Australian National University
Research School of Computer Science

COMP3600/COMP6466 in 2018 – Assignment Two

Due: 23:55 Friday, October 26
Late Penalty: 5% per working day, the cut off date October 31

Submit your work electronically through Wattle. The total mark is 50. *Note that the mark you received from each question is proportional to the quality of the solution you provided.*

Question 1 (**25 points**).

A complete graph is an undirected graph $G = (V, E)$ where there is an edge between any pair of nodes. Given a unit square in a 2D plane with its four corner coordinates are: $(0, 0)$ - the most left bottom coordinate, $(0, 1)$ - the most left top coordinate, $(1, 1)$ - the most top right coordinate, and $(1, 0)$ the most right bottom coordinate. We randomly choose n nodes within this square. Let node v_i have a coordinate $(x_i,\ y_i)$, where the values of both $x_i \geq 0$ and $y_i \geq 0$ are randomly drawn from an interval [0, 1], where x_i and y_i are random real numbers uniformly distributed between 0 and 1, $1 \leq i \leq n$.

Let $K(V) = (V, E)$ be a complete graph with $V = \{v_i \mid 1 \leq i \leq n\}$, the weight (or cost) assigned to each edge $(v_i, v_j) \in E$ is the Euclidean distance between nodes v_i and v_j, i.e., $w(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. We term $K(V)$ in this square area as a randomly generated complete graph. Denote by $L(n)$ the weighted sum of edges of a minimum spanning tree (MST) of a randomly generated complete graph $K(V)$.

Write code using one language such as C, C++, Java, or Python program that does the following tasks.

(i) Calculate the average weighted sum $\overline{L}(n)$ of MSTs of p randomly generated complete graphs with $|V| = n$, which is the sum of p MST costs divided by p, where p randomly generated complete graphs with each having n nodes need to be constructed and $p \geq 50$, using Kruskal's algorithms when $n = 100$, 500, 1,000, and 5,000, respectively **(10 points)**.

(ii) Observe the changing trend of the value of $\overline{L}(n)$ with the growth of n, and explain why this happens. **(2 points)**

(iii) Generate *a randomly connected graph* G of n nodes in a unit square as follows: choose n nodes randomly in the square where each node is represented by a coordinate (x_i, y_i) with $0 \leq x_i, y_i \leq 1$, then choose two nodes randomly (e.g., node u is represented by coordinate (x_i, y_i) and node v is represented by coordinate (x_j, y_j)

with $i \neq j$), add an edge (u, v) into G, and assign the edge (u, v) a weight, which is the Euclidean distance between them. This edge addition procedure continues until the graph becomes connected.

- Give the actual average running times of both Kruskal's and Prim's algorithms for finding MSTs in q randomly generated connected graphs with $q = 50$, when $n = 100, 500, 1,000$, and 5,000, respectively. **(10 points)**
- Observe the running time trends of both algorithms with the growth of the value of n in Part , and explain why there is such a difference between the running times of these two algorithms. Notice that you should NOT include the graph generation time in your running time calculation.**(3 points)**

Your program should have the following properties.

1. Do not read any input. Write one line of output for each n, and at most 5 extra lines of explanatory output.
2. Store the graph as a symmetric matrix (a complete graph) or linked list representations (a connected graph)
3. The minimum priority queue will be employed in the implementation of Prim's algorithm, and the disjoint set data structures will be adopted in the implementation of Kruskal's algorithm and testing whether a graph is connected. You can implement these two data structures by yourself, or make use of them from your program language library if they do exist.
4. To measure the running time of an algorithm, refer to a timing procedure in the exercise of the 2nd lab.

What to submit:

Upload your file named as `mst.c` or `mst.java` containing the program to Wattle.

You also need to submit a report in *the PDF format* that contains the source code and its outputs, and the answers to part (i), part (ii), and part (iii) of the question.

Question 2 (**15 points for COMP3600 and 10 points for COMP6466**).

Q2.(a) You are given a string of n characters $c_1c_2 \ldots c_n$, which is a corrupted text document, in which all punctuation have vanished. You need to reconstruct the document, using the following dictionary that is in the form of Boolean function $b(\cdot)$:

For any string s

$$b(s) = \begin{cases} true & \text{if } s \text{ is a valid word,} \\ false & \text{otherwise.} \end{cases}$$

- Devise an $O(n^2)$ time dynamic programming algorithm to determine whether a string $c_1c_2 \ldots c_n$ can be reconstructed as a sequence of valid words.
(**8 points for COMP3600 and 5 points for COMP6466**)
- Analyze that the running time of your algorithm indeed is $O(n^2)$. (**2 points**)

Q2.(b) Given an undirected, connected, weighted graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, assume that each edge is assigned a positive weight. There are only K distinct weights on its edges and K is a positive integer constant, i.e., for each edge $e \in E$, its weight is one of the values in $\{w_1, w_2, w_3, \ldots, w_K\}$, where $w_j > 0$ with $1 \leq j \leq K$. Devise an $O(n + m)$ time algorithm to find a minimum spanning tree in G.
(**5 points for COMP3600 and 3 points for COMP6466**)

Question 3 (**10 points for COMP3600 and 8 points for COMP6466**)

Given a social network that is represented by a directed graph $G = (V, E)$ where V is the set of people and E is the set of people trust relationships, there is one directed edge $(u, v) \in E$ from node u to node v if person u trusts person v with a certain degree trust, and this trust is measured by a value $p_{u,v}$ in the range between 0 to 1. The larger the trust value $p_{u,v}$ is, the higher the trustiness of u on v is. Notice that this trust relationship is not necessarily symmetric. In other words, if person $a \in V$ fully trusts person $b \in V$, e.g., $p_{a,b} = 1$, person b may not have any trust on person a, e.g., $p_{b,a} = 0$.

- Devise an efficient algorithm for finding a most trustable path P in G from a person $s \in V$ to another person $t \in V$ if path P does exist such that the trust value of P is maximized, where the trust value of a path Q consisting of nodes $v_1, v_2, \ldots, v_k$ is $\Pi_{i=1}^{k-1} p_{v_i,v_{i+1}}$. You can make use of either natural language or pseudo-code to describe your algorithm with brief explanations.
(**8 points for COMP3600 and 6 points for COMP646**)
- Analyze the running time of your algorithm. (**2 points**)

Question 4 (**7 points for COMP6466 only**)

Given a communication network represented by an undirected, weighted graph $G(V, E)$ and a non-negative integer delay bound $L > 0$, assume that for each edge $e \in E$, the communication cost and communication delay on it are $c(e)$ and $d(e)$, respectively, where $d(e)$ is a non-negative integer, a delay-constrained shortest path problem in G from a source node $s \in V$ to a destination node $t \in V$ is to find a simple path (each node and each edge appear in the path at most once) such that the total cost of the edges in path P is minimized (i.e., the value of $\sum_{e \in P} c(e)$ is minimized), while the accumulative delay on all edges in P is upper bounded by L (i.e., $\sum_{e \in P} d(e) \leq L$), assuming that the path exists.

- (a) Devise an efficient algorithm in terms of running time for this delay-constrained shortest path problem. (**3 points**)
- (b) Analyze the time complexity of your algorithm. (**2 points**)
- (c) For each edge $e \in E$ in G, if its delay $d(e)$ is a real number, instead of a non-negative integer, is your proposed algorithm in part (a) still applicable to the problem under this setting? If not, explain how to modify the proposed algorithm for this case briefly? (**2 points**)