FINAL EXAM PRACTICE PROBLEMS — CMSC 451 (Spring 2016)

The final exam will be on Thursday, May 12, from 8:00–10:00 am, at our regular class location (CSI 2117). It will be closed-book and closed-notes, except that you may bring notes written on (both sides of) a single sheet of letter-size paper, which you will be asked to turn in along with your exam.

The following problems may help you prepare for the exam. They are not to be handed in. Solutions are not available, but of course you are welcome to discuss these problems during office hours or on Piazza.

Disclaimer: Many of these problems are taken from exams in previous offerings of CMSC 451 when the emphasis of the course may have been different. The exam will not be as long as this set of practice problems, and this selection of problems does not necessarily reflect the difficulty level or distribution of material to be covered on the exam.

1. *Short answer questions.*

   For the following questions, explanations are required only when explicitly requested, but can always be given for partial credit.

   (a) Yes or No: Suppose an algorithm runs in $O(n2^{(2\log_2 n)})$ time. Is this a polynomial-time algorithm?

   (b) Suppose you are given an undirected graph which has n vertices, and each vertex has exactly six incident edges. As a function of n, what is the total number edges in this graph? (Give an exact answer for full credit, or an asymptotic answer for partial credit.)

   (c) You are given a digraph G that has edge weights with the values of either $+1$ or -1. It has no negative-cost cycles. Which of the following can be used to compute shortest paths in G from a single source vertex? (List all that apply): breadth-first search, Dijkstra's algorithm, Bellman-Ford algorithm.

   (d) You are given a connected, undirected graph $G = (V, E)$ in which each edge has a numeric edge weight, and all edge weights are distinct. Let e_1, e_2, and e_3 be the edges with smallest, second smallest, and third smallest weights among all the edges of G, respectively. Among these three edges, which (if any) *must* be in the minimum spanning tree (MST) of G, which (if any) *might* be in the MST, and which (if any) are definitely *not* in the MST?

   (e) Explain why Kruskal's MST algorithm made use of a Union-Find data structure.

   (f) True or False: Suppose that the capacities of the edges of an s-t network are integers that are all integer multiples of 3. Then there exists a maximum flow such that the flow on each edge is also integer multiples of 3.

   (g) Given a flow network G, let (X, Y) denote a cut. Let c denote the sum of the edge capacities of all edges (x, y), where $x \in X$ and $y \in Y$. What can be said about the maximum flow in G?

      i. The value of the maximum flow in G is at most c (but may be smaller)

      ii. The value of the maximum flow in G is at least c (but may be larger)

      iii. The value of the maximum flow in G is equal to c.

      iv. We cannot say anything about the maximum flow, because we failed to consider the capacities of the edges going from Y to X when defining c.

(h) True or False: It is possible to determine in polynomial time whether a graph has an independent set of size 100.

(i) Technically advanced aliens come to Earth and show us that some known NP-hard problem cannot be solved faster than $O(n^{100})$ time. Does this resolve the question of whether P = NP? (Explain briefly.)

(j) Suppose that $A \leq_P B$, the reduction runs in $O(n^2)$ time, and B can be solved in $O(n^4)$ time. What can we infer about the time needed to solve A? (Explain briefly.)

(k) True or False: If a graph G has a vertex cover of size k, then it has a dominating set of size k or smaller.

(l) Suppose that $A \leq_P B$, and there is a factor-2 approximation to problem B. Which of the following necessarily follows?

  i. There is a factor-2 approximation for A.

  ii. There is a constant factor approximation for A, but the factor might not be 2.

  iii. We cannot infer anything about our ability to approximate A.

(m) Recall that the Hadamard gate is described by the matrix $\frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$. Suppose the Hadamard gate is applied to a qubit in the state $\frac{1}{5}\left(\begin{smallmatrix} 3 \\ 4 \end{smallmatrix}\right)$ and the qubit is then measured in the computational basis. What is the probability of obtaining the outcome "0"?

2. *Minimum spanning trees and shortest paths.*

You are given a connected, undirected graph $G = (V, E)$, where each edge $(u, v) \in W$ is labeled with a nonnegative weight $w(u, v)$. Suppose that we run Prim's MST algorithm starting at some source node $s \in V$. Let T be the resulting tree. You would like to know whether T is also the shortest path tree for the single-source node s. Of course, you could run Dijkstra's algorithm and compare the two trees, but this would take additional $O(n \log n + m)$ time, where $n = |V|$ and $m = |E|$.

