Spring 2006

CS 462: Final Exam

Problem 0 (2 point; your total for the other problems is multiplied by half of your score for this question):

(a) Write your name at the top of this cover sheet. Staple all submitted pages together. You may insert extra pages if you need more space. Show all of your work.

(b) Sign your name under the declaration below.

I agree to be bound by the Loyola College Honor Code: “The Honor Code states that all students of the Loyola Community have been equally entrusted by their peers to conduct themselves honestly on all academic assignments. The students of this College understand that having collective and individual responsibility for the ethical welfare of their peers exemplifies a commitment to the community. Students who submit materials that are the products of their own minds demonstrate respect for themselves and the community in which they study. All outside resources or information should be clearly acknowledged. If there is any doubt or question regarding the use and documentation of outside sources for academic assignments, your instructor should be consulted. Any violations of the Honor Code will be handled by the Honor Council.”

I have not used any outside resources except the following:

(a) the textbook;
(b) my notes;
(c) material posted on the course web page;
(d) my graded homeworks, exams, and projects;
(e) a calculator;
(f) computer programs I wrote that do not access forbidden resources; and
(g) consultations with the instructor.

Signed,
Problem 1 (8 points): Draw the max-heap that results from running the BUILD-HEAP algorithm on an array that initially contains \(\{1, 4, 6, 5, 2, 3, 8, 7, 9\}\). Show intermediate results for partial credit.
Problem 2 (10 points): If the median is selected as the pivot for the partition stage of QUICKSORT, the worst case is $O(n \log n)$ if the inputs are distinct. What happens if the elements are not distinct? If the worst case is not $O(n \log n)$ when the inputs are not distinct, describe how PARTITION can be modified to improve the worst case in such cases.

Problem 3 (8 points): Give a tight asymptotic bound on $T(n)$ for each of the following recurrences. Assume that $T(0) = T(1) = 1$ in all cases and that the recurrences hold for $n \geq 2$.

(a) $T(n) = 2T(\frac{n}{2}) + \sqrt{n}$

(b) $T(n) = 4T(\frac{n}{2}) + n^3$

(c) $T(n) = 8T(\frac{n}{2}) + 5n^3 + n^2$

(d) $T(n) = 2T(\frac{n}{2}) + \log n$
Problem 4 (8 points): Strain’s algorithm for tree slicing uses a Flattened Fibonacci Partition (FFP). An FFP supports operations turn, push, and pull. Suppose there are two implementations of FFPs, one due to Kopf and one to Cerquetti. Their running times for each FFP operation are given below.

<table>
<thead>
<tr>
<th>operation</th>
<th>Kopf</th>
<th>Cerquetti</th>
</tr>
</thead>
<tbody>
<tr>
<td>turn</td>
<td>$O(\log V)$</td>
<td>$O(V)$</td>
</tr>
<tr>
<td>push</td>
<td>$O(V \log V)$</td>
<td>$O(\log V)$</td>
</tr>
<tr>
<td>pull</td>
<td>$O(\log V)$</td>
<td>$O(V^{3/2})$</td>
</tr>
</tbody>
</table>

Strain’s algorithm requires $O(E + V^{3/2})$ calls to turn, $O(E)$ calls to push, and $O(V^2)$ calls to pull.

(a) For a sparse graph ($E = O(V)$), does the Kopf implementation yield a better asymptotic running time for Strain’s algorithm or does the Cerquetti implementation?

(b) For a dense graph ($E = O(V^2)$), which is better?
Problem 5 (10 points): Define $F(n)$ by $F(1) = 1$, $F(2) = 3$, and for $n > 2$

$$F(n) = \min(\sum_{a=0}^{b} (F(a) + F(b)), \sum_{x=1}^{n-1} (2 + \frac{x}{n} \cdot F(x) + \frac{n-x}{n} \cdot F(n-x)))$$

Write an efficient algorithm to compute $F(n)$ and give its asymptotic running time.
Problem 6 (12 points):

(a) Show how to extend the 2-line factory scheduling problem to n lines. Assume that transfers can only happen between adjacent lines (so to get from line 1 to line 3 one could transfer from 1 to 2, go through a station on line 2, and then transfer from 2 to 3).

(b) What is the fastest way through a 3-line factory with the following station times and exit times? Assume all transfer times are 1.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th></th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Problem 7 (12 points):

(a) Write an efficient algorithm that, given a directed acyclic graph $G = (V, E)$ where all the vertices are colored red, blue, or green, identifies all the red vertices that can reach a green vertex. Give the asymptotic efficiency of your algorithm.

(b) Write an efficient algorithm that, given a directed graph (not necessarily acyclic) with vertices colored as above, identifies all the red vertices that can reach a green vertex along a path that contains no other red vertices. Give the asymptotic efficiency of your algorithm.
Problem 8 (12 points): A system of bus lines can be modeled as a weighted, directed graph where the vertices are the transfer points, the edges are the routes the busses take between those transfer points, and the weights are the times it takes to travel between the transfer points. In addition, each edge is labelled with the starting time and frequency of bus service along that edge (so if an edge is labelled with (10, 30) then busses leave every 30 minutes starting at 12:10). Show how to modify Dijkstra’s algorithm to compute the fastest way to get between two points given the time the trip is going to start (hint: let $d[v]$ be the earliest time one could get to point $v$).
Problem 9 (10 points): ALMOST-HP is the problem of, given an undirected graph with $n$ vertices, determining if that graph has a simple path that visits at least $n - 1$ vertices. Give a complete proof that ALMOST-HP is NP-complete. You may assume that HP is NP-complete.
Problem 10 (8 points):

(a) Show the result of running DFS on the following graph. Show the starting and finishing times for each vertex and shade the edges that correspond to the depth first search forest. Start the outer loop at vertex 1 and proceed down the page. When iterating through adjacent vertices, go through the outgoing edges clockwise starting from straight up.

(b) Show the result of running TOPOLOGICAL-SORT on the graph from (a).