Name(last, first): _____

CS570
Analysis of Algorithms
Fall 2005
Midterm Exam

Name: _____

Student ID: _____

	Maximum	Received
Problem 1	14	
Problem 2	2	
Problem 3	5	
Problem 4	10	
Problem 5	10	
Problem 6	15	
Problem 7	20	
Problem 8	15	
Problem 9	9	

Name(last, first): _____

1. 14 pts
   Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

   [TRUE/FALSE]
   A dynamic programming algorithm tames the complexity by making sure that no subproblem is solved more than once.

   [TRUE/FALSE]
   The memoization approach in dynamic programming has the disadvantage that sometimes one may solve subproblems that are not really needed.

   [TRUE/FALSE]
   A greedy algorithm finds an optimal solution by making a sequence of choices and at each decision point in the algorithm, the choice that seems best at the moment is chosen.

   [TRUE/FALSE]
   If a problem can be solved correctly using the greedy strategy, there will only be one greedy choice (such as "choose the object with highest value to weight ratio") for that problem that leads to the optimal solution.

   [TRUE/FALSE]
   Whereas there could be many optimal solutions to a combinatorial optimization problem, the value associated with them will be unique.

   [TRUE/FALSE]
   Let G be a directed weighted graph, and let u, v be two vertices. Then a shortest path from u to v remains a shortest path when 1 is added to every edge weight.

   [TRUE/FALSE]
   Given a connected, undirected graph G with distinct edge weights, then the edge e with the second smallest weight is included in the minimum spanning tree.

2. 2 pts
   In our first lecture this semester, which of the following techniques ware used to solve the Stable Matching problem? Circle all that may apply.
   A- Greedy
   B- Dynamic Programming
   C- Divide and Conquer

Name(last, first): _____

3. 5 pts
   A divide and conquer algorithm is based on the following general approach:

   - Divide the problem into 4 subproblems whose size is one third the size of the problem at hand. This is done in O(lg n)
   - Solve the subproblems recursively
   - Merge the solution to subproblems in linear time with respect to the problem size at hand

   What is the complexity of this algorithm?

4. 10 pts
   Indicate for each pair of expressions (A,B) in the table below, whether A is **O**, **Ω**, or **Θ** of B. Assume that k and c are positive constants.

A	B	**O**	**Ω**	**Θ**
$(\lg n)^k$	Cn			
2^n	$2^{(n+1)}$			
$2^n n^k$	$2^n n^{2k}$			

Name(last, first): _____

5. 10 pts
   The array A below holds a max-heap. What will be the order of elements in array A after a new entry with value 19 is inserted into this heap? Show all your work.
   A = {16, 14, 10, 8, 7, 9, 3, 2, 4, 1}

Name(last, first): _____

6. 15 pts
   Consider a max-heap H and a given number X. We want to find whether X is larger than the k-th largest element in the list. Design an O(k) time algorithm to do this.
   Note: you do not need to prove your algorithm but you need to prove the complexity.

Name(last, first): _____

7. 20 pts total
   Consider the following arcade game where the player starts at the lower left corner of an n by n grid to reach the top right corner. At each step the player can either jump to the right (x –axis) or jump up (y-axis). Depending on his/her energy level the player can jump either 1 or 2 squares. If his/her energy level is higher than E he/she can jump 2 squares, otherwise only one square. When landing on square (i,j) the player may gain or lose energy equal to $|X_{ij}|$. Positive X_{ij} indicates energy gain. In addition, every time the player jumps 2 squares he/she loses E/2 units of energy.

   a) Present a polynomial time solution to find the highest energy level the player can end the game with. (15 pts)

b) Describe how you could build the path that leads to the optimal solution. (5 pts)

Name(last, first): _____

8. 15 pts
   T is a spanning tree on an undirected graph G=(V,E). Edge costs in G are NOT guaranteed to be unique. Prove or disprove the following:
   If every edge in T belongs to SOME minimum cost spanning tree in G, then T is itself a minimum cost spanning tree.

Name(last, first): _____

9. 9 pts
   We have shown in class that the Integer Knapsack problem can be solved in O(nW) where n is the number of items and W is the capacity of the knapsack. The Fractional Knapsack problem is less restrictive in that it allows fractions of items to be placed in the knapsack. Solve the Fractional Knapsack problem (have n objects, weight of object i = W_i, value of object i = V_i , weight capacity of knapsack=W, Objective: Fill up knapsack with objects to maximize total value of objects in knapsack)

Name(last, first): _____

Additional space.

Name(last, first): _____

Additional space.

CS570 MIDTERM SOLUTION

Q 1:
1.True
2.False
3.True
4.False
5.True
6.False
7.True

Grading Policy : 2 points and all or none for each question.

Q 2: The answer is A

Grading Policy : Any other solution is wrong and get zero

Q 3:

$T(n) = 4T(\frac{n}{3}) + \lg n + cn$ by master theorem, $T(n)$ is $O(n^{\log_3 4})$

Grading Policy : 2 points for the correct recursive relation. 3 points for the correct result $O(n^{\log_3 4})$. If the student provide the correct result but doesn't give the recursive relation, get 4 points.

