

Algorithms and Complexity

Exam

14:00 - 17:00

June 23, 2004

Prof. Michael H. Böhlen

The exam consists of 6 exercises. The first exercise is a multiple choice exercise. Solve as many exercises as possible. It is important that you argue for your answers and that you present your solutions in a readable form. Fully specify all algorithms that you use as part of your solutions. During the exam no auxiliary material is permitted.

1 Multiple Choice Exercises (25 points)

1.1 What is the worst case asymptotic complexity of the following algorithms if run on an array with n elements? Check the corresponding table cells.

	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$
Bubble sort				
Heap sort				
Quick sort				
Binary search				
Linear search				

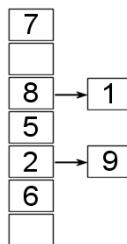
1.2 What approach is commonly used to implement the following algorithms? Check the corresponding table cells.

	Brute force	Divide and conquer	Dynamic programming
Longest common subsequence			
Linear search			
Binary search			
Merge sort			

1.3 Which of the following statements about a heap are true? Assume that the heap is implemented as an array.

- a) A heap is a complete binary tree.
- b) Each node in a heap is greater than or equal to all its children.
- c) It is possible to build a heap in linear time.
- d) Finding the parent of a node is fast.

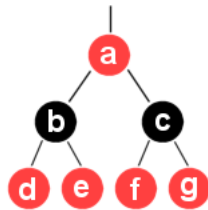
1.4 Consider the following hash table (in this example, element 7 is at position 0, element 6 is at position 5, etc).



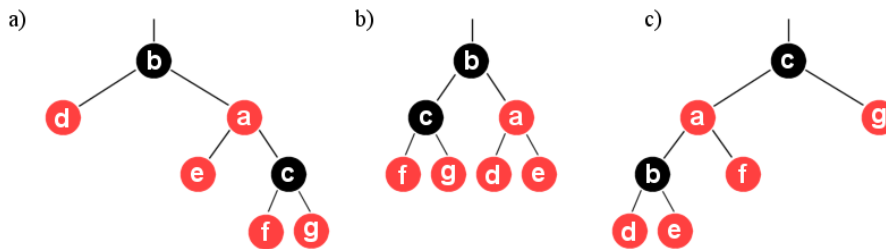
It was obtained by successively inserting the numbers 2, 7, 8, 6, 5, 9, 1 into the empty hashtable. Which hash function was used?

- a) $h(k) = 2k \pmod{7}$
- b) $h(k) = \lfloor 0.5k + 3 \rfloor \pmod{7}$
- c) $h(k) = k^2 \pmod{7}$

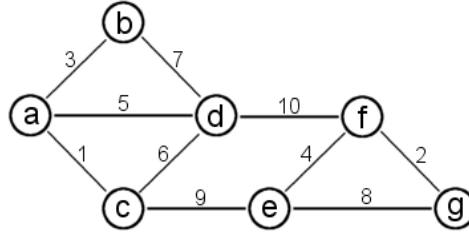
1.5 A right rotation was performed on the Red-Black subtree shown on the first picture below.



Which of three subtrees shown on the second picture is the result of the rotation?



1.6 Assume the following graph:



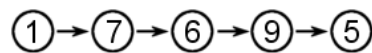
Which of the following listed edge sets is a result of running Kruskal's minimum spanning tree on the above weighted undirected graph?

- a) { fg, ab, ef, ad, cd, ce }
- b) { ac, fg, ab, ef, ad, ce }
- c) { ac, fg, ab, ef, ad, eg }
- d) { ab, ce, ac, fg, ad, ef }

1.7 Consider the following piece of code that manipulates a singly linked list:

```
p = head;
while(p->val != 6) { p = p->next; }
q = p->next;
p->next = p->next->next;
q->next = q->next->next;
p->next->next = q;
v = p->val;
p->val = p->next->val;
p->next->val = v;
while(q != NULL) {
    q->val = q->val + 1;
    q = q->next;
}
```

1.7.1 Before executing this code head points to the beginning of the following singly linked list:



What will be the result after the execution of the above code fragment?