Show that (once computed) it is possible to determine whether T is the shortest-path tree in only $O(n + m)$ time. Your algorithm takes as input the graph G, a source vertex s, and (as in the output of Prim's algorithm) the tree T is represented by the values $\text{pr}[v]$ for each $v \in V$. Your algorithm need only answer "yes" or "no". You may assume that all edge weights and shortest path lengths are distinct. Justify your algorithm's correctness and derive its running time.

3. *Scheduling to minimize average lateness.*

In class we presented a greedy algorithm for scheduling a set of n tasks, in which each task has a duration t_i and deadline d_i. We showed that scheduling the tasks in increasing order of deadline minimizes the maximum lateness. (Recall that if task i is scheduled at time s_i, then its lateness is $l_i = \max\{0, s_i + t_i - d_i\}$.) Define the average lateness to be $\frac{1}{n}\sum_{i=1}^{n} l_i$.

(a) Provide a counterexample to show that scheduling tasks in increasing order of deadline does not minimize average lateness. Briefly explain your example.

(b) Provide a counterexample to show that scheduling tasks in increasing order of duration does not minimize average lateness. Briefly explain your example.

(c) Suppose that we redefine lateness to be $l_i = s_i + t_i - d_i$ (ignoring the "max"). Thus, if the task finishes before the deadline, its lateness is negative. (Intuitively, this can be thought of as a bonus, which can be applied to reduce the positive lateness of some other task.) Prove that if tasks are scheduled in increasing order of duration, then average lateness (under this modified definition) will be minimized. (Hint: Use an exchange argument.)

4. *Counting mixed-parity inversions.*

   Given a list $A = (a_1, \ldots, a_n)$ of integers, a *mixed-parity inversion* is a pair a_i and a_j such that $i < j$, $a_i > a_j$, and a_i and a_j are of different parities. (In other words, it is an inversion in which one number is even and the other is odd.)

   Design an $O(n \log n)$-time algorithm that counts the number of mixed-parity inversions in a list A containing n elements. Justify your algorithm's correctness and derive its running time.

5. *Hamiltonian paths in tournaments.*

   A *tournament* is a digraph $G = (V, E)$ in which for each pair of vertices u and v, either there is an edge (u, v) or an edge (v, u), but not both. A directed *Hamiltonian path* is a path that visits every vertex in a digraph exactly once.

   (a) Prove that for all $n \geq 1$, every tournament on n vertices has a directed Hamiltonian path. (Hint: Use induction on the number of vertices.)

   (b) Give an $O(n^2)$ algorithm that, given a tournament, finds a Hamiltonian path. (You may assume either an adjacency list or an adjacency matrix representation of G.)

6. *Bucket redistribution.*

   You are given a collection of n blue buckets and n red buckets, denoted B_i and R_i for $0 \leq i \leq n-1$. Initially each of the blue buckets contains some number of balls and each red bucket is empty. The objective is to transfer all the balls from the blue buckets into the red buckets, subject to the following restrictions.

   The input to the problem consists of two sequences of integers, $(b_0, b_1, \ldots, b_{n-1})$ and $(r_0, r_1, \ldots, r_{n-1})$. Blue bucket B_i holds b_i balls initially, and at the end, red bucket R_i should hold exactly r_i balls. The balls from blue bucket B_i may be redistributed only among the red buckets R_{i-1}, R_i, and R_{i+1} (with indices taken modulo n). You may assume that $\sum_i b_i = \sum_i r_i$.

   Design a polynomial-time algorithm that, given the lists $(b_0, b_1, \ldots, b_{n-1})$ and $(r_0, r_1, \ldots, r_{n-1})$, determines whether it is possible to redistribute the balls from the blue buckets into the red buckets according to these restrictions.

7. *Shuffles.*

   You are given three strings $X = x_1 \ldots x_m$, $Y = y_1 \ldots y_n$, and $Z = z_1 \ldots z_p$, where $p = m + n$. We say that Z is a *shuffle* of X and Y if it is possible to interleave the individual symbols of X and Y (without changing their relative order within each string) to obtain Z. Present an efficient algorithm that, given strings X, Y, and Z, determines whether Z is a shuffle of X and Y. Give a short proof of your algorithm's correctness and derive its running time.

   (Hint: Use dynamic programming. It suffices to present just the recursive formulation. It is possible to do this in $O(mn)$ time.)

