15-750-S09: Final

May 4, 2009

Name:

Email:

Instructions

• Fill in the box above with your name and e-mail address. Do it, now!

• This exam is open book and open note. That is, you may use two books (Kozen and CLRS), any handouts from class, and your notes.

• Clearly mark your answers in the allocated space. If need be, use the back of a page for scratch space. If you have made a mess, cross out the invalid parts of your solution, and circle the ones that should be graded.

• Scan the test first to make sure that none of the 12 pages are missing. The problems are of varying difficulty, you might wish to pick off the easy ones first.

• With each problem requiring you to present an algorithm make sure include the following three bullets: Algorithm: Correctness: and Timing:

• You have 180 minutes. Good luck!

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td>160</td>
</tr>
</tbody>
</table>
Problem 1: Road Trip (20 pts.)

Professor Miller and the class are going on a road trip to San Francisco immediately after the (grueling) 15-750 final. To plan the trip, you have laid out a map of the U.S., and marked all the places you think might be interesting to visit along the way. However, the requirements are:

1. Each stop on the trip must be closer to SF than the previous stop.
2. The total length of the trip cannot be longer than \( D \).

Furthermore, we want to visit the most places subject to these conditions. As a first step, the course staff sorted the locations by distance to SF and labels them 1, 2, \ldots, n where 1 is SF and n is Pittsburgh. Let \( d(i, j) \) be the distance from location \( i \) to location \( j \).

Let \( M \) be a matrix (to be filled out) where \( M[i, k] \) is the length of the shortest legal path from location \( i \) to location 1 (San Francisco) that visits at least \( k \) places along the way, including \( i \) and SF. “Legal” just means that each new stop must have a lower index than the previous one. Let’s say that \( M[i, k] = \infty \) if it is not possible to visit that many places on the way.

1. Write a recurrence for \( M[i, k] \). Make sure you include the base cases.

2. What is the runtime and space usage of your recurrence when viewed as an algorithm using memoization?

3. How would you use \( M \) to figure out the number of locations we can visit?

4. How would you use \( M \) to actually produce the optimal path?
Problem 2: Backward Analysis (20 pts.)

Here’s a game that is played on a circular array of \( n \) cells. Initially all cells of the array are unmarked. A random unmarked cell is picked. The cell is marked. The cost incurred is the number of unmarked cells in the block of unmarked cells clockwise from the cell being marked. (In other words, we run around the array clockwise until we run into a marked cell.)

For example, say \( n = 4 \). Let the cells be labeled 0, 1, 2, 3 in clockwise order. Suppose we happen to mark the cells in this order: 0, 2, 1, 3. The cost of each of these marking steps is 3, 1, 0, 0, for a total of 4. On the other hand, if the order was 0, 1, 2, 3, then the costs would be 3, 2, 1, 0 respectively, for a total of 6.

Our goal is to compute the expected time to mark all \( n \) cells. This can be done using backwards analysis. Suppose that there are \( i \) marks, and we’re about to remove one of them. Let \( E(i) \) be the expected cost of removing one of these marks.

A. Write down the exact value of \( E(i) \) and a one sentence explanation why.

\[
E(i) = \text{.................................}
\]

B. In terms of \( E(i) \), write the expected cost of the whole process and a one sentence why:

\[
\text{.................................}
\]

C. The total expected cost is

\[
\text{.................................} = O(\text{.................................})
\]
D. Write the total expected cost

\[ E = \ldots + \Theta(n) \]

(Inside the formula should be a polynomial in \( n \) and \( \ln(n) \)).
Problem 3: Short Questions (40 pts.)

Give a **brief justification or counter-example** for your answers to each of the following problems.

(a) (5 pts.) [True or False] In the strongly connected component algorithm presented in class, the stack will never contain more than one complete connected component.

(b) (5 pts.) [True or False] The KMP string matching algorithm can be modified to count the number of matches of the pattern in the text and still run in linear time.

(c) (5 pts.) [True or False] The FFT algorithm works on $O(n \log n)$ time and requires $\Omega(n \log n)$ space.

(d) (5 pts.) [True or False] The FFT algorithm is a parallel algorithm with $O(\log n)$ run time and $O(n \log n)$ work.

(e) (5 pts.) Solution to the recurrence $T(n) = T(n/2) + \log(n)$ is $\Theta(\_\_\_\_\_\_\_)$.

(f) (5 pts.) [True or False] If $C_k$ is the graph consisting of a simple cycle of length $k$ then any elimination order for $C_k$ generates at most $k$ fill.

(g) (5 pts.) [True or False] The set of all graphs that are **not** 3-colorable is NP-hard and thus NP-complete.

(h) (5 pts.) If $M$ is an $\sqrt{n}$ by $\sqrt{n}$ square mesh, the commute time From the upper left corner to the lower right is: $\Theta(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)$.

(i) (5 pts.) [True or False] In class we gave an algorithm to find the inverse of an $n$ by $n$ matrix that was $O(\log^2 n)$ time and $O(n^{2.81})$ work.
Problem 4: Bounded-Degree MST is hard (20 pts.)

