Week-7

Dijkshatra's algorithm to print shortest path and distance given source to distination.

gorithm :→

1) START
2) input v
3) if i>=v goto STEP
4) if j>=v goto stp 7
5) input adj[i][j]
6) if j<v goto stp 4
7) if i<v goto step 3
8) input s
9) dijhstra(adj, v, s-1)
10) STOP

```
dijkstra (vector <vector <int>> mat, int v, int s){
    int dis[v];
    bool vis[v];
    int parent[v];
    int i,j;
    for (i=0; i<v; i++) {
     dis[i] = INT_MAX;
     vis[i] = false;
     parent[i] = -1;
    }
     dis[s] = 0;
     parent[s] = s;
    for(i=0; i<v; i++){
     m = minDistanceIndex(dis, vis, v);
     vis[m] = true;
     for(j=0; j<v; j++){
    if(dis[m] != INT_MAX && !vis[j] && mat[m][j]){
      if (dis[j] > dis[m] + mat[m][j]){
        dis[j] = dis[m] + mat[m][j];
```

```
        parent[j] = u;
    }
  }
}

for (i=0; i<V; i++) {
  if (i==s) {
    cout << i+1 << " : " << dis[i] << endl;
    continue;
  }
  cout << i+1;
  j = i;
  while (parent[j] != s) {
    cout << " " << parent[j]+1;
    j = parent[j];
  }
  cout << " " << s+1 << " : " << dis[i] << endl;
}
}

int minDistanceIndex(int * dis, bool * vis, int V) {
  int i;
  int minDis = INT_MAX;
  int minIndex = -1;
  for(i=0; i<V; i++) {
  if (vis[i] == false && dis[i] <= minDis) {
      minDis = dis[i];
      minIndex = i;
    }
  }
  return minIndex;
}
```

an algorithm and implement it using a program to solve question Problem using Bellman Ford's algorithm.

1) START
2) Input m
3) if i >= m goto step
4) if j >= m goto step 7
5) input graph[i][j]
6) if j < m goto step 4
7) if i < m goto step 3
8) input s
9) findpath(graph, m, s-1)
10) STOP

```
dpath(int** graph, int m, int s) {
vector<int> dis(m, INT_MA), pa(m,-1);
 dis[saur] = 0;
for(int k=0; k<m-1; k++) {
  for(int i=0; i<m; i++) {
      for(int j=0; j<m; j++) {
        if(graph[i][j] != 0) {
         if(dis[j] > dis[i] + graph[i][j])
            {
             dis[j] = dis[i] + graph[i][j];
                pa[j] = i;
```

```
for(int i=0; i<m; i++) {
   calculate(pa,i);
   cout<<" :" <<dis[i];
}
}
```

```
calculate (vector<int> &pa, int i) {
    cout << i+1 << " ";
    if (pa[i] >= 0)
    calculate (pa, pa[i]);
}
```

directed graph with two vertices. Design an algorithm & write a program to find weight of shortest path source to destination while exactly k edges on the path.

Algorithm :→

1) START
2) input ver
3) if i >= ver goto step
4) if j >= ver goto 7
5) input graph[i][j]
6) if j<ver goto step 4
7) if i<ver goto step 3
8) input u, v, k
9) ans = shortest_weight(graph, ver, u-1, v-1, k)
10) print ans
11) STOP

```
shortest_weight(int** graph, int ver, int u, int v, int k){
  if(k<=0)
    return INT_MAX;
  if(k==0 && u==v)
      return 0;
  if(k==1 && graph[u][v]!=INT_MAX)
      return graph[u][v];
  int res = INT_MAX;
  for(int i=0;i<ver;i++){
   if(graph[u][i]!=0 && u!=i && v!=i){
       recu = shortest_weight(graph,ver,i,v,k-1);
    if(recu!=INT_MAX)
      res = min(res, graph[u][i]+recu);
    }
  }
  return res;
}
```