Premier University

Assignment

Assignment Title	Applying Dijkstra and Knapsack Algorithm in Real Life Scenario
Course code	CSE 225
Course name	Algorithm Design and Analysis
Date of Submission	13/02/2024

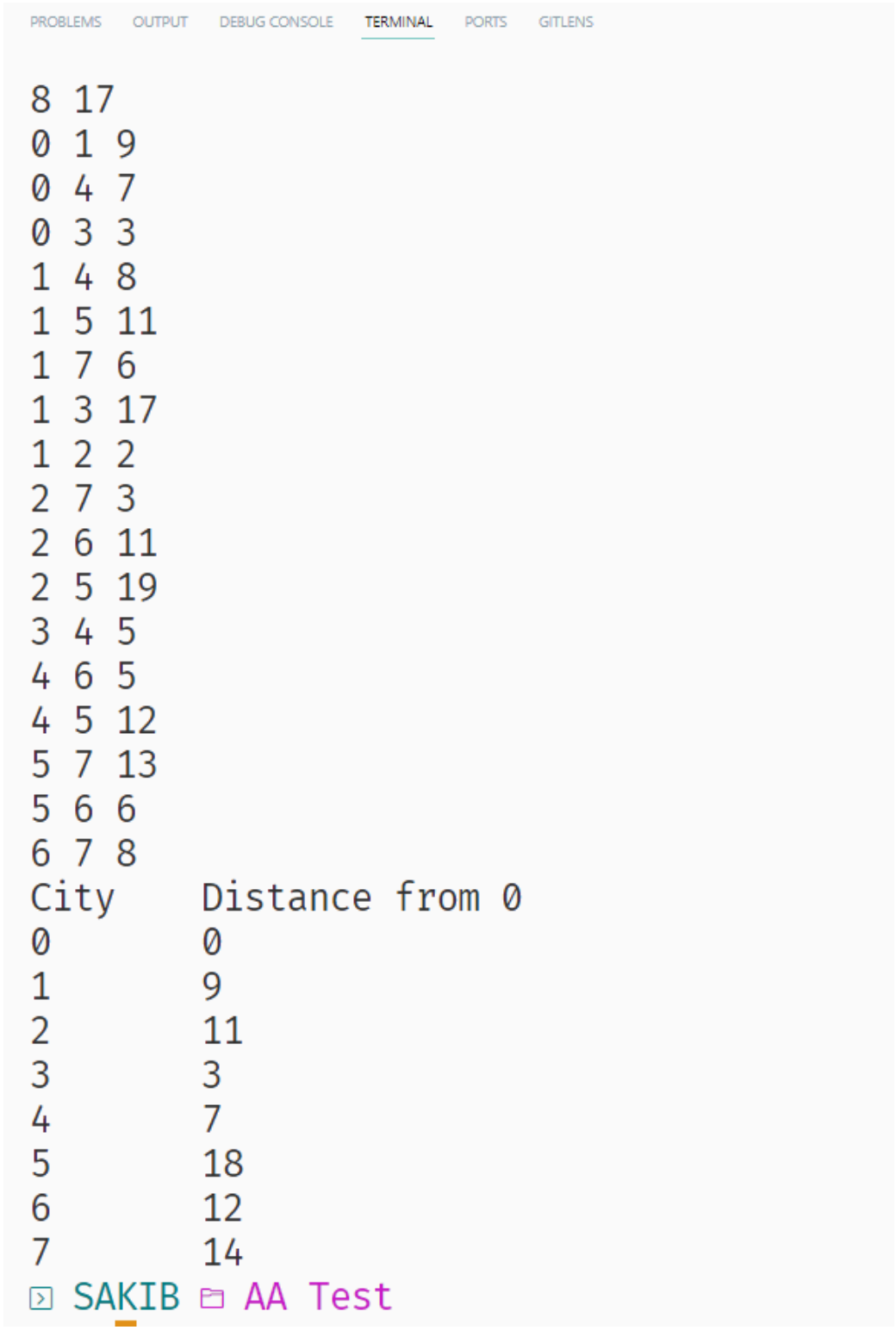
Submitted to
Fairuz Bilquis Khan Lecturer Department of CSE

Submitted by	
Name	Mohammad Hafizur Rahman Sakib
ID	0222210005101118
Section	C
Semester	4th
Session	Fall 2023

