
- ***This homework is optional.*** If you choose to submit solutions, we will provide feedback. Choosing to submit or not submit this homework will have no effect on your overall homework grade.

---

1. After a grueling algorithms midterm, you decide to take the bus home. Since you planned ahead, you have a schedule that lists the times and locations of every stop of every bus in Champaign-Urbana. Unfortunately, there isn't a single bus that visits both your exam building and your home; you must transfer between bus lines at least once.

   Describe and analyze an algorithm to determine the sequence of bus rides that will get you home as early as possible, assuming there are b different bus lines, and each bus stops n times per day. Your goal is to minimize your *arrival time*, not the time you actually spend traveling. Assume that the buses run exactly on schedule, that you have an accurate watch, and that you are too tired to walk between bus stops.

2. Describe an algorithm to compute the minimum spanning tree of an n-vertex *planar* graph in $O(n)$ time. *[Hint: Contracting an edge in a planar graph yields another planar graph. Any planar graph with n vertices has at most $3n - 6$ edges.]*

3. Negative edges cause problems in shortest-path algorithms because of the possibility of negative cycles. But what if the input graph has no cycles?

   (a) Describe an efficient algorithm to compute the shortest path between two nodes s and t in a given *directed acyclic graph* with weighted edges. The edge weights could be positive, negative, or zero.

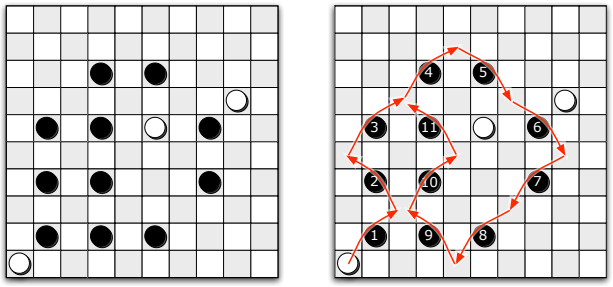
   (b) Describe an efficient algorithm to compute the *longest* path between two nodes s and t in a given directed acyclic graph with weighted edges.

   [Hint: Consider dynamic programming.]

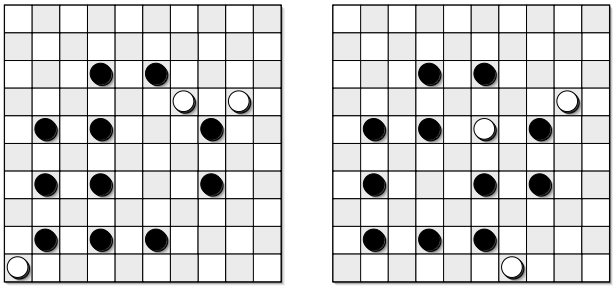
*4. Draughts/checkers is a game played on an $m \times m$ grid of squares, alternately colored light and dark. (The game is usually played on an 8×8 or 10×10 board, but the rules easily generalize to any board size.) Each dark square is occupied by at most one game piece (usually called a *checker* in 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.[1] 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.

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.



White wins in one turn.



White cannot win in one turn from either of these positions.

*[Hint: The greedy strategy—make arbitrary jumps until you get stuck—does **not** always find a winning sequence of jumps even when one exists. You may want to consider Euler tours. Parity, parity, parity.]*

[1] Most other variants of draughts have 'flying kings', which behave *very* differently than what's described here.