CS 170 Algorithms
Fall 2014 David Wagner
Final Exam

PRINT your name: _____, _____
(last) (first)

SIGN your name: _____

Your Student ID number: _____

Your Unix account login: cs170-_____

Name of the person
sitting to your left: _____

Name of the person
sitting to your right: _____

You may consult three double-sided sheet of notes. You may not consult other notes, textbooks, etc. Computers and other electronic devices are not permitted.

There are 16 questions. The last page is page 24.

Reminder: When we ask for an efficient algorithm, we want one whose asymptotic running time is as fast as possible.

Answer all questions. Be precise and concise. Write in the space provided. Good luck!

Do not turn this page until an instructor tells you to do so.

Problem 1. [True or false] (6 points)

Circle TRUE or FALSE. Do not justify your answers on this problem.

(a) TRUE or FALSE: If we perform DFS on an undirected graph, there are no cross edges.

(b) TRUE or FALSE: If the DFS tree has no back edges, then there are no cycles in the graph.

(c) TRUE or FALSE: Any tree with n vertices is guaranteed to have exactly $n-1$ edges.

(d) TRUE or FALSE: Suppose G is a directed graph $G=(V,E)$, and define the undirected graph G' by $G'=(V,E')$ where E' is the set of edges $\{u,v\}$ such that $(u,v)\in E$ or $(v,u)\in E$. Then, if G is *not* strongly connected, then it's guaranteed that G' is not connected.

(e) TRUE or FALSE: If G is a strongly connected directed graph, then there is guaranteed to exist a vertex such that deleting that vertex leaves you with a graph that is also strongly connected.

(f) TRUE or FALSE: Suppose we have a directed acyclic graph (dag) G with at least $n-1$ edges and where a topological sort yields the vertices $v_1,\ldots,v_n$, in that order. If we add the edge (v_n,v_1), then the resulting graph is guaranteed to be strongly connected.

(g) TRUE or FALSE: Dijkstra's algorithm will always work correctly on any graph that has at most two negative edges.

(h) TRUE or FALSE: If an undirected graph with n vertices has k connected components, then it must have at least $n-k$ edges.

(i) TRUE or FALSE: If a graph with integer lengths on the edges (some of them possibly negative) has no negative cycle reachable from s, then Dijkstra's algorithm works correctly for finding the length of the shortest path from s to every other vertex reachable from s.

(j) TRUE or FALSE: If we have a linear program where all of the coefficients and constants in every linear inequality are integers, then the optimum solution to this linear program must assign an integer value to every variable.

(k) TRUE or FALSE: If we have an undirected graph $G=(V,E)$ where $|V|=|E|$, then G is guaranteed to be a tree.

(l) TRUE or FALSE: If G is a graph with positive lengths on the edges, and if all edge lengths are distinct, and if s,t are two vertices, then there is a unique shortest path from s to t.

Problem 2. [Running time] (6 points)

Answer the following questions. No need to justify your answer. Use $\Theta(\cdot)$ notation.

(a) What is the asymptotic running time of decomposing a directed graph into strongly connected components, if the graph has n vertices and $n \lg n$ edges?

(b) What is the asymptotic running time of Dijkstra's algorithm, on a graph with n vertices and m edges (assuming all edges have non-negative length), if we implement the priority queue with a binary heap?

(c) What is the asymptotic running time of the Bellman-Ford algorithm, on a graph with n vertices and m edges?

(d) What is the asymptotic solution to the recurrence $F(n) = 3F(n/2) + \lg n$?

Problem 3. [Short answer] (6 points)

Answer the following questions, giving a short justification (a sentence or two).

(a) Suppose we have already trained two classifiers. Classifier A is a k-nearest neighbors classifier with a fixed $k = 25$, using a linear search implementation (not a k-d tree). Classifier B is a random forest classifier with a fixed $T = 25$ (i.e., 25 decision trees), where each decision tree has depth at most 10. Now we want to know the running time to classify 100 test points. If the number of observations in the training set is very large, which classifier will be faster at this task? In other words, if we consider the asymptotic running time for this task as a function of n, which classifier will have an asymptotically faster running time?

(b) Let n be a power of two. Suppose you don't know the coefficients of a degree $n-1$ polynomial f, but you have its values at the nth roots of unity. How quickly could you find the value of f at some given other point x? In other words, what would the asymptotic running time be, as a function of n? You can assume that it takes $O(1)$ time to add or multiply two numbers.

