Midterm Examination Problem Sheet

TIME: 04/24/2009, 15:00–17:30

This is a open-book exam. You can use any printed materials as your reference during the exam. Any electronic devices are not allowed.
Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or get negative scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
Both English and Chinese (if suited) are allowed for answering the questions. We do not accept any other languages.

Thank you for coming to the DSA Night. The night is prepared with the help of our eight divisions: Organize, Budget, Entrance, Dance, Drama, Music, Video and Band. Each division includes a few members (questions). There are 20 questions, each worth 10 points—the full credit is 200 points. For the 20 questions, 8 of them are marked with * and are supposedly simple; 8 of them are marked with ** and are supposedly regular; 4 of them are marked with *** and are supposedly difficult.

1 Organize: Post-order or In-order?

(1) (10%, *) Consider a full binary tree of depth 4 with nodes numbered from 1 to 15 such that node i has a left child $2i$ and a right child $2i+1$. Do a post-order traversal in the full binary tree above and print out the node numbers visited.

(2) (10%, **) Assume that the in-order traversal of one binary tree to be `GDHBIELJMACKF` and its post-order traversal to be `GHDILMJEBKFCA`. Please draw the binary tree.

2 Budget: Expression or Unlimited?

(1) (10%, *) Draw the expression tree of the following infix expression with usual precedence and left association. Note that the tree should contain only the binary operators and the operands.

$$a * b + c/d * (e - f) + g$$

(2) (10%, *) What is the prefix notation of the expression above?

3 Entrance: Queue or Stack?

(1) (10%, *) Considering representing two stacks with one array of size 6 by putting the bottoms of the stacks in the two ends of the array. List the contents of the array after doing all of the following operations orderly. For locations without contents, please use a special character to represent them.

```
push(stack 1, 5); push(stack 1, 5); push(stack 2, 6); push(stack 1, 6);
    push(stack 2, 3); pop(stack 1); push(stack 2, 5); pop(stack 1);
```

(2) (10%, **) Write down an algorithm that uses one queue (with `enqueue` and `dequeue` operations) and *at most* $O(1)$ *of additional memory* to simulate one stack.

4 Dance: Balanced or not?

Consider an ordered array a of N *nonzero* integers $a_0 \le a_1 \le a_2 \le \cdots \le a_{N-1}$. Define the unbalanced-ness of a to be

$$(\text{number of positive elements in } a) - (\text{number of negative elements in } a).$$

(1) (10%, **) Write down an $O(\log N)$-time algorithm that computes the unbalanced-ness of a. Briefly describe why the algorithm is correct. (*Hint: think about binary search.*)

(2) (10%, ***) Consider only the cases with $N = 2^k$, rigorously prove that your algorithm is indeed $O(\log N) = O(k)$-time by either a mathematical induction on k or any other proof of your choice.

5 Drama: Complex or not?

For this question, you can only use the definition of $O(\cdot)$ in the textbook:

$$f(n) = O(g(n)) \text{ if and only if}$$
$$\text{there exist positive constants } c \text{ and } n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \text{ such that } n \geq n_0.$$

The lg means $\log_2$ here.

(1) (10%, *) Prove that $n = O(2^n)$.

(2) (10%, **) If $f(n) = O(g(n))$ and $g(n) \geq 2$ for $n \geq 1$. Prove that $\lg f(n) = O(\lg g(n))$.

(3) (10%, *) Prove that $\lg n = O(n)$. (*Hint: check results above.*)

(4) (10%, ***) Consider some $f(n)$ and $g(n)$ such that $\lg f(n) = O(\lg g(n))$ and $g(n) \geq 2$ for $n \geq 1$. Construct a counter-example to disprove that $f(n) = O(g(n))$.