The decision version of the Bounded Degree Minimum Weight Spanning Tree Problem (Bounded Degree MST) is the following:

**Input:** An undirected graph $G = (V, E)$, edge weights $w : E \to \mathbb{Z}$ and integers $d$ and $k$.

**Question:** Does $G$ have a spanning tree of weight at most $k$ with degree at most $d$? (Recall the degree of a tree is the maximum degree of any node in the tree).

For both of the following problems, give reductions from known NP-complete problems. (*Hint: Use the same problem for both*)

1. Show that the Bounded Degree MST is NP-complete for the case when $d = 2$.

2. Show that the Bounded Degree MST is also NP-complete for any constant $d$. Partial credit will be given for the case when $d = 3$. 
Problem 5: List Access using Move Halfway to Front (20 pts.)

In class we considered the list update problem and showed that move-to-front on-line algorithm (MTF) was 4-competitive.

In this problem we consider a modified version of MTF where after an access of \( x \), we move \( x \) halfway to front (MHTF), i.e., if the item is at position \( k \), it is moved to position \( \lfloor k/2 \rfloor \). The goal of this problem is to determine the competitiveness of MHTF.

As in the analysis of MTF, let \( B \) be any off-line algorithm for the access problem. In the way of a hint, let’s define \( S \) and \( T \) in a way similar to the proof of MTF. Let \( J \) be the elements of the list that are “jumped over” by \( x \) when moving \( x \) halfway to front.

\[
S = \{ \ y \mid y \in J \text{ and } y \text{ is before } x \text{ in } B \} \\
T = \{ \ y \mid y \in J \text{ and } y \text{ is after } x \text{ in } B \}
\]

1. What potential function will you use for your analysis? You may want to work on the next part to determine the right potential function.

2. Determine the amortized cost for a single step of MHTF to access \( x \) and move \( x \). Write your answer as a function of the access cost for \( B \).
3. Determine a bound on the amortized cost for MHTF to do nothing and $B$ performs a swap.

4. Use the above parts to get a constant competitive bound for MHTF.
Problem 6: Fibonacci Farm (20 pts.)

The Fibonacci numbers can be computed by the recurrence $F_{i+2} = F_{i+1} + F_i$ with initial values $F_0 = 0$ and $F_1 = 1$. With the emerging trend in parallel computing, you are hired to design a fast parallel algorithm to compute the Fibonacci numbers.

Your Task: Give an algorithm which takes a number $n \in \mathbb{Z}_+$ as input and outputs the following array of $n$ elements:

$$[F_1, F_2, F_3, \ldots, F_n]$$

For full credit, your algorithm must run in $O(n)$ work and $O(\log n)$ depth. It might be helpful to represent the recurrence in the matrix form: for a suitable matrix $A$,

$$
\begin{pmatrix}
F_{i+1} \\
F_i
\end{pmatrix} = A 
\begin{pmatrix}
F_i \\
F_{i-1}
\end{pmatrix}.
$$

(More hint: In class we saw a parallel algorithm for computing all prefix-sums of an array of length $n$ with similar work and depth.)
Problem 7: An Approximation Algorithm for Multiway Cut (20 pts.)

Many real-world problems can be viewed as finding a graph cut to optimize certain objective functions. In this problem, we will study the Multiway Cut problem:

**Input:** a weighted undirected graph \( G = (V, E, w) \), and a set of terminals \( T = \{s_1, \ldots, s_k\} \subseteq V \). Assume that \( G \) is connected, and the weights are given by \( w : E \to \mathbb{R}_{\geq 0} \).

**Output:** a set of edges \( C \) of minimum weight whose removal disconnects all the terminals (i.e., for all pairs of \( i \neq j \), there is no path between \( s_i \) and \( s_j \) in \( G - C \)).

Since this problem is NP-hard, our goal will be to develop a good approximation algorithm which runs in polynomial time. As a starting point, consider the following simple algorithm:

**Step (1)** For \( i = 1, 2, \ldots, k \), compute the set of edges of minimum weight that disconnects \( s_i \) from the rest of the terminals. Call each of these sets \( C_i \).

**Step (2)** Output \( C = \bigcup_{i=1}^{k} C_i \).

You will show that this algorithm gives a 2-approximation. In the following steps, you will prove that \( C \) is a feasible solution and \( w(C) \leq 2w(C^*) \), where \( C^* \) is an optimal solution and \( w(\cdot) \) is the weight of the edge set (i.e., \( w(E') = \sum_{e \in E'} w(e) \)).

1. Give a polynomial-time algorithm for computing \( C_i \), the minimum-weight set of edges that disconnects \( s_i \) from the rest of the terminals.

2. Show that \( C \) is a feasible solution to this problem.
3. Let $C^*$ be an optimal multiway-cut solution. Consider that $G' = G - C^*$, the graph obtained by removing the edges in $C^*$ from $G$, has exactly $k$ components. Suppose $P_1, P_2, \ldots, P_k \subseteq V$ are the components of $G'$ such that $s_i \in P_i$. We define the following decomposition: let $C^*_i = \{(u, v) \in C^* : u \in P_i \text{ and } v \notin P_i\}$. Show that

$$
\sum_{i=1}^{k} w(C^*_i) = 2w(C^*).
$$

4. Conclude that the algorithm above is a factor-2 approximation, i.e., $w(C) \leq 2w(C^*)$. 