(c) In a strongly connected directed graph with integer weights (some of them possibly negative), is it possible that there are vertices s and t with no shortest path from s to t? Write "yes" or "no" and justify your answer in a sentence or two. Then, if you wrote "yes", name an algorithm we could use to detect whether this is the case; if you wrote "no", name an algorithm we could use to detect whether there are two vertices with no *longest* path from one to the other (i.e., the same task except with shortest replaced by longest).

Problem 4. [More running time] (2 points)

We have a directed graph $G = (V, E)$ with n vertices and m edges and integer lengths on all the edges. Only k of the edges have negative lengths; all the others have non-negative lengths. There are no negative cycles.

Your friend has a clever idea for computing the length of the shortest path from a vertex s to a vertex t. He will construct a new graph G' whose vertices are the endpoints of the negative edges of G, together with s and t. Each edge (u, v) in G' represents the length of the shortest path from u to v in G without using any negative-length edge. He'll use Dijkstra's algorithm to compute each of these lengths. Then, he'll add in all of the negative edges to G', and run Bellman-Ford on the result.

What is the asymptotic running time of your friend's algorithm, as a function of n and m and k? Give a short justification (a sentence or two).

Problem 5. [Algorithm design: sorted lists] (7 points)

Given two **sorted** lists of size m and n, we want to find the kth smallest element in the concatenation of the two lists. Fill in the blanks below so we get an algorithm whose running time is $O(\lg m + \lg n)$. Assume $k < m$ and $k < n$.

Find_kth_elem($A[1..m], B[1..n], k$):
1. Let $x := A[\lfloor k/2 \rfloor]$ and $y := B[\lceil k/2 \rceil]$.
2. If $x == y$:
3. Return _____
4. If $x > y$:
5. Return Find_kth_elem(A[_____.._____], B[_____.._____], _____)
6. If $x < y$:
7. Return Find_kth_elem(A[_____.._____], B[_____.._____], _____)

Now, answer the following question. Which algorithm design paradigm does this algorithm best represent?

1. Divide-and-conquer
2. Greedy algorithm
3. Dynamic programming
4. Linear programming
5. Reduce to network flow
6. None of the above

Problem 6. [Linear programming: space radar] (7 points)

A set of n space stations need your help in building a radar system to track spaceships traveling between them. The ith space station is located in 3D space at coordinates (x_i, y_i, z_i). The space stations never move. Each space station will have a radar of some power, say r_i for the ith space station, where r_i is to be determined.

You want to figure how powerful to make each space station's radar transmitter, so that whenever any spaceship travels in a straight line from one space station to another, it will always be in radar range of either the first space station (its origin) or the second space station (its destination). A radar with power r is capable of tracking space ships anywhere in the sphere with radius r centered at itself. Thus, a space ship is within radar range through its trip from space station i to space station j if every point along the line from (x_i, y_i, z_i) to (x_j, y_j, z_j) falls within either the sphere of radius r_i centered at (x_i, y_i, z_i) or the sphere of radius r_j centered (x_j, y_j, z_j). The cost of each radar transmitter is proportional to its power, and you want to minimize the total cost of all of the radar transmitters.

You are given all of the $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$ values, and your job is to choose values for $r_1, \ldots, r_n$. Express this problem as a linear program.

(a) Describe your variables for the linear program.

(b) Write out the objective function.

(c) Between each pair of space stations, say station i and station j, we need one constraint (one linear inequality). What is it?

Problem 7. [Algorithm design: pizza] (6 points)

Our spaceship's cook has baked a zero-gravity pizza and cut it into n slices, but the lack of gravity made him clumsy and the pizza wasn't evenly sliced: the n slices have size $s_1, s_2, \ldots, s_n$. There are n hungry space travelers on board who each want to eat a slice of pizza. Suppose the ith traveler would be happy with any slice whose size is at least t_i. Give an efficient algorithm to determine whether it is possible to distribute the pizza slices so everyone is happy.

(a) What algorithm design paradigm is most appropriate for this problem? Circle one of the following.

(a) Divide-and-conquer
(b) Greedy algorithm
(c) Dynamic programming
(d) Linear programming
(e) Reduce to network flow
(f) None of the above

(b) Show pseudocode for your algorithm. (No proof of correctness needed.)

(c) What is the asymptotic running time of your algorithm? You don't need to justify your answer.

