CS570
Analysis of Algorithms
Spring 2008
Exam I

Name: _____
Student ID: _____
Section:__2:00-5:00 or __5:00-8:00

	Maximum	Received
Problem 1	20	
Problem 2	16	
Problem 3	12	
Problem 4	16	
Problem 5	16	
Problem 6	20	
Total	100	

Note: The exam is closed book closed notes.

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

[TRUE/FALSE]
Given the shortest path P between two nodes A and B in graph G(V;E), there exists a minimum spanning tree T of G, such that all edges of path P is contained in T. F

[TRUE/FALSE]
If in a connected graph all edge weights are distinct, then for any two nodes A and B there exists a unique shortest path. F

[TRUE/FALSE]
O(E log E) and O(E log V) are equivalent regardless whether graph is dense or sparse. F

[TRUE/FALSE]
$2^n = \Theta(2^{n+4})$ T

[TRUE/FALSE]
Suppose $T_1(N) = O(F(N))$ and $T_2(N) = O(F(N))$. Then, $T_1(N) = O(T_2(N))$ F

[TRUE/FALSE]
Suppose that in an instance of the original Stable Marriage problem with n couples (so that every man ranks every woman and vice versa), there is a man M who is last on each woman's list and a woman W who is last on every man's list. If the Gale-Shapley algorithm is run on this instance, than M and W will be paired with each other. T

[TRUE/FALSE]
DFS takes O(|E| + |V |) time if the graph is represented as an adjacency matrix. |E| is the number of edges and |V | is the number of vertices(nodes). F/T

[TRUE/FALSE]
In a nested loop, if O(n) is a tight upper bound to the number of iterations in the outer loop and O(m) is a tight upper bound to the number of iterations in the inner loop, then O(mn) must be a tight upper bound on the number of times the inner loop is executed. F/T

[TRUE/FALSE]
In an algorithm that uses a priority queue, the worst case complexity of the algorithm can be dependent on the type heap the priority queue is implemented with T

[TRUE/FALSE]
Given a connected graph G, with at least two edges having the same cost, there will be at least two distinct minimum spanning trees F

2) 16 pts

What is the worst case complexity, in terms of n, of the following code fragment

a-

```
for (i = 1; i<n;  i + +)
    for (j = i + 1; j < n; j + +)
        s + +;
        k = 2 * (s - 1);
    endfor
endfor
```

b-

```
for (i = 1; i < n; i + +)
    for (j = 1; j <i**2; j + +)
        s + +;
        k = 2 _ (s _ 1);
    endfor
endfor
```

c- Consider the following pseudocode, where A is an array storing n integer values. What are the best and `worst case time complexities of the algorithm AddAndAdd?

```
Algorithm AddAndAdd
c = 0
i = 0
while c < 200 and i < n do
    c = c+A[i]
    i=i+1
    for k = 0 to n – 1
        c = c + A[k]
    endfor
endwhile
```

3) 12 pts

Indicate for each pair of expressions (A, B) in the table below, where A is O, Ω, or θ of B. Assume that k and c are positive constants. Mark each box with Y (yes) and N (no).

A	B	O	Ω	θ
n^k	C^n			
2^n	$2^{n/2}$			
lg(n!)	$lg(n_n)$			

4) 16 pts

Give an algorithm that takes a set of elements A[1] through A[n] and returns the maximum sum without neighbors. In other words, it computes the largest sum of elements in the array none of which are neighbors. (subsequent elements A(i) and A(i+1) are considered neighbors). The algorithm should run in polynomial time with respect to n.

5) 16 pts

There are three types of operations you can perform on an integer:

- If it's divisible by 3, divide it by 3.

- If it's divisible by 2, divide it by 2.

- Subtract 1.

Given an integer n, return the minimal number of operations needed to produce the number 1.

a) 20 pts

Suppose that you have a very old MP3 player. Songs cannot be shuffled, or organized by song-name. Instead, it lets users define a display-order for artists (they need not be alphabetical). All songs by a given artist will appear consecutively. The device scrolls down one song on its list when you whack it firmly on concrete or other hard surface. Another single button lets you reset to the top of the list. Thus, when you want to listen to songs by a particular group *X*, you push the reset, and then whack it enough times to scroll past all songs of all artists appearing before group *X*. Because it has an expected lifetime of only 20,000 whacks, you want to organize the artists so that on average it does not get whacked any more than necessary. You can formalize the problem this way: For each of *n* artists numbered $i = 1$ through *n*, a value s_i gives the number of songs stored for artist *i*, and a fraction $0 < f_i < 1$ for each *i* gives the frequency with which you intend to scroll to listen to songs by artist *i*. For example, see the following.

Artist	Number of songs	Access Frequency
A	15	1/4
B	8	1/3
C	12	1/6
D	5	1/4

Then, the average whacks needed to scroll to a desired artist's songs with the ordering (A,B,C,D) is 1/4·0+1/3·15+1/6·23+1/4·35. The average whacks for the reverse ordering (D,C,B,A) would be 1/4·0+1/6·5+1/3·17+1/4·25.

Describe an algorithm that, given a collection of artists 1, 2, … *n*, number of songs s_i for each, and frequency of access f_i for each, outputs an ordering of artists so that the average number of whacks is minimized.

Additional Space

Additional Space