- a) ①→⑦→⑥→⑨→⑤
- b) ①→⑦→⑥→⑤→⑩
- c) ①→⑦→⑤→⑥→⑩
- d) ①→⑦→⑤→⑥→⑨

1.7.2 Does the above algorithm work with all singly linked lists? If not describe the cases when it fails:

.....

2 Recurrence (20 points)

Consider the following recurrence:

$$T(1) = 4$$

$$T(n) = 2T(n/2) + 5n + 1 \quad (n \geq 2)$$

- What is the value of $T(n)$ for $n = 2, 4, 8$?
- Solve the above recurrence exactly by finding a closed form expression for it (without $T(n)$ on the left side of the expression).
- Find the simplest $f(n)$ such that $T(n) = \Theta(f(n))$.

3 Hashing (20 points)

Assume the hash function:

$$h(k, i) = (h_1(k) + ih_2(k)) \pmod m$$

with

$$h_1(k) = k \pmod m$$

$$h_2(k) = 1 + (k \pmod{(m-1)})$$

Assume you are using an array A of size m as a hash table to store positive integers. The hash table shall be used to insert elements into it and to delete elements from it. Explain the problems that might occur when deleting an element and describe how you can solve it.

1. Write a procedure for initializing the hash table.
2. Write a procedure for inserting an element into the hash table.
3. Write a procedure for deleting an element from the hash table.

4 Trees, C (15 points)

Assume arithmetic expressions that consist of integers, the four arithmetic operations +, -, *, and /, and parentheses. An example is the expression:

$$(51 + 6) * 72 - 9/3$$

1. Represent the above expression as a binary tree.
2. Give appropriate declaration in C to represent such a tree.
3. Write a C function that evaluates an arithmetic expression (represented as a tree) and returns the result. You can assume integer operations throughout.

5 Divide and Conquer (20 points)

Consider an array $A[a..b]$ of positive integers. Assume that $b - a + 1 = 2^k$. The tree representation of this array is a binary tree defined as follows. If $A[a] = A[a+1] = \dots = A[b] = k$ (i.e., all elements are equal) then the tree representation of A is a tree with just a root node with key value k and left and right subtree being equal to NULL. Otherwise the tree representation of A has a root with key value zero, the left subtree equal to the tree representation of the first half of the array ($A[a..(a+b-1)/2]$), and the right subtree equal to the tree representation of the second half of the array ($A[(a+b+1)/2..b]$).

Provide a divide and conquer algorithm that, given an array A , produces the tree representation of A . Your algorithm should run an instance of the BinTree ADT that has the following operations.

node* BinTreeInit(int val, node* left, node* right) Create and initializes a new binary tree B . The key value of the root node will be equal to val , the left subtree will be equal to $left$, and the right subtree will be equal to $right$.

int BinTreeVal(node* p) Returns the key of node pointed to by p . The result is undefined if $p = \text{NULL}$.

node* BinTreeLeft(node* p) Returns the left subtree of the node pointed to by p . If p is a leaf then the value is NULL.

node* BinTreeRight(node* p) Returns the left subtree of the node pointed to by p . If p is a leaf then the value is NULL.

6 Scheduling (30 points)

You have n jobs to be run on a single processor. Certain jobs depend on others, i.e., they cannot begin until all tasks they depend on are completed. For example, you may have jobs $\{1, 2, 3, 4, 5\}$ and have the information that 2 depends on 1 and 3, 4 depends on 1, and 5 depends on 3.

A *feasible schedule* is a permutation of the jobs such that when the jobs are performed in that order then whenever a job has started all those that it depends on have finished.

1. Give a general algorithm to find a feasible schedule if one exists, and run it on the aforementioned example. What is the complexity of your algorithm?
2. Give an example of a situation without a feasible schedule.
3. Assume that each job i is assigned a duration d_i . For instance

$$d_1 = 5, d_2 = 10, d_3 = 3, d_4 = 7, d_5 = 4$$

Give an algorithm to find the earliest possible time at which all jobs can be completed with an unlimited number of processors at your disposal. What is the complexity of your algorithm?