University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2006

Name:
Email:

- Print your name and email, *neatly* in the space provided above; print your name at the upper right corner of *every* page. Please print legibly.
- This is an *closed book* exam. You are permitted to use *only* two pages of "cheat sheets" that you have brought to the exam and a calculator. *Nothing else is permitted.*
- Do all the problems in this booklet. *Show your work!* You will not get partial credit if we cannot figure out how you arrived at your answer.
- Write your answers in the space provided for the corresponding problem. Let us know if you need more paper.
- Don't spend too much time on any single problem. The questions are weighted equally. If you get stuck, move on to something else and come back later.
- If any question is unclear, ask us for clarification.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Short Answer**

   For each problem below, give the answer in terms of simplest Θ. Please show your work where appropriate. (2 points each).

   (a) Time to find the minimum element in a balanced binary search tree *Solution:* $\Theta(\log n)$

   (b) $\log n!$ *Solution:* $\Theta(n \log n)$

   (c) $\sum_{i=1}^{n} (\ln n)/i$ *Solution:* $\Theta(\ln^2 n)$

   (d) Time to build a heap from an unsorted array *Solution:* $\Theta(n)$

   (e) Solution to the recurrence $T(n) = 2T(n/2) + \log n$ *Solution:* $\Theta(n)$

(f) Solution to the recurrence $T(n) = 4T(n/2) + n^2$ *Solution:* $\Theta(n^2 \log n)$

(g) Number of edges in a spanning tree for a connected graph $G = (V, E)$ *Solution:* $|V| - 1$

(h) Consider a graph $G = (V, E)$, with negative weight edges. What is the fastest time in which we can determine if there is a negative cycle in G? *Solution: We can do this with Bellman-Ford, taking* $\Theta(|V||E|)$ *time.*

(i) What is the expected amount of space used by a skip list storing n items? *Solution:* $\Theta(n)$

(j) Assume we are hashing n items into m bins with a good hash function. What is the expected number of colliding pairs of items? *Solution:* $\Theta(n^2/m)$

2. **Short Answer (10 points each) Where appropriate, circle your final answer.**

   (a) Solve the following recurrence using annihilators: $T(n) = 4T(n-2) + n$. Give the solution in general form i.e. do not solve for the constants

   Solution: The annihilator of the homogeneous part is $\boldsymbol{L}^2 - 4$ *which factors to* $(\boldsymbol{L}-2)(\boldsymbol{L}+2)$. *The annihilator of the non-homogeneous part is* $(\boldsymbol{L}-1)^2$. *This implies that the general solution is* $T(n) = c_1 2^n + c_2(-2)^n + c_3 n + c_4$

(b) Consider a data structure over an initially empty list that supports the following two operations. APPEND-NUMBER(x): Adds the number x to the beginning of the list; and REDUCE-LIST: Traverses the list, computing the sum of all the numbers traversed and then creates a new list that contains only one number, which is this sum.

- Assume an arbitrary sequence of n operations are performed on this data structure. What is the worst case run time of any particular operation? *Solution:* $\Theta(n)$. *First* $n-1$ *APPEND-NUMBER operations and then a REDUCE-LIST operation.*

- Show that the amortized cost of an operation is $O(1)$ using the potential method. Make sure to prove your potential function is valid. *Solution: Let* $\phi(D)$ *equal the number of items in the list. This is a valid potential function (Why?). The amortized cost of an APPEND-NUMBER operation is then* $c_i + \phi_i - \phi_{i-1} = 2$. *The amortized cost of a REDUCE-LIST operation is* $l_i + (1 - l_i) = 1$ *where* l_i *is the length of the list at time* i.

