COMP 3711 – Design and Analysis of Algorithms
2017 Fall Semester – Written Assignment # 4
Distributed: November 8, 2017 – Due: November 24, 2017

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address

Some Notes:

- Please write clearly and briefly.
- Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.
  In particular ***don't forget to acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- Please make a *copy* of your assignment before submitting it. If we can't find your answers, we will ask you to resubmit the copy.
- The default base for logarithms will be 2, i.e., $\log n$ will mean $\log_2 n$. If another base is intended, it will be explicitly stated, e.g., $\log_3 n$.
- Each problem is worth 25 points.
- As in the previous assignment, you must submit both a hardcopy (A4 size) and a PDF softcopy. The hardcopyshould be submitted to the COMP3711 assignment box and the softcopy via the CASS system. The PDF can be generated by Latex, from Word or a scan of a (legible) handwritten solution.
- For any question, please contact the TA in charge of this assignment at ckoutras@connect.ust.hk.

Problem 1: Minimum Spanning Tree

Let G be a connected undirected graph with weights on the edges. Assume that all the edge weights are distinct. Let e_i be the edge with the i-th smallest weight. Does the MST have to contain e_1? How about e_2 and e_3? If yes, give a proof; otherwise, give a counter example. You must prove your results from first principles, i.e., you cannot rely on the cut lemma or the correctness of Prim's or Kruskal's algorithm.

Problem 2: Bottleneck Spanning Tree

A Bottleneck Spanning Tree of an undirected graph $G(V, E, w)$ with weights on the edges, is a spanning tree of G, where the maximum edge weight is the minimum among all the spanning trees of G. Thus, the Bottleneck Spanning Tree T minimizes the bottleneck cost $c(T) = \max_{e \in T}\{w(e)\}$.

1. Show that every MST of G is a Bottleneck Spanning Tree.
2. Write a linear time algorithm where, given a graph $G(V, E, w)$ and an integer B, it decides whehter G has a spanning tree with bottleneck cost less or equal to B.
3. Write a linear time algorithm where, given a graph $G(V, E, w)$, it computes a Bottleneck Spanning Tree of G.

Problem 3: Road Network

Suppose a road network in the form of a graph $G(V, E, l)$, which connects a set of cities V. We assume that the network is directed and that every road $(u, v) \in E$ has a non-negative length $l(u, v)$. A new road is about to be constructed, so there is a list E' containing pairs of cities that it could connect. Every pair $(u, v) \in E'$ has a corresponding length $l'(u, v)$. We want to choose the pair of cities that succeeds the maximum reduction in distance between two cities $s, t \in V$. Write an efficient algorithm for this problem. Explain thoroughly the correctness and complexity of your algorithm.

Problem 4: Escape Problem

An $n \times n$ grid is an undirected graph consisting of n rows and n columns of vertices, as shown in the figure below. We denote the vertex in the i-th row and the j-th column by (i, j). All vertices in a grid have exactly four neighbors, except for the boundary vertices, which are the points (i, j) for which $i = 1$, $i = n$, $j = 1$, or $j = n$.

Given $m \leq n^2$ starting points $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$ in the grid, the *escape problem* is to determine whether or not there are m vertex-disjoint paths, i.e., the paths don't cross one another, from the starting points to

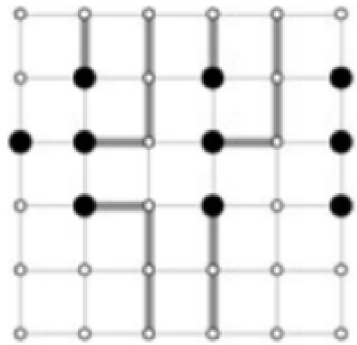

Figure 1: Grid for the escape problem. Starting points are black, and other grid vertices are white.

any m different points on the boundary. For example, the grid in figure 1 has an escape.

(1) The problem can be seen as a max-flow problem. Consider a flow network in which vertices, as well as edges, have capacities. That is, the total positive flow entering any given vertex is subject to a capacity constraint. Show that determining the maximum flow in a network with edge and vertex capacities can be reduced to an ordinary maximum-flow problem on a flow network of comparable size. More precisely, you need to convert a network $G = (V, E)$ with capacities on both vertices and edges, to another network $G' = (V', E')$ with capacities on the edges only, so that the maximum flows on the two networks are the same, and the new network you construct have $V' = O(V)$ vertices and $E' = O(E)$ edges. You can assume that the network is connected.

(2) Describe an efficient algorithm to solve the escape problem, and analyze its running time.