Midterm Exam 2

- Do not open this exam booklet until you are directed to do so. Read all the instructions on this page.

- When the exam begins, write your name on every page of this exam booklet.

- This exam contains 4 problems, some with multiple parts. You have 120 minutes to earn 70 points.

- This exam booklet contains 8 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your exam.

- This exam is closed book. You may use one handwritten $8\frac{1}{2} \times 11$ or A4 crib sheet. No calculators or programmable devices are permitted.

- Write your solutions in the space provided. If you need more space, you may use the scratch sheets at the back of the exam.

- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.

- Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.

- You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.

---

Name: _______________________________ ID: __________

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>15</td>
<td>70</td>
</tr>
<tr>
<td>Grade</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (Miscellaneous).

(a) Your friend has found a recursive algorithm for sorting an array of length $n$ that uses “only” $\Theta(n \log n)$ work at the top level plus three recursive calls on subarrays of length $\lceil n/3 \rceil$. What is the running time of your friend’s algorithm?

Answer:

Justification:

(b) Suppose you are given a directed graph $G = (V, E)$ with lengths $\ell : E \to \mathbb{R}$ for each edge, a source node $s$ and a list of candidate distances $d(v)$ (where $d(v)$ is the claimed to be the length of the shortest path from $s$ to $v$ in $G$). What is the running time of the fastest algorithm which checks whether the distances $d$ are correct?

Answer:

Describe the algorithm briefly:
(c) Suppose you are given a directed graph $G = (V, E)$ with capacities $c : E \rightarrow \mathbb{R}$ for each edge, and a flow $f : E \rightarrow \mathbb{R}$. What is the running time of the fastest algorithm which checks whether $f$ is a maximum flow?

**Answer:**

**Describe the algorithm briefly:**

(d) (Extra credit) Recall that in a directed graph $G$, a set of paths $p_1, \ldots, p_k$ from vertex $s$ to vertex $t$ is *edge-disjoint* if each edge in the graph is used by at most one of the paths (it could be that the paths share some vertices, but they have to use different edges).

The following fact is proved in the book: if we set the capacities of all edges to be 1, then the value of the maximum $s - t$ flow in $G$ is exactly the size of the largest collection of edge-disjoint paths from $s$ to $t$.

We say that $s$ is $k$-connected to $t$ if there are $k$ or more edge-disjoint paths from $s$ to $t$. Prove that if $u$ is $k$-connected to $v$ and $v$ is $k$-connected to $w$, then $u$ is $k$-connected to $w$. 

3
Problem 2. (Oscillating sequences) A subsequence \( i_1 < i_2 < \cdots < i_k \) of an array of numbers is oscillating if \( A[i_j] \leq A[i_{j+1}] \) when \( j \) is odd and \( A[i_j] \geq A[i_{j+1}] \) when \( j \) is even. Use dynamic programming to find a polynomial time algorithm which takes an array of numbers and finds the length of the longest oscillating subsequence. Faster (correct) algorithms are worth slightly more points than lower ones.

Clearly label the different parts of your solution as follows:

(a) Define the subproblems your algorithm will compute.
(b) Explain, in English, what the subproblems mean.
(c) Give a recursive algorithm (or formula, if it is easier) for computing the answer to a subproblem in terms of smaller subproblems.
(d) Explain why your recursive formulation is correct.
(e) Give pseudocode for your algorithm.
(f) Analyze its space usage and running time.

(You can use the next page if you need more space.)
(Extra space for answering question 2)
Problem 3. (Dining Problem) Several families are having a dinner party. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Show how to find a seating arrangement that meets this objective (or prove that no such arrangement exists) by reducing this problem to MAXIMUM FLOW. Assume that the dinner contingent has $p$ families, that the $i$th family has $a_i$ members, that $q$ tables are available and up to $b_j$ people can be seated at table $j$. Use the following to organize your answer:

(a) Explain how to construct a graph $G$ from the inputs $a_1, ..., a_p$ and $b_1, ..., b_q$.

(b) Explain how to use the answer to the maximum flow problem to construct a seating assignment.

(c) Explain why your algorithm is correct.

(d) Analyze the running time of the algorithm.
Problem 4. (Earthmover distance.) Currently, there are \( n \) piles of dirt scattered over a large construction site. Pile number \( i \) weighs \( w_i \) tons and is at position \( p_i \) in the plane. The dirt in these piles has to be moved into \( k \) new piles at positions \( q_j \) (for \( j = 1, \ldots, k \)) with weights \( v_j \) (also in tons).

The good news: You have a bulldozer.

The bad news: You have to figure out the cheapest way to move the dirt.

For any non-negative real number \( t \), the cost of moving \( t \) tons of dirt from position \( p \) to positions \( q \) is \( t \times \text{dist}(p, q) \) where \( \text{dist}(p, q) \) is the distance from \( p \) to \( q \). Note that \( t \) need not be an integer.

Your goal is to minimize the total cost of moving the dirt from its current locations to the new ones. The input consists of the points \( p_1, \ldots, p_n, q_1, \ldots, q_k \in \mathbb{R}^2 \) and weights \( w_1, \ldots, w_n, v_1, \ldots, v_k \in \mathbb{N} \). You may ignore the cost of driving around the construction site when not carrying dirt; you only need to account for the cost of moving dirt from pile to pile. Also, because no nuclear reactions will be involved, you may assume that mass is conserved, that is, \( \sum_{i=1}^{n} w_i = \sum_{j=1}^{k} v_j \).

Note: In what follows, the different parts of the problem are independent: you do not need to solve one to solve the others.

(a) Give a linear program that captures this problem.

Variables:

Constraints:

Objective function to be minimized:

(b) Prove or disprove the following conjecture: In an optimal solution, the paths taken by the bulldozer when moving dirt will never cross. [Hint: Think of an exchange argument.]
(c) Consider the following greedy algorithm: Repeat the following basic step until all the dirt has been moved: find the closest pair \((p_i, q_j)\) such that \(p_i\) still has dirt and \(q_j\) still needs dirt, and move a ton of dirt from \(p_i\) to \(q_j\). Is this greedy algorithm correct? Justify your answer.

(d) (Extra credit) Prove or disprove: if the weights of the initial and final piles are integers, there exists an optimal solution to the problem in which the bulldozer only moves integer amounts of dirt.