**Assignment DAA no-8**

Name :- Kapil Arun Patil

Reg.No :- 2020BIT153

**Implement Dynamic Algorithm for shortest path**

**1)Floyd Warshall Algorithm**

**Code:**

#include<bits/stdc++.h>

using namespace std;

#define INF 1000000000

int main(){

int n, m;

cin >> n >> m; // n - number of vertices, m - number of edges

vector<vector<int>> dist(n, vector<int>(n, INF));

// dist[i][j] stores the length of the shortest path between i and j

for(int i = 0; i < m; i++){

int u, v, w;

cin >> u >> v >> w; // input the edges and their weights

dist[u][v] = w;

}

for(int i = 0; i < n; i++) dist[i][i] = 0; // distance of a vertex to itself is 0

for(int k = 0; k < n; k++){

for(int i = 0; i < n; i++){

for(int j = 0; j < n; j++){

if(dist[i][k] < INF && dist[k][j] < INF)

dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);

}

}

}

// print the shortest paths

for(int i = 0; i < n; i++){

for(int j = 0; j < n; j++){

if(dist[i][j] == INF) cout << "INF ";

else cout << dist[i][j] << " ";

}

cout << endl;

}

return 0;

}

**2) Knapsack Algo. in cpp**

**Code:**

#include<bits/stdc++.h>

using namespace std;

int main(){

int n, W;

cin >> n >> W; // n - number of items, W - maximum weight of the knapsack

vector<int> w(n+1), v(n+1); // weights and values of the items

for(int i = 1; i <= n; i++) cin >> w[i] >> v[i];

vector<vector<int>> dp(n+1, vector<int>(W+1, 0));

// dp[i][j] stores the maximum value that can be obtained with items 1 to i and weight j

for(int i = 1; i <= n; i++){

for(int j = 1; j <= W; j++){

dp[i][j] = dp[i-1][j]; // exclude the i-th item

if(j >= w[i]) dp[i][j] = max(dp[i][j], dp[i-1][j-w[i]] + v[i]); // include the i-th item

}

}

cout << dp[n][W] << endl; // maximum value that can be obtained

return 0;

}