CS1356 Introduction to Information Engineering

Final, 2011/1/10
(3 pages, 11 questions, 110 points)

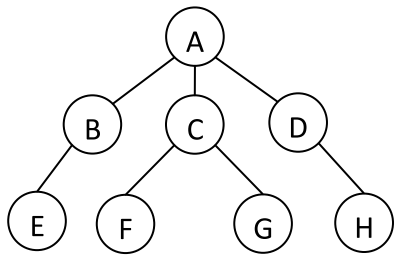
1. True or false (20pt)
   (a) The time complexity of using a linked list to implement the insertion sort is the same as that of using an array.
   True
   (b) In a binary tree, if the depth is *k*, the maximum number of nodes is 2^k-1.
   True (by the definition of ‘depth’ in textbook)
   (c) SQL (Structured Query Language) is a language for relational database to insert, remove, update and query data.
   True
   (d) Turing test is a software verification method.
   False
   (e) Interpreter does not need lexical analysis in language translation.
   False
   (f) The declarative programming language paradigm expresses the problem to be solved rather than the algorithm.
   True
   (g) All problems in class P are in class NP, and can be solved by Turing machines.
   True
   (h) Coupling means the internal binding within a module.
   False
   (i) Free software is software with zero charge.
   False
   (j) With the rapid improvement of artificial intelligence, one day the halting problem can be solved by Turing machines.
   False

2. Sort the following time complexity in the ascending order (6pt)
   $\Theta\ (n^{*}\log_2 n)$, $\Theta(n^{1.3})$, $\Theta\ (2^n)$, $\Theta\ (\log_2 n)$, $\Theta\ (n!)$, $\Theta\ (1)$

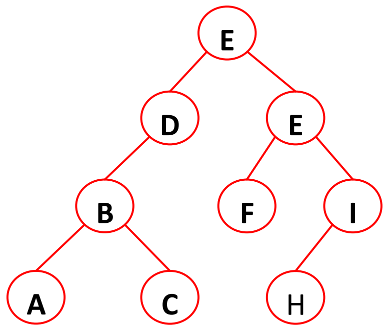
   $\Theta\ (1)$, $\Theta\ (\log_2 n)$, $\Theta\ (n^{*}\log_2 n)$, $\Theta(n^{1.3})$, $\Theta\ (2^n)$, $\Theta\ (n!)$

3. .Traverse the tree with (10%)
   (a) Breadth first order
   A B C D E F G H

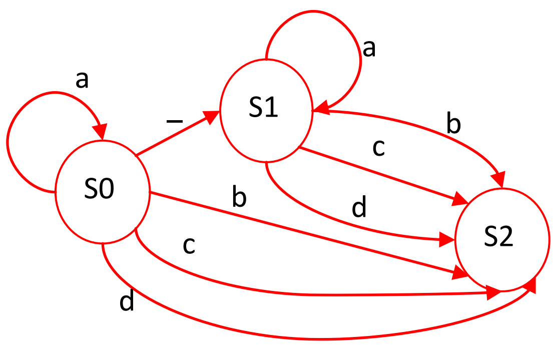
   (b) Depth first-postorder
   E B F G C H D A



4. The in-order traversal of a binary tree is ABCDEFGHI and the pre-order traversal is EDBACGFIH. Draw the tree. (10pt)



5. Draw the finite state machine for the regular expression (5pt)
`-?a*(b|c|d)`



6. Evaluate the prefix expression (5pt)
`- * / 15 - 7 + 1 1 3 + 2 + 1 1`

(15/(7– (1+1)))*3– (2+(1+1))=5

7. Please use the following statements to complete the circular queue implement. Assume the size of array is 5. (14 pt)
   (1) IsFull() is true
   (2) IsFull() is false
   (3) IsEmpty() is true
   (4) IsEmpty() is false
   (5) return Array[Head]
   (6) return Array[Tail]
   (7) Tail = Tail + 1
   (8) Head = Head + 1
   (9) Tail = (Tail+1)mod 5
   (10) Head = (Head+1)mod 5
   (11) Flag – 1
   (12) Flag + 1
   (13) (Flag + 1) mod 5

```
InitQueue() {
    Head = -1; Tail = 0;  Flag = 0;
}
IsEmpty() {
    If Flag is 0 then return true;
    Else return false;
}
IsFull() {
    If Flag is 5 then return true;
    Else return false;
}
Enqueue( input ){
    If (A) (1)IsFull() is true then
        return “Cannot Enqueue”;
    Else
        Flag = (B) (12)Flag + 1;
        Array[Tail] = input;
        (C) (9)Tail = (Tail+1)mod 5;
}
Dequeue(){
    If (D) (3) IsEmpty() is true then
        return “Cannot Dequeue”;
    Else
        Flag = (E) (11)Flag – 1;
        (F) (10)Head = (Head+1)mod 5;
        (G) (5)return Array[Head];
}
```

8. Given an unsorted integer array *A* of size *N*. Design a **recursive** algorithm that can report the number of elements in *A* which are larger than or equal to an integer *x*. For example, for *A*={3,1,-5,4,2}, *N*=5, *x*=3, the algorithm returns 2, because *A*[1]=3>=*x* and *A*[4]=4>*x*. (10pt) (partial credit will be given if your algorithm is not recursive.)

