Northern University Bangladesh (NUB)

LAB ASSIGNMENT 2

SUBMITTED BY

NAME: MAHMUDUL SAJID
ID NO: 41220300343
DEP- CSE
SEC- B
4ST SEMESTER

SUBMITTED TO

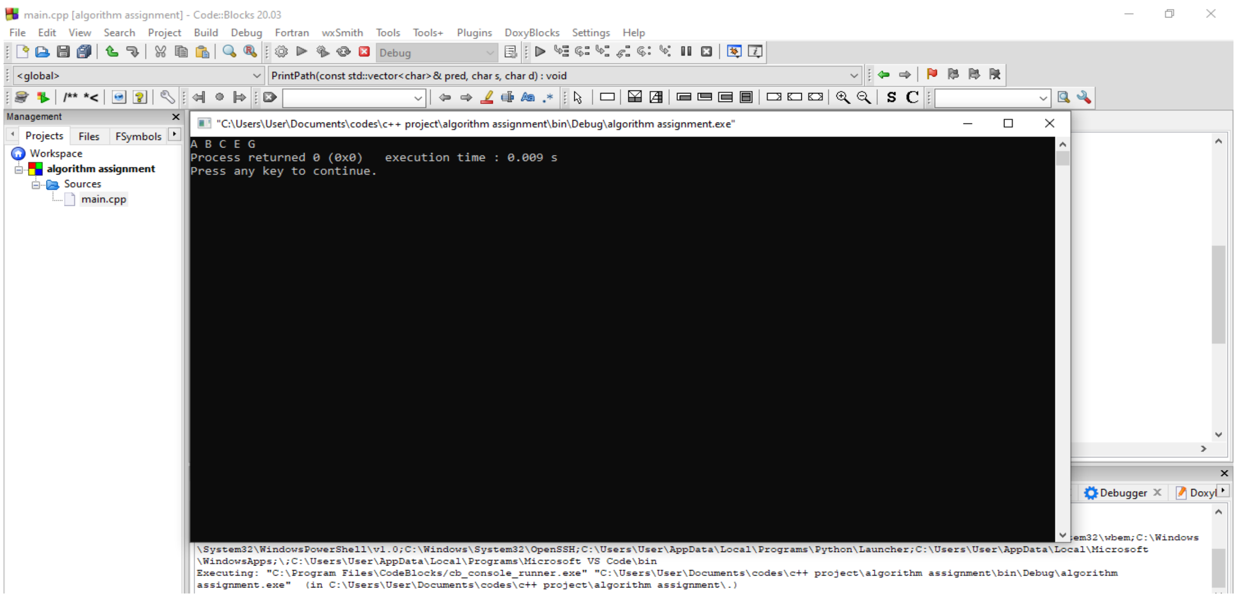
MD MAHADI HASAN,
SENIOR LECTURER
DEPARTMENT OF CSE

DATE OF SUBMISSION- 08/11/2023

Task-1

```
#include <bits/stdc++.h>
using namespace std;
void add_edge(vector<vector<char>>& adj, char src, char dest) {
adj[src - 'A'].push_back(dest);
adj[dest - 'A'].push_back(src);
}
bool BFS(const vector<vector<char>>& adj, char src, char dest, int v,
vector<char>& pred, vector<int>& dist) {
vector<bool> visited(v, false);
visited[src - 'A'] = true;
dist[src - 'A'] = 0;
queue<char> q;
q.push(src);
while (!q.empty()) {
char u = q.front();
q.pop();
for (char neighbor : adj[u - 'A']) {
if (!visited[neighbor - 'A']) {
visited[neighbor - 'A'] = true;
dist[neighbor - 'A'] = dist[u - 'A'] + 1;
pred[neighbor - 'A'] = u;
q.push(neighbor);
if (neighbor == dest)
return true;
}
}
}
return false;
}
void PrintPath(const std::vector<char>& pred, char s, char d) {
if (s == d) {
cout << s << ' ';
} else if (pred[d - 'A'] == '\0') {
cout << "No path from " << s << " to " << d << '\n';
} else {
```

```
PrintPath(pred, s, pred[d - 'A']);
cout << d << ' ';
}
}
int main() {
int v = 7;
vector<vector<char>> adj(7, vector<char>());
add_edge(adj, 'A', 'B');
add_edge(adj, 'B', 'C');
add_edge(adj, 'B', 'D');
add_edge(adj, 'C', 'E');
add_edge(adj, 'D', 'F');
add_edge(adj, 'E', 'G');
add_edge(adj, 'F', 'G');
char source = 'A', dest = 'G';
vector<char> pred(v, '\0');
vector<int> dist(v, -1);
if (BFS(adj, source, dest, v, pred, dist)) {
PrintPath(pred, source, dest);
} else {
cout << "No path from " << source << " to " << dest << '\n';
}
return 0;
}
```


main.cpp [algorithm assignment] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
PrintPath(const std::vector<char>& pred, char s, char d) : void
Management
Projects Files FSymbols
Workspace
algorithm assignment
Sources
main.cpp
Debugger
Doxyl

