Biyani Institute of Science and Management
I Internal Examination Sept. 2019
MCA (V Semester)
Subject- Analysis of Design and Algorithms

MM: 20 **Set:A** **Answer Key** **Time:** 1 ½Hrs

[I] Answer the following questions in one line only (2*1=2)

Q.1 What do you mean by graph coloring?

Ans: Assign color to each vertex of graph in such a way so that adjacent vertex color is different by using minimum number of colors.

Q.2 What do you mean by stack?

Ans: Stack is a linear data structure which follow LIFO discipline.

[II]Answer the following questions in 50 words (2*3=6)

Q.1 What do you mean by 0-1 knapsack problem.

Ans: Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item, or don't pick it (0-1 property).

Q.2 What is travelling salesmen problem?

Ans: The Travelling Salesman Problem describes a salesman who must travel between N cities. The order in which he does so is something he does not care about, as long as he visits each once during his trip, and finishes where he was at first. Each city is connected to other close by cities, or nodes. Each of those links between the cities has one or more weights (or the cost) attached. The cost describes how "difficult" it is to traverse this edge on the graph, and may be given, for example, by the cost of an airplane ticket or train ticket, or perhaps by the length of the edge, or time required to complete the traversal. The salesman wants to keep both the travel costs, as well as the distance he travels as low as possible.

The Traveling Salesman Problem is typical of a large class of "hard" optimization problems that have intrigued mathematicians and computer scientists for years. Most important, it has applications in science and engineering. For example, in the manufacture of a circuit board, it is important to determine the best order in which a laser will drill thousands of holes. An efficient solution to this problem reduces production costs for the manufacturer.

[III]Answer the following questions in 150 words. (2*6=12)

Q.1 What is multistage graph problem? Explain in detail with example.

Ans:

1. The multistage graph problem is to find a minimum cost from a source to a sink.
2. A multistage graph is a directed graph having a number of multiple stages, where stages element should be connected consecutively.
3. In this multiple stage graph, there is a vertex whose in degree is **0** that is known as the source. And the vertex with only one out degree is **0** is known as the destination vertex.
4. The one end of the multiple stage graphs is at **i** thus the other reaching end is on **i+1** stage.
5. If we denote a graph **G = (V, E)** in which the vertices are partitioned into **K >= 2** disjoints sets, **Vi**, **1 <= I <=K**. So that, if there is an edge **< u, v >** from **u** to **v** in E, the **u £Vi** and **v € v (i+1)**, for some **I**, **1 <= i <= K**. And sets **V1** and **Vk** are such that **|V1| = |Vk| = 1**.

Algorithm for Forward Approach

```
1. F graph (graph G, int K, int n, int p[])
2. {
3. Float cost [max size], int d [max size], r;
4. Cost [n] = 0.0
5. For (int j = n-1; j>= 1; j--)
6. {
7. Let r be a vertex such that  is an edge of G and C[j][r] +
cost[r] is minimum;
8. Cost [j] = C[j][r] + Cost[r]
9. D [j] = r
10.      }
11.      P [1] = 1 ,  P[k] = n
12.      For (j = 2 ; j <= K-1; j++)
13.      P[j] = d[P(j-1)];
14.      }
```

Input = input is a **K** stage graph **G = (V, E)** with **n** vertices indexed in order of stages.

E is a set of an edge.

