CS 473 ✧ Spring 2017
Final Exam
May 10, 2017

1. A ***three-dimensional matching*** in an undirected graph G is a collection of vertex-disjoint triangles. A three-dimensional matching is *maximal* if it is not a proper subgraph of a larger three-dimensional matching in the same graph.

   (a) Let M and M' be two arbitrary maximal three-dimensional matchings in the same underlying graph G. ***Prove*** that $|M| \leq 3 \cdot |M'|$.

   (b) Finding the *largest* three-dimensional matching in a given graph is NP-hard. Describe and analyze a fast 3-approximation algorithm for this problem.

   (c) Finding the *smallest maximal* three-dimensional matching in a given graph is NP-hard. Describe and analyze a fast 3-approximation algorithm for this problem.

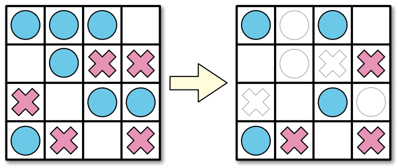
2. Let $G = (V, E)$ be an arbitrary dag with a unique source s and a unique sink t. Suppose we compute a random walk from s to t, where at each node v (except t), we choose an outgoing edge $v \rightarrow w$ uniformly at random to determine the successor of v.

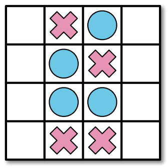
   (a) Describe and analyze an algorithm to compute, for every vertex v, the probability that the random walk visits v.

   (b) Describe and analyze an algorithm to compute the expected number of edges in the random walk.

   Assume all relevant arithmetic operations can be performed exactly in $O(1)$ time.

3. 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.

4. Suppose you are given a bipartite graph $G = (L \sqcup R, E)$ and a maximum matching M in G. Describe and analyze fast algorithms for the following problems:

   (a) Insert(e): Insert a new edge e into G and update the maximum matching. (You can assume that e is not already an edge in G, and that $G + e$ is still bipartite.)

   (b) Delete(e): Delete the existing edge e from G and update the maximum matching. (You can assume that e is in fact an edge in G.)

   Your algorithms should modify M so that it is still a maximum matching, faster than recomputing a maximum matching from scratch.

5. You are applying to participate in this year's Trial of the Pyx, the annual ceremony where samples of all British coinage are tested, to ensure that they conform as strictly as possible to legal standards. As a test of your qualifications, your interviewer at the Worshipful Company of Goldsmiths has given you a bag of n commemorative Alan Turing half-guinea coins, exactly two of which are counterfeit. One counterfeit coin is very slightly lighter than a genuine Turing; the other is very slightly heavier. Together, the two counterfeit coins have *exactly* the same weight as two genuine coins. Your task is to identify the two counterfeit coins.

   The weight difference between the real and fake coins is too small to be detected by anything other than the Royal Pyx Coin Balance. You can place any two disjoint sets of coins in each of the Balance's two pans; the Balance will then indicate which of the two subsets has larger total weight, or that the two subsets have the same total weight. Unfortunately, each use of the Balance requires completing a complicated authorization form (in triplicate), submitting a blood sample, and scheduling the Royal Bugle Corps, so you *really* want to use the Balance as few times as possible.

   (a) Suppose you *randomly* choose $n/2$ of your n coins to put on one pan of the Balance, and put the remaining $n/2$ coins on the other pan. What is the probability that the two subsets have equal weight?

   (b) Describe and analyze a randomized algorithm to identify the two fake coins. What is the expected number of times your algorithm uses the Balance? To simplify the algorithm, you may assume that n is a power of 2.

6. Suppose you are given a set L of n line segments in the plane, where each segment has one endpoint on the vertical line $x = 0$ and one endpoint on the vertical line $x = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which no pair of segments intersects.

Some Useful Inequalities

Let $X = \sum_{i=1}^{n} X_i$, where each X_i is a $0/1$ random variable, and let $\mu = \mathrm{E}[X]$.