8. *Partitioning.*

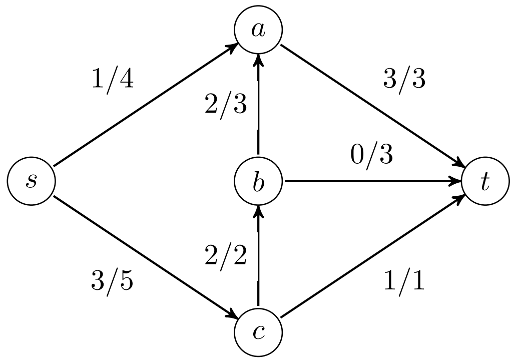
   Suppose a shipping company wants to ship n objects of weights $w_1, \ldots, w_n$. Each weight is a positive integer. The company wants to partition these objects between two different ships so that the total weight of the two ships is as similar as possible. In particular, if W_1 is the total weight of objects on Ship 1, and W_2 is the total weight on Ship 2, then the objective is to minimize the weight ratio, $\max\{W_1, W_2\}/\min\{W_1, W_2\}$. Observe that this ratio is never smaller than 1, and it equals 1 if and only if the two ships are carrying identical total weights.

Present an efficient algorithm that, given the weights $\{w_1, \ldots, w_n\}$, computes the optimum weight ratio. You can express your running time as a function of both n and the total weight $W = \sum_{i=1}^{n} w_i$.

(Hints: Use dynamic programming. You are not required to give the entire DP algorithm, just a recursive formulation. You need only compute the optimum weight ratio, not the actual partition. Justify your algorithm's correctness and derive its running time. Note that $O(nW)$ time is possible. It suffices to focus on computing the total weight carried by just one of the ships, since the other must carry all the remaining weight.)

9. *Max flow.*

Consider the flow network shown below, and suppose that we have already computed a flow f on this network. In the diagram, each edge e is labeled with $f(e)/c(e)$, where $f(e)$ is the flow on this edge and $c(e)$ is the capacity of the edge.



(a) Show the residual network corresponding to this flow.

(b) Show any augmenting path from s to t in the residual network.

(c) Show the new flow that results by pushing as much flow as possible through this augmenting path.

10. *Minimum cut with many terminals.*

You are given a directed graph $G = (V, E)$ with a source vertex s and a set of terminal vertices $T = t_1, \ldots, t_k$. Present a polynomial-time algorithm to determine the minimum number of edges to remove so that there is no path from s to any of the terminals. Prove that your algorithm is correct and analyze its running time.

11. *Can you hear me now? Part II.*

You are given a collection of n points $U = \{u_1, u_2, \ldots, u_n\}$ in the plane, each of which is the location of a cell phone user. You are also given the locations of m cell towers, $C = \{c_1, c_2, \ldots, c_m\}$. A cell-phone user can connect to a tower if it is within distance Δ of the tower. For the sake of fault-tolerance, each cell-phone user must be connected to at least three different towers. For each tower c_i you are given the maximum number of users, m_i, that can connect to this tower.

Give a polynomial-time algorithm that determines whether it is possible to assign all the cell-phone users to towers, subject to these constraints. Prove its correctness. (You may assume you have a function that returns the distance between any two points in $O(1)$ time.)

12. *Pharmacy scheduling.*

    Your local pharmacy has asked you to help set up the work schedule for the next month. There are n pharmacists on the staff and m work days in the month. Each pharmacist gives a list of the days of the month that he/she is available to work. For the ith pharmacist, this is given as a list A_i, where each number in the list is in the range from 1 to m. Let $d_i = |A_i|$ denote the number of days that pharmacist i is available to work. Then he/she should be scheduled to work at least $\lceil d_i/2 \rceil$ of these days. Each day there must be exactly 3 pharmacists on duty.

    Present an efficient algorithm that determines whether there exists a schedule that satisfies all the above requirements. Present a brief proof that your algorithm is correct. (Hint: Reduce to Max Flow.)

13. *Balanced 3-coloring.*

    Show that the following problem is NP-complete. Remember to show (1) that it is in NP and (2) that some known NP-complete problem can be reduced to it.

    Given a graph $G = (V, E)$, where $|V|$ is a multiple of 3, can G can be 3-colored such that the sizes of the 3 color groups are all equal to $|V|/3$? That is, can we assign an integer from $\{1, 2, 3\}$ to each vertex of G such that no two adjacent vertices have the same color, and such that all the colors are used equally often? (Hint: Reduction from the standard 3-coloring problem.)

14. *Angry knight seating.*

    King Arthur has a large round table around which all his knights should sit. Unfortunately, some knights hate each other and thus cannot be seated next to each other. There are n knights, $\{v_1, v_2, \ldots, v_n\}$, and the king has given you a list of pairs of the form $\{v_i, v_j\}$, which indicates that knights v_i and v_j hate each other.

    The king asks you to write a program to determine if it is possible to seat the knights about the table. Instead, prove that the problem is NP-complete.