Code for the Inhabitants :

```cpp
#include <bits/stdc++.h>
using namespace std;
const int inf = 1e7;
const int n = 1e3;
vector<pair<int, int>> g[n];
// { node , cost}
void dijkstra(int source)
{
    vector<int> distance(n, inf);
    vector<bool> visited(n, 0);
    set<pair<int, int>> st;
    // {cost , node} => kept cost on first value to sort based on lowest cost
    st.insert({0, source});
    distance[source] = 0;
    while (!st.empty())
    {
        auto node = *st.begin();
        // will give the minimum weighted pair {cost , node}
        int parent_node = node.second;
        int parent_node_cost = node.first;
        st.erase(st.begin());
        if (visited[parent_node])
        {
            continue;
        }
        visited[parent_node] = 1;
        // Traverse to the child of v,for Relaxation
        for (auto child : g[parent_node])
        {
            int child_node = child.first;
            int edge_cost = child.second;

            // Relaxation
            if ((parent_node_cost + edge_cost) < distance[child_node])
            {
                distance[child_node] = (parent_node_cost + edge_cost);
                st.insert({distance[child_node], child_node});
            }
        }
    }
    cout << "Node\tDistance from " << source << endl;
    for (int i = 0; i < n; ++i)
    {
        if (distance[i] != inf)
        {
            cout << i << "\t" << distance[i] << endl;
        }
    }
}
int main()
{
    int node, edge;
    cin >> node >> edge;
    for (int i = 0; i < edge; i++)
    {
        int u, v, cost;
        cin >> u >> v >> cost;
        g[u].push_back({v, cost});
        g[v].push_back({u, cost});
        // u/v indexed node connected with v/u node with cost
    }
    dijkstra(0);

    return 0;
}

```

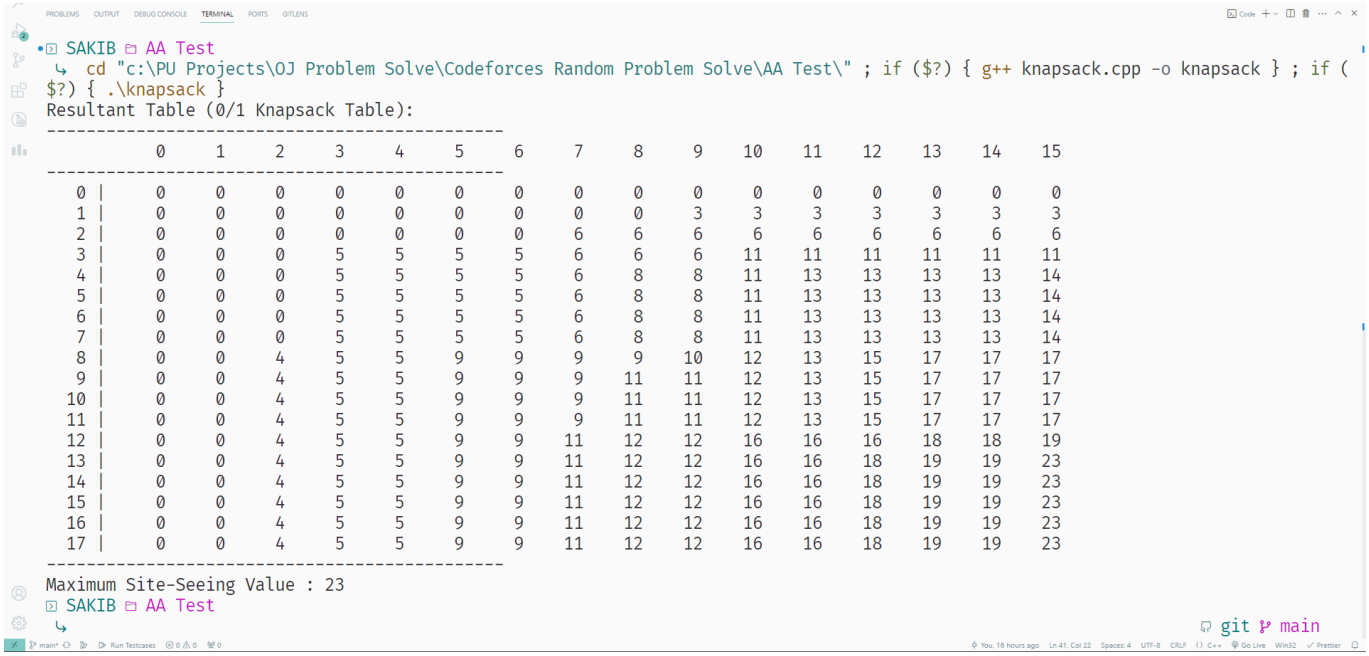
Minimum costs output for 0 number city's Inhabitants to travel into other cities using C++ :



