Fig. 6.9.1

6.10 Post's Correspondence Problem

AU : Dec.-07, 10, 11, 12, 14, Marks 8, May-13, Marks 6

In this section, we will discuss the undecidability of strings and not of Turing machines. The undecidability of strings is determined with the help of Post's Correspondence Problem (PCP). Let us define the PCP.

"The post's correspondence problem consists of two lists of strings that are of equal length over the input Σ The two lists are $A = w_1, w_2, w_3, \ldots, w_n$ and $B = x_1, x_2, x_3 \ldots x_n$ then there exists a non empty set of integers $i_1, i_2, i_3, \ldots i_n$ such that $w_1, w_2, w_3, \ldots w_n = x_1, x_2, x_3, \ldots x_n$"

To solve the post correspondence problem we try all the combinations of $i_1, i_2, i_3, \ldots, i_n$ to find the $w_i = x_i$ then we say that PCP has a solution.

6.6 Tractable and Possibly Intractable Problems : P and NP

AU : Dec.-12, 13, Marks 6

There are two groups in which a problem can be classified. The first group consists of the problems that can be solved in polynomial time. For example: searching of an element from the list $O(\log n)$, sorting of elements $O(\log n)$.

The second group consists of problems that can be solved in non-deterministic polynomial time. For example : Knapsack problem $O(2^{n/2})$ and Travelling Salesperson problem $(O(n^2 2^n))$.

- Any problem for which answer is either yes or no is called decision problem. The algorithm for decision problem is called **decision algorithm.**
- Any problem that involves the identification of optimal cost (minimum or maximum) is called optimization problem. The algorithm for optimization problem is called **optimization algorithm.**
- **Definition of P** - Problems that can be solved in polynomial time. ("P" stands for polynomial).

  Examples - Searching of key element, Sorting of elements, All pair shortest path.
- **Definition of NP** - It stands for "non-deterministic polynomial time". Note that NP does not stand for "non-polynomial".

Examples - Travelling Salesperson problem, Graph coloring problem, Knapsack problem, Hamiltonian circuit problems.

- The NP class problems can be further categorized into NP-complete and NP hard problems.

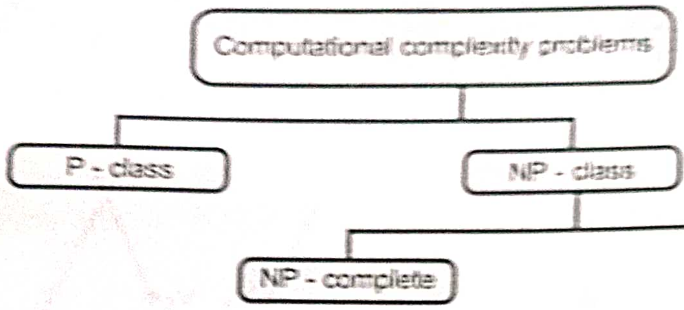


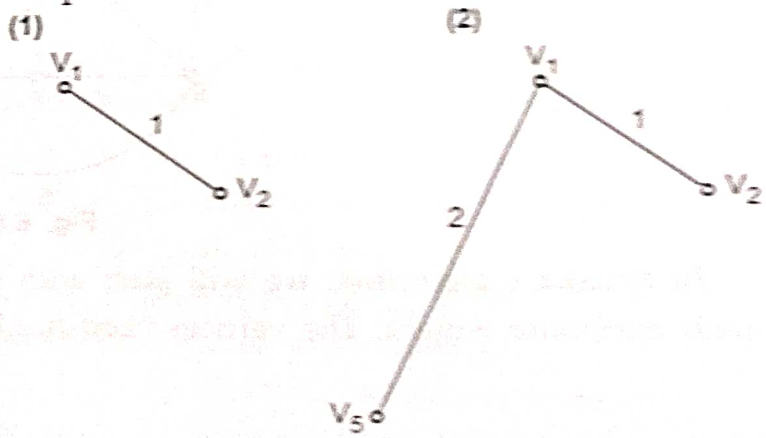
Fig. 6.6.1

- A problem D is called NP-complete if -
  i) It belongs to class NP
  ii) Every problem in NP can also be solved in polynomial time.
- If an NP-hard problem can be solved in polynomial time then all NP-complete problems can also be solved in polynomial time.
- All NP-complete problems are NP-hard but all NP-hard problems can not be NP-complete.
- The NP class problems are the decision problems that can be solved by non-deterministic polynomial algorithms.

