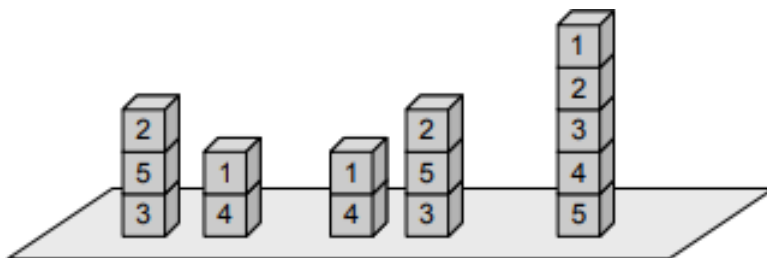


**Assignment No: 1**  
**Prn:2020BTECS00090**  
**Name:Vishal Chauhan**  
**Sub:AI-ML Lab**

1. The block's world is a well-known toy domain used in AI for planning problems related to robots. It consists of a set of blocks of various size, shape, and color, possibly identified by a letter or by a number, and placed on a surface (e.g., on a table); the blocks can be stacked into one or more piles, possibly with some constraints (depending, e.g., on their size); the goal is to form a target set of piles starting from a given initial configuration, by moving one block at a time, either to the table or atop another pile of blocks, and only if it is not under another block. Consider a simple version of this problem, in which there are five blocks with identical shape and size, numbered from 1 to 5.

The figure below shows two possible configurations of such blocks; note that the relative position of the piles does not matter, thus the configuration on the left is equivalent to the one in the middle. Every block can be either on the table (there are no constraints on the number of piles) or atop any another block. The goal is to stack the blocks in a single pile, in the order shown in the rightmost configuration in the figure, starting from any given set of piles, by moving the smallest possible number of blocks. Only blocks at the top of the current piles can be moved, and can be placed only on the table or a top another pile.



Formulate the above version of the block's world as a search problem, by precisely defining the state space, the initial state, the goal test, the set of possible actions and the path cost (based on cost of moving blocks).

Answer:-

The state space will be made up of set of distinct arrangements of the five blocks. The current state can be represented using the specific arrangement of the blocks in each pile. The possible or legal actions would be to move a single block from the top of the pile or ground and put it either on the top of the another pile or on the ground.

Here the goal state would be defined or said to be reached when in minimum number of moves we reach the target configuration and a path cost would be associated with it and the path cost will be given by the number of moves made to reach the goal state. Thus for each move the cost would be equal to 1.

2. You start with the sequence ABABAECCCEC, or in general any sequence made from A, B, C, and E. You can transform this sequence using the following equalities:  $AC = E$ ,  $AB = BC$ ,  $BB = E$ , and  $E * x = x$  for any  $x$ . For example, ABBC can be transformed into AEC, and then AC, and then E. Your goal is to produce the sequence E.

Q2. You start with the sequence ABABAECCCEC or in general any sequence made from A, B, C and E. You can transform this sequence made from A, B, C and E. You can transform this sequence using the following equalities:  $AC = E$ ,  $AB = BC$ ,  $BB = E$  and  $E * x = x$  for any  $x$ . For example ABBC can be transformed into AEC, and then AC and then E. Your goal is to produce the sequence E.

→ Answer:-

Approach 1:-

ABABAECCCEC

Step 1: Converting the AB to BC we get

ABBCAECCEC

Step 2: Converting the BB to E

AECAECCCEC

Step 3: Converting AE, EC, EC to correspond

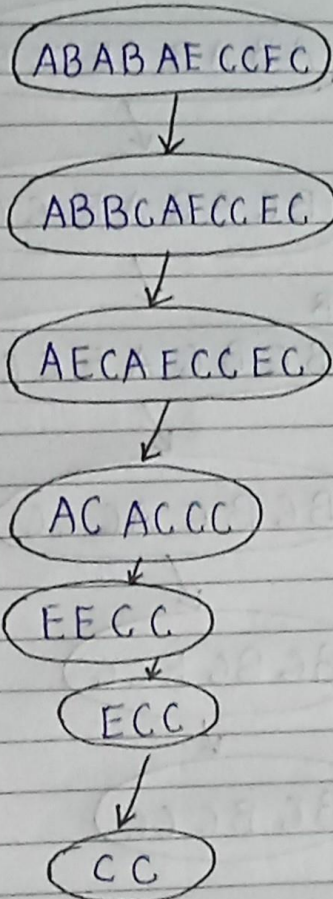
g A, E, E

AAEC ACACCC

Step 4: Converting AC to E

~~AEE~~ EEC

Approach I:-



Finally we get CC as the reduced string which is not our goal state hence the approach is discarded.



