CS 473: Undergraduate Algorithms, Spring 2009
Homework 6

Written solutions due Tuesday, March 17, 2009 at 11:59:59pm.

1. Let G be an undirected graph with n nodes. Suppose that G contains two nodes s and t, such that every path from s to t contains more than $n/2$ edges.

   (a) Prove that G must contain a vertex v that lies on *every* path from s to t.

   (b) Describe an algorithm that finds such a vertex v in $O(V+E)$ time.

2. Suppose you are given a graph G with weighted edges and a minimum spanning tree T of G.

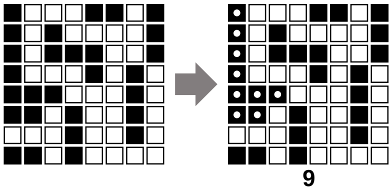
   (a) Describe an algorithm to update the minimum spanning tree when the weight of a single edge e is decreased.

   (b) Describe an algorithm to update the minimum spanning tree when the weight of a single edge e is increased.

   In both cases, the input to your algorithm is the edge e and its new weight; your algorithms should modify T so that it is still a minimum spanning tree. *[Hint: Consider the cases $e \in T$ and $e \notin T$ separately.]*

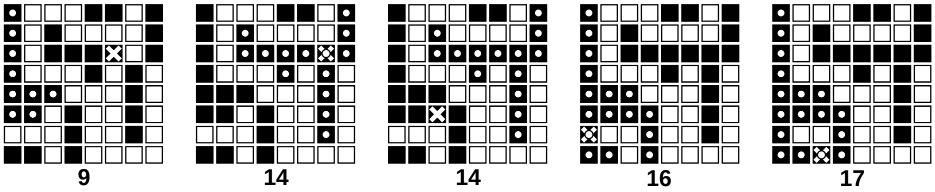
3. (a) Describe and analyze an algorithm to compute the size of the largest connected component of black pixels in an $n \times n$ bitmap $B[1..n, 1..n]$.

   For example, given the bitmap below as input, your algorithm should return the number 9, because the largest conected black component (marked with white dots on the right) contains nine pixels.



   (b) Design and analyze an algorithm Blacken(i, j) that colors the pixel $B[i, j]$ black and returns the size of the largest black component in the bitmap. For full credit, the *amortized* running time of your algorithm (starting with an all-white bitmap) must be as small as possible.

   For example, at each step in the sequence below, we blacken the pixel marked with an X. The largest black component is marked with white dots; the number underneath shows the correct output of the Blacken algorithm.



   (c) What is the *worst-case* running time of your Blacken algorithm?