Design the count_ge_x function as follows.

```
    int count_ge_x(int A, int N, int x) {
        if N==0, return 0;
        if (A[1]>=x) then
            return 1+ count_ge_x(A[2..N], N-1, x);
        else
            return count_ge_x(A[2..N], N-1, x);
    }
count = count_ge_x(A[1..N], N, x)
```

9. Design an algorithm that can swap the values of two integers, say *a* and *b*, without additional temporary variables. (5pt)

```
a = a – b;
b = b + a;
a = b – a;    // there are many other ways to do it.
```

10. The following algorithm finds the nearest pair of *N* numbers (15pt)

```
1. Sort N numbers in the ascending order and place them to an array A.
2. Let min = A[2]-A[1], pair = (A[1],A[2])
3. For i=2 to N-1
4.     If A[i+1]-A[i] < min then
5.         Let min = A[i+1]-A[i] and pair = (A[i],A[i+1])
6.     End If
7. End For
8. Return pair
```

(a) Prove the correctness of the algorithm.

We first consider the case that all numbers are distinct. We only need to prove that the nearest pair of numbers are adjacent to each other in the sorted array A. The following proof is by contradiction, which first makes an assumption that is opposite to what we want to prove, and then finds a contradiction from the inferred statements.

我們先考慮所有數都不相同。我們只須證明最近的兩個數在排序好的陣列 A 中是一定是相鄰的兩個數。以下證明是用歸謬法，先假設要證的敘述不對，然後從中找出矛盾之處。

Assumption: assume the nearest pair, say (x_i, x_j), are not adjacent to each other in A.

⇨ There is at least a number between them, say x_k.

⇨ Without loss of generality, we assume $x_i<x_k<x_j$, since they are sorted and they are all distinct.

⇨ But $|x_j – x_i|>|x_j – x_k|$ and $|x_j – x_i|>|x_k – x_i|$, which contradicts the assumption that $|x_j – x_i|$ is minimum.

⇨ Therefore, if (x_i, x_j) is the nearest pair, then they must be adjacent to each other in A.

假設: 在 N 個各不相同的數中，最相近的兩點，假設是(x_i, x_j)，不在排序好的陣列 A 中相鄰的位置。

⇨ 存在一個數 x_k 界於(x_i, x_j)之間。在不失一般性的原則下，我們假設 $x_i<x_k<x_j$。

⇨ 但是這會導致$|x_j - x_i| > |x_j - x_k|$ 且 $|x_j - x_i| > |x_k - x_i|$，和我們的假設$|x_j - x_i|$是最小矛盾。

⇨ 所以在 N 個各不相同的數中，最相近的兩點必在排序好的陣列 A 中相鄰的位置。

Now we consider the case that there are same numbers in the list, in which the nearest pair can be one of any pairs of equal numbers. Since the same number will be placed together in A, one of them will be reported by the algorithm.

考慮有相同大小的數。他們必定會被 A 排在一起，而且他們距離為 0，所以演算法會回報一對相同大小的數。

******** 許多人用歸納法證，先假設 N=k 時對，再證 N=k+1 時對。所以把 k+1 數拆成 k 和 1 然後說最接近的數對不是在前 k 個數中就是 (A[k],A[k+1])。這是不對的!!!! 或說是是不夠的。A 是排序好的這條件沒用到。必須要說明為什麼最接近的數不會是(A[i],A[k+1]), i<k。***

(b) What is the time complexity of the algorithm? Justify your answer.

1. Depending what kind of sorting algorithm you used. If you use the merge sort, then it's $\Theta(N \log N)$. If you use insertion sort, it is $\Theta(N^2)$.
2. Statement 2-8, the time is $\Theta(N)$.

Therefore, the entire algorithm is $\Theta(N\log N)$ or $\Theta(N^2)$, depending the used sorting algorithm.

(c) Design a $\Theta(N)$ algorithm that finds the furthest pair of N numbers.

1. Find the largest number: $\Theta(N)$
2. Find the smallest number: $\Theta(N)$
3. Report the pair (the smallest number, the largest number)

11.Given a tree of N nodes, design a $\Theta(N)$ algorithm to find the furthest pair of nodes. The distance between two nodes on a tree is the number of links connected them. (10pt)

1. Traverse the tree and find the node with the largest depth, say node x. $\Theta(N)$
2. Use x as root, and traverse the tree again to find the node with the largest depth, say node y. $\Theta(N)$
3. Report the pair (x,y)

If you are interested in the proof, please come to talk to me.