國立中山大學 102 學年度碩士暨碩士專班招生考試試題

科目名稱：資料結構【電機系碩士班丙組選考】　　　　題號：431004

※本科目依簡章規定「可以」使用計算機（廠牌、功能不拘）　　　　

注意事項

一、不可於試題紙上作答。

二、When you are asked to derive the running time of a program (or an algorithm) in the form of $O(f(n))$, the function $f(n)$ should be expressed in the ***simplest*** and ***tightest*** form. For example, if running time $T(n)=5n^2+4n$, you should write $T(n)=O(n^2)$. In other words, you will get 0 points if you write $T(n)=O(n^3)$ or $T(n)=O(5n^2+4n)$.

1. (a) [5 points] Given the size n of the input data, where n is a positive integer, we assume that the running time of a program is $O(f(n))$. State the *formal* definition of $O(f(n))$.

   (b) [5 points] Let $T_1(n)=\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}=\sum_{x=1}^{n}\frac{1}{x}$. Tell us the value of $\lim_{n\to\infty}T_1(n)$. Note that you will get 0 points if you just provide the answer. In other words, you *must* show your reasons.

   (c) [5 points] Given the size n of the input data, where n is a positive integer, we assume that a program requires the running time $T_1(n)=O(f(n))$. Derive the function $f(n)$.

   (d) [5 points] Let n be the radius of a circle C and $A(n)$ be the area of C. Let $T_2(n)=|\sin\theta|\times\log A(n)=O(f(n))$. Derive the function $f(n)$.

2. [5 points] Consider the following function **F** written in a C-like pseudo-code, which takes an array **A** of n positive integers and an initially-empty stack **S** as input parameters:

```
int F(A: array, S: stack) {
   int i, t=0;

   for (i=0; n-1; i++) {
      if (A[i]%2==0)
        push(S, A[i]);
      else
        while (S is not empty)
           t = t + pop(S);
   } // end of for-loop
   while (S is not empty)
      t = t + pop(S);
   return t;
}
```

What is the output (returned value) of the function **F** for the array `A = {2, 14, 4, 7, 11, 18, 10, 15, 6, 23, 12, 8}`?

3. Consider the following weighted graph $G(V,E)$ presented by the adjacency matrix.

$$G = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 40 & \infty & \infty & 7 & 21 \\ 40 & 0 & 5 & 12 & \infty & 11 \\ \infty & 5 & 0 & 10 & \infty & \infty \\ \infty & 12 & 10 & 0 & 18 & 14 \\ 7 & \infty & \infty & 18 & 0 & 33 \\ 21 & 11 & \infty & 14 & 33 & 0 \end{bmatrix} \end{array}$$

(1) [10 points] Please find the vertex sequence derived by DFS and BFS respectively. Note that we assume that node A is the root.
(2) [10 points] Please apply Kruskal's algorithm to drive the minimum cost spanning tree. Note that you *must* show your actions step by step.

4. The array `A` shown below is used to represent the complete binary tree.

i	1	2	3	4	5	6
A[i]	23	15	9	6	11	2

Please answer the following three questions:
(1) [5 points] Draw the corresponding binary tree T.
(2) [5 points] Is T a max heap? Is T a min heap? Explain your reasons.
(3) [15 points] Perform the following three heap operations sequentially: `INSERT(18)`, `INSERT(27)`, `DELETE` on T. Draw the resultant tree after each operation.

5. (a) [5 points] Given an unsorted integer array of size n, does the binary search algorithm outperform the sequential search algorithm? Use the big-O notation to justify your answer.

(b) [5 points] Given an integer array `A` of size n, the following pseudo-code shows the quick-sort algorithm. Note that we assume that all array elements in `A` are distinct and the function `medium(A)` will return the value `x` in `A` such that the number of integers that are larger than `x` is equal to the number of integers that are smaller than `x`. Besides, we assume that the running time of the function `medium(A)` is $O(n)$. Derive the worst case running time of `quicksort(A)` in terms of big-O notation. (Note that you will get 0 points if you just give the answer directly.)

```
void quicksort(A: array) {
  if (length(A)==0)
    return;
  x = medium(A);
  S = { y | y∈A and y < x };
  L = { z | z∈A and z > x };
  quicksort(S);
  print x;
  quicksort(L);
}
```

(c) [5 points] The following `enhanced_quicksort` algorithm written in a C-like pseudo-code tries to first split the array **A** into three partitions and then recursively sorts the arrays **S**, **M**, and **L**. Note that we carefully design two selection functions, `first_selection_function(A)` and `second_selection_function(A)`, to guarantee that `min(A)` < `x` < `y` < `max(A)`, where `min(A)` and `max(A)` denote the minimum element and the maximum element in array **A**, respectively. Moreover, we assume that the running time of each selection functions is $O(n)$. Please tell us whether the `enhanced_quicksort` algorithm is more efficient than the `quicksort` algorithm? Use the big-O notation to justify your answer.

```
void enhanced_quicksort(A: array) {
  if (length(A)==0)
    return;
  x = first_selection_function(A);
  y = second_selection_function(A);
  /* The above two selection functions together guarantee that
     min(A) < x < y < max(A).  */
  S = { a | a∈A and a < x };
  M = { b | b∈A and x < b < y };
  L = { c | c∈A and c > y };
  enhanced_quicksort(S);
  print x;
  enhanced_quicksort(M);
  print y;
  enhanced_quicksort(L);
}
```

6. (1) [5 points] Please convert the infix expression shown below into postfix form.

$A \times (B/(C-D)+E) \times \times F - G$

(2) [10 points] Tell us the results of preorder and postorder traversals for the following binary tree.