Step 5:- Converting EC to C  
 ECC

Step 6:- Converting EC to C  
 CC

So we can conclude as this is the least possible reduced by this method so we can understand that this method doesn't give solution so we need to try an alternative method.

Approach 2:-

ABABAECCEC

Step 1:- Converting AB to BC  
 BCABAECC

Step 2:- Converting AB to BC  
 BCBCAECC

Step 3:- Converting EC to C  
 BCBC ACCC

~~The we cannot~~

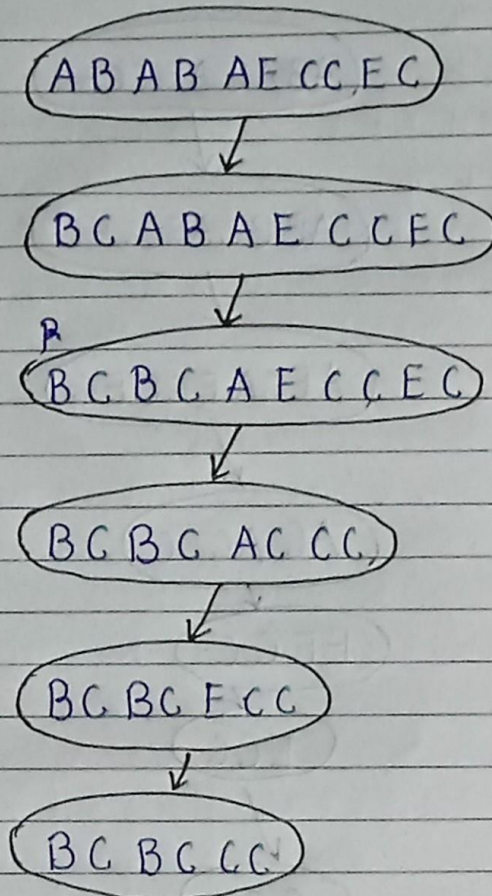
Step 4:- Converting AC to E  
 BCBC ECC

Step 5:- Converting EC to C

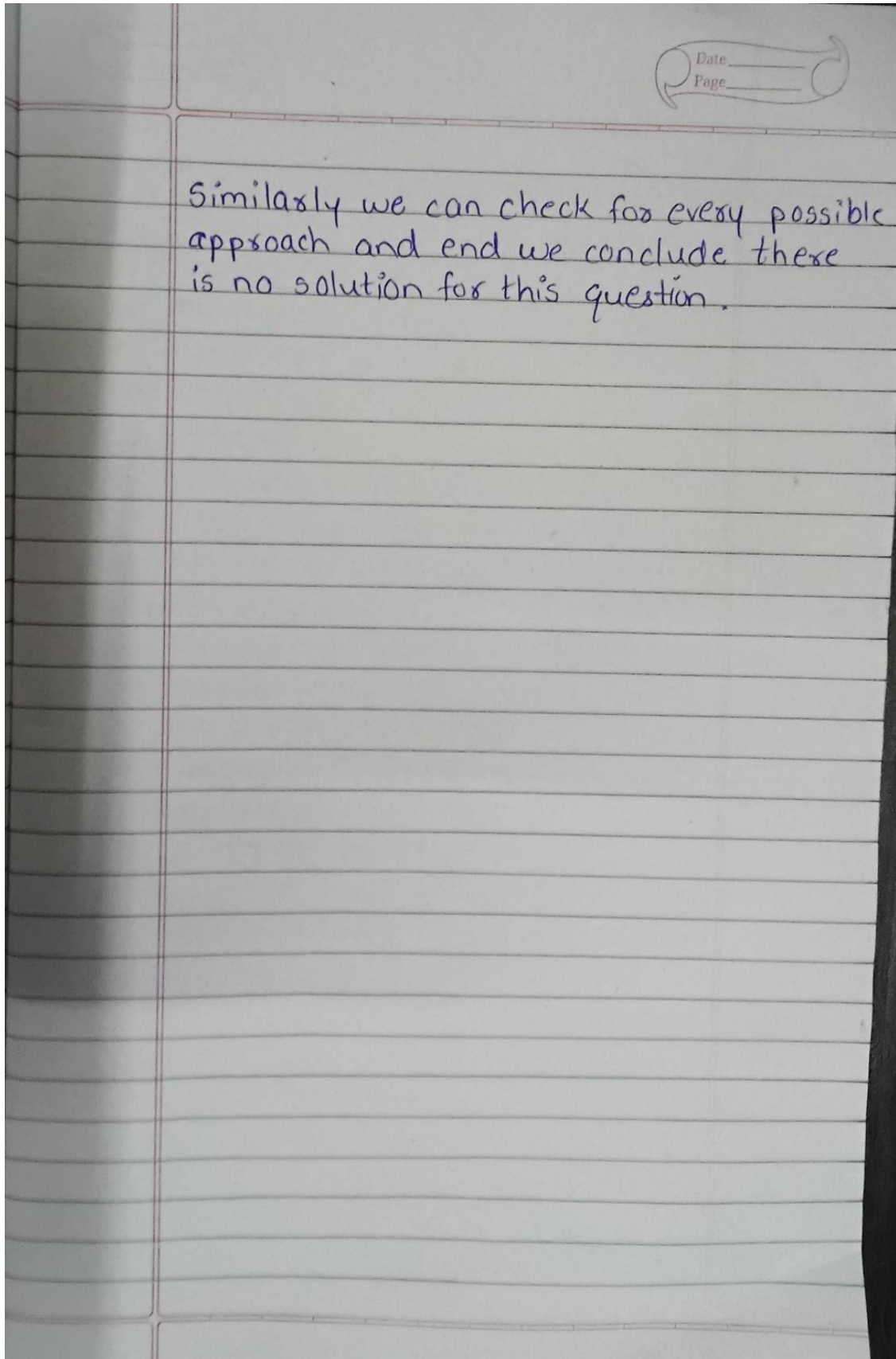
BCBCCC

We cannot further reduce the string

Approach 2:-

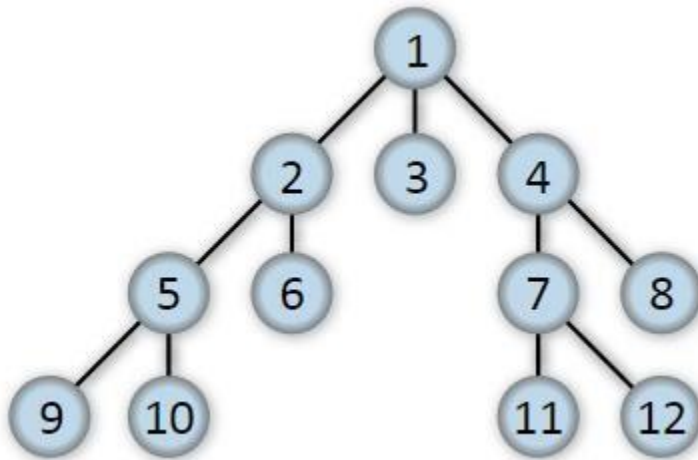


Finally we get BCBC,CC as the reduced string and we cannot further reduce it hence this approach is also discarded as it is not equal to E which is our goal state.





3. For the tree given below:



If starting state is 1 and goal state is 7, implement BFS and DFS search strategies for above tree. Which searching strategy will be best for this problem? Justify your answer with proper explanation.

DFS Output:-

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

int main(){
    int n,m;
    cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    int key;
    cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);
    for(int i=0;i<m;i++){
        int a,b;
        cin>>a>>b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

    cout<<"Enter the links between the nodes"<<endl;
    stack<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
        int data = q.top();
