AMERICAN UNIVERSITY
LIBRARY
OF BEIRUT

American University of Beirut

CMPS 256

Advanced Algorithms and Data Structures

Spring 2004

Final Exam

Date: June 7th 11:00am – 1:00pm. **Name: ----------**

Instructor: Chiraz Ben Abdelkader

- This exam is closed-book and closed notes. You are allowed to use nor more than **2 pages of notes**, front and back.
- There are **7 problems** over **8 pages** and for a total of **105 points**.
- Please write as legibly as you can, otherwise I will NOT mark your answer!
- **Important note**: Justify all your answers. Only partial credit will be given to unjustified answers.

	Prob. 1	*Prob. 2*	*Prob. 3*	*Prob. 4*	*Prob. 5*	*Prob. 6*	*Prob. 7*	*Total*
Max Grade	15	8	7	15	20	25	15	105
Your Grade								

AMERICAN UNIVERSITY
LIBRARY
OF BEIRUT

Question 1. (12-15 pts) Sorting, Selection, Hash Tables

a) (3 pts) Bucket-Sort runs in $O(n)$ time and QuickSort in $O(n\log n)$, but why is it that QuickSort is more widely used in practice?

b) (4 pts) We want to design efficient algorithms for the following operations on a hash-table of n elements: (i) Extract-Max, and (ii) Inorder-walk (i.e. printing all elements in increasing order). Briefly explain how you would design each algorithm and determine its running time.

c) (5-8 pts) Do exactly one of the following two problems (I will only grade one problem – there is no extra credit for doing both or part of each).

Hint: Both problems may be solved by using the SELECT algorithm somehow …

(i) (5 pts) Give an $O(n)$ algorithm which, given a list $A[1..n]$ of numbers, a positive integer k ($1 \le k \le n$) and number X, outputs the k numbers of A that are closest to X in value. (These numbers may be greater than, less than, or equal to X.) For example, suppose we have: $A = \{9.5, 1.3, 5.2, 0.1, 7.5, 6.6, 2.5, 9.1\}$, $k=4$ and $X=8.2$. The algorithm should output the set $\{9.5, 7.5, 6.6, 9.1\}$ (in any order). Briefly explain your algorithm and show that it runs in $O(n)$.

(ii) (8 pts) Suppose you are given an unsorted array A of n distinct integers in the range $0..n$ except for one integer, denoted the *missing number*. For example, if $A = \{2,4,0,3\}$ then the *missing number* is 1. Design a $O(n)$ divide-and-conquer algorithm to find the *missing number* of an input array A. Briefly explain the correctness of your algorithm and analyze its running time.

Question 2. (8 pts) Recurrences

Solve the following recurrence using the iteration method:

$$T(n) = T(n-1) + 3\log n, \quad T(1) = 5$$

Your answer should be given in asymptotic notation (and not as a summation).

Question 3. (7 pts) Asymptotic notation

Use formal definitions of asymptotic notation to prove that: if $f(n) = O(g(n))$ then $f(n)+g(n)=\Theta(g(n))$

Question 4. (15 pts) Code Analysis

a) (6pts) Give the worst-case asymptotic running time of the following code. You may assume that A is an array of integers.

```
maxSubsequenceSum(A)
n = length(A)
maxSum = 0
for i=1 to n do
      thisSum = 0
      for j=i to n do
            thisSum = thisSum + A[j]
            if (thisSum > maxSum) then
                  maxSum = thisSum
return maxSum
```

b) (9pts) Consider **Dijkstra's** algorithm below which is based on the adjacency list graph representation and a heap-implementation of set Q. Its running time is *O(ElogV+VlogV)*. Suppose we use an **adjacency matrix** graph representation instead. Give the running time of the algorithm, and clearly indicate on the code below the part(s) of the algorithm which caused the change in running time.

1. $Q \leftarrow V[G]$
2. $S \leftarrow \varnothing$
3. **for each** vertex $u \in Q$
4. **do** $d[u] \leftarrow \infty$
5. $\pi[u] \leftarrow$ NIL
6. $d[s] \leftarrow 0$
7. **while** $Q \neq \varnothing$
8. **do** $u \leftarrow$ Extract-Min(Q)
9. $S \leftarrow S \cup \{u\}$
10. **for each** $v \in adj[u]$ **do**
11. Relax (u,v,w)
12. Heapify(Q,v) //because value of d[v] may have changed after Relax()

Question 5. (20 pts) Graph Algorithms – Short Answer

a) Why does a shortest path contain at most |V| vertices and |V|-1 edges in any graph G=(V,E)?

b) In the DFS of a directed graph, let *(u,v)* be any edge in the graph and suppose d[u] < d[v]. What are the possible types of this edge? (tree, back, forward, cross.)

c) Student Cluelessenberg ran Dijkstra's algorithm on a graph containing a negative-cost cycle. **True or False**: the algorithm may go into an infinite loop. Justify your answer.

