CISC-235
Assignment 4
March 21, 2018

In this assignment you will experiment with Breadth-First Search and Prim's MST algorithm. BFS has already been covered in class – Prim's algorithm is being covered this week.

Your work on hashing for the HOTNCU has given you the reputation as the new "go-to" person for tech problems in the Canadian entertainment community (which is ironic considering that all well-educated programmers know that "GOTO" is considered harmful).

HOTNCU is entering its budgeting phase, and they have recruited you to solve a particular problem for them. Like any proper Cinematic Universe the HOTNCU movies are intended to interconnect, with characters that appear in multiple films. The writers have been working out the possible cross-overs, to the point where they have identified between 1 and 8 possible links for each movie. Now the accountants have stepped in and worked out the cost of each proposed cross-over (in millions of dollars, of course).

The goal of course is to link all the movies together, directly or indirectly, at minimum total cost.

The HOTNCU management understand that the optimal solution to their problem can be found by applying Prim's algorithm, but they are hesitant to pay your consultant's fee for implementing Prim. In fact, they are willing to settle for a "good" solution if it can be found be a faster algorithm such as BFS, for which they already have an implementation.

You of course want to convince them that the MST found by Prim's algorithm will be significantly better than the spanning tree found by BFS.

HOTNCU is not willing to release the actual set of links but they have created a number of test cases. Your task is to run BFS and Prim on each case and report on how much better (if at all) the MST found by Prim is in comparison to the spanning tree found by BFS.

The test cases are all in a text file called “Test_Cases.txt”. Each case is defined by a sequence of lines:

first line:
Case# #ofMovies (ex. 1 32)

detail lines:
MovieID #ofLinks Link1ID weight Link2ID weight .... LinkxID weight
(ex. remnants 3 dactylic 17 bookbind 24 swirling 9)

the number of detail lines will exactly match the #ofMovies

There are 25 sample cases for each size of graph

Your study will consist of

- for each case
  - construct a representation of the graph
  - find a spanning tree using BFS, and
  - find a spanning tree using Prim’s MST algorithm, and
  - compute the percentage difference between the total weights of the two spanning trees

- report the average percentage difference as a function of graph size

- project the percentage difference for the full set of 2500 film projects

Now the details:

Part 1:

Implement the Breadth First Search algorithm, modifying it so that it randomly chooses a start vertex, and also returns the total of the weights of the edges that it selects. The key part of this modification can look something like this:

```
...
for each neighbour y of x:
     if y is not visited and y has not been added to Q:
          Q.append(y)
          mark y as "added to Q"
          total += weight(x,y)
```

There are still a number of design and implementation details for you to work out!

Test your implementation on the graph that has the following adjacency matrix. The first row and the first column contain the vertex identifiers (1, 2, 3, 4, 5, 6) and the rest of the matrix represents the costs of the edges. A value of 0 means the edge does not exist.

	1	**2**	**3**	**4**	**5**	**6**
1	0	15	0	7	10	0
2	15	0	9	11	0	9
3	0	9	0	0	12	7
4	7	11	0	0	8	14
5	10	0	12	8	0	8
6	0	9	7	14	8	0

You are not required to use an Adjacency Matrix to represent this or any other graph in this assignment. Please feel free to use Adjacency Lists if you wish, since they are more efficient for these algorithms – the matrix was just the easiest way to represent the graph on this page.

Part 2:

Implement Prim's Minimum Spanning Tree algorithm, so that it returns the total of the weights of the edges that it selects.

Test your implementation on the small graph given in Part 1.

Part 3:

Put the pieces together to conduct the following experiment:

for each of the test cases:

```
generate the graph to represent the test case
use BFS to find a spanning tree (let its total weight be B)
use Prim to find a spanning tree (let its total weight be P)
compute Diff = (B/P – 1) * 100      # Diff is the percentage by which
                                    #  B is larger than P
```

Compute the average of the values of Diff for each of the given graph sizes

Report the average value of Diff for each of the given graph sizes

Estimate the expected value of Diff if the graph has 2500 vertices.

Appendix (for interest only):

You may wonder how the test cases were created. The construction of random graphs is a huge topic and many approaches have been developed – including one that involved a toothbrush! For this assignment I used the following very simple method:

Let the set of vertices be {1, 2, ..., n}

Let Available = {1}

```
for i = 2 .. n:
    x =  a small random integer        # x is the number of neighbours
                                       # that vertex i starts with
    let S be a randomly selected sample of x vertices from Available
    for each s in S:
        w = a randomly chosen weight for the edge
        add an edge between vertex i and vertex s, with weight w
        if s now has 8 neighbours, remove s from Available
    if vertex i has fewer than 8 neighbours, add i to Available
```

Note that in rare instances this may fail – if we happen to raise the degree of all vertices in Available to 8, Available will become empty and no further vertices can be added. In this case we simply start again. This never happened during the creation of the test cases.

Logistics:

You may complete the programming part of this assignment in Python, Java, C or C++.

You must submit your source code, properly documented according to standards established in CISC-121 and CISC-124, and taking into consideration any feedback you received on previous Assignments. You must also submit a PDF file summarizing the results of your experiments and containing your conclusions. Both files must contain your name and student number, and must contain the following statement: "I confirm that this submission is my own work and is consistent with the Queen's regulations on Academic Integrity."

You are required to work individually on this assignment. You may discuss the problem in general terms with others and brainstorm ideas, but you may not share code. This includes letting others read your code or your written conclusions. The course TAs will be available to advise and assist you regarding this assignment.

The due date for this assignment is 11:59 PM, April 9, 2018.