        cout<<data<<" ";
        if(data==key){
            break;
        }
        q.pop();
        reverse(adj[data].begin(),adj[data].end());
        for(auto it:adj[data]){
            if(!visited[it]){
                visited[it] = 1;
            }
        }
    }
}
```



```
        q.push(it);
    }
}

return 0;
}
```

```
Question_3_DFS.cpp - AI Assignments - Visual Studio Code
C++ Question_3_BFS.cpp C++ Question_3_DFS.cpp X
C++ Question_3_DFS.cpp > main()
7   cin>>n>>m;
8   int key;
9   cout<<"Enter the element you want to find: "<<endl;
10  cin>>key;

PROBLEMS OUTPUT TIMELINE JUPYTER DEBUG CONSOLE TERMINAL
7
1 2
1 3
1 4
2 5
2 6
5 9
5 10
4 7
4 8
7 11
7 12
Enter the links between the nodes
1 2 5 9 10 6 3 4 7
PS D:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer

Ln 11, Col 26 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier 2:26 PM 8/24/2022
```

## BFS Output:-

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

int main(){
    int n,m;
    cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    int key;
    cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    cout<<"Enter the links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        int a,b;
        cin>>a>>b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

    queue<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
```

```
int data = q.front();  
cout<<data<<" ";  
if(data==key){  
    break;  
}  
q.pop();  
for(auto it:adj[data]){  
    if(!visited[it]){  
        visited[it] = 1;  
        q.push(it);  
    }  
}  
}  
return 0;  
}
```