3. **Graph Theory**

Assume you are given a set of cities and the highways between them in the form of an undirected graph $G = (V, E)$. Each edge $e \in E$ connects two cities and has a length $l(e)$. You want to get from city s to city t. However, there is one problem: your car can only hold enough gas to travel L miles and there is a gas station in every city, but *not* between the cities. Therefore you can only take a route e if $l(e) \leq L$

- Given the limitation on your car's fuel tank, show how to determine in linear time (i.e. $O(|V| + |E|)$) whether there is a feasible route from s to t. *Solution: Remove all edges from the graph with length greater than L and then do a DFS starting at s and see if t can be reached*

- You are now planning on buying a new car and want to know the minimum fuel tank capacity needed to get from s to t. Give a $O((|V| + |E|) \log |V|)$ time algorithm to do this. Hint: Make two small changes to an algorithm we discussed in class. Your solution to this problem need be no more than three sentences. *Solution: Define an edge u, v to be tense if* $\max(dist(u), w(u,v)) < dist(v)$ *and relax a tense edge (u, v) by setting $dist(v)$ to be* $\max(dist(u), w(u,v))$. *Then run Dijkstra's with these new changes.*

4. **Palindromes**

A sequence is a palindrome if it is the same whether read left to right or right to left. For instance, the sequence:

$$A, C, T, G, T, C, B, Q, B, A$$

has several palindromic subsequences including: C, T, T, C and A, C, T, G, T, C, A (on the other hand, T, C, B is not palindromic). Devise an algorithm that takes a sequence $x[1..n]$ and returns the length of the longest palindromic subsequence. Its running time should be $O(n^2)$. Hint: For integers $1 \leq i < j \leq n$, let $P(i, j)$ be the length of the longest palindrome in the subsequence $x[i..j]$.

Solution: For $i = j$, set $P(i, j) = 1$. For $i > j$, define $P(i, j) = 0$. For all other i and j, if $x[i] = x[j]$, let

$$P(i, j) = \max(P(i + 1, j), P(i, j - 1), P(i + 1, j - 1) + 2)$$

otherwise, let

$$P(i, j) = \max(P(i + 1, j), P(i, j - 1))$$

From the recurrence, the dynamic program follows directly.

5. **Holiday Shopping**

You've just finished all your Holiday shopping at Page One and are now faced with the formidable task of putting n holiday gifts in bags to take home (where n is a large number). More precisely, assume that you have n items, $x_1, x_2, ..., x_n$ with weights $w_1, w_2, ...w_n$. Each bag can hold total weight 1 and you want to minimize the number of bags used to hold all the items. In other words, you want to partition the n items into the smallest number of sets such that each set has total weight no more than 1. Assume that for all i, $w_i \leq 1$.

Consider the following greedy algorithm. We put x_1 in the first bag. The for $i = 2, ..., n$, we put x_i in the last bag if there is room for it or start a new bag if there is no room. For example if $w_1 = .2$, $w_2 = .4$, $w_3 = .6$ and $w_4 = .3$, the greedy algorithm will put the first two items in a bag together and the last two items in a separate bag.

- Show that this greedy algorithm is non-optimal by giving an input for which it does not use the smallest number of bags. *Solution: Let the weights of the items be 2/3,2/3,1/3,1/3. Then the optimal number of bags is two but the algorithm requires three bags.*

- Show that the greedy algorithm has a ratio bound of two. In other words, show that the number of bags used by greedy is no more than $2 * OPT + O(1)$ where OPT is the minimum number of bags.

  Solution: Let $b_1, b_2, ..., b_x$ be the bags used by greedy. For each odd number i between 1 and $x - 1$, consider the bags b_i and b_{i+1}. The sum of the weights of the items in b_i and b_{i+1} must be at least equal to 1 because of the greedy property. In particular, the first item placed in b_{i+1} could not fit in b_i and so the sum of the weight of this item and all of the items in b_i must be greater than 1. We know that $OPT \geq \sum_i w_i$ and so the previous observation implies that the number of bags used by greedy is no more than $2 * OPT + 1$. The 1 is added due to the possibility of having one unpaired bag if x is an odd number.