Algorithms in the Wild: Making Change

Question 1. *Our recent focus in class has been* greedy *algorithms. Here we will examine a very practical problem that affords a greedy solution in some cases, but doesn't in others. Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.*

(a) *Describe a greedy algorithm to make change for n cents using US quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.*

(b) *Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.*

Some Problems from Jeff Erickson

Question 2. *A* feedback edge set *of a graph G is a subset F of the edges such that every cycle in G contains at least one edge in F. In other words, removing every edge in F makes the graph G acyclic. Describe and analyze a fast algorithm to compute the minimum weight feedback edge set of a given edge-weighted graph.*

Question 3. *Suppose you are given a graph G with weighted edges and a minimum spanning tree T of G.*

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

(b) *Describe and analyze 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 $e \in T$ and $e \notin T$ separately.

Question 4. *An Euler tour of a graph G is a cycle through G that traverses every edge of G exactly once.*

(a) *Prove that a connected graph G has an Euler tour if and only if every vertex has even degree.*

(b) *Describe and analyze a linear time algorithm to compute an Euler tour in a given graph, or correctly report that no such graph exists.*

Question 5 (checkers)**.** *Draughts/checkers is a game played on an $m \times m$ grid of squares, alternately colored light and dark. The game is usually played on an 8×8 or 10×10 board, but the rules easily generalize to any board size. Each dark square is occupied by at most one game piece (called a checker in the U.S.), which is either black or white; light squares are always empty. One player (White) moves the white pieces; the other (Black) moves the black pieces.*

Consider the following simple version of the game, essentially American checkers or British draughts, but where every piece is a king. Pieces can be moved in any of the four diagonal directions, either one or two steps at a time. On each turn, a player either moves one of her pieces one step diagonally into an empty square, or makes a series of jumps with one of her checkers. In a single jump, a piece moves to an empty square two steps away in any diagonal direction, but only if the intermediate square is occupied by a piece of the opposite color; this enemy piece is captured and immediately removed from the board. Multiple jumps are allowed in a single turn as long as they are made by the same piece. A player wins if her opponent has no pieces left on the board. See Figure 1 for an example.

Describe an algorithm that correctly determines whether White can capture every black piece, thereby winning the game, in a single turn. The input consists of the width of the board (m), a list of positions of white pieces, and a list of positions of black pieces. For full credit, your algorithm should run in $O(n)$ time, where n is the total number of pieces.

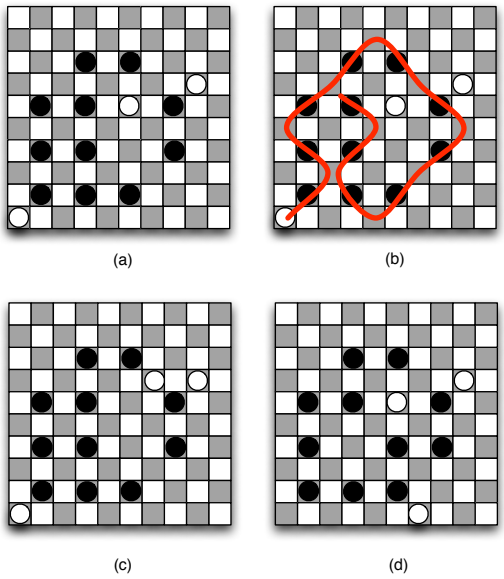

Figure 1: (a) White can win in one turn using the jumps given in (b). Both (c) and (d) have no one-turn wins.

Extra Credit

Question 6 (maze generation)**.** *One can use MST algorithms to generate mazes. There's a nice Wikipedia entry on the topic:*

`http://en.wikipedia.org/wiki/Maze_generation_algorithm`

Read the sections about randomized Kruskal's and randomized Prim's algorithms and implement a maze generation program using one of the algorithms. Your program should take three parameters, an integer width, an integer height, and a filename and output a maze in `PNG` *format. For example*

```
genmaze 30 40 mymaze.png
```

should generate a 30×40 *maze in the file* `mymaze.png`*. Hand in a sample maze generated from your program and turn in your source code using* `turnin -c 256`

Question 7 (this question is very hard)**.** *In Question 2 we considered feedback edge sets (FES). Similarly, a* feedback vertex set *(FVS) in an undirected graph* $G = (V, E)$ *is a set* $F \subseteq V$ *of vertices such that every cycle in the graph* G *contains some vertex in* F*. The* minimum feedback vertex set *problem is as follows: Given an undirected graph* $G = (V, E)$ *with a non-negative cost function on its vertices,* $c : V \to \mathbb{R}^+$*, find a minimum cost subset of vertices that is an FVS in* G*. While FES is easy to solve in undirected graphs, FVS appears to be difficult in undirected graphs. In fact, there is no known polynomial time algorithm to solve the problem. However, it is possible to approximate the minimum FSV in polynomial time. Develop and analyze an algorithm that finds a FSV that is always within a factor of 3 of the optimal FSV.*