Visual Studio Code interface showing the execution of a C++ program. The editor displays 'Question\_3.cpp' with the following code:

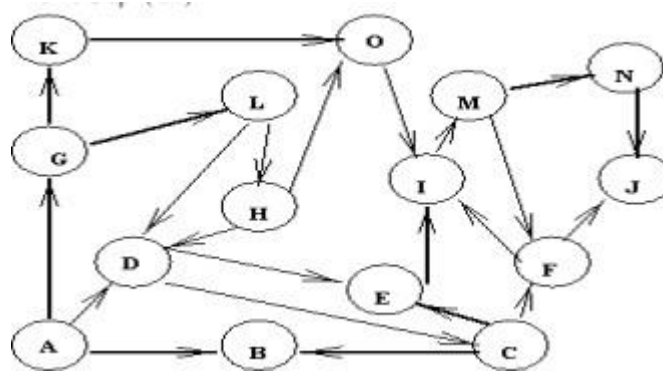
```
13     adj[b].push_back(a);  
14 }  
15  
16  
17 queue<int> q;
```

The terminal window shows the output of the program:

```
1 4  
2 5  
2 6  
5 9  
5 10  
4 7  
4 8  
7 11  
7 12  
1 2 3 4 5 6 7 8 9 10 11 12  
PS D:\D Drive Data\Vyankatesh B  
ce and Engineering\Third Year\A
```

The status bar at the bottom indicates the file is 'Question\_3.cpp' at line 16, column 1, using UTF-8 encoding and C++ language.

4. Consider the traffic map of certain city is given below



Identify different paths using

a. BFS with start state A and goal state N

### CPP Implementation:-

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

void printpath(vector<int>& path)
{
    char ch='A';
    map<int,char>mp;

    for(int i=0;i<15;i++)
    {
        mp[i]=ch;
        ch++;
    }

    int size = path.size();
    for (int i = 0; i < size; i++)
        cout << mp[path[i]] << " ";
    cout << endl;
}

int isNotVisited(int x, vector<int>& path)
{
    int size = path.size();
    for (int i = 0; i < size; i++)
        if (path[i] == x)
            return 0;
    return 1;
}

void findpaths(vector<vector<int> >&g, int src,int dst, int v)
{
    queue<vector<int> > q;

    vector<int> path;
```

```
path.push_back(src);
q.push(path);
while (!q.empty()) {
    path = q.front();
    q.pop();
    int last = path[path.size() - 1];

    if (last == dst)
        printpath(path);

    for (int i = 0; i < g[last].size(); i++) {
        if (isNotVisited(g[last][i], path)) {
            vector<int> newpath(path);
            newpath.push_back(g[last][i]);
            q.push(newpath);
        }
    }
}

int main()
{
    int n=16;
    vector<vector<int> > g(n+1);
    // number of vertices

    int m=23;

    for(int i=0;i<m;i++)
    {
        char c1,c2;
        cin>>c1>>c2;

        int u=c1-'A';
        int v=c2-'A';

        // cout<<u<<" "<<v<<endl;

