Practice Final Exam (version A) **15-451/651 Algorithms**

Name: _____ **Section:** _____ **Andrew ID:** _____

There are 6 problems, for a total of 180 points. You have 180 minutes. One sheet of notes is allowed. No calculators, cell phones, internet, texting, tweeting, etc.

Problem	1	2	3	4	5	6	total
Score							
Out of	30	30	30	30	30	30	180

---

1. Short-answer questions. (30 pts)

(a) Define (i) a convex function, and (ii) a convex set.

(b) Given an unsorted list of n distinct elements, how quickly can you output the largest $\sqrt{n}$ elements in sorted order?

$\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(\sqrt{n}\log\sqrt{n})$ $\Theta(n)$ $\Theta(n\log n)$

(c) Given three points a, b, and c in the plane, write an expression whose value is 0 if the three points are on the same line, and > 0 if c is on the right side of the line from a to b, and < 0 otherwise.

(Use vector algebra to write your answer concisely. Recall that if $a = (a_x, a_y)$ and $b = (b_x, b_y)$ then $a - b = (a_x - b_x, a_y - b_y)$, $a \times b = a_x b_y - a_y b_x$ (cross product) and $a \cdot b = a_x b_x + a_y b_y$ (dot product).)

(d) Given two arrays `A` and `B` both of length n, the convolution operation `C = A*B` is defined as

`C[i]` = _____ for all $i = 0, 1, \ldots$ _____.

(e) Suppose you have the option of running one of two algorithms to solve a given problem. On an input of size n (assume n is a power of 4 and $T(1) = 1$):

- Algorithm A breaks it into 3 pieces of size $n/4$, recursively solves each piece, and then combines the solutions in time $n^{1.5}$.
- Algorithm B breaks it into 4 pieces of size $n/4$, recursively solves each piece, and then combines the solutions in time n.

The faster algorithm is algorithm _____ with running time $\Theta($_____$)$.

(f) Build a suffix tree for the string `ababcab$`.

(g) Take the following LP

$$\begin{aligned} &\text{minimize } 7x_1 + 51x_2 \\ &\text{subject to } 5x_1 - 2x_2 \geq 18 \\ &\qquad\qquad 3x_1 + 8x_2 \geq 1 \\ &\qquad\qquad x_1 \geq 0, \quad x_2 \geq 0 \end{aligned}$$

And write down its LP dual:

$$\begin{aligned} &\text{\_\_imize} \qquad \text{\_\_\_} y_1 + \text{\_\_\_} y_2 \\ &\text{subject to } \text{\_\_\_} y_1 + \text{\_\_\_} y_2 \ \square \ \text{\_\_\_} \\ &\qquad\qquad \text{\_\_\_} y_1 + \text{\_\_\_} y_2 \ \square \ \text{\_\_\_} \\ &\qquad\qquad y_1 \geq 0, \quad y_2 \geq 0 \end{aligned}$$

Each blank except the first gets a number, the boxes get either $\leq$ or $\geq$ or $=$.

2. You be the TA (30 pts)

Having finished 15-451 you are now being put to work as a TA. Below is a series of purported proofs. For the first three parts, either say "correct" or "wrong", and if wrong, *explain the fundamental bug in the argument*. For the final part, give a counterexample as explained.

(a) Claim: The following matrix game has no minimax optimal strategy for the row player (numbers represent payoffs to the row player, who wants to maximize).

6	3
2	1

Proof: To solve for the minimax optimal strategy, we let p be the probability on row 1, let $1 - p$ be the probability on row 2, and solve for p such that the payoff if the column player plays column 1 is the same as the payoff if the column player plays column 2. Specifically, $6p + 2(1 - p) = 3p + (1 - p)$ which solves to $p = -1/2$. Since one cannot have negative probabilities, there is no minimax optimal strategy.

(b) Claim: The problem "Given a graph G, does it require *more* than 3 colors to color correctly" is in **NP**.

Proof: Given a YES-instance, a proof that it requires more than 3 colors is just a listing of all possible 3-colorings of the vertices, along with for each of these identifying an edge that has both endpoints the same color. This proof can be easily checked in time linear in the length of the proof. Therefore, this problem is in **NP**.

(c) Claim: Let h_0 be a hash function that maps everything to 0, and let h_1 be a hash function that maps everything to 1. Then $H = \{h_0, h_1\}$ is a universal hash family from U to $\{0, 1\}$.

