2, v_3, v_4\}$	29,51 49,43
5	$\{s, v_1, v_2, v_3, v_4\}$	27

Thus the minimum capacity of an (s,t)-cut is 27. Therefore, by the Max-Flow Min-Cut Theorem, the maximum possible flow has value 27.

PRACTICAL SESSIONS

As we have mentioned in the course introduction, you need to gain some practical experience of some of the algorithms learnt in this course. Keeping in view of this, we have included a practical component for this course. You will recall that in the 70% marks earmarked for the term end examination, 50% is for the theory and 20% is for the practical. In the 30% marks for continuous assessment, 20% is of the assignment and 10% is for the practical component.

This course has 8 practical sessions of 3 hours each. The language to be used for writing the programs is *C*. You must already be familiar with *C* -programming from the course MMT-001. Apart from this, the course MMTE-002, which you will studying along with this course also will help you to write the programs.

While writing the program, you should add comments in the program and in the functions, documenting what the program or function does. Indent your programs properly so that anyone can read and understand them easily. If you are working on linux, the indent program may be available on your computer. For example, if you want to indent your program in the style followed in the book by Kernighan and Ritchie, you can invoke indent using the command.

indent –kr file_name. c

Indent offers many other options. Read the documentation for the program if necessary.

Also, write a friendly interface, wherever required, for the user to interact with the program. Your program should also check if the input is appropriate and print an error if it is not. For writing and compiling of your programs you can download the Code: Blocks software, from the website https://www.codeblocks.org .

Now we list the practical sessions for this course. For each of the problems given below for practical work you are expected to do the following:

(1) Write the algorithm in pseudocode or make a flow chart and show it to your counselor.

(2) After the counselor approves, implement the algorithm in C-language, i.e. write a C-program, debug it, compile it and run it for the sample graph given and show it to the counselor.

(3) The counselor may ask you questions about the program to test your understanding and also may suggest changes.

(4) After the counselor approves the program, print or write neatly the source code of the program, and get it signed by the counselor.

(5) **Maintain you program in a file**. You continuous assessment will be based on this file.

(6) You should also produce the file at the time of practical examination.

Now we list the practical sessions for this course.

Session 1:

The aim of this session is to give you some experience of how to write programs for some of the basic concepts of graph theory. As a first step we consider how to represent a graph as an object in computer programming.

The most common way to represent a graph G is by its adjacency matrix $A(G)$. You know that when $A(G)$ is symmetric, G is undirected, and when $A(G)$ has any entry greater than 1, then G is either a multigraph or a weighted graph. In C programming, it is easy to declare the adjacency matrix of any graph (or multigraph) G. For instance, to declare the adjacency matrix of a graph of order 10, just invoke the command

```
int A[10][10];
```

Then you can read and print A or perform any operations on A as usual.

Now you can do the following:

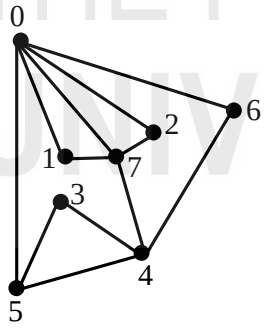
Program 1: Write a program that constructs the adjacency matrix A of an undirected, unweighted graph, given a sequence of edges as input. The program must perform the following operations:

- print the degree sequence of the graph.
- print the neighbours of a given vertex.
- compute A^k for any positive integer k given by the user.
- print the number of walks of length k, between any two vertices.

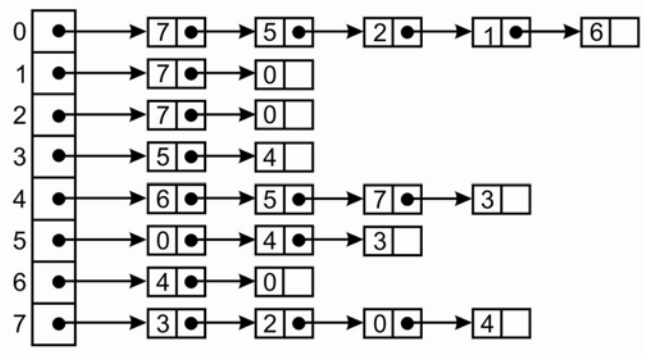
Program 2: Write a program to check whether a given graph is connected or not. (See Theorem 5 of Unit 2.)

