Mid-term

2016年5月30日 23:13

by Chris

Department of Computing and Information Systems

COMP90038 Algorithms and Complexity

Mid Semester Test, Semester 1 2016

Student Number:

Tutorial (Day/Time):

Test Duration: 40 minutes.
Reading Time: 5 minutes.
Marks Available: 20 marks.
Length: This paper has 5 pages including this cover page.
Authorized Materials: None.

Instructions to Invigilators: Students will write all of their answers on this test paper. Students may not remove any part of the test paper from the examination room.

Instructions to Students: This paper counts for 10% of your final grade. All questions should be answered in the spaces provided on the test paper. You may make rough notes, and prepare draft answers, on the reverse of any page, and then copy them neatly into the boxes provided.

Calculators: Calculators are not permitted.

Q1:	Q2:	Q3:	Q4:

Question 1 (4 marks).

a. Order the following functions by increasing order of growth:

n^2 $n \log n$ $\sqrt{n}$ 2^n

③ ② ① ④

b. What is the time complexity of insertion sort on an already sorted, or almost-sorted, array of n elements?

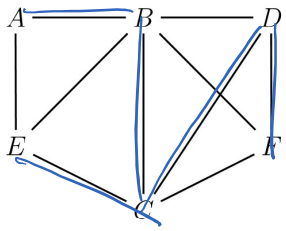
n

c. Show the result after one pass of Shellsort on the following list, if the gap sequence k=3.

16 4 17 13 9 6 8 5 10 11 7 3

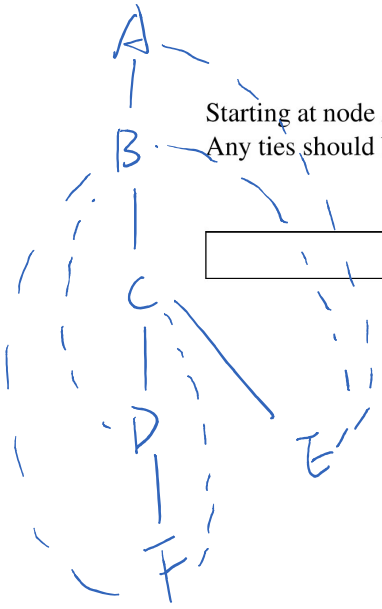
8 4 3 | 11 5 6 | 13 7 10 | 16 9 17

d. Consider the following graph:



Starting at node A, write down the order in which a depth-first search traverses the graph. Any ties should be resolved by taking nodes in alphabetical order.

A B C D F E



Question 2 (4 marks).

Consider the following algorithm:

function MYSTERY(A, n)
  //Input: Array A and integer n. A has size of least n
  if $n < 1$ **then**
    return 0
  else
    $temp \leftarrow 3 \times A[n-1] +$ MYSTERY$(A, n-2)$
    return $temp$

a. Write down recurrence equations that describe the running time of the algorithm. Assume that the basic operation has a cost k.

$$T(n) = \begin{cases} T(n-2)+k & n \geq 1 \\ 0 & n < 1 \end{cases}$$

b. Solve these recurrence equations and give the asymptotic time complexity of the algorithm. Use Big-O notation for your answer.

Without loss of generality, assume that n is even.

$T(n) = T(n-2)+k = T(n-4)+2k$
$= \cdots$
$= T(n-n)+\frac{n}{2}k$
$= \frac{n}{2}+k$

$T(n) \in O(n)$

Question 3 (6 marks).

a. Complete the following algorithm in pseudocode:

function FIND_VERTEX(G)
// Input: $G[0..n-1][0..n-1]$ the adjacency matrix of a directed graph.
Note: the index position number corresponds to the vertex number/label.
// Output: the vertex number of any vertex that has an out degree greater than zero.
If no such vertex exists, return -1
// Side Effects: none

```
for i ← 0 to n-1
    for j ← 0 to n-1
        if G[i][j] = 1 then
            return i
return -1
```

b. You have written a search application that uses binary search on a sorted array. In this application, all keys are unique. A co-worker has suggested speeding up access during searching by keeping track of recent unsuccessful searches in a *cache*, or store. He suggests implementing the cache as an unsorted linked-list.

Is this a good idea or a bad idea? Explain your answer.

Bad idea. Because items in an unsorted linked-list cannot be accessed randomly and therefore we need to check every item before we know whether a search fail recently or not.

Question 4 (6 marks).

A bitonic array is defined as an array `A[]` of size n in which the elements are strictly increasing up to and including index position m and then strictly decreasing. For example, the array `A=`$[2, 5, 11, 24, 8, 1]$ is bitonic where $n = 6$ and $m = 3$. Note `A[`m`]` is the maximum element in the array. You may assume that all elements in the array `A[]` are unique.

Write an algorithm in pseudocode to find the maximum element of a bitonic array. The input to your algorithm (function) will be the array `A[]` and two integer values `lo` and `hi` corresponding to the first and last positions in the array. The algorithm will return the value of `A[`m`]`.

Maximum marks will be awarded if your algorithm is correct and runs in $O(\log n)$ time. If your algorithm is correct and runs in $O(n)$ time you will be awarded 2 marks.

```
function BinaryFindMax(A[ ], lo, hi)
    if lo < hi-1
        m ← ⌊(lo+hi)/2⌋
        if A[m] > A[m+1] and A[m]> A[m-1] then
            return m
        else if A[m] > A[m+1] then
            BinaryFindMax(A, lo, m)
        else
            BinaryFindMax(A, m, hi)
    else
        return -1  // return -1 if it's not a bitonic array.

invoke the function by using BinaryFindMax(A, 0, n-1)
initially.
```