Q 4: Click ✓on $(1,1),(2,1),(2,2),(2,3),(3,1)$ and write $\times$ on others . Here (i,j) means the $i-th$ row and $j-th$ column in the blank form.

Grading Policy : 1 points for each entry in the blank form

Q 5: The process of a new entry with value 19 inserted is as follows:

(1) A={16,14,10,8,7,9,3,2,4,1,19}
(2) A={16,14,10,8,19,9,3,2,4,1,7}
(3) A={16,19,10,8,14,9,3,2,4,1,7}
(4) A={19,16,10,8,14,9,3,2,4,1,7}

Grading Policy : It's OK if represent A by a graph. 2.5 points for each step.

Q 6:

Algorithm: The max head is a tree and we start from the root doing the BFS(or DFS) search. If the incident node is bigger than x, we add

Date: Oct 17, 2005.

it into the queue. By doing this if we already find k nodes then x is smaller than at least k nodes in the heap, otherwise, x is bigger than $k-th$ biggest node in the heap.

running time: During the BFS(or DFS), we traverse at most k nodes. Thus the running time is $O(k)$

Grading Policy : 9 points for the algorithm and 6 points for the correct running time. A typical mistake is that many students assumed that the max heap was well sorted and just pick the k-th number in the heap. In this case they get 3 points at most

Q 7:

a) The recursive relation is as follows:

$$OPT(m,n) = max \begin{cases} OPT(m,n-1) + X_{mn} \\ OPT(m,n-2) + X_{mn} - E/2, & \text{if OPT}(m,n-2) > \frac{E}{2} \\ OPT(m-1,n) + X_{mn} \\ OPT(m-2,n) + X_{mn} - E/2, & \text{if } OPT(m-2,n) > \frac{E}{2} \end{cases}$$

And the boundary condition is $OPT(1,1) = E_0$. Clearly to find the value of $OPT(n,n)$, we need to fill an $n \times n$ matrix. It takes constant time to do the addition and compare the 4 numbers in the recursive relation. Hence the running time of our algorithm is $O(n^2)$

b)After the construction of the $n \times n$ table in a), we start right at the upper right corner (n,n). We compare the value with the 4 previous value in the recursive relation mentioned in a). If we get $OPT(n,n)$ from the previous position (i,j) in the recursive relation, store the position (i,j) in the path and repeat the process at (i,j) until we get to $(1,1)$

Grading Policy : For part a), 10 points for the correct recursive relation and 5 points for the correct running time. If the student didn't solve this question by dynamic programming, he gets at most 4 points as the partial credit.

For part b), make sure they know the idea behind the problem.

Q 8: False. The counter example is as follows: Consider the graph $(a,b),(b,c),(a,c)$ where $w(a,b) = 4, w(b,c) = 4, w(c,a) = 2$. Clearly (a,b) is in the MST $((a,b),(c,a))$ and (b,c) is in the MST $((b,c),(c,a))$ but $((a,b),(b,c))$ is not MST of the graph.

Grading Policy : If the answer is "True", at most get 5 points for the whole question. If the answer is "False" but didn't give a counter example, get 9 points. If the answer is "false" and give a correct counter example, but didn't point out the spanning tree in which every edge is in some MST but the spanning tree is not MST, get 12 points.

Q 9: First we calculate $c_i = V_i/W_i$ for $i = 1, 2, ..., n$, then we sort the objects by decreasing c_i. After that we follows the greedy strategy of always taking as much as possible of the objects remaining which has highest c_i until the knapsack is full. Clearly calculating c_i takes $O(n)$ and sorting c_i takes $O(n \log g)$. Hence the running time of our algorithm is $O(n) + O(n \log n)$, that is, $O(n \log n)$. We prove our algorithm by contradiction. If $S = \{O_1, ..., O_m\}$ is the objects the optimal solution with weight $w_1, ..., w_m$. The total value in the knapsack is $\sum_{i=1}^{m}(w_i/W_i)V_i$.Assume there is i, j(without loss of generality assume that $i < j$) such that $V_i/W_i > V_j/W_j$ and there is V_i with weight w_i' left outside the knapsack. Then replacing $min\{w_i', w_j\}$ weight of O_j with $min\{w_i', w_j\}$ weight of O_i we get a solution with total value $\sum_{k=1}^{m} w_k/W_kV_k + min\{w_i', w_j\}/W_iV_i - min\{w_i', w_j\}/W_jV_j$. $w_i' > 0, w_j > 0$ and thus $min\{w_i', w_j\} > 0$. Note that $V_i/W_i > V_j/W_j$, hence $\sum_{k=1}^{m}(w_k/W_k)V_k + (min\{w_i', w_j\}/W_i)V_i - (min\{w_i', w_j\}/W_j)V_j > \sum_{k=1}^{m}(w_k/W_k)V_k$. we get a better solution than the optimal solution. Contradiction!

Grading Policy : 5 points for the algorithm, 2 points for the proof and 2 points for the running time.

Dynamic Programming will not work here since the fractions of items are allowed to be placed in the knapsack. If the student tried solving this problem by dynamic programming and both the recursive relation and time complexity are correct, they get 4 points at most.