C [i][j] is the cost or weight of the edge **[i][j]**

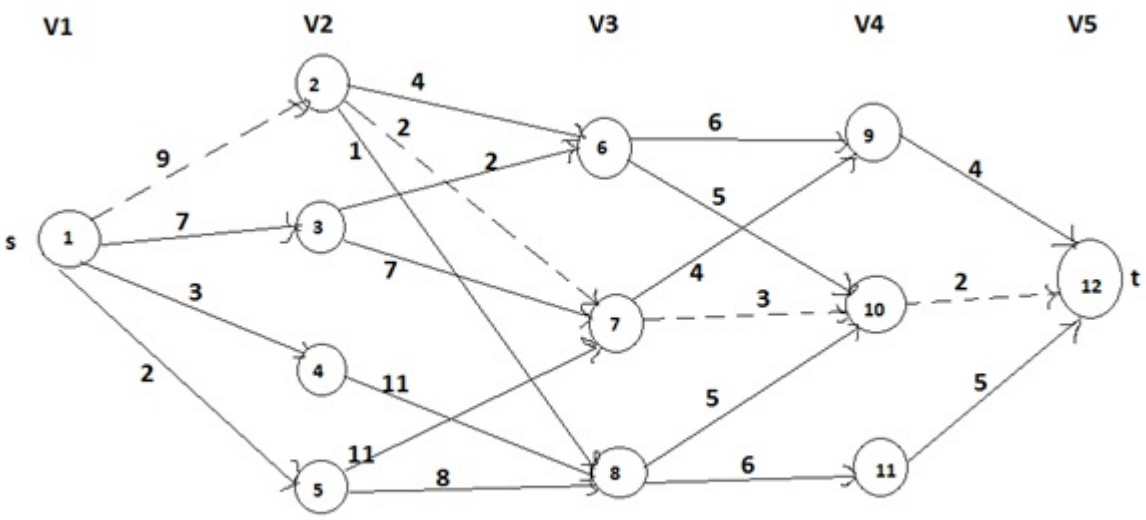
Algorithm for Backward Approach

```
1. Algorithm BGraph (G, K, n, p)
2. // some function as FGraph
3. {
4. B cost [1] = 0.0;
5. For j = 2 to n do
6. {
```

```
7. // compute b cost [j].
8. Let r be such that  is an edge of
9. G and b cost [r] + c [r, j];
10.      D [j] = r;
11.      }
12.      // find a minimum cost path
13.      P [1] = 1;                p [k] = n;
14.      For j = k-1 to 2 do p[j] = d [p (j+1)];
15.      }
```

If we consider **s** as the vertex in **V1** and **t** as the vertex in **Vk**. The vertex **s** is supposed as the source and vertex **t** as the sink. The cost of a path from **s** to **t** is the sum of the costs of the edges on the path.

Here, each set **Vi** defines a stage in the graph. Each path starts from stage 1 goes to stage 2 then to stage 3 and so on, because of constraints on **E**.



MULTI STAGE GRAPH

The minimum cost of **s** to **t** path is indicated by a dashed line. This method can be used for solving many problems such as allocating some resources to given number of projects with the intent to find the maximum profit.

Q.2 Explain All pair shortest path problem with example.

Ans: Given a directed graph G = (V,E), where each edge (v,w) has a nonnegative cost C[v,w], for all pairs of vertices (v,w) find the cost of the lowest cost path from v to w.
We will consider a slight extension to this problem: find the **lowest cost path** between each pair of vertices.
Floyd-Warshall Algorithm is used to solve APSP problem.
This problem can be solved by using following recursive formula.
(k) (k-1) (k-1) (k-1)
$A_{ij} = \text{Min } [A_{ij} ; A_{ik} + A_{kj}]$

Floyd's Algorithm

Floyd's algorithm takes as input the cost matrix C[v,w]
- C[v, w] = 00 if (v,w) is not in E

It returns as output
- a distance matrix D[v,w] containing the cost of the lowest cost path from v to w
  - initially D[v,w] = C[v,w]
- a path matrix P, where P[v,w] holds the intermediate vertex k on the least cost path between v and w that led to the cost stored in D[v,w].

We iterate N times over the matrix D, using k as an index. On the k_{th} iteration, the D matrix contains the solution to the APSP problem, where the paths only use vertices numbered 1 to k.
On the next iteration, we compare the cost of going from i to j using only vertices numbered 1..k (stored in D[i,j] on the k_{th} iteration) with the cost of using the $k+1_{th}$ vertex as an intermediate step, which is D[i,k+1] (to get from i to k+1) plus D[k+1,j] (to get from k+1 to j).
If this results in a lower cost path, we remember it.
After N iterations, all possible paths have been examined, so D[v,w] contains the cost of the lowest cost path from v to w using all vertices if necessary.