15. *High-degree independent set.*

    In the High-Degree Independent Set (HDIS) problem, you are given a graph $G = (V, E)$ and an integer k, and you want to know whether G has an independent set V' of size k such that each vertex of V' is of degree at least k.

    (a) Show that HDIS is in NP.

    (b) Show that HDIS is NP-hard. (Hint: Use the standard independent set problem.)

16. *Acyclic subgraph.*

    In the acyclic subgraph problem, you are given a directed graph $G = (V, E)$ and an integer k, and are asked to determine whether G contains a subset V' of k vertices such that the induced subgraph on V' is acyclic. (Recall that the induced subgraph on V' is the subgraph $G' = (V', E')$ whose vertex set is V', and for which $(u, v) \in E'$ if $u, v \in V'$ and $(u, v) \in E$. Hint: Try a reduction from Independent Set. Think of a reduction that maps undirected edges to directed cycles.)

17. *Hamiltonian paths.*

Given an undirected graph $G = (V, E)$, a *Hamiltonian path* is a simple path (not a cycle) that visits every vertex in the graph. The Hamiltonian Path problem is the problem of determining whether a given graph has a Hamiltonian path.

(a) Show that Hamiltonian Path is in NP.

(b) Observe that if a graph has a Hamiltonian Cycle, then it also has a Hamiltonian Path. Consider the following attempt to prove that Hamiltonian Path is NP-hard: given a graph G for the Hamiltonian Cycle problem, simply output a copy of this graph. Explain why this reduction is incorrect.

(c) Prove that Hamiltonian Path is NP-hard. (Hint: You can give a reduction from the Hamiltonian Cycle problem.)

18. *Approximate partitioning.*

Recall the Partition Problem (Problem 8 above). Consider the following heuristic for this problem. First, sort the objects by decreasing order of their weights. Initially both ships have 0 weight. Repeatedly, take the next object from the sorted list and put it on the ship whose total weight is the smaller of the two. (If they are tied, put it on Ship 1.) Update the weight of this ship.

(a) Briefly explain how to implement this in $O(n \log n)$ time.

(b) Give a small example that shows that this is not optimal.

(c) Prove that for any input, this heuristic achieves an approximation ratio of at most 2. In other words, show that the weight ratio produced by this heuristic is at most twice as large as the weight ratio of the optimal solution.

19. *Set cover approximation.*

The set cover optimization problem is as follows: Given a pair (X, S), where X is a finite set and a $S = \{s_1, s_2, \ldots, s_n\}$ is a collection of subsets of X, find a minimum-sized collection of these sets whose union equals X. Consider a special version of the set cover problem in which each element of X occurs in at most three sets of S. Present an approximation algorithm for this special version of the set cover problem with a ratio bound of 3. Briefly derive the ratio bound of your algorithm.

20. *Bottleneck TSP.*

Suppose you have a sequence of points $X = x_1, \ldots, x_n$ sorted from left to right along a line. The distance between two points x_i and x_j is just their absolute difference $|x_j - x_i|$. The *bottleneck TSP* is the following: Find a cycle that visits each point exactly once, such that maximum distance traveled between any two consecutive points is as small as possible.

Consider the following alternating heuristic for this problem: Travel from x_1 to x_n, skipping every other point. Then return from x_n to x_1, visiting the skipped points.

Prove that this heuristic provides a factor-2 approximation to the bottleneck TSP. (Hint: Let L be the maximum distance between any two consecutive points. Relate the costs of the optimum path and the heuristic path to L.)

21. *Randomized approximation algorithm for 3-coloring.*

Suppose you are given an undirected graph $G = (V, E)$ and are asked to color each of its vertices with one of three colors. We say an edge $\{u, v\} \in E$ is *satisfied* if the colors assigned to u and v are different. In the optimization version of 3-coloring, your goal is to maximize the number of satisfied edges. Let c^* denote the maximum possible number of satisfied edges. Design a polynomial-time randomized algorithm with the property that the expected number of satisfied edges is at least $\frac{2}{3}c^*$. Establish your algorithm's running time and prove that it is correct.

22. *Applying Karger's algorithm to the s-t cut problem.*

Recall that Karger's contraction algorithm finds the global minimum cut by repeatedly contracting random edges of a given input graph. To apply the same strategy to the s-t cut problem, we might try the following. At any iteration of the algorithm, let v_s and v_t denote the vertices corresponding to the subsets that contain vertices s and t, respectively. Since s and t should be on opposite sides of the cut, we delete any edges connecting v_s and v_t and select a random edge to contract among the remaining edges. Give an example showing that the probability of this algorithm finding a minimum s-t cut can be exponentially small.