CS 577: Introduction to Algorithms Spring 2017

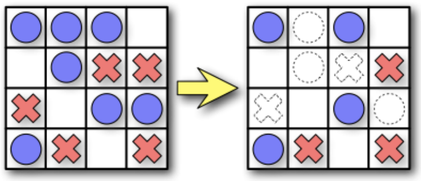
Practice Final Exam

Guidelines:

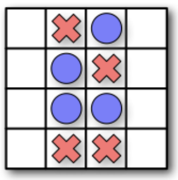
- You are required to answer each of the four questions. The actual exam will be slightly shorter. Aim to finish this practice exam in 2.5 hours.
- You can use all the results we showed in class or in the homework. Clearly state the results you use. Substantiate all other claims you make.
- The exam is closed book and closed notes, but you are allowed to bring a "cheat-sheet"—one page of notes (front and back).

GOOD LUCK!

1. Consider the following solitaire game. The puzzle consists of an $n \times m$ grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions: (1) every row contains at least one stone, and (2) no column contains stones of both colors. For some initial configurations of stones, reaching this goal is impossible.



A solvable puzzle and one of its many solutions.



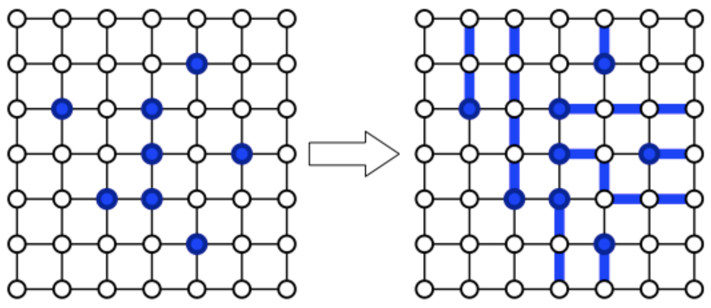
An unsolvable puzzle.

Prove that it is NP-Hard to determine, given an initial configuration of red and blue stones, whether the puzzle can be solved.

2. An $n \times n$ grid is an undirected graph with n^2 vertices organized into n rows and n columns. We denote the vertex in the ith row and the jth column by (i, j). Every vertex in the grid has exactly four neighbors, except for the boundary vertices, which are the vertices (i, j) such that $i = 1$, $i = n$, $j = 1$,or $j = n$.

   Let $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$ be distinct vertices, called terminals, in the $n \times n$ grid. The escape problem is to determine whether there are m vertex-disjoint paths in the grid that connect the terminals to any m distinct boundary vertices. (See picture on back page.) Describe and briefly analyze an efficient algorithm to solve the escape problem.

3. In this problem we will build an unrooted tree node-by-node using a random process. We start with a single node and no edges. At step i for $i \geq 1$, we have i nodes and $i - 1$ edges already in the tree. We will add one new node (called node $i + 1$) and one new edge. The new edge connects node $i+1$ to a node selected independently and uniformly at random from


A positive instance of the escape problem, and its solution.

all nodes added previously (i.e. nodes 1 through i). The process ends when the tree has n nodes. At the end of the process, we are interested in all the nodes that are leaves, that is, have only one edge incident on them.

For example, suppose that $n = 5$, and the following sequence of events happens. At step 2, node 2 attaches to node 1. At step 3, node 3 attaches to node 2. At step 4, node 4 attaches to node 3. At step 5, node 5 attaches to node 2. At this point the process ends. Then the tree has three leaves at the end: nodes 1, 4, and 5.

(a) What is the probability that node i, for some i in $\{1, 2, \cdots, n\}$, is a leaf at the end of the process? Your answer should be a function of i and n.

(b) What is the expected number of leaves in the tree at the end of the process? Your answer should be a function of n, the number of nodes in the tree.

4. You manage n billboards along a popular jogging path. A client of yours wishes to purchase advertising space on your billboards such that every jogger using the path will see at least one of the billboards that carry its advertisement. You are given for each of m joggers an interval in $\{1, \cdots, n\}$ — jogger i runs past billboards $\{a_i, \cdots, b_i\}$ for some $1 \leq a_i < b_i \leq n$. Your goal is to find the smallest set of indices $S \subseteq \{1, \cdots, n\}$ such that for each i, there is some $j \in S$ with $a_i \leq j \leq b_i$.

   Design and briefly analyze an efficient algorithm for this problem.