- **Markov's Inequality:** $\Pr[X \ge x] \le \mu/x$ for all $x > 0$.
- **Chebyshev's Inequality:** If $X_1, X_2, \dots, X_n$ are pairwise independent, then for all $\delta > 0$:

$$\Pr[X \ge (1+\delta)\mu] < \frac{1}{\delta^2\mu} \quad \text{and} \quad \Pr[X \le (1-\delta)\mu] < \frac{1}{\delta^2\mu}$$

- **Chernoff Bounds:** If $X_1, X_2, \dots, X_n$ are fully independent, then for all $0 < \delta \le 1$:

$$\Pr[X \ge (1+\delta)\mu] \le \exp\left(-\delta^2\mu/3\right) \quad \text{and} \quad \Pr[X \le (1-\delta)\mu] \le \exp\left(-\delta^2\mu/2\right)$$

Some Useful Algorithms

- **Random(k):** Returns an element of $\{1, 2, \dots, k\}$, chosen independently and uniformly at random, in $O(1)$ time. For example, Random(2) can be used for a fair coin flip.
- **Ford and Fulkerson's maximum flow algorithm:** Returns a maximum (s,t)-flow f^* in a given flow network in $\boldsymbol{O(E \cdot |f^*|)}$ ***time***. If all input capacities are integers, then all output flow values are also integers.
- **Orlin's maximum flow algorithm:** Returns a maximum (s,t)-flow in a given flow network in $\boldsymbol{O(VE)}$ ***time***. If all input capacities are integers, then all output flow values are also integers.
- **Orlin's minimum-cost flow algorithm:** Returns a minimum-cost flow in a given flow network in $\boldsymbol{O(E^2 \log^2 V)}$ ***time***. If all input capacities, costs, and balances are integers, then all output flow values are also integers.

Some Useful NP-hard Problems:

3Sat: Given a boolean formula in conjunctive normal form, with exactly three distinct literals per clause, does the formula have a satisfying assignment?

MaxIndependentSet: Given an undirected graph G, what is the size of the largest subset of vertices in G that have no edges among them?

MaxClique: Given an undirected graph G, what is the size of the largest complete subgraph of G?

MinVertexCover: Given an undirected graph G, what is the size of the smallest subset of vertices that touch every edge in G?

MinSetCover: Given a collection of subsets $S_1, S_2, \dots, S_m$ of a set S, what is the size of the smallest subcollection whose union is S?

MinHittingSet: Given a collection of subsets $S_1, S_2, \dots, S_m$ of a set S, what is the size of the smallest subset of S that intersects every subset S_i?

3Color: Given an undirected graph G, can its vertices be colored with three colors, so that every edge touches vertices with two different colors?

HamiltonianCycle: Given a graph G (either directed or undirected), is there a cycle in G that visits every vertex exactly once?

FeasibleILP: Given a matrix $A \in \mathbb{Z}^{n \times d}$ and a vector $b \in \mathbb{Z}^n$, determine whether the set of feasible integer points $\max\{x \in \mathbb{Z}^d \mid Ax \le b, x \ge 0\}$ is empty.

HydraulicPress: And here ve go!

Common Grading Rubrics

(For problems out of 10 points)

General Principles:

- Faster algorithms are worth more points, and slower algorithms are worth fewer points, typically by 2 or 3 points (out of 10) for each factor of n. Partial credit is scaled to the new maximum score, and all points above 10 are recorded as extra credit.
- A clear, correct, and correctly analyzed algorithm, no matter how slow, is always worth more than "I don't know". An incorrect algorithm, no matter how fast, may be worth nothing.
- Proofs of correctness are required on exams if and only if we explicitly ask for them.

Dynamic Programming:

- 3 points for a **clear English specification** of the underlying recursive function = 2 for describing the function itself + 1 for describing how to call the function to get your final answer. We want an English description of the underlying recursive *problem*, not just the algorithm/recurrence. In particular, your description should specify precisely the role of each input parameter. **No credit for the rest of the problem if the English description is is missing; this is a Deadly Sin.**
- 4 points for correct recurrence = 1 for base case(s) + 3 for recursive case(s). **No credit for iterative details if the recursive case(s) are incorrect.**
- 3 points for iterative details = 1 for memoization structure + 1 for evaluation order + 1 for time analysis. Complete iterative pseudocode is *not* required for full credit.

Graph Reductions:

- 4 points for a complete description of the relevant graph, including vertices, edges (including whether directed or undirected), numerical data (weights, lengths, capacities, costs, balances, and the like), source and target vertices, and so on. If the graph is part of the original input, just say that.
- 4 points for other details of the reduction, including how to build the graph from the original input, the precise problem to be solved on the graph, the precise algorithm used to solve that problem, and how to extract your final answer from the output of that algorithm.
- 2 points for running time of the overall algorithm, expressed as a function of the original input parameters, *not* just the number of vertices and edges in the graph.

NP-hardness Proofs:

- 3 points for a complete description of the reduction, including an appropriate NP-hard problem to reduce from, how to transform the input, and how to transform the output.
- 6 points for the proof of correctness = 3 for the "if" part + 3 for the "only if" part.
- 1 points for "polynomial time".