Data Structures And Algorithm Digital Assignment

Aadhitya Swarnesh I

Question No 2 : The Great Escape

Code :

```
// Breadth First Traversal of a Non-directional Unweighted Graph
#include<iostream>
using namespace std;

template<class temp>
class Queue
{
    public :
        int size, front, rear;
        temp *arr;

        Queue()
        {
            size = 50;
            front = 0;
            rear = -1;
            arr = new temp[size];
        }

        void enqueue(temp ele)
        {
            arr[++rear] = ele;
        }

        temp dequeue()
        {
            temp ele;
            ele = arr[front++];
            return(ele);
        }

        bool isEmpty()
        {
            if(rear == -1 || front == rear+1)
            {
```

```
            return(true);
        }
        return(false);
    }

    bool isFull()
    {
        if(rear == size-1)
        {
            return(true);
        }
        return(false);
    }
};


template<class tempName>
class Node
{
    public :
        tempName name;
        tempName *adj;
        int n;
        Node()
        {
            adj = new tempName[20];
            n=0;
        }

        void insertAdjascentNode(tempName ins_node)
        {
            bool flag = true;
            for(int i=0;i<n;i++)
            {
                if(adj[i] == ins_node)
                {
                    flag = false;
                    break;
                }
            }
            if(flag)
            {
                adj[n] = ins_node;
                n++;
            }
        }
};


template<class tempName>
class Graph
{
    public :
        tempName name;
        Node<tempName> *nodeList;
        int n;
        Graph()
        {
            nodeList = new Node<tempName>[20];
            n=0;
```

```
}

void getInputByEdges()
{
   int numEdges;
   cout<<"Enter the number of pairs of buildings that are close : ";
   cin>>numEdges;
   tempName s1;
   tempName s2;
   bool f1, f2;
   for(int i=0;i<numEdges;i++)
   {
      cout<<(i+1)<<") Enter the name of the close buildings : ";
      cin>>s1>>s2;
      f1 = false;
      f2 = false;
      if(s1!=s2)
      {
         for(int j=0;j<n;j++)
         {
            if(!(f1) && nodeList[j].name==s1)
            {
               nodeList[j].insertAdjascentNode(s2);
               f1 = true;
            }
            if(!(f2) && nodeList[j].name==s2)
            {
               nodeList[j].insertAdjascentNode(s1);
               f2 = true;
            }
         }
         if(f1 == false)
         {
            Node<tempName> new_node;
            new_node.name = s1;
            nodeList[n] = new_node;
            nodeList[n].insertAdjascentNode(s2);
            n++;
         }
         if(f2 == false)
         {
            Node<tempName> new_node;
            new_node.name = s2;
            nodeList[n] = new_node;
            nodeList[n].insertAdjascentNode(s1);
            n++;
         }
      }
   }
}

void printGraph()
{
   cout<<"Graph : "<<endl;
   for(int i=0;i<n;i++)
   {
      cout<<nodeList[i].name<<" : ";
      for(int j=0;j<nodeList[i].n;j++)
      {
         if(j < nodeList[i].n-1)
```

```
                {
                    cout<<nodeList[i].adj[j]<<", ";
                }
                else
                {
                    cout<<nodeList[i].adj[j];
                }
            }
            cout<<endl;
        }
    }
};

int main()
{
    typedef int tempName;
    Graph<tempName> graph;


cout<<"-------------------------------------------------------------------------------------------------------"<<e
ndl;
    cout<<"Welcome"<<endl;

cout<<"-------------------------------------------------------------------------------------------------------"<<e
ndl;

    int numBuildings;
    cout<<"Enter the number of buildings : ";
    cin>>numBuildings;

    graph.getInputByEdges();


cout<<"-------------------------------------------------------------------------------------------------------"<<e
ndl;
    graph.printGraph();

cout<<"-------------------------------------------------------------------------------------------------------"<<e
ndl;

    tempName src, dest;
    cout<<"Enter the source node : ";
    cin>>src;
    cout<<"Enter the destination node : ";
    cin>>dest;

    Queue<tempName> queue;
    queue.enqueue(src);
    tempName ele;

    Queue<int> numqueue;
    numqueue.enqueue(0);
    int number = 0;

    tempName* visited = new tempName[20];
    int i, j, n = 0, pos;
    bool flag1, flag2, flag;
    flag = false;

    while(!queue.isEmpty())
```

```
{
    ele = queue.dequeue();
    number = numqueue.dequeue();
    flag1 = true;
    for(i=0;i<n;i++)
    {
        if(visited[i] == ele)
        {
            flag1 = false;
            break;
        }
    }
    if(flag1 == true)
    {
        pos = 0;
        visited[n] = ele;
        n++;
        for(i=0;i<graph.n;i++)
        {
            if(graph.nodeList[i].name == ele)
            {
                pos = i;
                break;
            }
        }
        for(i=0;i<graph.nodeList[pos].n;i++)
        {
            flag2 = true;
            for(j=0;j<n;j++)
            {
                if(visited[j] == graph.nodeList[pos].adj[i])
                {
                    flag2 = false;
                    break;
                }
            }
            if(flag2 == true)
            {
                if(dest == graph.nodeList[pos].adj[i])
                {
                    flag = true;
                    break;
                }
                queue.enqueue(graph.nodeList[pos].adj[i]);
                numqueue.enqueue(number+1);
            }
        }
    }
    if(flag == true)
    {
        break;
    }
}

if(flag == true)
{
    cout<<"The Hero needs "<<(number+1)<<" hops to reach his destination.";
}
else
{
```

```
        cout<<"The Hero cannot reach his destination at all.";
    }

    cout<<endl;

cout<<"-----------------------------------------------------------------------------------------------------"<<e
ndl;
}
```

```
(base) Aadhityas-MacBook-Air:DSA-Course aadhitya$ g++ GreatEscape.cpp
(base) Aadhityas-MacBook-Air:DSA-Course aadhitya$ ./a.out
-----------------------------------------------------------------------------------------------------
Welcome
-----------------------------------------------------------------------------------------------------
Enter the number of buildings : 5
Enter the number of pairs of buildings that are close : 5
1) Enter the name of the close buildings : 1
3
2) Enter the name of the close buildings : 2
3
3) Enter the name of the close buildings : 1
2
4) Enter the name of the close buildings : 3
5
5) Enter the name of the close buildings : 4
5
-----------------------------------------------------------------------------------------------------
Graph :
1 : 3, 2
3 : 1, 2, 5
2 : 3, 1
5 : 3, 4
4 : 5
-----------------------------------------------------------------------------------------------------
Enter the source node : 1
Enter the destination node : 4
The Hero needs 3 hops to reach his destination.
-----------------------------------------------------------------------------------------------------
```