Session 2:

Another straightforward method for representing a graph is to use an array of linked lists, called adjacency lists. We keep a linked list for each vertex, with a node for each neighbour of that vertex. For undirected graphs if there is a node for j in i's list, then there must be node for i in j's list. For, example, consider the graph given below.



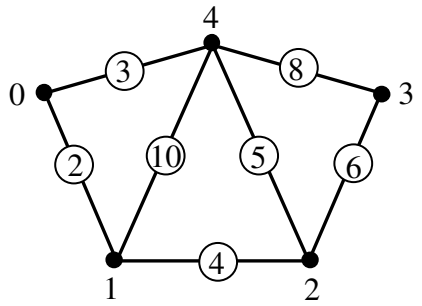
Its adjacency list representation is given below.



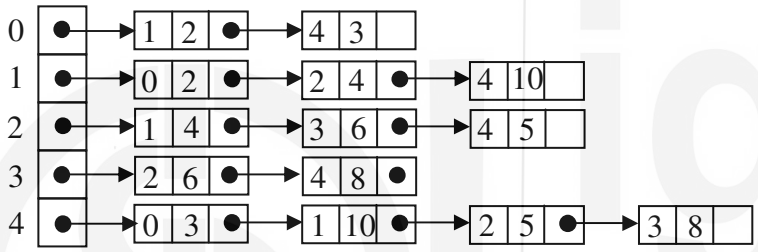
Program 3: Write a program that uses the adjacency lists for storing a graph. The program takes a sequence of edges as input, and must perform the following tasks:

- print the degree sequence of the graph
- print all the neighbours of a given vertex
- check whether any two given vertices are adjacent or not

Suppose we which to store a weighted graph in a computer's memory, using adjacency lists. So, we declare an array of linked lists. But this time, each node of the linked list has three members-vertex name, its weight, and a link for the next node. For instance, consider the following weighted graph.



Its adjacency list representation would be:



Program 4: Write a program that stores a weighted graph using adjacency list representation, and prints the weight of any input subgraph of it.

Session 3:

Recall that a graph (or multigraph) can also be represented through an incidence matrix. Let $G = (V, E)$ be a multigraph, where $V = \{v_1, v_2, \ldots, v_n\}$, and $E = \{e_1, e_2, \ldots, e_n\}$. Then the incidence matrix $B(G)$ of G is an $n \times m$ matrix with its ij^{th} entry b_{ij} defined as

$$b_{ij} = \begin{cases} 1, & \text{if } v_i \text{ is an endpoint of } e_j, \\ 0, & \text{otherwise} \end{cases}$$

If A and B are adjacency and incidence matrices, respectively, of a graph G, then you can show that $A = BB^T - D$, where D is a diagonal matrix whose diagonal entries are the degrees of G.

Program 5: Write a program that constructs and prints the incidence matrix of a given undirected graph. The program takes a list of pairs (v_i, e_j) as input, where v_i is an endpoint of e_j. The program must perform the following tasks:

- print the degree sequence of the graph
- print the adjacency matrix of the graph using the relation $A = BB^T - D$.

Session 4: Kruskal's Algorithm

In this section we want you to implement Kruskal's Algorithm that we have discussed in Unit 4 of Block 1.

Program 6: Write a program that implements Kruskal's Algorithm to find a minimum weight spanning tree in a weighted connected graph.

Using the program find a minimum spanning tree in some connected weighted graphs.

Session 5: BFS Algorithm

Program 7: Write a program that uses the BFS Algorithm to find a rooted spanning tree in a given connected graph. The program should print the list of edges of the spanning tree.

Sessions 6: Dijkstra's Algorithm

Program 8: Using Dijkstra's Algorithm write a program to find the distances from a root vertex to all the vertices in a connected weighted graph. (Use the adjacency list representation for storing the graph.)

Run your program for some small graph.

Session 7: Ford-Fulkerson Algorithm

You are already familiar with Ford-Fulkerson Algorithm from Unit 10 of Block 3.

Program 9:

i) Write a function namely ff_algorithm (N, f) which takes as input a network N, a flow f and returns an f-augmenting chain or a set S.

ii) Write a program that calls the function ff_algorithm (N, f) repeatedly and computes the maximum flow in a given network. Run your program for some small networks.