CS570

Analysis of Algorithms

Summer 2009

Exam I

Name: _____

Student ID: _____

	Maximum	Received
Problem 1	20	
Problem 2	10	
Problem 3	10	
Problem 4	20	
Problem 5	15	
Problem 6	10	
Problem 7	15	
Total	100	

2 hr exam
Close book and notes

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE] T
An array with following sequence of terms [20, 15, 18, 7, 9, 5, 12, 3, 6, 2] is a max-heap.

[TRUE/FALSE] T
Complexity of the following shortest path algorithms are the same:
- Find shortest path between S and T
- Find shortest path between S and all other points in the graph

[TRUE/FALSE] T
In an undirected weighted graph with distinct edge weights, both the lightest and the second lightest edge are in the MST.

[TRUE/FALSE] F
Dijkstra's algorithm works correctly on graphs with negative-cost edges, as long as there are no negative-cost cycles in the graph.

[TRUE/FALSE] T
Not all recurrence relations can be solved by Master theorem.

[TRUE/FALSE] F
Mergesort does not need any additional memory space other than that held by the array being sorted.

[TRUE/FALSE] T
An algorithm with a complexity of $O(n^2)$ could run faster than one with complexity of O(n) for a given problem.

[TRUE/FALSE] F
There are at least 2 distinct solutions to the stable matching problem--one that is preferred by men and one that is preferred by women.

[TRUE/FALSE] F
A divide and conquer algorithm has a minimum complexity of *O(n log n)* since the height of the recursion tree is always *O(log n)*.

[TRUE/FALSE] T
Stable matching algorithm presented in class is based on the greedy technique.

2) 10 pts

a) Arrange the following in the increasing order of asymptotic growth. Identify any ties.

$$lg\ n^{10},\ 3^n,\ lg\ n^{2n},\ 3n^2,\ lg\ n^{lg\ n},\ 10^{lg\ n},\ n^{lg\ n},\ n\ lg\ n$$

If lg is log2 (convention),

$$lg\ n^{10} < lg\ n^{2n} < lg\ n^{2n} = n\ lg\ n < 3n^2 < 10^{lg\ n} < n^{lg\ n} < 3^n$$

If lg is log10 (from ISO specification)

$$lg\ n^{10} < lg\ n^{2n} < 10^{lg\ n} < lg\ n^{2n} = n\ lg\ n < 3n^2 < n^{lg\ n} < 3^n$$

b) Analyze the complexity of the following loops:

i- x = 0
```
for i=1 to n
    x= x + lg n
end for
```
O(n)

ii- x=0
```
for i=1 to n
    for j=1 to lg n
        x = x * lg n
    endfor
endfor
```
O(nlogn)

iii- x = 0
```
k = “some constant”
for i=1 to max (n, k)
    x= x + lg n
end for
```
O(n)

iv- x=0
```
k = “some constant”
for i=1 to min(n, k)
    for j=1 to lg n
        x = x * lg n
    endfor
endfor
```
O(logn)

3) 10 pts

a) For each of the following recurrences, give an expression for the runtime T (n) if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply.

$T(n) = T(n/2) + 2^n$

theta(2^n)

$T(n) = 16T(n/4) + n$

theta(n^2)

$T(n) = 2T(n/2) + n \log n$

More general case 2
If F(n) = theta((n^logb(a))*(logn)^k) with k >= 0, then T(n) = theta((n^logb(a))*(logn)^(k+1))

theta(n(logn)^2) from case 2

$T(n) = 2T(n/2) + \log n$

theta(n)

$T(n) = 64T(n/8) - n^2 \log n$

Does not apply (f(n) is not positive)

4) 20 pts

You work for a small manufacturing company and have recently been placed in charge of shipping items from the factory, where they are produced, to the warehouse, where they are stored. Every day the factory produces n items which we number from 1 to n in the order that they arrive at the loading dock to be shipped out. As the items arrive at the loading dock over the course of the day they must be packaged up into boxes and shipped out. Items are boxed up in contiguous groups according to their arrival order; for example, items 1… 6 might be placed in the first box, items 7…10 in the second, and 11… 42 in the third.

Items have two attributes, *value* and *weight,* and you know in advance the values *v1 … vn* and weights *w1…wn* of the *n* items. There are two types of shipping options available to you:

Limited-Value Boxes: One of your shipping companies offers insurance on boxes and hence requires that any box shipped through them must contain no more than V units of value. Therefore, if you pack items into such a "limited-value" box, you can place as much weight in the box as you like, as long as the total value in the box is at most V.