6 Music: KMPlayer or not?

The key idea in the KMP string matching algorithm is the failure function of a pattern $p_0 p_1 \cdots p_{n-1}$.

$$f_p(j) = \begin{cases} \text{largest } i < j \text{ such that } p_0 p_1 \cdots p_i = p_{j-i} p_{j-i+1} \cdots p_j & \text{if such } i \geq 0 \text{ exists;} \\ -1 & \text{otherwise.} \end{cases}$$

(1) (10%, **) Given a failure function f_p for a pattern p with length 8 and consider $q = p_2 \cdots p_{n-1}$ (the tail of p) with length 6. Consider the following f_p.

j	0	1	2	3	4	5	6	7
$f_p(j)$	-1	-1	0	1	2	0	1	-1

What is f_q? Briefly justify your answers.

(2) (10%, **) Consider the same f_p above. Assume that we know, additionally, some elements of p.

j	0	1	2	3	4	5	6	7
p	a						b	d
$f_p(j)$	-1	-1	0	1	2	0	1	-1

What is the full pattern string p? Briefly justify your answers.

7 Video: Compress or not?

Run-length encoding is a very common way of data compression. Consider a vector

$$a = (0, 1, 1, 1, 3, 3, 3, 3, 5, 5, 6, 6, 1).$$

Its run-length encoding is an array of pairs like

$$[(1, 0), (3, 1), (4, 3), (2, 5), (2, 6), (1, 1)],$$

which means there is one "0", followed by three "1", followed by four "3", followed by two "5", followed by two "6", followed by one "1".

(1) (10%, *) Given the run-length encoding (as a length-M dense array of pairs) of a vector a of length N, write down an algorithm that computes its average $\frac{1}{N} \sum_{i=0}^{N-1} a_i$.

(2) (10%, **) Given the run-length encoding of two vectors a and b, both of length N, write down an algorithm that computes their inner product $\sum_{i=0}^{N-1} a_i b_i$.

8 Band: Factorize or not?

Decomposing a positive integer into multiplications of prime integer factors (so-called factorization) is important in many areas. For instance, $12 = 2 * 2 * 3$. Assume that there are N integers $1, 2, \cdots, N$. One way to represent the factorization is to maintain a linked list of those factors for every integer. For instance, 12 would be associated with a linked list $(2, 2, 3)$ and 6 would be associated with a linked list $(2, 3)$. As you can see, there is a waste in memory because the tail part of the linked list of 12 is essentially the same as the linked list of 6.

Dr. Fact thought of a data structure to eliminate the memory waste. He lets N link to M if and only if N/M is the smallest prime factor of N. So for instance, 12 links to 6, which links to 3, which links to 1, which links to nothing. Thus, starting from 12, we can get all its prime factors by moving in the order of $12, 6, 3, 1$. The links for integers 1 to 12 are as follows.

```
1->nothing; 2->1; 3->1; 4->2; 5->1; 6->3; 7->1; 8->4; 9->3; 10->5; 11->1; 12->6;
```

We will represent the links within an array *next* where $next[i]$ stores the integer that i links to.

(1) (10%, *) Given the *next* array of size N, write down an algorithm that determines whether a given integer X between 2 and N is a prime.

(2) (10%, **) The following algorithm computes the value of $next[X]$ for a given X.

```
COMPUTE-NEXT(integer X)
for i ← 2 to ··· do
  if X mod i = 0 then
    return X/i
  end if
end for
return 1
```

Describe the tightest upper bound for i to make the algorithm correct and briefly justify your answer.

(3) (10%, ***) For $X = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$ where p_i are primes, its number of positive divisors is

$$(n_1 + 1)(n_2 + 1) \cdots (n_k + 1).$$

Given the *next* array of size N, write down an efficient algorithm that computes the number of positive divisors for a given integer X between 2 and N.

(4) (10%, ***) Given the *next* array of size N and two integers X, Y that are both between 2 and N. Write down an algorithm that print out the links from $X \cdot Y$ (which can be larger than N) to 1. For instance, if $N = 20$, $X = 6$, $Y = 10$, the algorithm should print out `60->30->15->5->1`.