Question No 5 : Dividing Sequences

Code :

```
#include<iostream>
using namespace std;

int main()
{
   int n, i, j, maxleng=1, ma;

   cout<<"Enter the number of elements in the sequence : ";
   cin>>n;
   int arr[n], parr[n];
   cout<<"Enter the sequence of numbers : ";
   for(i=0;i<n;i++)
   {
      cin>>arr[i];
   }

   //parr[0] = 1;
   for(i=0;i<n;i++)
   {
      ma = 0;
      parr[i] = 0;
      for(j=0;j<i;j++)
      {
         if(arr[i]%arr[j]==0 && parr[j]>ma)
         {
            ma = parr[j];
         }
      }
      parr[i] = ma+1;
      if(maxleng < parr[i])
      {
         maxleng = parr[i];
      }
   }

   cout<<"The length of the longest fully dividing subsequence of the input sequence is :
"<<maxleng<<endl;
}
```

```
(base) Aadhityas-MacBook-Air:DSA-Course aadhitya$ g++ DividingSequences.cpp
(base) Aadhityas-MacBook-Air:DSA-Course aadhitya$ ./a.out
Enter the number of elements in the sequence : 14
Enter the sequence of numbers : 2
11
16
12
36
60
71
17
29
144
288
129
432
993
The length of the longest fully dividing subsequence of the input sequence is : 5
```