CISC-235
Assignment 4
March 26, 2017

In this assignment you will experiment with Breadth-First Search and Prim's MST algorithm. BFS has already been covered in class and in the notes – Prim's algorithm will be covered on Monday March 27.

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

CISC's parent organization, Knowing Information Needs Global Secrets (KINGS) has asked you to solve a problem for them. They run a large number of networks of agents all across the world. Each network has between 20 and 60 agents. For each pair of agents in any given network, there is a known cost of maintaining the communication link between those agents. KINGS wants to minimize the total communication costs within each of their networks.

Their existing software uses Breadth First Search to find a spanning tree of each network – they use the edges of these spanning trees as their communication links. They are considering switching over to using Minimum Spanning Trees, but they know that Prim's MST algorithm is slower than the BFS algorithm. Before they commit to a slower algorithm, they want data on how much lower the costs of the Minimum Spanning Trees will be than the costs of the Breadth First Search trees. Your task is to gather that information.

Your study will consist of

- generating a large number of random graphs on numbers of vertices ranging from 20 to 60
- for each graph
    - finding a spanning tree using BFS, and
    - finding a spanning tree using Prim's MST algorithm, and
    - computing the percentage difference between the total weights of the two spanning trees
- reporting the average percentage difference for different sizes of graph

Now the details:

Part 1:

Implement an algorithm that will create a random connected graph with a specified number of vertices, and random weights on all the edges, such that the edge-weights all come from a specified range of integers. Your algorithm should return either an adjacency matrix for the graph, or the adjacency lists for the graph – your choice.

Please see the end of this assignment for a very simple algorithm for generating random graphs.

Part 2:

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

```
for each neighbour y of x:
     if y is not visited:
          mark y visited
          Q.append(y)
          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, since they are more efficient for both of these algorithms – the matrix was just the easiest way to represent the graph on this page.

Part 3:

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 same graph as was specified in Part 2.

Part 4:

Put all the pieces together to conduct the following experiment:

```
for n = 20, 30, 40, 50, 60:
    repeat k times:        # you need to decide how large k should be!
        generate a random graph with n vertices
        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 this value of n
```

Report the average value of Diff for each value of n.

Can you draw any conclusion from your data regarding the relative performance of BST and Prim with respect to the total weight of the spanning trees they construct?

Constructing Random Graphs

The construction of random graphs is a huge topic and many approaches have been developed – including one that involved a toothbrush! For the purposes of this assignment, I suggest the following very simple method:

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

```
for i = 2 .. n:
    x = randint(1,i-1)      # x is a random integer in the range [1 .. i-1]
    let S be a randomly selected sample of x values from the set {1, ..., i-1}
    for each s in S:
        w = random(10,100)
        add an edge between vertex i and vertex s, with weight w
```

Note that this works equally well with adjacency matrices and with adjacency lists.

You should think about why this is guaranteed to produce a connected graph.

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, 2017.