Problem 8. [Algorithm design: sums] (7 points)

You're given a positive integer K and a set of n different positive integers $A = \{a_1, a_2, \ldots, a_n\}$. Your job is to determine whether there exists two sets S, T such that (1) S and T are subsets of A, (2) S and T are disjoint (i.e., $S \cap T = \emptyset$), (3) $S \cup T = A$, and (4) $(\sum_{s \in S} s) - (\sum_{t \in T} t) = K$.

Design an efficient algorithm for this problem.

(a) What algorithm design paradigm is most appropriate for this problem? Circle one of the following.

   (a) Divide-and-conquer
   (b) Greedy algorithm
   (c) Dynamic programming
   (d) Linear programming
   (e) Reduce to network flow
   (f) None of the above

(b) Show pseudocode for your algorithm. (No proof of correctness needed.)

(c) What is the asymptotic running time of your algorithm from part (b)? Write your answer as a function of n, K, and/or L, where we define $L = a_1 + a_2 + \cdots + a_n$. You don't need to justify your answer.

Problem 9. [Algorithm design: knapsack] (7 points)

Consider the discrete knapsack problem where all items have the same value 42. In other words, we have n items, where w_i is the weight of the ith item and every item has value 42, and we have a knapsack of capacity C. (This is knapsack without repetition: each item can be chosen at most once.) We want to find a set of items whose total weight is at most C and whose total value is as large as possible.

Devise an efficient algorithm for this problem, then answer the following two questions about it.

(a) What algorithm design paradigm is most appropriate for this problem? Circle one of the following.

   (a) Divide-and-conquer
   (b) Greedy algorithm
   (c) Dynamic programming
   (d) Linear programming
   (e) Reduce to network flow
   (f) None of the above

(b) What is the asymptotic running time of your algorithm? Justify your answer in a sentence or two.

Problem 10. [Algorithm design: phone numbers] (4 points)

You're the manager for the phone company on board a new space station, and you're trying to assign numbers to various services. To increase dialing speed, you want a call to be placed as soon as a valid number is dialed. For example, if 911 is the number for emergency services, then entering those three digits starts the call immediately, so you can't assign the number 9113 to some other service, since anyone trying to dial it would end up calling 911.

There are n phone services. You have statistics for each service from other space stations; you know that the ith service is used on average f_i times per year. Given $f_1, \ldots, f_n$, you want to assign phone numbers to the services (using only the digits 0–9) so that services which are used more often will have shorter numbers. In particular, you want to minimize the average number of digits dialed (with the average taken over all calls to a phone service).

There's an algorithm we've seen in class that is perfect for solving this problem, if you make a modification. Which one? What modification do you need to make?

Problem 11. [Algorithm design: catch the thief] (7 points)

Oh no! Someone has stolen the draft of Rohit's paper proving $\mathsf{P} = \mathsf{NP}$. The thief is trying to get out of Berkeley before getting caught. The Berkeley police have asked you to help them block the thief's escape.

The roads and intersections are represented as an unweighted directed graph $G = (V, E)$. We're also given a subset $C \subseteq V$ of possible current locations of the thief's car and a set $P \subseteq V$ of all exit points out of Berkeley. The police want to set up roadblocks on a subset of the edges to prevent the thief from escaping (i.e., from reaching one of the exit points). Clearly the police could just put a roadblock at all the edges to each exit point, but there might be a better way: for example, if the thief is known to be at a particular intersection, they could just block all roads coming out of it. You may assume that roadblocks can be set up instantaneously.

Design an efficient algorithm to find a way to stop the thief using the smallest possible number of roadblocks.

(a) What algorithm design paradigm is most appropriate for this problem? Circle one of the following.

   (a) Divide-and-conquer
   (b) Greedy algorithm
   (c) Dynamic programming
   (d) Linear programming
   (e) Reduce to network flow
   (f) None of the above

(b) Describe your algorithm. (You don't need to prove it correct.)

(c) What is the asymptotic running time of your algorithm in part (b)? Your answer can be a function of $|C|$, $|P|$, $|V|$, and/or $|E|$. You don't need to justify your answer.

Problem 12. [Algorithm design: maze] (7 points)

You've landed on an alien planet, in the middle of some strange maze, and you want to find the quickest way out.

The maze can be thought of as a $m \times n$ grid where some of the squares in the grid are blocked. You know the entire maze (including the size of the grid, which squares are blocked, and your current location). You can drive north, south, west, or east from each square in your hovercar, but not diagonally.

