

Moderator: **vuangels**

Muhammad Kashif Ahmed : Student of BS(CS)

<http://groups.yahoo.com/group/vuangels>Kashif_risingstar@yahoo.com

CS502 Fundamentals of Algorithms

Final Term Examination - August 2004

Time Allowed: 150 Minutes

Instructions

Please read the following instructions carefully before attempting any of the questions:

1. Attempt all questions. Marks of every question are shown adjacent to it.
2. Do not ask any questions about the contents of this examination from anyone.
 - a. If you think that there is something wrong with any of the questions, attempt it to the best of your understanding.
 - b. If you believe that some essential piece of information is missing, make an appropriate assumption and use it to solve the problem.

****WARNING: Please note that Virtual University takes serious note of unfair means. Anyone found involved in cheating will get an 'F' grade in this course.**

Very Important:

Some results you may need:

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^n i^2 = \frac{2n^3 + 3n^2 + n}{6}, \quad \sum_{i=1}^n x^i = \frac{x^{(n+1)} - 1}{x - 1}$$

So please copy this in the word file for later use.

Total Marks: 120

Total Questions: 12

Question No. 1

Marks : 5

It is impossible to design a sorting algorithm based on comparison of keys worst-case run time is in $O(n)$.

- ☐ True
- ☐ False

Question No. 2

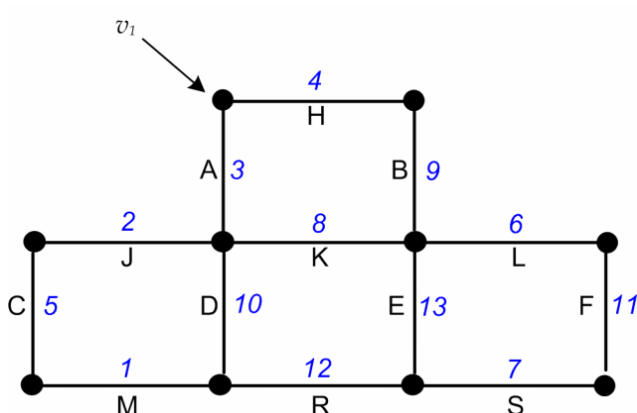
Marks : 15

Consider the following problem: You are given a sequence of n integers that contains $\log_2 n$ different integers. For example, $n = 8$, $\log(8) = 2$: $\{3, 5, 3, 5, 3, 5, 5, 5\}$; the list of 8 numbers is made up of only two distinct integers 3 and 5. Design an algorithm to sort this sequence using $O(n \log \log n)$ element comparisons (which is better than $O(n \log n)$). Prove that your algorithm is indeed $O(n \log \log n)$ [Hint: use variation of heap sort.]

Question No. 3

Marks : 20

In the following graph, edges are labeled with upper case letters. Edge weights are given as numbers next to edges.



Recall that Kruskal's algorithm greedily adds edges in a way that avoids cycles. For the graph shown above, list the edges in the order chosen by Kruskal's algorithm.

Question No. 4

Marks : 5

Greedy algorithms are called "greedy" because they often take a lot of time.

- ☐ True
- ☐ False

Question No. 5

Marks : 5

Kruskal's algorithm (choose best non-cycle edge) is better than Prim's (choose best tree edge) when the graph has relatively few edges.

- ☐ True
- ☐ False

Question No. 6

Marks : 5

Quick sort has a good average run time and a poor worst-case run time.

- ☐ True
- ☐ False

Question No. 7

Marks : 15

In the following algorithm, ". . ." stands for some simple calculations that take constant time, i.e., $\Theta(1)$.

```
procedure(n)
for k from 1 to n do
... /* produces a number j */
if k divides j, then mergesort an n-long list
...
end for
...
end
```

Note: Think of j as a random integer, so the probability that "k divides j" is "1/k".