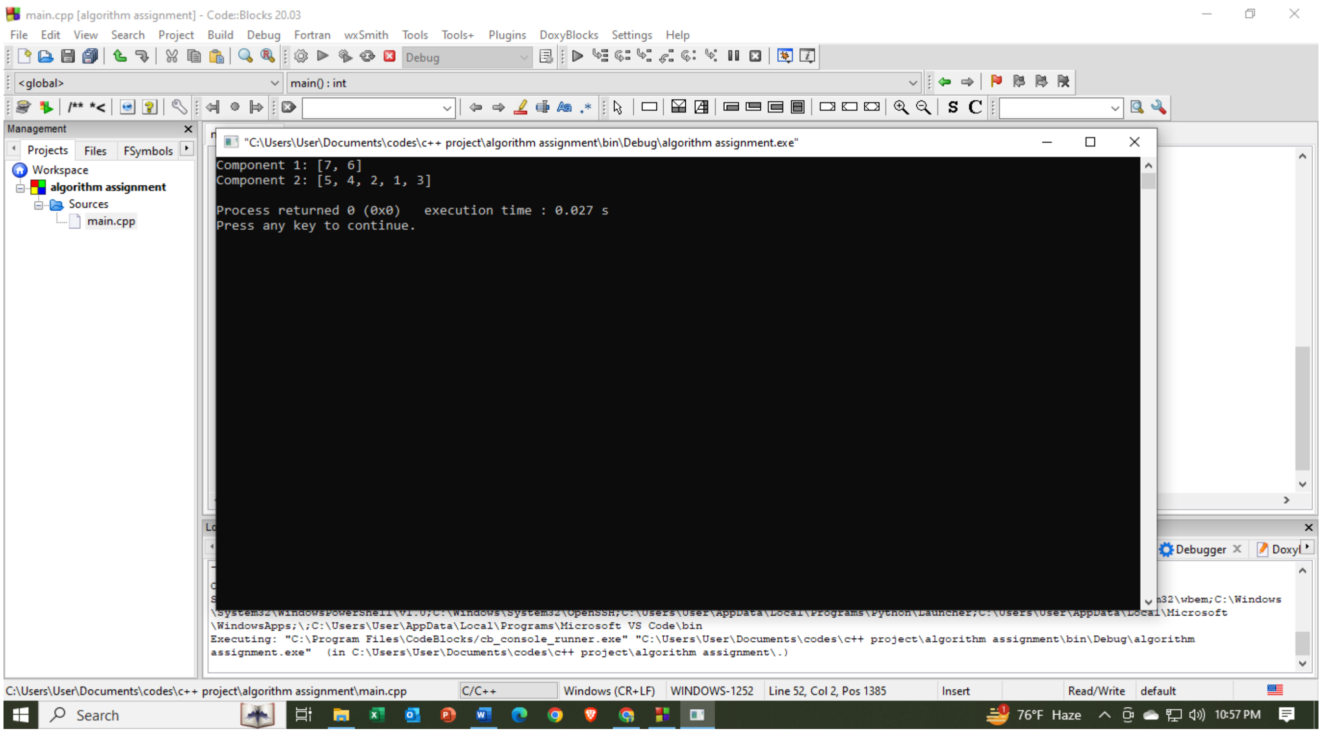
"C:\Users\User\Documents\codes\c++ project\algorithm assignment\bin\Debug\algorithm assignment.exe"

```
A B C E G
Process returned 0 (0x0)   execution time : 0.009 s
Press any key to continue.
```

```
em32\wbem;C:\Windows
\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Users\User\AppData\Local\Programs\Python\Launcher;C:\Users\User\AppData\Local\Microsoft
\WindowsApps;\;C:\Users\User\AppData\Local\Programs\Microsoft VS Code\bin
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\User\Documents\codes\c++ project\algorithm assignment\bin\Debug\algorithm
assignment.exe"  (in C:\Users\User\Documents\codes\c++ project\algorithm assignment\.)
```