6.6.1 Example of P Class Problem

Kruskal's Algorithm: In Kruskal's algorithm the minimum weight is obtained. In this algorithm also the circuit should not be formed. Each time the edge of minimum weight has to be selected, from the graph. It is not necessary in this algorithm to have edges of minimum weights to be adjacent. Let us solve one example by Kruskal's algorithm.

Example 1 :



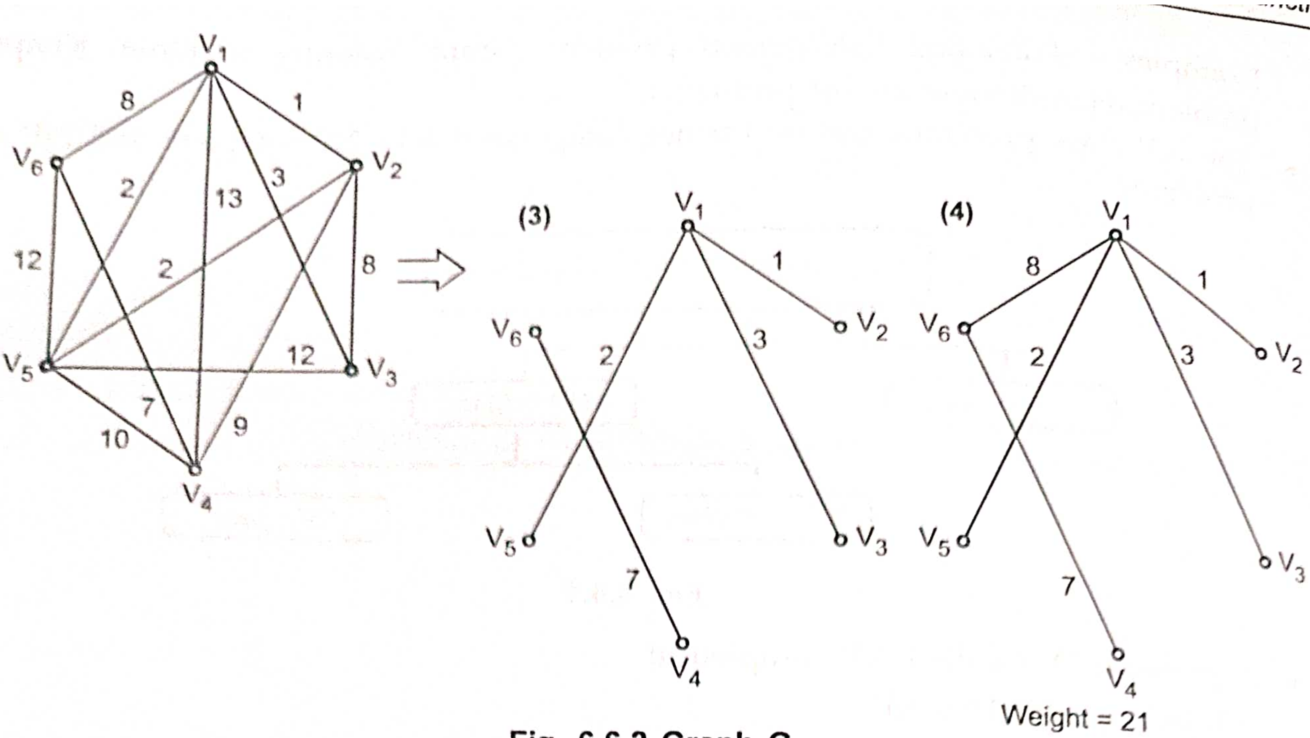


Fig. 6.6.2 Graph G

Example 2 : Find the minimum spanning tree for the following figure using Kruskal's algorithm.

