101 學年度第 2 學期資格考 演算法

1. (15%) Explain the following terms
   (a) Greedy method
   (b) Longest common subsequence (LCS) problem.
   (c) Optimal substructure property
   (d) NP-complete
   (e) Minimum spanning tree

2. (15%) Answer the following questions:
   (a) Define smooth function.
   (b) Prove that $f(n) = n^k \log_b n$ is a smooth function.
   (c) If g is a non-decreasing function, f is a smooth function, and $g(n) = \Theta(f(n))$ for n a power of b, then $g(n) = \Theta(f(n))$.

3. (20%) Answer the following questions:
   (a) What is dynamic programming?
   (b) Write a dynamic-programming algorithm to calculate $C(n, k)$, the number of k-combinations (i.e., k-element subsets) of an n-element set. Use the formulas
   $$C(n, k) = C(n - 1, k - 1) + C(n - 1, k)$$
   valid for $1 \le k \le n - 1$, and
   $$C(n, n) = 1 = C(n, 0)$$
   valid for $n \ge 0$.
   (c) What is the worst-case time of your algorithm?
   (d) Trace your algorithm for $C(7, 5)$.

4. (15%) Shortest Path Problem:
   (a) **(5%)** Explain why greedy method cannot work on Shortest Path Problem by given a counter example.
   (b) **(5%)** Explain why we can use dynamic programming strategy to solve the problem. That is, please explain why the problem satisfies the optimal substructure property.
   (c) **(5%)** Can we use same strategy (dynamic programming) to solve Longest Path Problem, which tries to find a longest simple path between two nodes in a graph? Why or Why not?

5. (35%) Finding middle elements of an array:
Given a numeric array of size n X=$(x_1, x_2, ..., x_n)$, let the array Y=$(y_1, y_2, ..., y_m)$ is the resulting array if we sort over the array X in ascending order. Then, the middle elements of the array X can be defined as the element Y[$\lfloor n/2 \rfloor$], if n is odd. Or, the two elements Y[n/2 -1], and Y[n/2], if n is even. For example if X=(1,3,5,4,6,8,7) then the middle element is 5; if X=(1,3,5,4,6,7,9) then the middle elements are 5 and 6. Now, given two **sorted** array of size n A=$(a_1, a_2, ..., a_n)$ and B=$(b_1, b_2, ..., b_n)$, $a_i \le a_j, b_i \le b_j$ if $i \le j$, define C=A||B=$(c_1, c_2, ..., c_{2n}) = (a_1, a_2, ..., a_n, b_1, b_2, ..., b_n)$. Please find the middle elements of the array C. For example, let A=(1,3,5) and B=(3,4,5), then C=(1,3,5,3,4,5). The middle elements of the array C is 3 and 4.

(Note: you don't need to solve the sub-problems in order. You can skip some sub-problems and jump directly to those you try to solve.)

(i) **(3%)** The problem can be solved easy by sorting the array C first and find the middle elements. Please write a pseudo code to get the result based on the sorting. What is your algorithm complexity (in O (upper bound) or Θ (exact bound) notation).

(ii) **(4%)** You can improve the complexity of the algorithm to linear time O(n) by a better sorting algorithm, given the fact that both array A and array B are sorted. What is the algorithm (note: you cannot use the algorithm in (iii)). (hint: using merge)

(iii) **(28%)** You can improve the algorithm even further to O(log n) by prune strategy as follows.

A. Let the middle of array A is m_{a1} and m_{a2} (if the size of array A is odd then $m_{a1}=m_{a2}$). Similarly, let the middle of array B is m_{b1} and m_{b2}. Assume $m_a=(m_{a1}+m_{a2})/2$ and $m_b=(m_{b1}+m_{b2})/2$. Let the middle elements of C is m_{c1} and m_{c2}, $m_{c1} \le m_{c2}$. Show that $m_a \le m_{c1} \le m_{c2} \le m_b$ if $m_a \le m_b$, and $m_b \le m_{c1} \le m_{c2} \le m_a$ if $m_b \le m_a$. (6%)

B. Based on the definition of above (A), write a pseudo code for function **AvgMiddle(A, p, q)** with complexity O(1), which given a sorted array A, return $m_a=(m_{a1}+m_{a2})/2$. (m_{a1},m_{a2}) is the two middle elements of the sub-array A'=(A[p], A[p+1], A[p+2], …, A[q]) if size of A' (=q-p+1) is even, or $m_{a1}=m_{a2}$= the middle element of the sub-array A' if size of A' is odd. (3%)

C. Based on the definition of above ((A) to (B)), write a pseudo code for function **lowerHalfArrayBound(A, p,q)** with complexity O(1), which given a sorted array A, return the smallest index q', A[q'] $\ge m_a$ = **AvgMiddle(A, p, q)**,

return -1 if no such an element exist. Similarly, write a pseudo code for function **upperHalfArrayBound (A, p,q)** , which return the largest index p', A[p'] $\leq m_a$, return -1 if no such an element exist. (4%)

D. Based on the fact presented above ((A) to (C)), we, each time, can either remove lower half part of the elements in array A and upper half part of the elements in array B, or remove upper half part of the elements in array A and low half part of the elements in array B . The problem can be solve by algorithm FindMiddle(A, p1, q1, B, p2, q2) with initial input FindMiddle(A, 1, n, B, 1, n). Please complete the pseudo code of the algorithm FindMiddle(A, p1, q1, B, p2, q2). (6%)

```
FindMiddle(A, p1, q1, B, p2, q2){
    N1=(q1-p1+1); N2=(q2-p2+1);
    If(N1≤1 || N2≤1 || (N1+N2) ≤ K) //K is some small constant
    { solve it directly by (i) (sorting) or (ii) (merge), return the result.}
    ma= AvgMiddle(A, p1, q1); mb= AvgMiddle(B, p2, q2);
    if(ma≤mb) {
        q2'= lowerHalfArrayBound(B, p2,q2);
        p1'= upperHalfArrayBound (A, p1,q1)
        //some code here
    }else{
        // some code here
    }
        // some code here
};
```

E. Explain that in the algorithm FindMiddle(A, p1, q1, B, p2, q2) (in above (D)), if (N1+N2) ≤ K (K is a small constant value) (**Line 4**), the complexity to solve the problem, either by sorting (like (i)) or by merging (like (ii)), is O(1) (constant). (3%)

F. Let the complexity of FindMiddle(A, 1, n, B, 1, n) is F(n), show that F(n)=F(n/2)+O(1). (3%)

G. Base on (F), please show that the complexity of FindMiddle(A, 1, n, B, 1, n) is O(log n). (3%)