CS 4349.400 Homework 9

Due Wednesday December 5th, ***by 1:15pm***, in class

Please answer each of the following questions.

1. Suppose we are given an n-digit integer X. Repeatedly remove one digit from either end of X (your choice) until no digits are left. The *square-depth* of X is the maximum number of perfect squares that you can see during this process. For example, the number 32492 has square-depth 3, by the following sequence of removals:

$$32492 \to \underline{\mathbf{3249}}\cancel{2} \to \underline{\mathbf{324}}\cancel{9} \to \cancel{3}24 \to \cancel{2}\underline{\mathbf{4}} \to \cancel{4}.$$

Suppose we are given an integer X, represented as an array $X[1 \,..\, n]$ of n decimal digits. Further suppose we have access to a subroutine IsSquare that determines whether a given k-digit number (represented by an array of digits) is a perfect square ***in $O(k^2)$ time***.

   (a) For any i, j such that $1 \le i \le n$ and $1 \le j \le n$, let $SquareDepth(i, j)$ be the square-depth of $X[i \,..\, j]$ if $i \le j$ and let it be 0 if $j < i$. Give a ***recursive definition*** of $SquareDepth(i, j)$ that can be used for a dynamic programming algorithm. Don't forget the base case(s)!

   (b) In what kind of ***memoization data structure*** should we store the solutions to all subproblems $SquareDepth(i, j)$?

   (c) What is a good ***evaluation order*** for solving the subproblems so each subproblem is solved after the ones it is dependent upon?

   (d) What will be the final ***space*** and ***time*** complexity of the dynamic programming algorithm? Don't forget it takes $O(k^2)$ time to run IsSquare on a k-digit number.

   (e) Write the iterative algorithm that computes the square-depth of X.

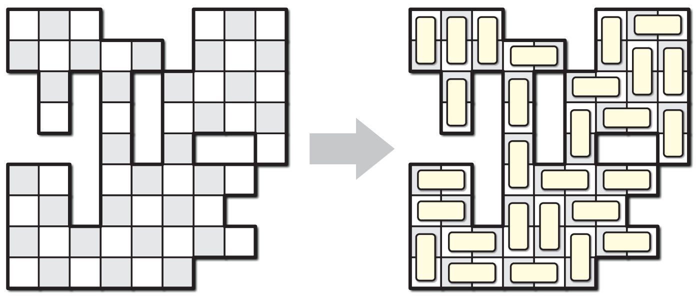
2. Most classical minimum spanning tree algorithms use the notions of "safe" and "useless" edges described in lecture and Jeff Erickson's lecture notes, but there is an alternative formulation. Let G be a weighted undirected graph where the edge weights are distinct. We say that an edge e is ***dangerous*** if it is the longest edge in some cycle in G and ***useful*** if it does not lie in any cycle in G.

   (a) Prove that the minimum spanning tree of G contains every useful edge. *[Hint: Spanning trees are connected subgraphs containing every vertex.]*

   (b) Prove that the minimum spanning tree of G does not contain any dangerous edge. *[Hint: Give an exchange argument. How can we decrease the weight of a spanning tree containing a dangerous edge?]*

(Kruskal described an alternative minimum spanning tree algorithm based on these concepts in the same 1956 paper where he proposed "Kruskal's algorithm": Examine the edges of G in *decreasing* order; if an edge is dangerous, remove it from G. This is just some trivia; you don't need to do anything else for this problem.)

3. Suppose you are given an $n \times n$ checkerboard with some of the squares deleted. You have a large set of dominos, each just the right size to cover two neighboring squares of the checkerboard. *Tiling* the board with dominos means placing dominos so each domino covers exactly two squares and each undeleted square is covered by exactly one domino.



A checkerboard tiled with dominos.

We will design an algorithm to determine whether one can tile a board with dominos by reducing to maximum matching in a bipartite graph.

(a) A bipartite graph G has two sets of vertices U and W. What should we use for each of the sets U and W? *[Hint: Our goal is to cover all the undeleted squares of the checkerboard. You may want to check what the squares of a checkerboard look like.]*

(b) What should we use for our set of edges? *[Hint: A matching is a subset of edges that is incident to each vertex at most once.]*

(c) What should the size of the maximum matching be if we can tile the given checkerboard?

(d) ***In terms of n***, how long does it take to determine if we can tile the checkerboard using this reduction?

4. **(Extra credit worth 1 homework problem. No "I don't know" credit or credit for citations is available.)** The problem 11Color is defined as follows: Given an undirected graph G, determine whether we can color each vertex with one of eleven colors so that every edge touches two different colors.

Our goal is to prove that 11Color is NP-complete.

(a) Give a short argument that 11Color is in NP.

(b) To prove 11Color is NP-hard, we should be able to reduce *from* known NP-hard problem 3Color *to* 11Color. Give an input graph $G = (V, E)$ for the 3Color problem, describe how to construct a graph $G' = (V', E')$ so that G' has a proper 11-coloring if and only if G has a proper 3-coloring. *[Hint: Add eight additional vertices to G. What edges should you add?]*

(c) Prove that G' is 11-colorable if G is 3 colorable.

(d) Prove that G is 3-colorable if G' is 11-colorable.

(e) How long does it take to construct G' in terms of V and E? NP-hardness reductions must run in polynomial time.