COMP 3711H – Honors Design and Analysis of Algorithms
2014 Fall Semester – Written Assignment # 3
Distributed: November 3, 2014 – Due: November 16, 2014

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

Information:

- Please write clearly and briefly.
- Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page. Don't forget to *acknowledge* individuals who assisted you, or sources where you found solutions.
- 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.
- Your solution should be submitted as a PDF. This can be generated by Latex, from Word or a scan of a (legible) handwritten solution, etc..
- Your solution should be submitted via the CASS system by 11:59PM on November 16, 2014. The class web page has reminders on how to use CASS.

Problem 1: [20 points]

The *Fan Graph* F_n of Figure (A) has $n+1$ vertices $v_0, v_1, \ldots v_n$ with node v_0 connected to all of the other nodes and nodes $v_1, \ldots, v_n$ forming a straight line, each vertex connected to its left and right neighbors on the line (if the neighbors exist),

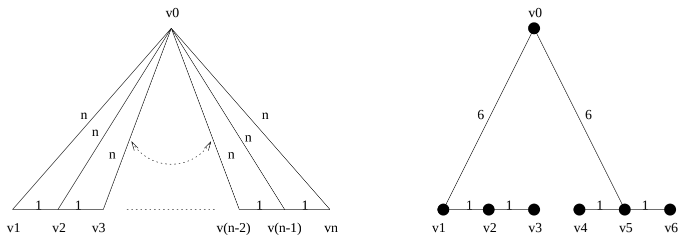
The graph edges will have the following costs.

$$c(v_0, v_i) = n, \qquad i = 1, 2, \ldots n,$$

and

$$c(v_i, v_{i+1}) = 1, \qquad i = 1, 2, \ldots, n-1$$

(edges with no defined costs are not in the graph).



(A) **(B)**

In the problems below solve for all $n \geq 2$.

(a) Describe a Minimum Spanning Tree of F_n. To describe the tree draw a picture and list its edges. What is the cost of the Minimum Spanning Tree as a function of n?

(b) Describe a Shortest Path Tree of F_n with source node v_0. To describe the tree draw a picture and list its edges. What is the cost of the Shortest Path Tree as a function of n?

(c) Describe a Shortest Path Tree of F_n with source node v_2. To describe the tree draw a picture and list its edges. What is the cost of the Shortest Path Tree as a function of n?

Recall that the cost of a tree is the sum of the costs of the weights of its edges. For example, the tree in Figure (B), which is neither a minimum spanning tree or a shortest path tree, has cost 16.

Problem 2: [20 points]

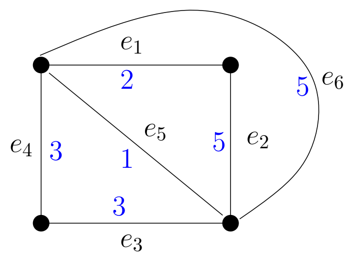
We saw in class that if up-trees with union-by-height are used to implement UNIONs and FINDs then a sequence of m UNION/FIND operations on a universe of n items will take at most $O(m \log n)$ time. In this problem you will show that that bound is tight.

Assume that you start with n items each in their own separate tree. Describe a sequence of $n-1$ UNIONs and n FINDs that requires $\Omega(m \log n)$ time where $m = 2n-1$. Justify your answer.

Note: You have complete freedom to choose the order in which the operations are performed.

Problem 3: [20 points]

Recall that Kruskal's algorithm starts by sorting the edges by non-decreasing cost. If some edges have the same cost, that's a *ties,* and ties can be broken arbitrarily. Since the spanning tree produced depends upon the initial order of the edges, different tie-breaking *may* result in different spanning trees being produced. As an example, consider the graph below: with $c(e_5) = 1$,



$c(e_1) = 2$, $c(e_3) = c(e_4) = 3$ and $c(e_2) = c(e_6) = 5$. Note that there are 4 ways of sorting the edges:

$e_5, e_1, e_3, e_4, e_2, e_6 \quad e_5, e_1, e_3, e_4, e_6, e_2, \quad e_5, e_1, e_4, e_3, e_2, e_6, \quad e_5, e_1, e_4, e_3, e_6, e_2$

Note that for the first two orders Kruskal's algorithm produces the MST $\{e_5, e_1, e_3\}$; for the last two orders Kruskal's algorithm produces $\{e_5, e_1, e_4\}$.

These are the ONLY two possible MSTs for this graph (convince yourself).

For this problem *you must prove that for each minimum spanning tree T of G that exists, there is a way to sort the edges of G (with appropriate tie-breaking) so that Kruskal's algorithm returns T.*

Note: This immediately implies that if all edges have different weights the graph has a unique minimum spanning tree (since there are no ties to break).

Problem 4: [40 points]

Let $G = (V, E)$ be an undirected weighted graph (each edge (u, v) is given weight $w(u, v)$ as part of the input) with no negative edge costs. The *bottleneck* value of a path

$$(u_0, u_1), (u_1, u_2), \ldots, (u_{n-2}, u_{n-1}), (u_{n-1}, u_n)$$

is $\min_{1 \le i \le n} w(u_{i-1}, u_i)$.

For intuition, think of the edges as being water pipes and the weights as the maximum-flow that can pass through a pipe every hour. The *maximum* amount that can flow through a path in an hour is the value of the *minimum-weight* edge on the path. This is the bottleneck value of the path.

Design an algorithm that, given s and t, finds a path from s to t with *maximum* bottleneck value among all $s - t$ paths.

Argue the correctness of your algorithm and analyze its running time.

Hint: There are multiple ways of solving this problem. One of them requires using a Max-Heap. A Max-Heap is like a Min-heap except that it allows extracting the largest item rather than the smallest one. It can be implemented almost exactly like a Min-heap with the same running times.