COMP 3711 Design and Analysis of Algorithms
Fall 2015 Assignment 4

1. 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; if no, 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.

2. 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 that consists of only edges cheaper than e.

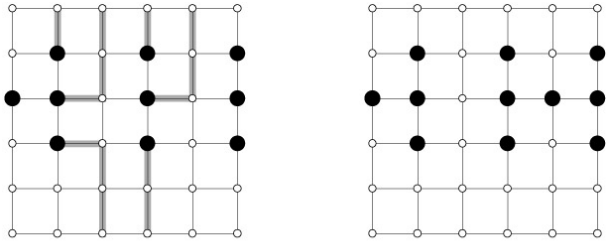
   Prove this theorem (you can use the cut lemma and the cycle property in the tutorial). Then give the $O(E)$-time algorithm.

3. The longest path problem introduced in the lecture is somewhat unnatural to model jobs and the dependencies. In a more natural structure, vertices would represent jobs and edges would represent dependencies; that is, edge (u, v) would indicate that job u must be performed before job v. We would then assign weights to vertices, not edges. Let $w(v)$ be the weight of vertex u. Give an algorithm to find the longest path from a source vertex s to a destination vertex t, where the weight of a path is the sum of the weights of its vertices.

4. 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.

   (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.



(a) (b)