University of M'sila
Faculty of Mathematics and Computer Science
Department of Computer Science

جامعة المسيلة
كلية الرياضيات و الإعلام الآلي
قسم الإعلام الآلي

Exam : Data Structures and Algorithms 3 (DSA 3)

Duration: 1h 30 - Documents: not allowed **Academic year : 2023 / 2024**

Date : 17/01/2022 (10 : 30 – 12 : 00) **Corrected** **Level : L2** **Semester : 5**

Exercise 1 : (1.75 Points / 10 Minutes)

Consider the following function and the snapshot of its local variables in the stack frame (activation record) at **Stack Frame Snapshot**:

```
void main() {
 int i = 3, *p = 0, **r = 0;
 double d = 2.0, e = 0.0, *q = &d;
 p = &i;                    (0.25 pts)
 r = &p;                    (0.25 pts)
 **r = 4;                   (0.25 pts)
 // Stack Frame Snapshot
 printf("Line 1: %p %p %p\n", &i, &d, &e);
 printf("Line 2: %d %lf %p\n", *p, *q, *r);
 printf("Line 3: %p %p %p\n", p, q, r);
 printf("Line 4: %p %p %p\n", &p, &q, &r);
}
```

Stack Frame of main()

at Stack Frame Snapshot

Memory Address	Stack	Stored Variable
0x007AF960	0x007AF96C	p
0x007AF964	0x007AF960	r
0x007AF968	0x007AF970	q
0x007AF96C	4	i
0x007AF970	2.000000	d
0x007AF978	0.000000	e

Q1) Fill up the blank instructions in the program **(0.25 × 3 = 0.75 pts)**

Q2) Write the output of the function in the blank lines below. **(0.25 × 4 = 1 pt)**

Line 1: 0x007AF96C 0x007AF970 0x007AF978 **(0.25 pts)**

Line 2: 4 2.000000 0x007AF96C **(0.25 pts)**

Line 3: 0x007AF96C 0x007AF970 0x007AF960 **(0.25 pts)**

Line 4: 0x007AF960 0x007AF968 0x007AF964 **(0.25 pts)**

Exercise 2 : (5 Points / 20 Minutes)

Each of the two functions **P1** and **P2** calculates the value of a polynomial $\boldsymbol{p(x) = a_1 + a_1 x + a_2 x^2 + \cdots + a_n x^n}$ for a value of $\boldsymbol{x}$.

The degree $\boldsymbol{n}$ is an integer and the coefficients $\boldsymbol{a_i}$ are stored in an array $\boldsymbol{a}$[] of type float.

```
float P1 (int a[] , int n , float x){
     s = a[0] ;
     for (int i=1 ; i<=n ; i++)
          s+ = a[i]* pow(x,i) ;
     return s ;
}
```

```
float P2 (int a[] , int n , float x){
     s = a[0] ; q = 1;
     for (int i=1 ; i<=n ; i++) {
          q = q * x ;
          s += a[i]*q ;
     }
     return s ;
}
```

Q1) Calculate the number of multiplications $T1(n)$ that function **P1** performs as a function of n and then express its complexity in **Big-O notation**. **(1 pt)**

$$T1(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} = O(n^2).$$

Q2) Calculate the number of multiplications $T2(n)$ that function **P2** performs as a function of n and then express its complexity in **Big-O notation**. **(1 pt)**

$$T2(n) = 2 + 2 + \cdots + 2 = 2n = O(n).$$

Q3) What can we deduce? **(0.5 pts)**

P2 is better than P1.

Q4) Write a function **P3** that gives the same result in n multiplications. **(1.5 pts)**

```
float P3 (int a[] , int n , float x){
     s = a[n] ;
     for (int i=n ; i > 0 ; i--)
          s = s*x + a[i-1] ;
     return s ;
}
```

Q5) Show that $\boldsymbol{log(n!) \in \mathcal{O}(n\, log\, n)}$. **(1 pt)**

$$log(1) \leq log\, n$$

$$\log(2) \leq \log n$$

$$\vdots \quad \vdots \quad \vdots$$

$$log(n) \leq log\, n$$

Adding up both the left solution and the right solution, we get

$$log(1) + log(2) + \cdots + log(n) \leq log\, n + log\, n + \cdots + log\, n$$

$$log(1 \times 2 \times \cdots \times n) \leq n\, log\, n$$

$$log(n!) \leq n\, log\, n$$

Hence $\boldsymbol{log(n!) \in \mathcal{O}(n\, log\, n)}$

<u>Exercise 3 :</u> (2.75 Points / 15 Minutes)

This is the beginning of a sorting algorithm trace:

10	56	-2	52	-8	41	13
-8	56	-2	52	10	41	13
-8	-2	56	52	10	41	13
-8	-2	10	52	56	41	13

Q1) Which algorithm is it? **(0.75 pts)**

This algorithm is a **selection sort** : at each stage, the smallest unordered element is put in its place and exchanged with the one whose place it has taken.