Unfortunately, your hovercar was damaged during the rough landing onto the planet, and it can't turn left or make U-turns: the hovercar can only go straight or turn right. For instance, if you leave the current square by going north, you'll only be able to leave the next square by going north or east; west is not possible, because that would require a left turn, and south is not possible either, because would require a U-turn. Your hovercar was landed facing north, so on your first move you will need to drive north.

Find an efficient algorithm to find the shortest route to escape from the maze. We want you to do this by creating an appropriate directed graph G and then using breadth-first search on G. Answer each of the following questions. You don't need to justify any of your answers.

(a) Explain how to create the graph G.

(b) How many vertices does your graph G have?

(c) How many edges does your graph G have? (at most)

(d) What is the asymptotic running time of the resulting algorithm, as a function of m and n?

Problem 13. [Stock trading] (7 points)

By exploiting a handy wormhole to travel through time, Daniel knows what the stock price of Intergalactic Starships, Inc. will be on every day over the next n days: on day i, each share of stock costs p_i. Daniel starts out with \$1000. He plans to pick one day to buy as many shares of stock as he can afford and then sell them all on some subsequent day. Design an efficient algorithm to help Daniel figure out what's the maximum profit he can achieve. (Warning: Stock prices for starship companies can fluctuate a lot. There is no guarantee that the day with the lowest stock price will come before the day with the highest stock price.)

(a) Show your pseudocode. You don't need to prove your pseudocode correct.

(b) What is the asymptotic running time of your algorithm? Justify your answer concisely.

(c) Now suppose Daniel is willing to do at most k trades, where each trade constitutes buying as many shares as he can afford on some day, or selling all his shares on some day. Daniel wants an efficient algorithm to calculate the maximum profit he can achieve.

What algorithm design paradigm is most appropriate for this problem? Circle one of the following.

(a) Divide-and-conquer

(b) Greedy algorithm

(c) Dynamic programming

(d) Linear programming

(e) Reduce to network flow

(f) None of the above

You do not need to describe your algorithm; just answer the above question.

Problem 14. [Algorithm design: wormholes] (7 points)

The inhabitants of a far off star system have developed a transportation system between their planets. There are n planets in the system. Some planets have a wormhole between them, which allows you to travel between planet i to planet j in $t_{i,j}$ seconds. All times $t_{i,j}$ are positive (you can't travel back in time). Some pairs of planets may not have a wormhole between them.

You are currently at planet 0 and wish to reach planet $n-1$. However, your ship's navigation commands are starting to malfunction. When you are at a planet and tell the ship to go through a certain wormhole, it may misinterpret the command and instead choose a completely random outgoing wormhole instead. Luckily, you know that you that this will happen at most once. You don't know when the malfunction might happen, but you are going to assume it will happen at the most inconvenient time.

You want to choose a strategy that will minimize your total travel time in the very worst case (no matter when the malfunction happens or which outgoing wormhole it takes you on). Design an efficient algorithm for this. Specifically, your algorithm should output at what time you can be guaranteed to reach planet $n-1$, if we use the optimal strategy.

You don't need to prove your algorithm correct or justify the stated running time. You may assume it's possible to reach planet $n-1$ from any other planet via some sequence of wormholes.

(a) Describe your main idea, in enough detail that another CS 170 student could implement your algorithm. (No pseudocode or proof of correctness needed.)

(b) What's the asymptotic running time of your algorithm from part (a), if there are n planets and m wormholes? No justification needed.

Problem 15. [Algorithm design: disease] (7 points)

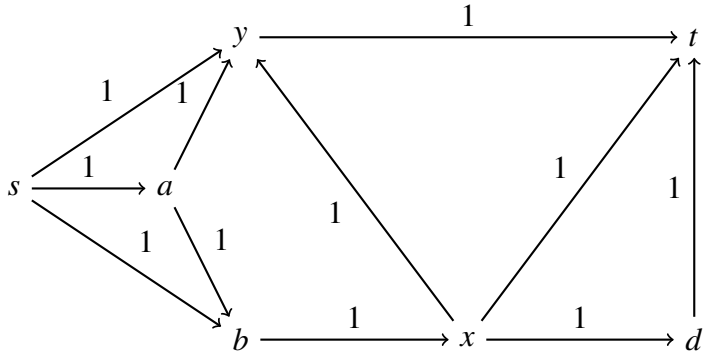
You are a government bureaucrat responsible for preventing some new disease from spreading from planet to planet through the galactic empire and making its way to the galactic capital. You have the power to require medical screening on any travel routes between the planets you want. The goal is to find a minimal subset of routes to ensure any new disease will be detected before it can reach the capital, without subjecting anyone to medical screening twice.