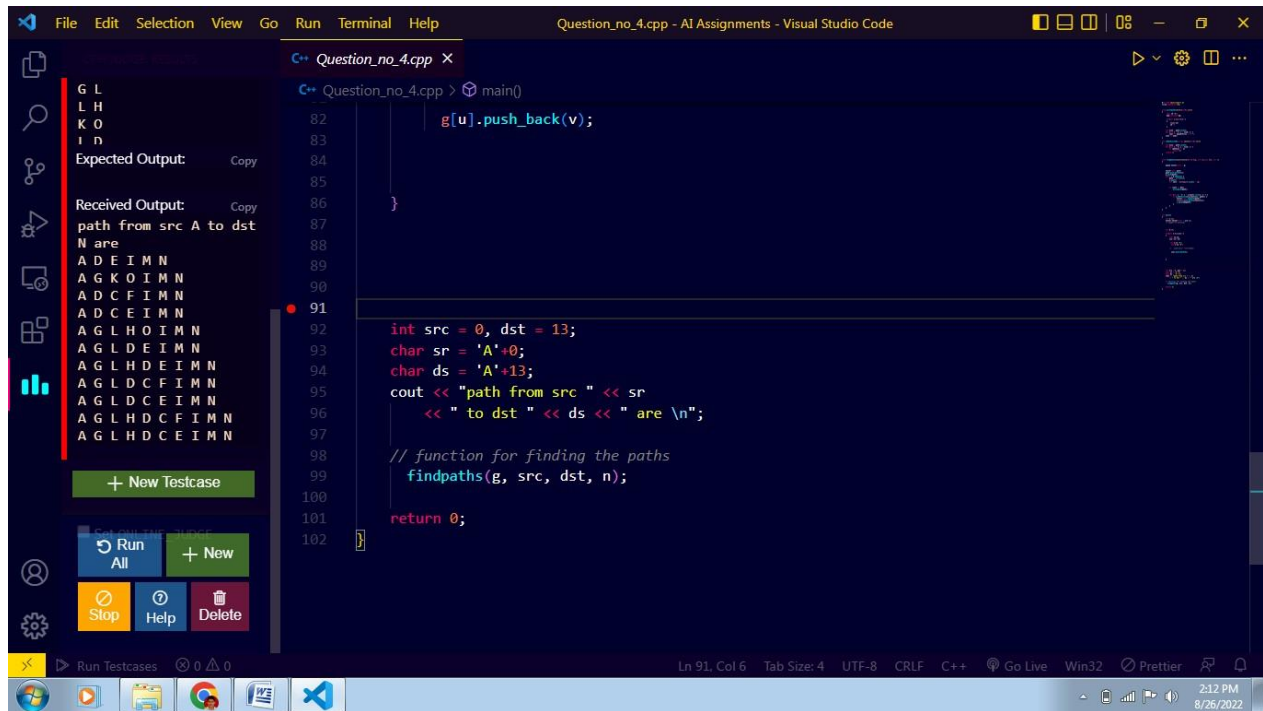
        g[u].push_back(v);
    }

    int src = 0, dst = 13;
    char sr = 'A'+0;
    char ds = 'A'+13;
    cout << "path from src " << sr
        << " to dst " << ds << " are \n";

    // function for finding the paths
    findpaths(g, src, dst, n);

    return 0;
}
```





The screenshot shows a Visual Studio Code editor with a C++ file named `Question_no_4.cpp`. The code implements a Depth-First Search (DFS) algorithm to find paths from a source 'A' to a destination 'N' in a 15x15 grid. The grid contains obstacles represented by 'X'. The program uses a recursive `findpaths` function to explore all possible paths. The output shows the expected and received paths.

```
g l
l h
k o
i n
Expected Output: Copy
Received Output: Copy
path from src A to dst
N are
A D E I M N
A G K O I M N
A D C F I M N
A D C E I M N
A G L H O I M N
A G L D E I M N
A G L H D E I M N
A G L D C F I M N
A G L D C E I M N
A G L H D C F I M N
A G L H D C E I M N
+ New Testcase
Run All + New
Stop Help Delete
Run Testcases 0 0 0
Ln 91, Col 6 Tab Size: 4 UTF-8 CRLF C++ Go Live Win32 Prettier 2:12 PM 8/26/2022
```