Q2) Complete the table with the following steps, until the table is completely sorted. **(0.25 × 3 = = 0.75 pts)**

10	56	-2	52	-8	41	13	
-8	56	-2	52	10	41	13	
-8	**-2**	56	52	10	41	13	
-8	-2	**10**	52	56	41	13	
-8	-2	10	**13**	56	41	52	**(0.25 pts)**
-8	-2	10	13	**41**	56	52	**(0.25 pts)**
-8	-2	10	13	41	**52**	56	**(0.25 pts)**

Q3) Write a function sorting the array $\boldsymbol{Tab}$ of size $\boldsymbol{n}$ using this algorithm. **(1 pt)**

```
void selection_sort(int Tab[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int min_index = i;
        for (int j = i + 1; j < n; j++) {
            if (Tab[j] < Tab[min_index]) {
                min_index = j;
            }
        }
        swap(Tab[i], Tab[min_index]);
    }
}
```

Q4) What is the complexity of this algorithm as a function of $\boldsymbol{n}$? **(0.25 pts)**

The complexity of selection sort is $\mathcal{O}(\boldsymbol{n^2})$.

Exercise 4 : (5 Points / 20 Minutes)

We want to write a function `void separation(int Tab[], int n, int x)`, which applied to an array $\boldsymbol{Tab}$ of $\boldsymbol{n}$ integers, moves the elements of $\boldsymbol{Tab}$ such that :

- All elements less than $\boldsymbol{x}$ are placed at the beginning of the array $\boldsymbol{Tab}$
- All elements greater than $\boldsymbol{x}$ are placed at the end of array $\boldsymbol{Tab}$
- All elements equal to $\boldsymbol{x}$ are placed between these two zones.

For example, if $\boldsymbol{Tab} = [\mathbf{0},\mathbf{4},\mathbf{2},\mathbf{3},\mathbf{1},\mathbf{0},\mathbf{2},\mathbf{1},\mathbf{0},\mathbf{3},\mathbf{4},\mathbf{3},\mathbf{1},\mathbf{2},\mathbf{0}]$ and $\boldsymbol{x} = \mathbf{3}$ after calling `separation(Tab,15,3)` we get $\boldsymbol{Tab} = [\mathbf{0},\mathbf{0},\mathbf{2},\mathbf{1},\mathbf{0},\mathbf{2},\mathbf{1},\mathbf{0},\mathbf{2},\mathbf{1},\mathbf{3},\mathbf{3},\mathbf{4},\mathbf{4}]$.

Each element of $\boldsymbol{Tab}$ will be observed **only once** during the execution of the `separation` function.

Q1) Under what conditions will the `separation` function produce **fewer** than **3** different zones in the modified $\boldsymbol{Tab}$ array? **(0.25 × 4 = 1 pt)**

Assuming that the array $\boldsymbol{Tab}$ contains **more than 2 different** values, the separation function will produce fewer than 3 zones if : **(0.25 pts)**

- $\boldsymbol{x}$ is the minimum of the elements in $\boldsymbol{Tab}$ **(0.25 pts)**
- $\boldsymbol{x}$ is the maximum of the elements in $\boldsymbol{Tab}$ **(0.25 pts)**
- $\boldsymbol{x}$ is not an element of $\boldsymbol{Tab}$ **(0.25 pts)**

Q2) Complete the code for the `separation` function below (You can use the **`swap`** function without needing to redefine): **(2.5 pts)**

```
1. void separation (int Tab[], int n, int x){
2.   int i=0,j=0,k=n-1;
3.   while ( j<=k ){                          (0.5 pts)
4.        if (Tab[j]<x){
5.             swap(Tab[j], Tab[i]);          (0.5 pts)
6.             j++;                           (0.25 pts)
7.             i++;                           (0.25 pts)
8.        }
9.        else{     //1
10.            if (Tab[j]>x) {
11.                 swap(Tab[j], Tab[k]);          (0.5 pts)
12.                 k--;                           (0.25 pts)
13.            }
14.            else
15.                 j++;                           (0.25 pts)
16.           } //else 1
17.       }
18. }
```

Q3) What is the property verified by integer ***i*** throughout the execution of the `separation` function? **(0.5 pts)**

i is the index of the first element of Tab (in ascending index order) that is not less than ***x***.
$Tab[i-1] == x \ if(i-1) >= 0$

Q4) Same question for ***j***. **(0.5 pts)**

j is the index of the first element of Tab not yet processed by the algorithm.

Q5) Same question for ***k***. **(0.5 pts)**

k is the index of the last element of Tab not yet processed by the algorithm.

$Tab[k+1] > x \ if \ k+1 <= n-1$ When $j > k$ all elements of Tab are processed and in their place.