We'll represent this as a graph problem: we have a directed acyclic graph $G = (V, E)$ with a source vertex s and sink vertex t. Call a set U of edges *unavoidable* if every path from s to t has exactly one edge that's in U (at least one edge of the path is in U, but the path doesn't contain more than one edge that's in U). Your job is to design an efficient algorithm to find the smallest unavoidable set U possible.

(Each vertex other than s represents a planet, t is the galactic capital, and s is an artificial vertex with an edge from s to every planet where a new infectious disease might arise.)

(a) Suppose we treat every edge of G as having capacity 1, find a minimum (s,t)-cut $(S, V-S)$ for this graph, and let $T = \{(u,v) \in E : u \in S, v \notin S\}$ be the set of edges that cross this cut in the $s \to t$ direction. Is T guaranteed to be an unavoidable set? Why or why not?

(b) Consider the graph G shown below. Suppose we want to find the minimum (s,t)-cut $(S, V-S)$ in G such that either $x \in S$ or $y \notin S$ (i.e., such that (x,y) doesn't cross the cut in the $t \to s$ direction). This can be done by adding one edge to get a new graph G', giving the new edge an appropriate capacity, and then finding a minimum (s,t)-cut in G'. Draw below the edge you should add to guarantee that this approach will give a correct answer, and label it with its capacity.



(c) We want an efficient algorithm for the following problem: given a dag G, source vertex s, and sink vertex t, find an unavoidable set U containing as few edges as possible.

What algorithm design paradigm is most appropriate for this problem? Circle one of the following.

(a) Divide-and-conquer

(b) Greedy algorithm

(c) Dynamic programming

(d) Linear programming

(e) Reduce to network flow

(f) None of the above

(d) Show pseudocode for the problem in part (c). (You don't need to prove your answer correct.)

(e) What is the asymptotic running time of your algorithm? Justify your answer concisely (a sentence or two should suffice).

Problem 16. [Algorithm design: rockets] (7 points)

You're leading an exploration of space, and it is your job to design a spaceship that will go as far as possible. To do this, you have n available rockets. The ith rocket is capable of accelerating the ship by a rate of a_i meters per second squared and can do this for a total of s_i seconds. The ship is initially at rest (its initial velocity is zero).

Your job is to determine the order in which to fire the rockets. You will choose one rocket, fire it until s_i seconds have elapsed, then immediately switch to another unused rocket. Each rocket can only be used once, and you may ignore the time it takes to switch between rockets.

Given the accelerations and durations of each rocket, design an efficient algorithm to determine the optimal order to fire the rockets, to maximize the total distance traveled.

Physics background: if our velocity is v before we start firing a single rocket whose acceleration is a then, after t seconds, we will have traveled a total $vt + \frac{1}{2}at^2$ meters during that time, and our final velocity will be $v' = v + at$.

(a) Suppose we have two rockets, with $a_1 = 2, s_1 = 1, a_2 = 3, s_2 = 2$. How far do we travel if we fire rocket 1 then rocket 2?

How far do we travel if we fire rocket 2 then rocket 1?

(b) (Part (b) is optional. You can skip it. Only do it if you get stuck on parts (c) and (d) and need a hint.)

Suppose we have two rockets, with $a_1 = 6, s_1 = 3, a_2 = 4, s_2 = 7$. How far do we travel if we fire rocket 1 then rocket 2? Leave your answer as an unevaluated expression.

How far do we travel if we fire rocket 2 then rocket 1? Leave your answer as an unevaluated expression.

Circle the terms in common to both answers. Can they be ignored for our purposes?

(c) What algorithm design paradigm is most appropriate for finding the optimal order to fire the rockets? Circle one of the following.

(a) Divide-and-conquer
(b) Greedy algorithm
(c) Dynamic programming
(d) Linear programming
(e) Reduce to network flow
(f) None of the above

(d) Design an algorithm to determine the optimal order to fire the rockets. Show your pseudocode. (You don't need to provide anything other than your pseudocode.)

(e) What is the running time for your algorithm in part (d)? (No justification needed.)

You are done!