```
Question_no_4.cpp X
C++ Question_no_4.cpp > main()
82     g[u].push_back(v);
83
84
85
86
87
88
89
90
91
92     int src = 0, dst = 13;
93     char sr = 'A'+0;
94     char ds = 'A'+13;
95     cout << "path from src " << sr
96         << " to dst " << ds << " are \n";
97
98     // function for finding the paths
99     findpaths(g, src, dst, n);
100
101     return 0;
102 }
```

b. DFS with start state A and goal state N

Cpp Implementation:-

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

void printpath(vector<int>& path)
{
    char ch='A';
    map<int,char>mp;

    for(int i=0;i<15;i++)
    {
        mp[i]=ch;
        ch++;
    }

    int size = path.size();
    for (int i = 0; i < size; i++)
        cout << mp[path[i]] << " ";
    cout << endl;
}

int isNotVisited(int x, vector<int>& path)
{
    int size = path.size();
    for (int i = 0; i < size; i++)
        if (path[i] == x)
            return 0;
    return 1;
}
```

```
void findpaths(vector<vector<int> >&g, int src,int dst, int v)
{
    stack<vector<int> > q;

    vector<int> path;
    path.push_back(src);
    q.push(path);
    while (!q.empty()) {
        path = q.top();
        q.pop();
        int last = path[path.size() - 1];

        if (last == dst)
            printpath(path);

        for (int i = 0; i < g[last].size(); i++) {
            if (isNotVisited(g[last][i], path)) {
                vector<int> newpath(path);
                newpath.push_back(g[last][i]);
                q.push(newpath);
            }
        }
    }
}

int main()
{
    int n=16;
    vector<vector<int> > g(n+1);
    // number of vertices

    int m=23;

    for(int i=0;i<m;i++)
    {
        char c1,c2;
        cin>>c1>>c2;

        int u=c1-'A';
        int v=c2-'A';

        // cout<<u<<" "<<v<<endl;

        g[u].push_back(v);

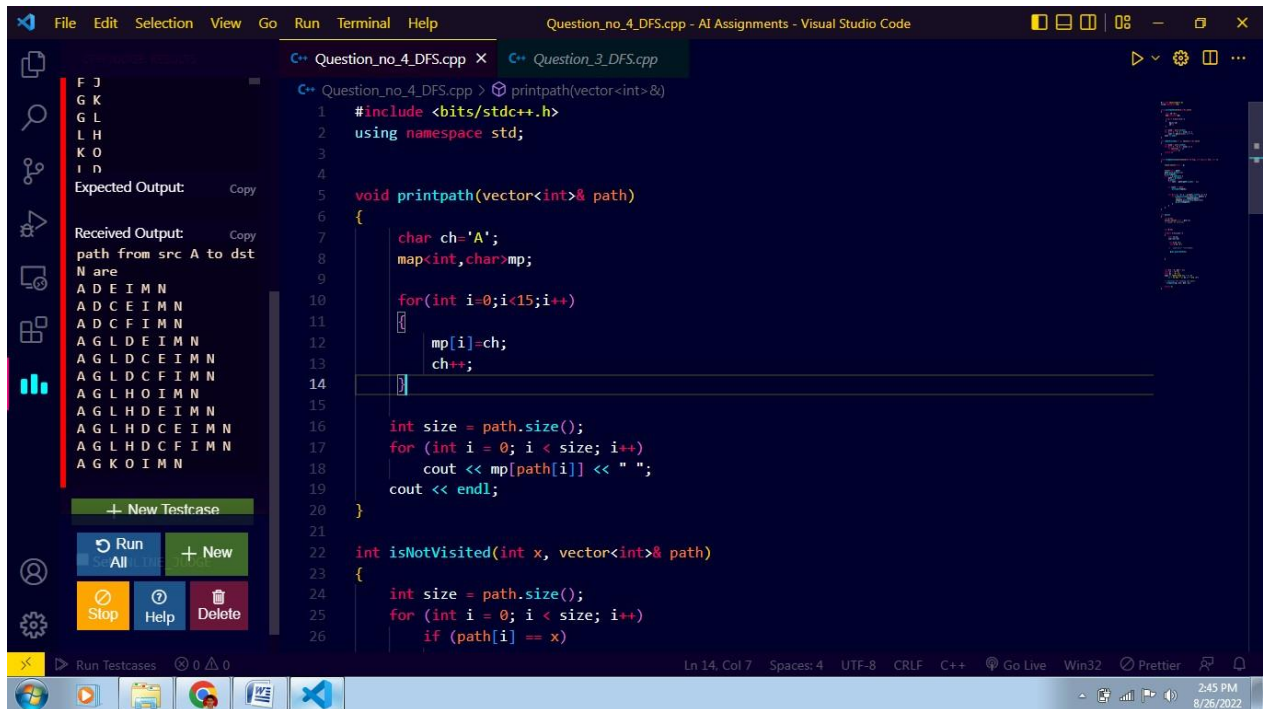
    }

    int src = 0, dst = 13;
    char sr = 'A'+0;
    char ds = 'A'+13;
    cout << "path from src " << sr
```

```
<< " to dst " << ds << " are \n";

// function for finding the paths
findpaths(g, src, dst, n);

return 0;
}
```



```
File Edit Selection View Go Run Terminal Help
Question_no_4_DFS.cpp - AI Assignments - Visual Studio Code
C++ Question_no_4_DFS.cpp x C++ Question_3_DFS.cpp
C++ Question_no_4_DFS.cpp > printpath(vector<int>& path)
1 #include <bits/stdc++.h>
2 using namespace std;
3
4
5 void printpath(vector<int>& path)
6 {
7     char ch='A';
8     map<int,char>mp;
9
10    for(int i=0;i<15;i++)
11    {
12        mp[i]=ch;
13        ch++;
14    }
15
16    int size = path.size();
17    for (int i = 0; i < size; i++)
18        cout << mp[path[i]] << " ";
19    cout << endl;
20 }
21
22 int isNotVisited(int x, vector<int>& path)
23 {
24     int size = path.size();
25     for (int i = 0; i < size; i++)
26         if (path[i] == x)
```

