Sample exam
2016年5月30日 23:15

by Chris

The University of Melbourne
Department of Computing and Information Systems

COMP90038
Algorithms and Complexity
Sample Exam 2016*

**This is only a sample exam. The number of questions asked in each section in the final exam may vary. The distribution of marks could change depending on the questions.*

Identical examination papers: None

Exam duration: Three hours

Reading time: Fifteen minutes

Length: This paper has 5 pages including this cover page.

Authorized materials: No materials are authorized. Calculators are not permitted.

Instructions to invigilators: Students may not remove any part of the examination paper from the examination room.

Instructions to students: This paper counts for 70% of your final grade. ***All questions should be answered*** **by writing a brief response or explanation in the ruled boxes under the question**. The reverse side of any page may be used to make rough notes, or prepare draft answers.

Students ID number:

Examiners' use only:

Total [70]

Section A (13 marks): Answer all the questions

1. **(1 mark)** What is the fundamental difference between the *stack* ADT and a *queue* ADT?

a stack implements Last In First Out or LIFO policy, whereas a queue implements First In First Out or FIFO policy

2. **(2 marks)** Consider the following algorithm.

```
Algorithm MIN1(A[0..n-1])
    If n = 1 then return A[0]
    else temp ← MIN1(A[0..n-2])
        if temp <= A[n-1] then return temp
        else return A[n-1]
        end if
    end if
```

1. the smallest item of the array A[0..n-1]
2. Comparison
3. n-1
4. T(n)=T(n-1)+1=T(n-2)+2=……=T(1)+n-1=n-1

i. What does this algorithm compute?
ii. What is its basic operation?
iii. How many times is the basic operation executed?
iv. What is the efficiency class of this algorithm? Show your working.

θ(n)

3. **(2 marks)** Use the Master Theorem to find the order of growth for the recurrence
$T(n) = 4T(n/2) + n^2$

$T(n)=aT(n/b)+f(n)$, $f(n)\in\Theta(n^d)$

$T(n)=\begin{cases}\Theta(n^d) & \text{if } a<b^d\\ \Theta(n^d\log n) & \text{if } a=b^d\\ \Theta(n^{\log_b a}) & \text{if } a>b^d\end{cases}$ → $a=4$, $b=2$, $d=2$ → $a=b^d$ → $\Theta(n^2\log n)$

4. **(1mark)** What is the worst case complexity of selection sort? Use big-Oh notation. $O(n^2)$

5. **(1 mark)** What is the best case complexity of merge sort? Use big-Oh notation. $O(n\log n)$

6. **(4 marks)** For each of the following statements, indicate whether it is true or false:
   a. "Heap sort is a stable sorting algorithm". false
   b. "The worst case complexity of merge sort is $O(n^2)$. false
   c. "Insertion sort is a stable algorithm". true
   d. "The height of a complete binary tree with N nodes is N" false ($\log_2 N$)

7. **(2 marks)** Explain briefly when a sorting algorithm is said to be *in-place*

in-place if it does not require additional memory except, perhaps, for a few units of memory.
See Lecture 5 slides

Section B (32 marks): Answer all the questions

8. **(3 mark)** Sequential search can be used with about the same efficiency whether a list is implemented as an array or a linked list. Is this statement true or false for binary search? Justify your answer.

False. The time complexity of binary search using an array is O(log n); however, that of binary search using a linked listed is O(n), because unlike an array, where any element can be accessed in constant time, reaching the middle element in a linked list is a O(n) operation. So their time complexity are different.
https://umutzafer.files.wordpress.com/2012/01/solu4.pdf

9. **(3 marks)** You are managing a software project that involves building a computer-assisted instrument for medical surgery. The exact placement of the surgical knife is dependent on a number of different parameters, usually at least 25, sometimes more. Your programmer has developed two algorithms for positioning the cutting tool, and is seeking your advice about which algorithm to use:

Algorithm A has an average – case run time of n, and a worst case run time of n^4, where n is the number of input parameters.
Algorithm B has an average case run time of $n(\log n)^3$, and a worst case of n^2.

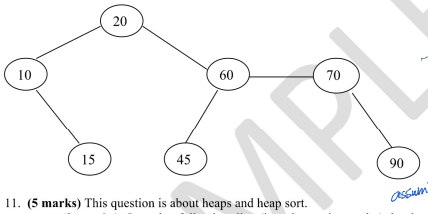
Which algorithm would you favour for inclusion in the software? Justify your answer.

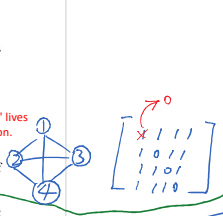
B. Surgery is related to life risk. We need to guarantee a better worst-case complexity, to avoid affect patients' lives as much as possible. Considering b has better worst-case performance, I will favour for inclusion in the situation.