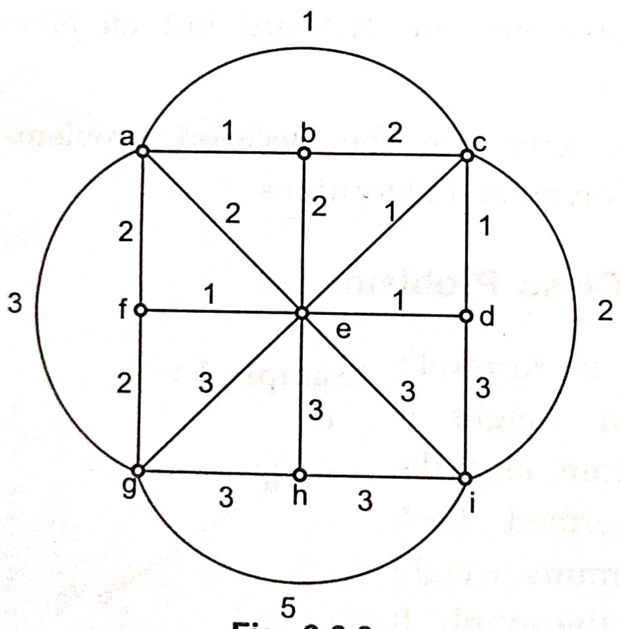


Fig. 6.6.3

In Kruskal's algorithm, we will start with some vertex and will cover all the vertices with minimum weight. The vertices need not be adjacent.

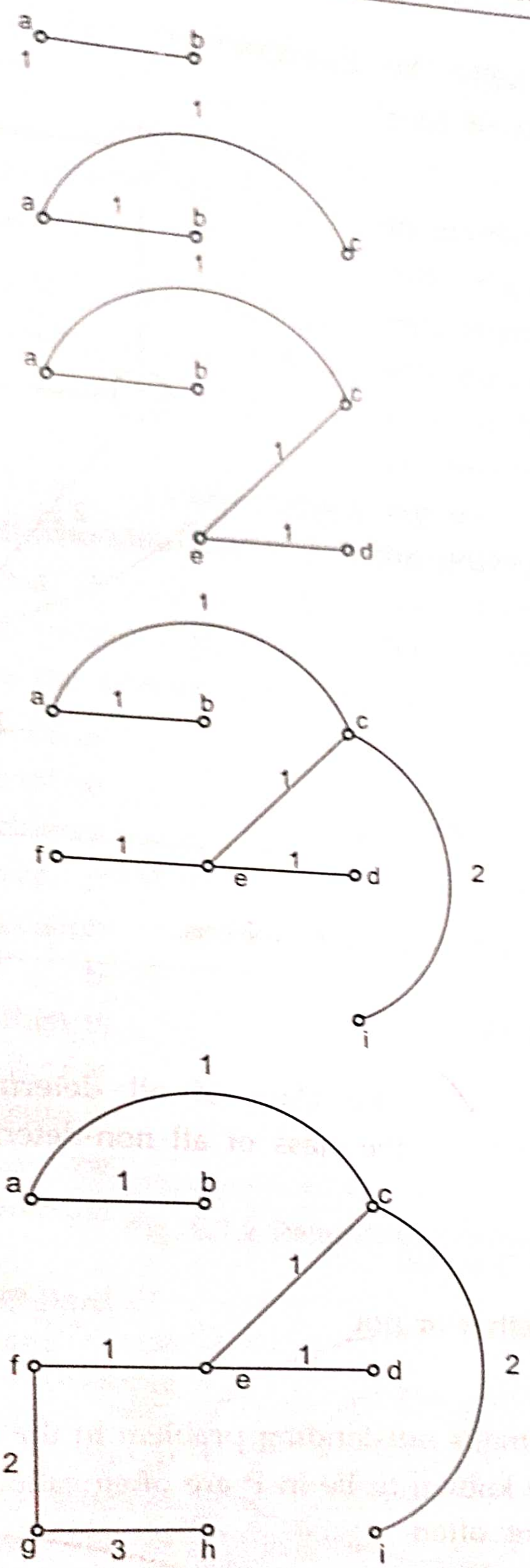


Fig. 6.6.4

6.6.2 Example of NP Class Problem

Travelling Salesman's Problem (TSP) : This problem can be stated as " Given a set of cities and cost to travel between each pair of cities, determine whether there is a path that visits every city once and returns to the first city. Such that the cost travelled is less".