- (a) Suppose the sorting were free (which it is not). What is the complexity class for the average running time of this algorithm. You MUST give a reason for your answer. (The class should be of the form $\Theta(f(n))$ where $f(n)$ is a simple function.)
- (b) Suppose that the basic operation is a comparison in merge sort. What is the complexity class for the average running time of this algorithm. (You may give your answer in the form $\Theta(\sum f(k))$ where $f(k)$ is a simple function and the sum runs from 1 to n .) You MUST give a reason for your answer.
- (c) Use (a) and (b) to find the complexity class for the average running time of this algorithm. You MUST give a reason for your answer.

Question No. 8

Marks : 5

There is a good greedy algorithm for the 0-1 Knapsack Problem.

- ☐ True
- ☐ False

Question No. 9

Marks : 10

Consider the following two problems. In P1 we are given as input a set of n squares (specified by their corner points), and a number k . The problem is to determine whether there is any point in the plane that is covered by k or more squares.

In P2 we are given as input an n -vertex graph, and a number k ; the problem is to determine whether there is a set of k mutually adjacent vertices. (E.g. for $k = 3$ we are just looking for a triangle in the graph.).

Obviously, the problems are both in NP. There exists a simple translation from P1 to P2: just make a graph vertex for each square, and add an edge between a pair of vertices if the corresponding two squares overlap.

(a) If P1 is NP-complete, would this translation imply that P2 is NP-complete?

(b) If P2 is NP-complete, would this translation imply that P1 is NP-complete?

(Give your Answer in Yes or NO only)

Question No. 10

Marks : 5

Dynamic programming uses a top-down approach.

- ☐ True
- ☐ False

Question No. 11

Marks : 10

Solve the following recurrence relation with full history:

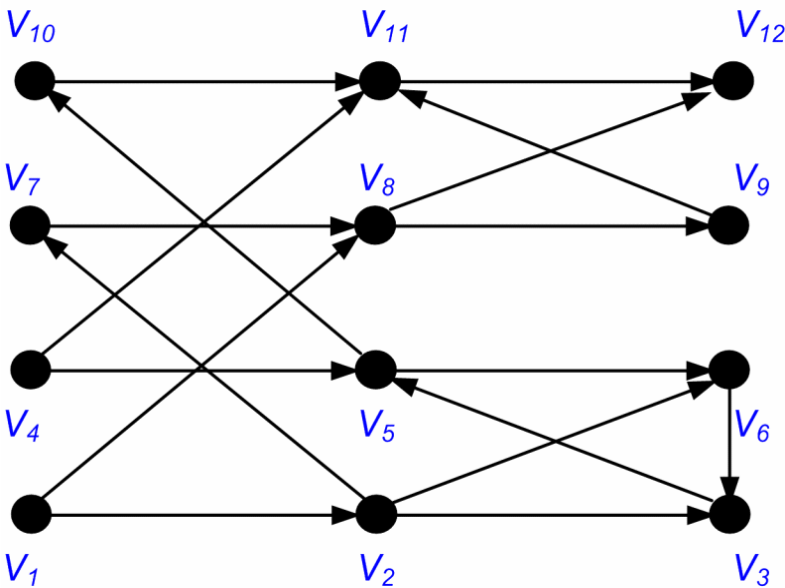
$$T(n) = \sum_{i=1}^{n-1} T(i), \quad T(1) = 1$$

Hint: subtract $T(n)$ from $T(n+1)$.

Question No. 12

Marks : 20

Consider the directed graph G below:



- (a) Run DFS on this graph and write down the values of $d[v]$ and $f[v]$ for all vertices v . Assume that in outer for-loop of the DFS, vertices of G are processed in numeric order of their labels. For example, DFS will choose v_1 initially. Also assume that in DFS-Visit, vertices adjacent to each vertex u are processed in increasing numeric label. For example, if v_{10} , v_5 , v_8 are adjacent to some vertex, they will be processed in the order v_5 , v_8 , v_{10} .
- (b) Identify tree edges, back edges, forward edges and cross edges for the DFS.