Final Exam

- Do not open this exam booklet until 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.
- You have 180 minutes to earn 180 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed three 1-page cheat sheets.** No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a clear description in English will suffice. Pseudo-code is not required unless asked for.
- **Pay close attention to the instructions for each problem.** Depending on the problem, partial credit may be awarded for incomplete answers.
- If you need to, make any reasonable assumptions and state those assumptions.
- **Blank answers will receive approximately 25% of the credit for the problem.** If your answer shows no understanding of the problem, you will receive less than 25%. So, if you have no idea how to attack a problem, you should leave it blank.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parts</th>
<th>Points</th>
<th>Grade</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>180</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name: ____________________________

Circle your recitation:

- R01 Gurtej Ashwin 10AM
- R02 Gurtej Ashwin 11AM
- R03 Jennifer Quanquan 12PM
- R04 Jennifer Quanquan 1PM
- R05 Arvind Vlad 2PM
- R06 Arvind Vlad 3PM
- R07 Justin Sherwin 11AM
- R08 Justin Sherwin 12PM
Problem 1. (13 parts) [39 points] True/False and Justify

Circle True or False for each statement below, and provide a brief justification for your answer. Your justification is worth more points than your true/false designation.

(a) T F [3 points] If \( f(n) = O(g(n)) \) and \( g(n) = O(h(n)) \), then \( f(n) = O(h(n)) \).

(b) T F [3 points] The Master Theorem can be used to solve the recurrence

\[
T(n) = 2T(n/2) + \frac{n}{\log n}.
\]

(c) T F [3 points] It is possible to sort any \( n \) integers between 0 and \( n^{100} \) in \( O(n) \) time.

(d) T F [3 points] Given \( n/2 \) sorted elements \( A[0 : n/2] \) and \( n/2 \) sorted elements \( A[n/2 : n] \) in a single array \( A[0 : n] \) (as in merge sort), the merge algorithm presented in lecture sorts \( A \) by moving the elements in \( A \) while using \( O(1) \) additional space (‘in-place’).
(e) T F [3 points] Quadratic probing (the open addressing scheme using hash/collision function \( h(k, i) = h(k) + c_1 i + c_2 i^2 \) for constants \( c_1, c_2 \)) satisfies the \textit{uniform hashing assumption}: the probe sequence of each key \( k \) is equally likely to be any of the \( m! \) permutations of the slots \( \{0, 1, \ldots, m - 1\} \).

(f) T F [3 points] In a hash table maintained with cuckoo hashing, the worst-case running time to search for an element is \( O(1) \).

(g) T F [3 points] Consider a hash table with open addressing and linear probing. Any table resulting from insertions and deletions into an empty table can also be created just by inserting into an empty table.

(h) T F [3 points] Given a directed graph \( G = (V, E) \), we can (using algorithms presented in 6.006) determine whether \( G \) has a directed cycle in \( O(V + E) \) time.
(i) T F [3 points] Given a weighted directed graph $G = (V, E, w)$ with no negative-weight edges, we can (using algorithms presented in 6.006) compute the shortest-path weight from node $s$ to node $t$ in $O(V + E)$ time.

(j) T F [3 points] In a weighted graph $G = (V, E, w)$ with no cycles and all edge weights negative, we can run Dijkstra's algorithm on some graph $G'$ to find the maximum possible weight of a path from node $s$ to node $t$ in $G$.

(k) T F [3 points] If you are told that SAT (Boolean satisfiability problem) is NP-complete, and you find a polynomial-time reduction from SAT to Hamiltonian Path, then you can conclude that Hamiltonian path is NP-hard.

(l) T F [3 points] Given two integers $a, b$ where $a < m$, we can (using algorithms presented in 6.006) compute $a^b \mod m$ in $O((\lg m)^{\lg 3} \lg b)$ bit operations.
(m) T F [3 points] Given an array $A$ of $n$ nonzero distinct integers, we can compute in $O(n)$ time a **stable sign sort**, that is, a permutation $A'$ of $A$ such that all negative integers appear before all positive integers, and all integers of the same sign appear in the same order as they did in $A$. 
Problem 2. AVL Practice [5 points]

Suppose you have the AVL tree shown below. Draw the AVL tree resulting from inserting 11 into the tree. There is no need to show intermediate work: we will be looking only at the final tree configuration.
At one time, Prof. Dynamic thought that every dynamic program could be viewed as finding the shortest path from some node $s$ to some node $t$ in the directed acyclic graph (DAG) where the vertices represent subproblems and edges represent dependencies between subproblems. But this is not true.