Code for maximizing site-seeing values for tourists:

```cpp
#include <bits/stdc++.h>
using namespace std;

vector<int> ss_value = {3, 6, 5, 8, 3, 2, 6, 4, 2, 3, 3, 7, 7, 6, 5, 2, 6};
vector<int> cost = {9, 7, 3, 8, 11, 6, 17, 2, 3, 11, 19, 5, 5, 12, 13, 6, 8};
int budget = 15; // Tourists Budget

int knapsack(int W)
{
    int n = cost.size();
    vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));
    for (int i = 1; i <= n; i++)
    {
        for (int w = 1; w <= W; w++)
        {
            if (cost[i - 1] <= w)
            {
                dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - cost[i - 1]] + ss_value[i - 1]);
            }
            else
            {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }

    // Printing the resultant table
    cout << "Resultant Table (0/1 Knapsack Table):" << endl;
    cout << "-----------------------------------------------" << endl;
    cout << setw(6) << " ";
    for (int w = 0; w <= W; w++)
    {
        cout << setw(6) << w;
    }
    cout << endl;
    cout << "-----------------------------------------------" << endl;
    for (int i = 0; i <= n; i++)
    {
        cout << setw(4) << i << " |";
        for (int w = 0; w <= W; w++)
        {
            cout << setw(6) << dp[i][w];
        }
        cout << endl;
    }
    cout << "-----------------------------------------------" << endl;

    return dp[n][W];
}

int main()
{
    cout << "Maximum Site-Seeing ss_valueue : " << knapsack(budget) << endl;

    return 0;
}

```

Resultant table and output for the tourists :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

SAKIB  AA Test
 ↳  cd "c:\PU Projects\OJ Problem Solve\Codeforces Random Problem Solve\AA Test\" ; if ($?) { g++ knapsack.cpp -o knapsack } ; if (
$?) { .\knapsack }
Resultant Table (0/1 Knapsack Table):
-----------------------------------------------
          0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15
-----------------------------------------------
   0 |    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
   1 |    0     0     0     0     0     0     0     0     0     3     3     3     3     3     3     3
   2 |    0     0     0     0     0     0     0     6     6     6     6     6     6     6     6     6
   3 |    0     0     0     5     5     5     5     6     6     6    11    11    11    11    11    11
   4 |    0     0     0     5     5     5     5     6     8     8    11    13    13    13    13    14
   5 |    0     0     0     5     5     5     5     6     8     8    11    13    13    13    13    14
   6 |    0     0     0     5     5     5     5     6     8     8    11    13    13    13    13    14
   7 |    0     0     0     5     5     5     5     6     8     8    11    13    13    13    13    14
   8 |    0     0     4     5     5     9     9     9     9    10    12    13    15    17    17    17
   9 |    0     0     4     5     5     9     9     9    11    11    12    13    15    17    17    17
  10 |    0     0     4     5     5     9     9     9    11    11    12    13    15    17    17    17
  11 |    0     0     4     5     5     9     9     9    11    11    12    13    15    17    17    17
  12 |    0     0     4     5     5     9     9    11    12    12    16    16    16    18    18    19
  13 |    0     0     4     5     5     9     9    11    12    12    16    16    18    19    19    23
  14 |    0     0     4     5     5     9     9    11    12    12    16    16    18    19    19    23
  15 |    0     0     4     5     5     9     9    11    12    12    16    16    18    19    19    23
  16 |    0     0     4     5     5     9     9    11    12    12    16    16    18    19    19    23
  17 |    0     0     4     5     5     9     9    11    12    12    16    16    18    19    19    23
-----------------------------------------------
Maximum Site-Seeing Value : 23
SAKIB  AA Test
 ↳                                                                                                  git  main
```