Expected Output: Copy

Received Output: Copy

path from src A to dst

N are

A D E I M N

A D C E I M N

A D C F I M N

A G L D E I M N

A G L D C E I M N

A G L D C F I M N

A G L H O I M N

A G L H D E I M N

A G L H D C E I M N

A G L H D C F I M N

A G K O I M N

+ New Testcase

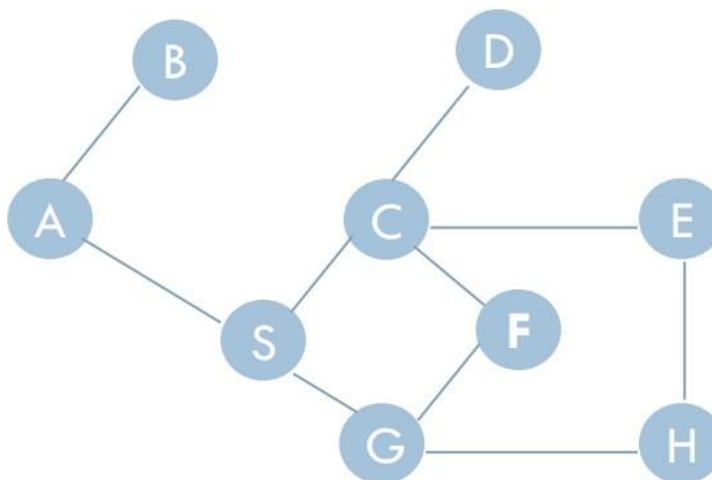
Run All + New

Stop Help Delete

Run Testcases 0/0 0

Ln 14, Col 7 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier 2:45 PM 8/26/2022

5. For the graph given below



Implement BFS and DFS technique to identify a path from start node A to goal node E.

Answer:

BFS Traversal:

Implementation:

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

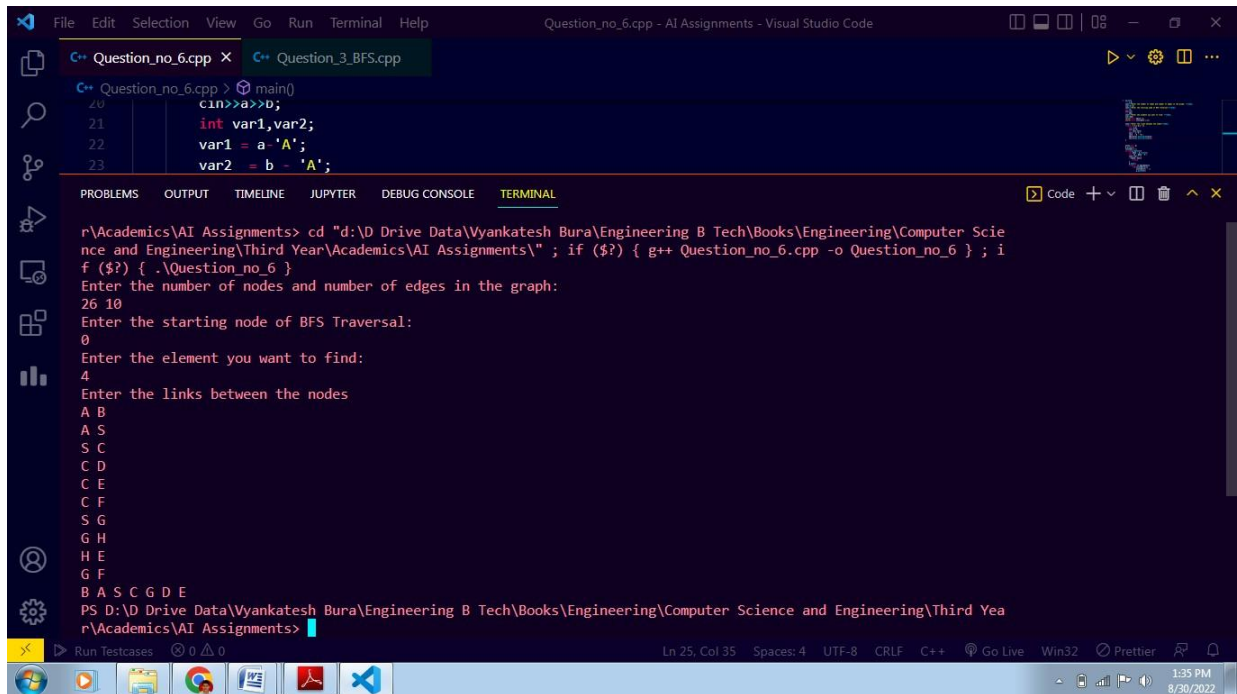
int main(){
    int n,m;
    cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    cout<<"Enter the starting node of BFS Traversal:"<<endl;
    int st;
    cin>>st;
    int key;
    cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    cout<<"Enter the links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        char a,b;
        cin>>a>>b;
        int var1,var2;
        var1 = a-'A';
        var2 = b - 'A';
        adj[var1].push_back(var2);
        adj[var2].push_back(var1);
    }

    queue<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
        int data = q.front();
        char ch = data +'A';
        cout<<ch<<" ";
        if(data==key){
            break;
        }
        q.pop();
        for(auto it:adj[data]){
            if(!visited[it]){
                visited[it] = 1;
                q.push(it);
            }
        }
    }
    return 0;
}
```



Output:



```
File Edit Selection View Go Run Terminal Help
Question_no_6.cpp - AI Assignments - Visual Studio Code

C++ Question_no_6.cpp x C++ Question_3_BFS.cpp

C++ Question_no_6.cpp > main()
21 cin>>a>>b;
22 int var1,var2;
23 var1 = a-'A';
var2 = b - 'A';

PROBLEMS OUTPUT TIMELINE JUPYTER DEBUG CONSOLE TERMINAL
r\Academics\AI Assignments> cd "d:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer Science and Engineering\Third Year\Academics\AI Assignments\"; if ($?) { g++ Question_no_6.cpp -o Question_no_6 }; if ($?) { .\Question_no_6 }
Enter the number of nodes and number of edges in the graph:
26 10
Enter the starting node of BFS Traversal:
0
Enter the element you want to find:
4
Enter the links between the nodes
A B
A S
S C
C D
C E
C F
S G
G H
H E
G F
B A S C G D E
PS D:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer Science and Engineering\Third Year\Academics\AI Assignments>
```

DFS Traversal:

Implementation:

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

int main(){
    int n,m;
    // cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    // cout<<"Enter the starting node of BFS Traversal:"<<endl;
    int st;
    cin>>st;
    int key;
    // cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    // cout<<"Enter the Links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        char a,b;
        cin>>a>>b;
        int var1,var2;
        var1 = a-'A';
        var2 = b - 'A';
        adj[var1].push_back(var2);
        adj[var2].push_back(var1);
    }
}
```

```
stack<int> q;  
q.push(1);  
visited[1]=1;  
while(!q.empty()){  
    int data = q.top();  
    char ch = data + 'A';  
    cout<<ch<<" ";  
    if(data==key){  
        break;  
    }  
    q.pop();  
    for(auto it:adj[data]){  
        if(!visited[it]){  
            visited[it] = 1;  
            q.push(it);  
        }  
    }  
}  
return 0;  
}
```

Output:

The screenshot displays the Visual Studio Code interface. The main editor shows a C++ file named 'Question\_no\_6B.cpp' with the following code:

```
1 int main(){  
2     int n,m;  
3     // cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;  
4     cin>>n>>m;  
5     // cout<<"Enter the starting node of BFS Traversal:"<<endl;  
6     int st;  
7     cin>>st;  
8     int key;  
9     // cout<<"Enter the element you want to find: "<<endl;  
10    cin>>key;  
11    vector<int> adj[n+1];  
12    vector<int> visited(n+1,0);  
13  
14    // cout<<"Enter the Links between the nodes"<<endl;  
15    for(int i=0;i<m;i++){  
16        char a,b;  
17        cin>>a>>b;  
18        int var1,var2;  
19        var1 = a - 'A';  
20        var2 = b - 'A';  
21        adj[var1].push_back(var2);  
22        adj[var2].push_back(var1);  
23    }  
24  
25    stack<int> q;  
26    q.push(1);  
27    visited[1]=1;  
28    while(!q.empty()){  
29        int data = q.top();  
30        char ch = data + 'A';  
31        cout<<ch<<" ";  
32        if(data==key){  
33            break;  
34        }  
35        q.pop();  
36        for(auto it:adj[data]){  
37            if(!visited[it]){  
38                visited[it] = 1;  
39                q.push(it);  
40            }  
41        }  
42    }  
43    return 0;  
44 }
```

On the left sidebar, the 'Testcase' panel shows a single test case that has passed. The input is:

```
26 10  
0 4  
A B  
A S  
S C  
S G  
C D  
C E  
C F  
G F  
G H  
H E
```

The expected output is 'B A S G H E' and the received output is also 'B A S G H E'. Below the test case, there are buttons for 'Run All', 'Run', 'Stop', 'Help', and 'Delete'. The status bar at the bottom indicates 'Ln 11, Col 13', 'Spaces: 4', 'UTF-8', 'CRLF', 'C++', 'Go Live', 'Win32', 'Prettier', and the time '1:43 PM 8/30/2022'.