Which of the following dynamic programs from lecture can be solved by computing shortest paths in the subproblem dependency DAG (with appropriately weighted edges)? Circle “can” or “cannot” in each case. You do not need to justify your answer.

(a) Fibonacci numbers  
can  cannot

(b) text justification  
can  cannot

(c) parenthesizing / matrix multiplication  
can  cannot

(d) edit distance  
can  cannot

(e) knapsack  
can  cannot

(f) piano/guitar fingering  
can  cannot
Problem 4. Chaining vs. Linear Probing [12 points]

Profs. Sauron and Saruman disagree on how to best organize a hash table storing their records on the \( n \) people of Middle Earth. They both construct a table with \( m = 2n \) slots, and use the hash function \( h(k) = k \mod m \). Prof. Sauron insists that chaining is a better conflict resolution strategy, while Prof. Saruman insists that linear probing is better.

For each of the following sequences, compute the total asymptotic running time (using \( \Theta \) notation) for inserting all \( n \) items into each of the two hash tables. For chaining, assume that inserting into a chain of length \( k \) costs \( \Theta(k) \) time (to check for duplicates).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Chaining</th>
<th>Linear Probing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3, \ldots, n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( m, 2m, 3m, \ldots, nm )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1, 2, \ldots, ( \frac{n}{2} ), ( m + 1 ), ( m + 2 ), \ldots, ( m + \frac{n}{2} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 1, m + 1, 2, m + 2, \ldots, \frac{n}{2}, m + \frac{n}{2} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 5. Failure to Launch (2 parts) [12 points]

Suppose you have a hash table with \( n \) items in \( m \) slots maintained with open addressing with the uniform hashing assumption: the probe sequence of a key is equally likely to be any of the \( m! \) permutations of the slots \( \{0, 1, \ldots, m - 1\} \).

(a) [6 points] If you insert one more element, what is the probability that the first probe fails, i.e., finds an occupied slot?

(b) [6 points] What is the probability that the second probe finds an empty slot \textbf{given} that the first probe failed?
Problem 6. I’m Hungry [16 points]

You are the mayor of the infinitely large city Aleph Town, a 2D plane with infinitely many restaurants: for all integers $i, j$, there is a restaurant at coordinates $(10^i, 10^j)$. In addition, there are $n$ houses at integer points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Each restaurant $(10^i, 10^j)$ serves customers from all houses $(x_i, y_i)$ within Manhattan distance 10: $|10^i - x_i| + |10^j - y_i| \leq 10$. Describe and analyze an $O(n)$-time algorithm to find a restaurant serving the most houses.

In the example above, the ovals represent the restaurants in this portion of the grid, the two squares represent houses, and the numbers on edges denote the Manhattan distances between the corresponding house–restaurant pairs. In this example, Restaurant F serves the most houses, as it is the only restaurant serving both houses on the grid.
Problem 7. Ternary Search (2 parts) [10 points]

Consider the following algorithm for searching for an element \( x \) in a sorted array \( A[i : k] \). Instead of breaking the array in half and searching in one of the halves, like binary search, it breaks the array into thirds and searches in one of the thirds. (For clarity, we assume that \( n = 3^\ell \) for an integer \( \ell = \log_3 n \).)

**Ternary-Search** \((x, A, i, j)\)

1. // Assumption: \( A[i] \leq x < A[j] \)
2. if \( j - i \leq 1 \):
   3. \textbf{return} \( i \)
4. \( p = \frac{2}{3} i + \frac{1}{3} j \)
5. \( q = \frac{1}{3} i + \frac{2}{3} j \)
6. if \( x < A[p] \):
   7. \textbf{return} Ternary-Search\((x, A, i, p)\)
8. elseif \( A[p] \leq x < A[q] \):
   9. \textbf{return} Ternary-Search\((x, A, p, q)\)
10. elseif \( x \geq A[q] \):
11. \textbf{return} Ternary-Search\((x, A, q, j)\)

(a) [5 points] Write down a recurrence for \( T(n) \), the exact number of calls to Ternary-Search when starting with Ternary-Search\((x, A, 0, n)\) (counting the initial call).