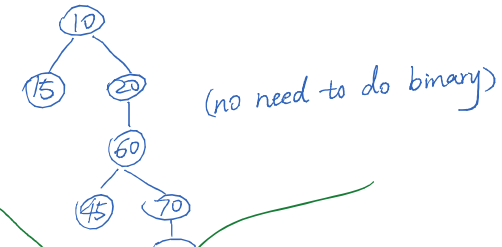
10. **(4 marks)** This question is about graph algorithms

   a. **(2 marks)** Sketch a graph that could be used to model the friendship network of a group of four people. Draw the corresponding adjacency matrix representation of your graph.

   b. **(2 marks)** Traverse the graph by Breadth First Search (BFS). You may start the traversal at vertex 10 and construct the corresponding BFS forest:

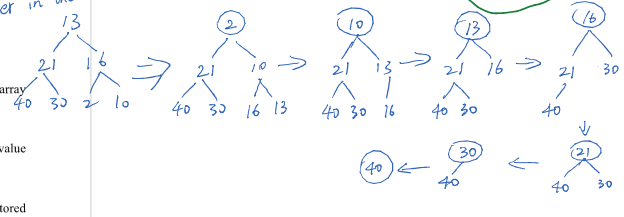






(no need to do binary)

here in form of a tree
remember in the exam should be in the form of an array.

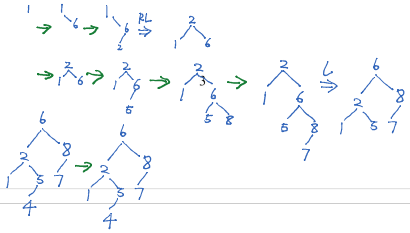


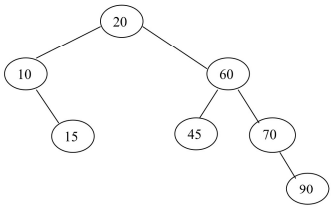
assuming using bottom-up

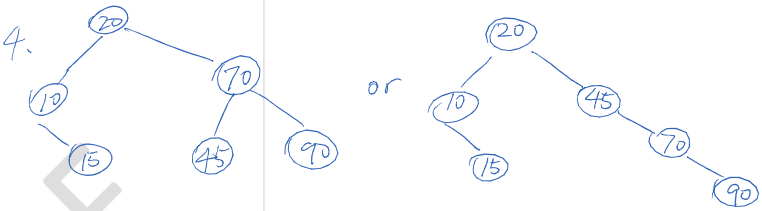
11. **(5 marks)** This question is about heaps and heap sort.
   a. **(3 marks)** Sort the following list (into increasing order) by heap sort, using array representation of heaps. Show your workings. 13, 21, 16, 40, 30, 2, 10
   b. **(2 marks)** State whether the following statements are true or false:
      i. A heap is a complete binary tree. true
      ii. The heap structure must satisfy either the structural constraint or the value relationship constraint. false. both

12. **(8 marks)** This question is about search trees
   a. **(2 marks)** What is the relationship between the value stored at a node and the values stored at its children in a Binary Search tree?
   b. **(2 marks)** Suggest one limitation of using a Binary Search Tree in search applications?
   c. **(2 marks)** Build an AVL tree for the list of items: 1 6 2 1 5 8 7 4 2
   d. **(2 marks)** Show one possible binary search tree that can result from deleting 60 in the following binary search tree:

1. Greater than its left child node but smaller than its right child node.
2. If the sequence of keys keep increasing/decreasing, the BST will be a stick, and it's the worst case, whose time complexity for searching is O(n). Then the benefit of BST cannot be taken advantage of, and its structure is more complicated than an array without showing better performance.
3. Assuming the AVL tree handles duplicates like a set, and duplicated keys will not be inserted.







13. **(9 marks)** This question is about hashing.
   a. Insert items with the following keys into a hash table of size 7
      12 7 6 8
      Use the hash function h(k) = k(k+3) mod 7.
      i. **(2 marks)** Use separate chaining for collision resolution. Show the hash value you have calculated for each input key, and show the hash table after all items have been inserted.
      ii. **(4 marks)** Use linear probing for collision resolution. Show the hash value you have calculated for each input key, and show the hash table after all items have been inserted.
      iii. **(1 mark)** For the resulting hash table using linear probing, how many probes are needed in a search for the key 5.
   b. **(2 marks)** Why is it not a good idea for a hash function to depend on just one letter (say the first one) of a natural language word?

5 0 5 4

i.

0	1	2	3	4	5	6
7				8	12	
					6	

ii.

0	1	2	3	4	5	6
7				8	12	6

iii. 4

No. 26 different addresses can such a hash function produce. it doesn't distribute keys evenly.

Textbook P269

- A hash table's size should not be excessively large compared to the number of keys, but it should be sufficient to not jeopardize the implementation's time efficiency (see below).
- A hash function needs to distribute keys among the cells of the hash table as evenly as possible. (This requirement makes it desirable, for most applications, to have a hash function dependent on all bits of a key, not just some of them.)