The tour path will be **a-b-d-e-c-a** and total cost of tour will be **16.**

This problem is NP problem as there may exist some path with shortest distance between the cities. If you get the solution by applying certain algorithm then Travelling Salesman problem is **NPComplete Problem.** If we get no solution at all by applying an algorithm then the travelling salesman problem belongs to **NP hard class.**

For example :

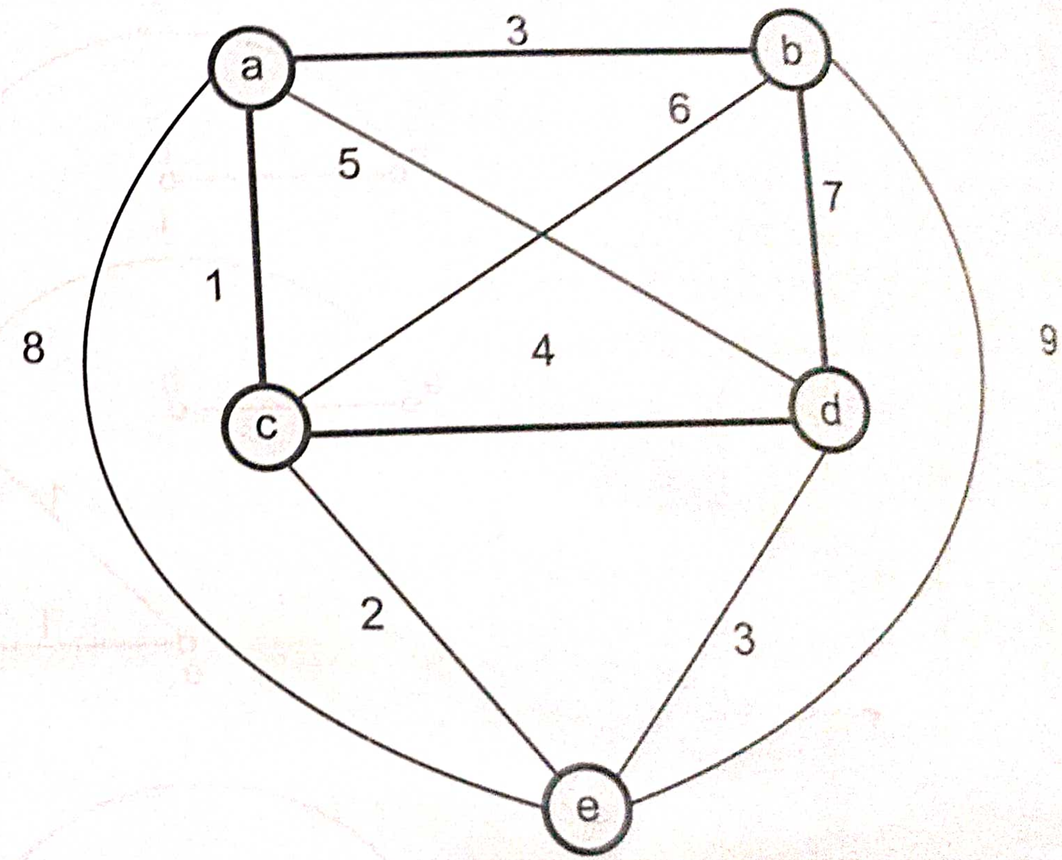


Fig 6.6.5

Review Question

Example 6.10.1 *Consider the correspondence system as given below -*

$A = (1; 0; 010; 11)$ and $B = (10; 10; 01; 1)$. The input set is $\Sigma = \{0, 1\}$. find the solution.

Solution : A solution is 1; 2; 1; 3; 3; 4. That means $w_1w_2w_1w_3w_3w_4 = x_1x_2x_1x_3x_3x_4$.

The constructed string from both lists is 10101001011.

