COMP 3711H – Honors Design and Analysis of Algorithms
2016 Fall Semester – Written Assignment # 4
Distributed: November 18, 2016– Due: December 2 2016

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.
  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., $\log_3 n$.
- As in the previous assignment, you must submit both a hardcopy and a PDF softcopy. The hardcopyshould be submitted to the COMP3711H 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.

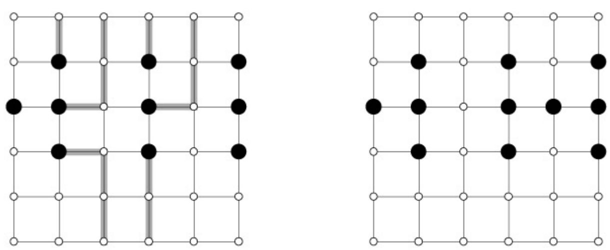
Problem 1: [20 pts] Let G be a connected undirected graph with distinct weights on the edges. Given an edge e of G, can you decide whether e belongs to the MST in $O(E)$ time? If you compute the MST and then check whether e belongs to the MST, this would take $O(E \log V)$ time. To design a faster algorithm, you will need the following theorem:

Edge $e = (u, v)$ does not belong to the MST if and only if there is a path from u to v in G that consists of only edges cheaper than e.

Prove this theorem (you can use any statement or algorithm we taught in class or the tutorial as long as you reference it). Then give the $O(E)$-time algorithm.

Hint: One approach would be to consder running Kruskal's algorithm and its implications for e.

Problem 2: [30 pts] 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 from the starting points to any m different points on the boundary. For example, the grid in (a) has an escape, but the grid in (b) does not.



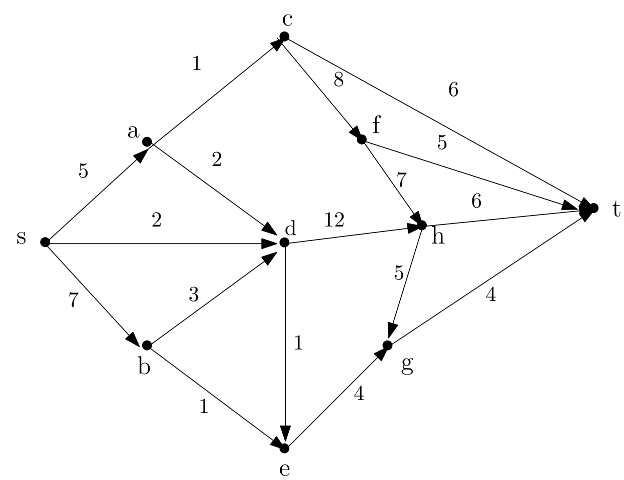
(a) (b)

(1) 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.

Problem 3: [20 pts] Start with a flow that has $f(e) = 0$ for every edge and run the Ford-Fulkerson Max Flow Algorithm on the graph below to find a max-flow from s to t.



(a) Show every step of the algorithm. That is, for every step show the current flow, its associated residual graph and the augmenting path you choose.

(b) Show your final flow and provide a cut with capacity equal to that of the flow.

Problem 4: [20 pts] *Arbitrage* is the use of discrepancies in currency-exchange rates to make a profit. For example, there may be a small window of time in which 1 U.S. dollar buys 0.75 British pounds, 1 British pound buys 2 Australian dollars and 1 Australian dollar buys 0.70 U.S. dollars. Then, a smart trader can trade one U.S. dollar and end up with $0.75 \times 2 \times 0.7 = 1.05$ U.S. dollars, a profit of 5% (note that this assumes there are no trading costs).

Suppose that there are n currencies $c_1, \ldots, c_n$ and an $n \times n$ table R of exchange rates such that one unit of currency c_i buys $R[i, j]$ units of currency c_j. The *value* of a series of exchanges $c_{i_1}, c_{i_2}, \ldots, c_{i_t}$ is

$$R[i_1, i_2] \cdot R[i_2, i_3] \cdots R[i_{t-1}, i_t].$$

which is the number of units of currency c_{i_t} that result when starting with one unit of currency c_{i_1}, exchanging it into currency c_{i_2}, exchanging what results into c_{i_3}, etc. until ending by exchanging units of $c_{i_{t-1}}$ in currency c_{i_t}.

A series of exchanges yields a profit if it both starts and ends in the same currency and the value of the series is greater than 1.

Devise and analyze a dynamic programming algorithm that, given the $R[\ ]$ table, determines whether or not it is possible to make a profit by trading currencies. You must prove that your algorithm is correct and also give the running time of the algorithm using $O()$ notation.

Problem 5: [10 pts] (From CLRS) Show that a maximum flow in a network $G = (V, E)$ can always be found by a sequence of at most $|E|$ augmenting paths. (Hint: Determine the paths AFTER knowing the maximum flow).