Task-2

```
#include <bits/stdc++.h>
using namespace std;
void dfs(int node, unordered_map<int, vector<int>>& adjacencyList, vector<int>&
component, unordered_set<int>& visited) {
visited.insert(node);
component.push_back(node);
for (int neighbor : adjacencyList.at(node)) {
if (visited.find(neighbor) == visited.end()) {
dfs(neighbor, adjacencyList, component, visited);
}
}
}
vector<vector<int>> findConnectedComponents( unordered_map<int, vector<int>>&
adjacencyList) {
unordered_set<int> visited;
vector<vector<int>> connectedComponents;
for (const auto& entry : adjacencyList) {
int node = entry.first;
if (visited.find(node) == visited.end()) {
vector<int> component;
dfs(node, adjacencyList, component, visited);
connectedComponents.push_back(component);
}
}
return connectedComponents;
}
int main() {
unordered_map<int, vector<int>> adjacencyList = {
{1, {2, 3}},
{2, {1, 4}},
{3, {1}},
{4, {2, 5}},
{5, {4}},
{6, {7}},
{7, {6}}
};
vector<vector<int>> connectedComponents = findConnectedComponents(adjacencyList);
// Print the result
int componentNumber = 1;
for (const vector<int>& component : connectedComponents) {
cout << "Component " << componentNumber << ": [";
for (int i = 0; i < component.size(); ++i) {
cout << component[i];
if (i < component.size() - 1) {
cout << ", ";
}
}
cout << "]" << endl;
componentNumber++;
}
return 0;
}
```


main.cpp [algorithm assignment] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Files FSymbols
Workspace
algorithm assignment
Sources
main.cpp
Debugger
C:\Users\User\Documents\codes\c++ project\algorithm assignment\main.cpp
C/C++ Windows (CR+LF) WINDOWS-1252 Line 52, Col 2, Pos 1385 Insert Read/Write default
Search
76°F Haze 10:57 PM

"C:\Users\User\Documents\codes\c++ project\algorithm assignment\bin\Debug\algorithm assignment.exe"

```
Component 1: [7, 6]
Component 2: [5, 4, 2, 1, 3]

Process returned 0 (0x0)   execution time : 0.027 s
Press any key to continue.
```

```
a32\wbem;C:\Windows
\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Users\User\AppData\Local\Programs\Python\Launcher;C:\Users\User\AppData\Local\Microsoft
\WindowsApps;\;C:\Users\User\AppData\Local\Programs\Microsoft VS Code\bin
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\User\Documents\codes\c++ project\algorithm assignment\bin\Debug\algorithm
assignment.exe"  (in C:\Users\User\Documents\codes\c++ project\algorithm assignment\.)
```

Task-3

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

int minDistance(const vector<int>& dist, const set<int>& sptSet) {
    int minDist = INT_MAX, minIndex = -1;
    for (int v = 0; v < dist.size(); v++) {
        if (!sptSet.count(v) && dist[v] < minDist) {
            minDist = dist[v];
            minimum index = v;
        }
    }
    Return the minimum index;
}

void printPath(const vector<int>&parent, int dest) {
    if (parent[target] ==-1) {
        cout << target;
        return;
    }
    printPath(parent, parent[target]);
    cout <<"->"<<target;
}
vector<int> dijkstra(const vector<vector<int>>& graph, int src, int dest) {
    int V = graph.size();
    Vector<int> dist(V, INT_MAX);
    Vector<int> parent(V, -1);
    set<int> sptSet;

    distance[source] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, sptSet);
        sptSet.insert(u);

        for (int v = 0; v < V; v++) {
            if (!sptSet.count(v) && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
                parent[v] = u;
            }
        }
    }

    Returns parent.
}
int main() {
    vector<vector<int>> graph = {
        {0, 5, 0, 0, 0, 0, 0},
        {0, 0, 3, 7, 0, 0, 0},
        {0, 0, 0, 0, 4, 0, 0},
        {0, 0, 0, 0, 0, 8, 0},
        {0, 0, 0, 0, 0, 0, 6},
        {0, 0, 0, 0, 0, 0, 5},
        {0, 0, 0, 0, 0, 0, 0}
    };

    int start = 0;
    int target = 6;

    vector<int> parent = dijkstra(graph, origin, destination);

    if (parent[destination] == -1) {
        cout << "The path from "<<start<<" to "<<target<<" was not found."<< endl;
    } different {
        cout <<"The shortest path from "<<Start<<" to "<<Destination<<" is: ";
        printPath(parent, target);
        cout << endl;
    }

    return 0;
}
```