Proof: H is universal because given any $x \in U$, there is a 50% chance that x will hash to 0 and a 50% chance that x will hash to 1.

(d) A student claims to have simplified the Ford-Fulkerson max-flow algorithm. Recall that FF repeats the following 2 operations:

1. Find an arbitrary path P in the residual graph from s to t that can support some $k > 0$ units of flow (halt if none exists). Push k units of flow on P.
2. Create a new residual graph by subtracting k from the capacity of all forward edges (u, v) in P and adding k to the capacity of all back edges (v, u) (viewing edges that do not exist as edges of capacity 0).

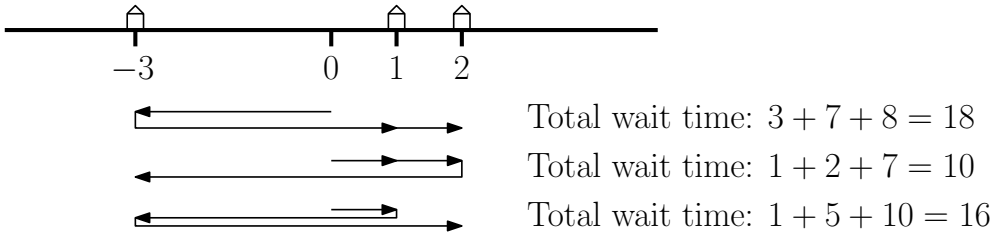
The student simplifies Step 2 to only reduce the capacity of forward edges by k but *not* add k to the back edges. The student claims this is still guaranteed to find a maximum flow. Give a counterexample to this claim. Specifically, show a directed graph with capacities on edges, along with a sequence of paths P chosen that cause the modified algorithm to fail to find a maximum flow.

3. You've got mail! (30 pts)

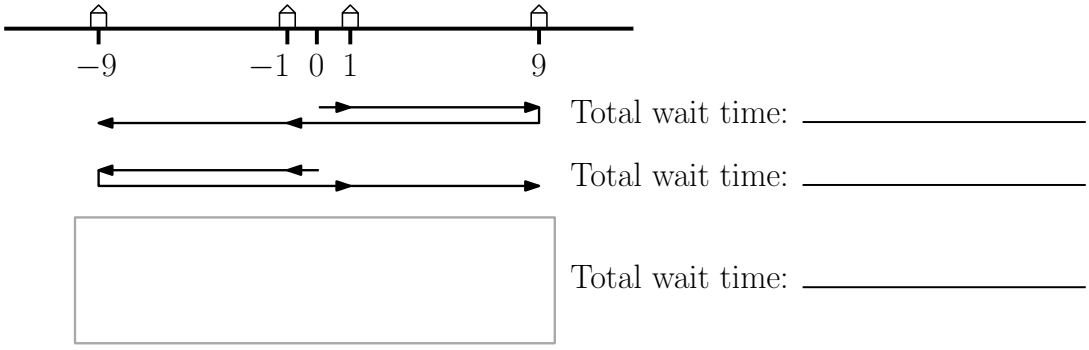
Picov Andropov is the mailman in a small town. The town consists of n houses on a single long street. Say the houses are at integer locations $h_1, h_2, \ldots, h_n$ on the real line, sorted so that $h_1 < h_2 \ldots < h_n$. The postoffice is at the origin, and assume $h_1 \leq 0 \leq h_n$.

Each day, Picov brings mail to all the houses. He enjoys his job because he can easily find the fastest route to do this: he first goes left from 0 to h_1, delivering mail on the way, then goes back to 0; then he goes right from 0 to h_n, delivering mail along the way, then returns back to 0. The total distance traveled is just $2(h_n - h_1)$. In other words, the Traveling Salesman Problem on a line is easy.

Now, Picov's boss comes to him and says that instead of minimizing *his* travel time, he should be minimizing the total time that the town's *residents* need to wait for their mail. Formally, if Picov starts at time 0 and travels at unit speed, then the waiting time of house i is the time at which Picov first reaches the house. Here's an example with 3 houses at locations $-3, 1, 2$; the second route is optimal.



(a) On the example below, what is the total wait time for the two simplest kinds of routes? What is an optimal route minimizing total wait time? (Draw this route in the box and calculate the total wait time.)



(b) We can solve for Picov's optimal strategy using Dynamic Programming. Define the *cost* of a proposed solution is the total waiting time in this solution. Also, assume there is a house i^* located at the origin (i.e., $h_{i^*} = 0$) since this will not change the cost. Finally, for $i < j$ define $n_{ij} = n - (j - i)$.