Section C (25 marks) Answer all the questions

14. (6 marks) Write a correct and detailed algorithm that takes as input a Binary Search Tree (BST) T and finds the second largest value in the given BST. You may assume that the BST has at least two nodes, and that all of the node values are distinct. You must use clear, appropriately commented pseudo-code, Java or C to write your algorithm. Then, analyze the time and space complexity of your algorithm. Show your working for the complexity analysis.

```
n ← head
while (n.right != null)
    p ← n
    n ← n.right.
if n.left = null
    return p.
else
    n ← n.left
    while n.right!=null
        n ← n.right.
return n.
```

time complexity O(logn)
space: O(1)

15. (9 marks) Building the east-west road link created a budget blowout for the Victorian Government even before the project started. However, in the quest to provide better road infrastructure for suburbs in Melbourne, the Victorian Government in partisan with the road authorities, have now decided to lay bus routes connecting suburbs between the east and west of Melbourne. The road authorities have asked the project manager to start work on this project. The project manager has called on you – the software engineer – to design and implement an algorithm that manages a bus router problem between suburbs in Melbourne. The road authorities want to have bus routes in the suburb such that there is at least one ~~bus route connecting every pair of~~ suburbs. Furthermore, as laying roads is very expensive, the road authorities want to minimize the total amount of bus routes they wish to lay. Your algorithm must provide the project manager with a shorted bus route layout such that all suburbs between the eastern and western suburbs are connected, thus lowering costs.

   a. What A*bstract Data Type (ADT)* should be used to help manage the bus-router problem?
   b. Write a clear, correct and detailed algorithm to solve this problem. Make sure you design the algorithm to accept relevant input and; produces an appropriate output that the project manager can use to identify the shortest bus route layout connecting suburbs. You must use clear, appropriately commented pseudo-code, Java or C to write your algorithm.

Min spanning tree. Prim ALgo

```
1 heap
2 function Prim(<V,E>)
        for each v ∈ V do
                cost[v] ← ∞
                prev[v] ← nil
        pick initial node v0
        cost[v0] ← 0
        Q ← InitPriorityQueue(V) ⊲ priorities are cost values
        while Q is non-empty do
                u ← EjectMin(Q)
                for each (u,w) ∈ E do
                        if weight(u,w) < cost[w] then
                        cost[w] ← weight(u,w)
                        prev[w] ← u
                        Update(Q,w,cost[w]) ⊲ rearranges priority queue
```

16. (10 marks) Let *A[0 ..n-1]* be an array of *n* distinct numbers. If *i < j* and *A[i] > A[j]*, then the pair *(i, j)* is called an ***inversion*** of A.

2-1 3-1 8-6 8-1 6-1
So (0,4) (1,4) (2,3) (2,4) (3,4)

a) List the inversions of the array (2, 3, 8, 6, 1)
b) What arrays of size *n* have the largest number of inversions and what is this number? Decreasing arrays. (n-1)*n/2
c) What arrays of size *n* have the smallest number of inversions and what is this number? increasing arrays. 0
d) Write two different algorithms that you could use to count the number of inversions in a list of *n* elements, one based on each of the following design strategies:
   (i) Brute force
   (ii) Divide and conquer
   You must use clear, appropriately commented pseudo-code, Java or C to write your algorithm.
e) Analyze the complexity of each algorithm and determine the efficiency class. You must show how you arrived at your answer.

```
Function BruteForceInversionCount(A[0..n-1])
        Count <- 0
        For i <- 0 to n-2 do
                For j <- I to n-1 do
                        If A[i] > A[j]
                                Count <- count+1
```

O(n^2)

```
function Mergesort(A[0..n − 1])
        Count <- 0
        if n > 1 then
                copy A[0..⌊n/2⌋ − 1] to B[0..⌈n/2⌉ − 1]
                copy A[⌈n/2⌉..n − 1] to C[0..⌈n/2⌉ − 1]
                R1 <- Mergesort(B[0..⌈n/2⌉ − 1])
                R2 <- Mergesort(C[0..⌈n/2⌉ − 1])
                R <- Merge(B,C,A)
                Count <- R1+R2+R
        Return count

function Merge(B[0..p − 1],C[0..q − 1],A[0..p + q − 1])
        Count <- 0
        i ← 0; j ← 0; k ← 0
        while i < p and j < q do
                if B[i] ≤ C[j] then
                        A[k] ← B[i]
                        i ← i + 1
                else
                        A[k] ← C[j]
                        Count <- count +(p-i)
                        j ← j + 1
                        k ← k + 1
        if i = p then
                copy C[j..q − 1] to A[k..p + q − 1]
        else
                copy B[i..p − 1] to A[k..p + q − 1]
```