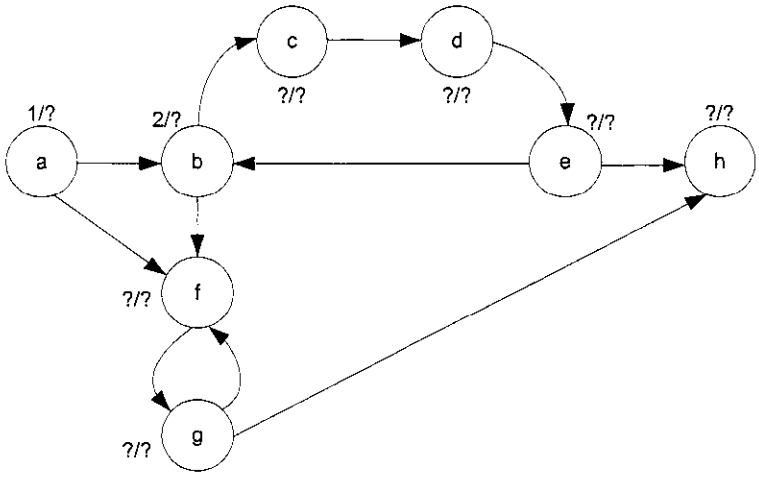
d) In the DFS of a directed acyclic graph, let *(u,v)* be any edge in the graph. **Show that** we must have f[v] < f[u].

Hint: Use proof by contradiction; Suppose f[v] > f[u] and give all possible cases for the relative positions of intervals (d[u],f[u]) and (d[v],f[v]). Then show that there is a contradiction in each case.

ANSWERS (use back side if necessary)

Question 6. (25 pts) Strong Connected Components

Consider the following directed graph G = (V,E):



a) Give all the strong components of this graph. Also clearly label each component with a super-vertex on the Figure above.

b) Assuming the set of vertices are stored in *V* in the following order: <a,b,c,d,e,f,g,h>, show the discovery and finishing times computed by DFS on this graph by filling in the question marks in the Figure above (d/f). Also mark all *tree* edges with a pencil.

c) Draw the **component graph** and clearly indicate the finishing time of each strong component on this graph. (Recall that we define the finishing time of a strong component C as $f(C) = \text{Max}_{u \in C}\{f([u]\})$.

d) Let (U,V) be any edge in the component graph. Explain why there cannot be an edge (V,U) in this graph.

e) Draw the **transpose** of the component graph (i.e. the graph obtained by reversing all its edges). Explain why <u>all</u> edges of this graph go from a vertex with a smaller finishing time to a vertex with a larger finishing time. In other words, there are <u>no</u> edges going from a vertex with a larger finishing time to a vertex with a smaller finishing time.

f) Based on **e)**, explain why when we run DFS G^T in topological order (i.e. Step 4 of StrongComp algorithm), the <u>first</u> DFS tree constructed contains only the strong component with the largest finishing time.

Question 7. (15 pts) Dynamic Programming and Greedy Algorithms

a) Recall that in the activity scheduling problem, we are given a set of n activities that are to be scheduled to use some resource, where each activity is defined by a start/finish time interval (s_i, f_i). The objective is to schedule the maximum number of non-overlapping activities. Recall also that an optimal solution for this problem is based on the *earliest-finish-first* greedy selection strategy.

Now suppose we instead use a *shortest-activity-first* strategy as follows: Sort the activities in increasing order of duration $(f_i - s_i)$. Give a counter-example to show that this would not lead to an optimal solution for the problem.

b) Dynamic programming is *recursion without repetition*. What does this mean?

c) Show the execution of Huffman's algorithm on the following set of six symbols and associated probabilities.

Symbol	Probability	Symbol	Probability
A	.15	D	.05
B	.50	E	.15
C	.10	F	.05

Show the final Huffman tree, as well as the final prefix code for your tree. (Intermediate steps are not required, but help in assigning partial credit.)

ANSWERS (use back side if necessary)