

Due February 27, 2:45pm

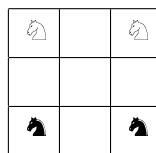
**Instructions:** Write your name, the username for your instructional account, your TA's name, and your discussion section time (where you want the homework to be returned) prominently on the first page of your homework. Also list your study partners for this homework, or "none" if you had no partners. *If you do not write your username, you risk receiving no credit for the homework.* Your homework will be returned at the discussion section of the TA you named; if you do not write your TA's name and section time, the graded homework *will not be returned to you.*

When asked for an algorithm, please provide: (a) the pseudocode of your algorithm, (b) a brief explanation of the main idea/intuition, (c) a proof of correctness, (d) the asymptotic running time of the algorithm, using  $O(\cdot)$  notation, (e) justification for your running time calculation.

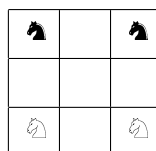
**1. (20 pts.) The four-knight puzzle, reprise**

Remember Alice from HW3? She's back. Thanks to you, she has learned that the four-knight puzzle is solvable. Now she wants to know what is the shortest sequence of moves that solves the puzzle: in other words, what is the minimum number of moves that it takes to solve the puzzle. What algorithm that we saw in class could be used to answer this question? Briefly explain your answer. You don't need to give pseudocode, a proof of correctness, or a running time analysis.

As a reminder, here is her puzzle. She sets up a  $3 \times 3$  chessboard, with two white knights in the upper corners and two black knights in the lower corners, like this:



At first, she asked you whether it was possible to find a sequence of legal chess moves, starting from the above position, so that you reach the following position:



**2. (20 pts.) BFS?**

Suppose we modify BFS to record `pre` and `post` times for each vertex visited. In particular, `pre[v]` records the time at which  $v$  was first added to the queue, and `post[v]` records the time

when we finished examining all edges out of  $v$ . You may assume that BFS is modified to restart exploration as needed to ensure that every vertex is visited, just like was done with DFS.

Now suppose we want to do topological sorting as described in the book, but using BFS instead of DFS. In other words, we run BFS, recording  $\text{pre}[v]$  and  $\text{post}[v]$  as above, and we finish by sorting the vertices according to decreasing  $\text{post}$ -value. Does this produce a true topological sorting of the vertices? Either prove that the resulting algorithm is correct (i.e., that it is guaranteed to output a valid linearization of the dag), or show a counterexample (a dag where the algorithm fails).

### 3. (20 pts.) Travel planning

You are given a set of  $n$  cities and an  $n \times n$  matrix  $M$ , where  $M(i, j)$  is the cost of the cheapest direct flight from city  $i$  to city  $j$ . You live in city A and want to find the cheapest and most convenient route to city B, but the direct flight might not be the best option. As far as you're concerned, the best route must have the following properties: the sum of the costs of the flights in the route should be as small as possible, but if there are several possible routings with the same total cost, then you want the one that minimizes the total number of flights. Design an efficient algorithm to solve this problem. Your algorithm should run in  $O(n^2 \lg n)$  time.

### 4. (30 pts.) Sail trip

Help Carol plan a trip down the Eastern seaboard in her sailboat. Each night she plans to stay at a cove and anchor overnight, then the next day she will sail on to a cove further down the seaboard. There are  $n$  coves on the Eastern seaboard. Also, for each cove, you are given a list  $L_i$  of other coves further down the coast that Carol can safely reach in a single day's sail starting from cove  $i$ . Carol wants to start at cove 1 and end at cove  $n$ .

- Suppose Carol wishes to minimize the number of days her trip will take. Name an algorithm we've seen in class that she could use for this purpose. Briefly explain your answer. You don't need to give pseudocode, a proof of correctness, or a running time analysis.
- Here's a twist. At each cove, the sea bottom is at a different depth. Carol is going to need a long-enough anchor rope to ensure that she can safely anchor at each cove she visits. Assume that you are given the list  $r_1, \dots, r_n$ , where  $r_i$  denotes the minimum length of anchor rope that is required to safely anchor at cove  $i$ . (Assume that Carol wisely refuses to stay at any cove where her anchor rope is too short to safely anchor.) Therefore, Carol will need to buy an anchor rope that's long enough for her to anchor at every cove along her route. If rope costs \$1 per foot and Carol is willing to choose any route as long as it minimizes the necessary cost of her anchor rope, what's the least she can pay to reach her destination? Design an efficient algorithm to solve this problem, given  $n$ ,  $L_1, \dots, L_n$ , and  $r_1, \dots, r_n$ . It is enough to output the minimum anchor rope length that will allow Carol to reach cove  $n$ ; you don't need to bother with outputting the route, if that simplifies your algorithm.

This can be solved in  $O((n+m) \lg n)$  time, where  $m = |L_1| + \dots + |L_n|$ , but we will also accept other algorithms that are roughly as efficient.

### 5. (10 pts.) Reliable routing in computer networks

Suppose we have a directed graph  $G = (V, E)$  describing a computer network, where vertices correspond to hosts or routers and edges correspond to network links. Also assume that for each network link  $(u, v) \in E$ , we are given a measure  $r(u, v)$  of the reliability of this link: specifically,

$r(u, v)$  = the probability that a packet sent across the link  $(u, v)$  will not be lost while it is transiting that link. You may assume that these probabilities are independent. So, if we have a path  $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ , the probability that a packet sent along this path makes it from  $v_0$  to  $v_k$  successfully is given by  $r(v_0, v_1) \times r(v_1, v_2) \times \dots \times r(v_{k-1}, v_k)$ .

Given the graph  $G$ , the reliability measure  $r(\cdot, \cdot)$ , and vertices  $s, t \in V$ , your job is to find a path from  $s$  to  $t$  of maximum reliability. Design an efficient algorithm to solve this problem.

Hint: This is *almost* a shortest-path problem, but we are maximizing instead of minimizing, and we have products instead of sums. But can we “modify” the problem slightly to use a shortest-path algorithm?

**6. (5 pts.) Optional bonus question: fast shortest paths**

This is an *optional* challenge problem, worth 5 bonus points if you solve it. Don't attempt to solve this one unless you have solved all of the other homework questions; this question is intended to be more challenging than the others.

Let  $G = (V, E)$  be a weighted, directed graph with a special property: no strongly connected component has more than 10 vertices, and all edge weights are positive. Given  $G$  and a vertex  $s \in V$ , your task is to label each vertex  $v \in V$  with the length of the shortest path from  $s$  to  $v$ . Design a linear-time algorithm to solve this problem. In other words, the running time of your algorithm should be  $O(|V| + |E|)$ .