Student name: ____ Student ID: __________________

Mid-term exam

Cairo University
Faculty of Computers and Information
Subject: Algorithms
Subject Code: **CS316**
Examiner(s): **Sherif Khattab and Basheer Youssef**

Semester: 2nd
Date: 7/5/**2014**
Duration: 1 **hour**

Answer as much as you can. Max. grade is 20.

Question 1 [7 marks] Answer the following questions:

1. Recall the Partition subroutine used in both QuickSort and RSelect. Suppose that the following array has just been partitioned around some pivot element: 3, 1, 2, 4, 5, 8, 7, 6, 9. Which of the elements could **NOT** have been the pivot element?

all elements except 4, 5, and 9.

2. Given the following array of ten integers: 5 3 8 9 1 7 0 2 6 4. Suppose we run MergeSort on this array. What is the number in the 5th position of the partially sorted array after the outermost two recursive calls have completed (i.e., just before the very last Merge step)? Assume that counting positions starts at 1.
9

3. What is the asymptotic worst-case running time of MergeSort, as a function of the input array length n?

O(n log n)

4. What is the running time of depth-first search, as a function of n and m, if the input graph G=(V,E) is represented by an adjacency matrix (i.e., NOT an adjacency list), where as usual n=|V| and m=|E| ?

$O(n^2)$.

5. Consider a directed graph G=(V,E) with non-negative edge lengths and two distinct vertices s and t of V. Let P denote a shortest path from s to t in G. If we add 10 to the length of every edge in the graph, **would** P remain a shortest path? **why**?

P wouldn’t remain the shortest path. A path that has fewer edges than P will increase by a smaller amount than P and may become the shortest.

6. What is the asymptotic running time of Randomized QuickSort on arrays of length n, in expectation (over the choice of random pivots) **and** in the worst case, respectively?

O(n logn) on average
$O(n^2)$ in the worst case.

7. Suppose that the running time of an algorithm is governed by the recurrence $T(n)=7*T(n/2)+n^2$. **<u>What's</u>** the overall asymptotic running time (i.e., the value of T(n))? **<u>Name</u>** an algorithm that has this running time.

a = 7
b = 2
d = 2
Case 3 of Master method.
$T(n) = O(n^{\log_b a}) = O(n^{\log_2 7}) = O(n^{2.8})$
Strassen's matrix multiplication algorithm.

Question 2 [5 marks] True or False? <u>correct the wrong statements.</u> ***[1 mark each]***
1. Depth-first search can be used to compute a topological ordering of a directed acyclic graph in $O(m^2)$ time.

False. In O(m+n).
2. Breadth-first search can be used to compute the strongly connected components of a directed graph in O(m+n) time.

False. Depth-first search can.

3. Breadth-first search can be used to compute the connected components of an undirected graph in O(m+n) time.

True.

4. Depth-first search can be used to compute shortest paths in O(m+n) time (when every edge has unit length).

False. Breadth-first search can be used to compute shortest paths in O(m+n) time when every edge has unit length.

5. Dijkstra's shortest-path algorithm is guaranteed to correctly compute shortest-path distances (from a given source vertex to all other vertices) for input graphs that might have some negative edge lengths.

False. Dijkstra's shortest-path algorithm is not guaranteed to be correct when edges have negative lengths.

Question 4 [5 marks] Write pseudo-code for an efficient algorithm to compute the strongly connected components of a directed graph.

1. Let Grev = G with all arcs reversed
2. Run DFS-Loop on Grev ← Goal : compute "magical ordering" of nodes

   Let f(v) = "finishing time" of each v in V

1. Run DFS-Loop on G ← Goal : discover the SCCs one-by-one

   processing nodes in decreasing order of finishing times

[SCCs = nodes with the same "leader"]

<u>DFS-Loop (graph G)</u>

Global variable t = 0 (For finishing times in 1st pass)

[# of nodes processed so far]

Global variable s = NULL (For leaders in 2nd pass)

[current source vertex]

Assume nodes labeled 1 to n

For i = n down to 1

    if i not yet explored

        s := i

        DFS(G,i)

DFS (graph G, node i) (For rest of DFS-Loop)

-- mark i as explored

-- set leader(i) := node s

-- for each arc (i,j) in G :

    -- if j not yet explored

        -- DFS(G,j)

-- t++

-- set f(i) := t (i's finishing time)

Question 5 [5 marks] **Prove** the average running time of randomized selection.

Notation : Rselect is in phase j if current array size between $(\frac{3}{4})^{j+1} \cdot n$ and $(\frac{3}{4})^{j} \cdot n$

-X_j = number of recursive calls during phase j

Note : running time of RSelect $\leq \sum_{phases\ j} X_j \cdot c \cdot (\frac{3}{4})^{j} \cdot n$

- X_j: # of phase j subproblems
- $(\frac{3}{4})^{j} \cdot n$: <= array size during phase j
- $c \cdot (\frac{3}{4})^{j} \cdot n$: Work per phase j subproblem

Tim Roughgarden

Proof II: Reduction to Coin Flipping

X_j = # of recursive calls during phase j → Size between $(\frac{3}{4})^{j+1} \cdot n$ and $(\frac{3}{4})^{j} \cdot n$

Note : if Rselect chooses a pivot giving a 25 – 75 split (or better) then current phase ends ! (new subarray length at most 75 % of old length)

Recall : probability of 25-75 split or better is 50%

So : $E[X_j]$ <= expected number of times you need to flip a fair coin to get one "heads"

(heads ~ good pivot, tails ~ bad pivot)

Tim Roughgar

Let N = number of coin flips until you get heads.

(a "geometric random variable")

Note : E[N] = 1 + (1/2)*E[N]

1st coin flip — Probability of tails — # of further coin flips needed in this case

Solution : E[N] = 2 (Recall $E[X_j] <= E[N]$)

Expected running time of RSelect

$$\leq E[cn \sum_{phase\ j} (\frac{3}{4})^j X_j] \quad (*)$$

$$= cn \sum_{phase\ j} (\frac{3}{4})^j E[X_j] \quad \text{[LIN EXP]}$$

= E[# of coin flips N] = 2

$$\leq 2cn \sum_{phase\ j} (\frac{3}{4})^j$$

geometric sum, <= 1/(1-3/4) = 4

$$\leq 8cn = O(n)$$

Q.E.D.