Limited-Weight Boxes: Another of you shipping companies lacks the machinery to lift heavy boxes, and hence requires that any box shipped through them must contain no more than *W* units of weight. Therefore, if you pack items into such a "limited-weight" box, you can place as much value in the box as you like, as long as the total weight inside the box is at most *W*.

Please assume that every individual item has a value at most V and a weight at most W. You may choose different shipping options for different boxes. Your job is to determine the optimal way to partition the sequence of items into boxes with specified shipping options, so that shipping costs are minimized.

Suppose limited-value and limited-weight boxes each cost $1 to ship. Describe an *O(n)* greedy algorithm that can determine a minimum-cost set of boxes to use for shipping the *n* items. Justify why your algorithm produces an optimal solution.

We use a greedy algorithm that always attempts to pack the largest possible prefix of the remaining items that still fits into some box, either limited-value or limited weight. The algorithm scans over the items in sequence, maintaining a running count of the total value and total weight of the items encountered thus far. As long as the running value count is at most V or the running weight count is at most W, the items encountered thus far can be successfully packed into some type of box. Otherwise, if we reach a item j whose value and weight would cause our counts to ***exceed V and W*** , then prior to processing item j we first package up the items scanned thus far (up to item j-1) into an appropriate box and zero out both counters. Since the algorithm spends only a constant amount of work on each item, its running time is O(n).

Why does the greedy algorithm generate an optimal solution (minimizing the total number of boxes)? Suppose that it did not, and that there exists an optimal solution different from the greedy solution that uses fewer boxes. Consider, among all optimal solutions, one which agrees with the greedy solution in a maximal prefix of its boxes. Let us now examine the sequence of boxes produced by both solutions, and consider the first box where the greedy and optimal solutions differ. The greedy box includes items i…j and the optimal box includes items i…k, where $k < j$ (since the greedy algorithm always places the maximum possible number of items into a box). In the optimal solution, let us now remove items k+1…j from the boxes in which they currently reside and place them in the box we are considering, so now it contains the same set of items as the corresponding greedy box. In so doing, we clearly still have a feasible packing of items into boxes and since the number of boxes has not changed, this must still be an optimal solution; however, it now agrees with the greedy solution in one more box, contradicting the fact that we started with an optimal solution agreeing maximally with the greedy solution.

5) 15 pts

For two unsorted arrays x and y, each of size n, give a divide and conquer algorithm that runs in *O(n)* time and computes the maximum sum *M,* as given below:

$M = Max\ (x_i + x_{i+1} - y_i);\ i=1 \text{ to } n-1$

Divide: divide problem into 2 equal size subproblems at each step.
Conquer: solve the subproblems recursively until the subproblem become trivial to solve. In this case problem size of 2 is trivial. In that case the solution is the sum of the 2 numbers.
Combine: We need to merge the solutions to the two subproblems S1 and S2. At each step we need to evaluate the following 3 cases:
Case 1: Max is in S1 (subproblem to the left)
Case 2: Max is in S2 (subproblem to the right)
Case 3: Max is on the border of S1 and S2

In case 3, you need to calculate X(the last element of S1) + X(the first element of S2) – Y(the last element of S1)

6) 10 pts

Suppose we are given an instance of the Shortest Path problem with source vertex s on a directed graph G. Assume that all edges costs are positive and distinct. Let P be a minimum cost path from s to t. Now suppose that we replace each edge cost c_e by its square, c_e^2, thereby creating a new instance of the problem with the same graph but different costs.

Prove or disprove: P still a minimum-cost s - t path for this new instance.

Let G have edges (s,v), (v,t), and (s,t), when the first two of these edges have cost 3 and the third has cost 5. Then the shortest path is the single edge (s,t), but after squaring the costs the shortest path would go thourgh v.

7) 15 pts

We are given two arrays of integers A[1..n] and B[1..n], and a number X. Design an algorithm which decides whether there exist i, j 2 {1, . . . , n} such that A[i] + B[j] = X. Your algorithm should run in time O(n log n).

Sort array B
For int i = 0 to n
  Temp = X – a[i]
  Do binary search to find an element whose value is equal to X – A[i] in array B
  If it finds return true
End of For