For any $1 \leq i < j \leq n$, define

- $\mathsf{cost}_L(i, j)$ to be the cost of the optimal solution to a smaller problem in which all houses h_k for $i < k < j$ have been removed, and Picov begins at location h_i. (Note there are n_{ij} unvisited houses, viewing i as just visited.)
- $\mathsf{cost}_R(i, j)$ to be the cost of the optimal solution to a smaller problem in which all houses h_k for $i < k < j$ have been removed, and Picov begins at location h_j. (Again there are n_{ij} unvisited houses, viewing j as just visited.)

Fill in the blanks in the following recursive expression for $cost_L$ and $cost_R$ in terms of the subproblems where *fewer houses remain* (i.e., in terms of subproblems for which $j - i$ is greater).

$\mathsf{cost}_L(i, j) =$ _____

$\mathsf{cost}_R(i, j) =$ _____

Base case: $\mathsf{cost}_L(1, n) = \mathsf{cost}_R(1, n) =$ _____.

(c) Implementing this recursion using Dynamic Programming, what is the overall running time as a function of n?

Running time: _____

4. Set Packing (30 pts)

Consider the SET PACKING decision problem: *given a collection of subsets $S_1, S_2, \ldots, S_n$ of $\{1, \ldots, m\}$ and a number K, do there exist K disjoint sets in the collection?*

Show that SET PACKING is *NP*-complete.

SET PACKING is in _____ because a verifier can efficiently check a proposed solution to see if the sets chosen are indeed disjoint. If I is a _____-instance, such a solution exists and if I is a _____-instance then no such solution exists.

To prove that SET PACKING is **NP**-hard, we can reduce from

$Q =$ _____ to $Q' =$ _____.

Specifically, given an instance I of Q we create an instance $f(I)$ of Q' as follows:

_____

_____

_____

This reduction takes _____ time.

We want to show that

I is a _____-instance of $Q \iff f(I)$ is a YES-instance of Q'.

If I is a _____-instance of Q, then $f(I)$ is a YES-instance of Q' because

_____

_____

Similarly, if $f(I)$ is a YES-instance of Q', then I is a _____-instance of Q because

_____

_____

QED.

5. A Matter of Degree (30 pts)

You are the chief engineer for Graphs-R-Us, a company that makes graphs to meet all sorts of specifications.

(a) A client comes in and says he needs a 4-node *directed graph* in which the nodes have the following in-degrees and out-degrees:

$$\begin{array}{llll} d_{1,in} = 0, & d_{2,in} = 1, & d_{3,in} = 2, & d_{4,in} = 3 \\ d_{1,out} = 2, & d_{2,out} = 2, & d_{3,out} = 1, & d_{4,out} = 1 \end{array}$$

Is there a directed graph, with no multi-edges or self loops, that meets this specification? If so, what is it? If not, give a brief reason why.

(b) What about a 3-node graph (again with no multi-edges or self loops) with these in-degrees and out-degrees?

$$\begin{array}{lll} d_{1,in} = 2, & d_{2,in} = 2, & d_{3,in} = 1 \\ d_{1,out} = 2, & d_{2,out} = 2, & d_{3,out} = 1 \end{array}$$

(c) This type of specification, in which the in-degrees and out-degrees of each node are given, is called a *degree sequence*. The question above is asking whether a given degree sequence is *realizable* — that is, whether there exists a directed graph having those degrees.

Find an efficient algorithm that, given a degree sequence, will determine whether this sequence is realizable, and if so will produce a directed graph with those degrees. The graph should not have any self-loops, and should not have any multi-edges (i.e., for each directed pair (i, j) there can be at most one edge from i to j, though it is fine if there is also an edge from j to i).

Hint: max flows. (In an actual exam we'll give you another blank sheet to write the solution — this problem would be spread over two pages.)

6. Finding the Triangle of Largest Area (30 pts)

You've given a set S of n points in the plane. Give an algorithm for computing the triangle of largest area whose vertices are among the points of S.

Let $T(n)$ denote the running time of your algorithm. If $T(n) = \Theta(n^3)$ you will get 2 (out of 30 points) — and the number of points will increase (possibly non-linearly) as $T(n)$ decreases, until you will get full points for an algorithm with $T(n) = O(n^2)$.