(b) [5 points] Solve for \( T(n) \) exactly. (For partial credit, write the solution using asymptotic \( \Theta \) notation.)
Problem 8. Blind Mountain Climbing with a Transporter [12 points]

Suppose you are given an array $A$ of $n$ distinct numbers $A[0], A[1], \ldots, A[n-1]$, and define $A[-1] = A[n] = -\infty$. Assume you must work in the comparison model, i.e., all you are allowed to do with the numbers is compare them (with <, >, =). Prove that, in the worst case, it takes $\Omega(\lg n)$ comparisons to find a peak, i.e., an index $i$ such that $A[i-1] < A[i] > A[i+1]$. 
Problem 9. Binary Search as a DP (2 parts) [17 points]

Prof. Bottoms-Up has discovered a new way to do binary search: dynamic programming! To search for a number $x$ in a sorted array $A[0 : n]$, he uses the following dynamic program:

1. **Subproblems:** For all $0 \leq i \leq j \leq n$, $BS(i, j) =$ the index of $x$ (or its predecessor) in the interval $A[i : j]$.

2. **Guessing:** None.

3. **Recurrence:**

   \[
   \begin{align*}
   BS(i, j) &= 1 \quad m = \left\lceil \frac{i+j}{2} \right\rceil. \\
   &= 2 \quad \text{if } j - i \leq 1; \\
   &= 3 \quad \text{return } i \\
   &= 4 \quad \text{elseif } x < A[m]; \\
   &= 5 \quad \text{return } BS(i, m) \\
   &= 6 \quad \text{else return } BS(m, j)
   \end{align*}
   \]

4. **Ordering Loop:**

   \[
   \begin{align*}
   &1 \quad \text{for } d = 1, 2, \ldots, n: \\
   &2 \quad \text{for } i = 0, 1, \ldots, n - d: \\
   &3 \quad \text{solve } BS(i, i + d)
   \end{align*}
   \]

5. **Original problem:** $BS(0, n)$

(a) [7 points] If Prof. Bottoms-Up implements this dynamic program in a bottom-up style (expanding the Ordering Loop with the Recurrence, but replacing recursive calls with table lookups), what would be the resulting asymptotic running time?
(b) [10 points] If Prof. Bottoms-Up implements this dynamic program in a top-down recursive style (implementing the Recurrence as a recursive algorithm with memoization), what would be the resulting asymptotic running time?
Problem 10. [15 points] Must-See T.V.

Suppose you are given a weighted directed graph $G = (V, E, w)$ with nonnegative edge weights, and you are given a source node $s \in V$ and a destination node $t \in V$. In addition, there are two red edges $e_1, e_2$ that must be traversed at least once each, although the order doesn’t matter.

Give an efficient algorithm to find the weight of a shortest path from $s$ to $t$ that contains both of the red edges (or $\infty$ if there is no such path).
Problem 11. Flying Home for Winter Break  (3 parts)  [30 points]

You’ve just acquired the weighted graph $G = (V, E, w)$ representing the airline route network: vertices represent cities, edges represent flight routes, and edge weights represent ticket prices. (Sadly, all edge weights are positive.) You’d like to travel via a directed path from $s = \text{Boston} \in V$ to $t = \text{Springfield} \in V$ of minimum total price, but you’re only willing to take 3 flights. Your goal is to compute, in $O(V + E)$ time, a minimum-cost path from $s$ to $t$ among paths using at most 3 edges. (Assume that there is such a path.)

(a) [10 points] Your first approach is to run 3 iterations of Bellman-Ford. That is, you start with distance estimates $d[v] = \infty$ except for $d[s] = 0$; then you relax all edges in $E$; then you relax all edges in $E$; and then you relax all edges in $E$. Give a counterexample where $d[t]$ won’t then contain the weight of a minimum-cost path from $s$ to $t$ using at most 3 edges.

(b) [10 points] Your second approach is to run BFS to locate all vertices reachable from $s$ by a path of at most 3 edges, remove all other vertices and their incident edges, and then run Dijkstra on the resulting subgraph. Give a counterexample where $d[t]$ won’t then contain the weight of a minimum-cost path from $s$ to $t$ using at most 3 edges.
(c) [10 points] Give an efficient algorithm to compute the weight of a minimum-cost path from s to t using at most 3 edges. For full credit, your algorithm should run in $O(V + E)$ time.