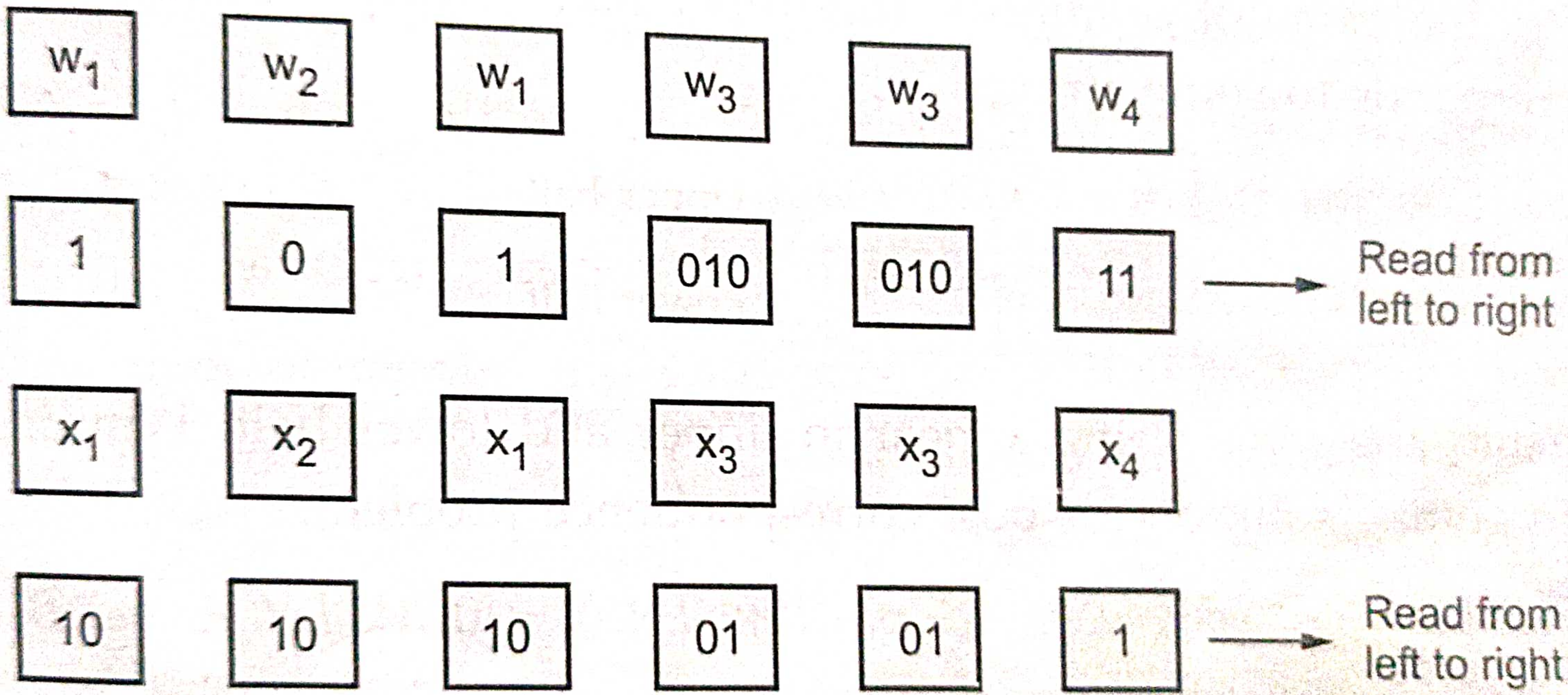


Fig. 6.10.1

Example 6.10.2 *Obtain the solution for the following correspondence system.*

A = {ba, ab, a, baa, b}, B = {bab, baa, ba, a, aba}. The input set {a, b}.

Solution : To obtain the corresponding system the one sequence can be chosen. Hence we get $w_1w_5w_2w_3w_4w_4w_3w_4 = x_1x_5x_2x_3x_4x_4w_3w_4$. This solution gives a unique string babababaabaaabaa = babababaabaaabaa. Hence solution is 15234434.

Example 6.10.4 *Obtain the solution for the following system of posts correspondence problem*

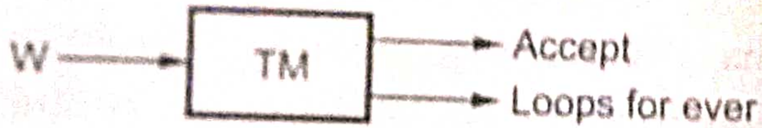
A = {ba, abb, bab} B = {bab, bb, abb}.

Solution : Now to consider 1, 3, 2 the string bababab from set A and bababbbb from set B thus the two strings obtained are not equal. As we can try various combinations from both the sets to find the unique sequence but we could not get such a sequence. Hence there is no solution for this system.

it is said that

Q.8 Differentiate between recursive and recursively enumerable languages. AU : May-07

Ans. :

Sr. No.	Recursive languages	Recursively Enumerable languages (RE)
1.	A language is said to be recursive if there exists a turing machine that accepts every string of language and every string is rejected if it is not belonging to that language. 	A language is said to be recursively enumerable if there exists a turing machine that accepts it. If the string does not belong to a language then TM will enter in infinite loop on accepting this string. 
2.	The recursive languages are called turing decidable languages.	The RE are called turing acceptable languages.
3.	Recursive language will halt on every input.	RE may not halt on every input, it may fall into an infinite loop.
4.	Every recursive language is also a recursively enumerable.	Every recursively enumerable language is not recursive.
5.	If L is recursive language then its complement L' is also recursive.	There exist RE language L whose complement L' may not be RE. If L and L' both are recursively enumerable then that L is definitely a recursive language