CSCE 340 PROJECT 3 BACKTRACKING ALGORITHMS GRAPH ISOMORPHISM

GODFREY MUGANDA

1. The Assignment

You are to use the backtracking framework introduced in lecture to write a program that determines whether two graphs specified via adjacency matrices are isomorphic. If so, your program is to print out a function that witnesses for isomorphism, otherwise, it should print out that the two graphs are not isomorphic.

Two graphs $G_1 = (V_1, E_1)$ and $G_1 = (V_1, E_1)$ are *isomorphic* if there exists a one-to-one onto function $f : V_1 \to V_2$ that sends edges to edges in the sense that for all pairs of vertices u and v in V_1, we have that $u \to v$ is an edge in G_1 if and only if $f(u) \to f(v)$ is an edge in G_2.

2. Input Format

A shell will be given to perform graph input and output, and to call a function for finding an isomorphism if one exists. This frees you to concentrate on the isomorphism part. However, you should understand the input format so you can build test cases.

An input file starts with a single word, one of `directed` or `undirected`, that indicates whether the input graph is directed or undirected. This is followed by

(1) an integer n, the number of vertices in the graph,
(2) an integer m, the number of edges in the graph,
(3) m pairs of strings, the names of the vertices that are the initial and terminal endpoints of the m edges.

For example, the complete graph on 4 vertices a, b, c, d might be represented like this

```
undirected
4
6
a b  a c  a d
b c b d  c d
```

The string names of the vertices are replaced with integers in the range 0 to $n - 1$ upon input, and translated back to the original names for output.

A single run of the program will read two input files, each containing data for a single graph.

3. Program Requirement

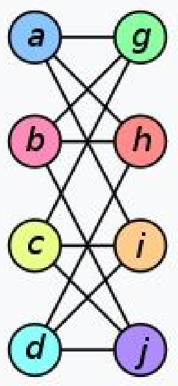
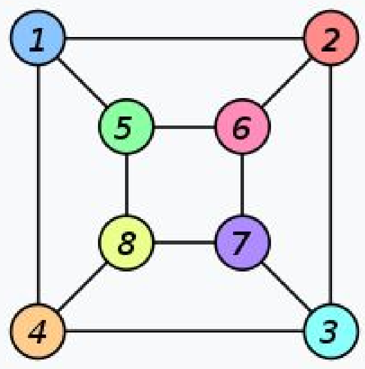
Your job will be to write a function

```
boolean
extend(LinkedList<Integer> p, boolean [ ][ ] m1, boolean [ ][ ] m2 )
```

which is passed an empty list p and two adjacency matrices for the two graphs being tested for isomorphism. The function returns **true** if an isomorphism is found and placed in p, and returns **false** otherwise.

4. Sample Run

Here is a sample run with graphs taken from the Wikipedia page on graph isomorphism.

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

```
This program looks for an isomorphism between two graphs.
Select the input file for the first graph.
a : [g, h, i]
b : [g, h, j]
c : [g, i, j]
d : [h, i, j]
g : [a, b, c]
h : [a, b, d]
i : [a, c, d]
j : [b, c, d]

Select the input file for the second graph.
1 : [2, 4, 5]
2 : [1, 3, 6]
3 : [2, 4, 7]
4 : [1, 3, 8]
5 : [1, 6, 8]
6 : [2, 5, 7]
7 : [3, 6, 8]
8 : [4, 5, 7]
```

```
Here is the isomorphism:

a -> 1
g -> 2
h -> 4
i -> 5
b -> 3
j -> 7
c -> 6
d -> 8
```

Notice that this run finds a different isomorphism than the one in the Wikipedia example.

The input graphs are provided on the K-drive as adjMatrix1.txt and adjMatrix2.txt.

You should make up additional graph pairs, both isomorphic and non isomorphic to test your program. To make up an isomorphic pair, create the first graph. To get a second graph isomorphic to the first, simply assign new names to the vertices, possibly in a different order.

To create a pair of graphs that are not isomorphic, simply create two different graphs. Chances are that the graphs are not isomorphic.

Here is another sample run for adjMatrix4.txt and adjMatrix3.txt

```
This program looks for an isomorphism between two graphs.
Select the input file for the first graph.
1 : [4, 2]
2 : [4]
3 : [7]
4 : [2]
6 : [3, 7]
7 : [3, 5]
8 : [1]

Select the input file for the second graph.
a : [b]
b : [c, d]
c : [d]
d : [c]
e : [f, g]
f : [g, h]
g : [f]

Here is the isomorphism:
1 -> b
4 -> c
2 -> d
3 -> g
7 -> f
6 -> e
5 -> h
8 -> a
```

5. Due Date

March 11 2020.