Exercise 5 : (5.5 Points / 20 Minutes)

Q1) Suppose a binary tree is **5** layers deep. What is the maximum number of nodes that can fit in this tree before a new layer needs to be added? **(0.5 pts)**

$$L = 5 \Longrightarrow 2^{L+1} - 1 = 2^{5+1} - 1 = 63$$

OR

$$1 + 2 + 4 + 8 + 16 + 32 = 63$$

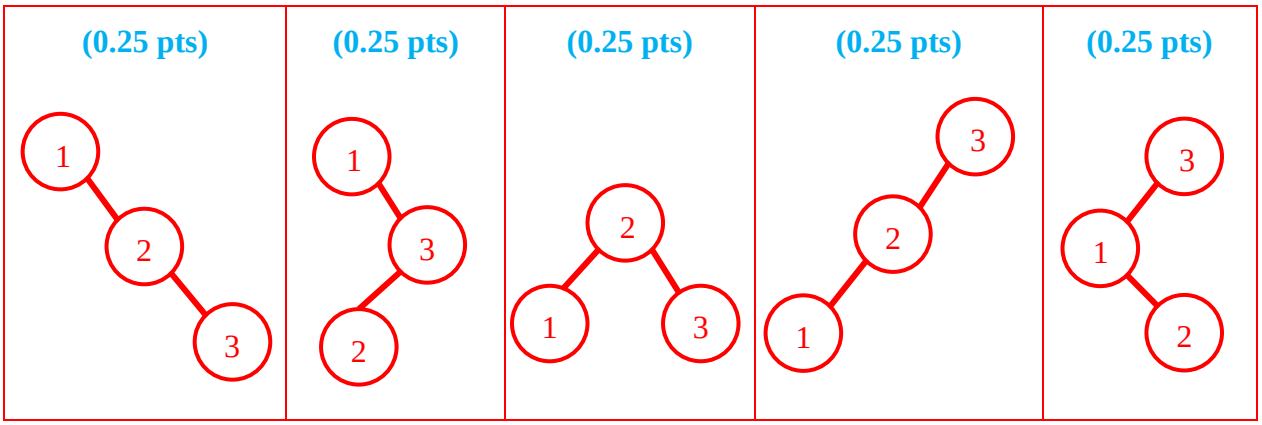
Q2) What is the minimum height of a binary tree containing **35** nodes? **(0.25 pts)**

5

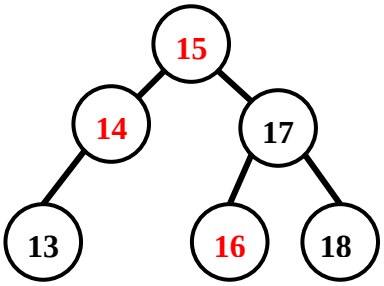
Q3) What is the minimum height of a general tree containing **30** nodes? **(0.25 pts)**

1

Q4) Draw all the binary search trees on the keys **1**, **2** and **3**. **(0.25 × 5 = 1.25 pts)**



Consider the following (incomplete) binary search tree that stores different integer values :



Q5) Assuming the top node is referenced by root,

what is the value of : `root->left->left->data` **(0.25 pts)**

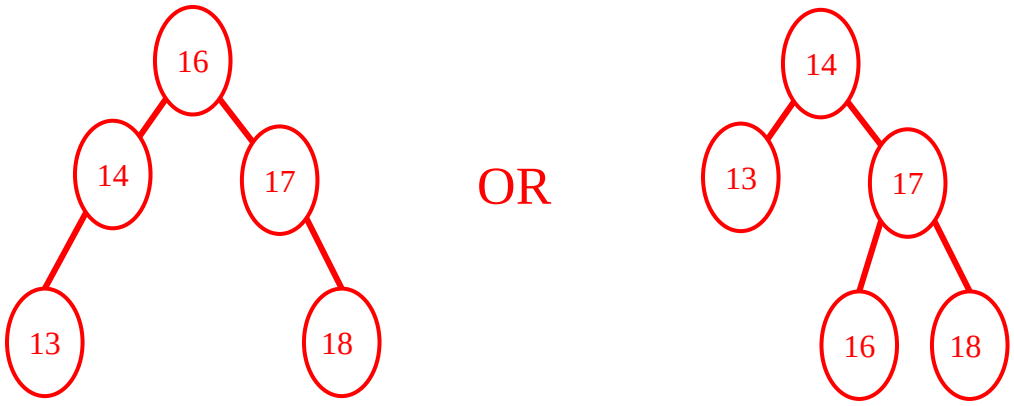
13

Q6) Fill in the missing integer numbers into the blank nodes. **(0.5 pts)**

Q7) Write down the sequence of vales produced by a post-order traversal, assuming values have been filled in as necessary. **(0.5 pts)**

13, 14, 16, 18, 17, 15

Q8) Remove the root, and draw the tree. **(0.5 pts)**



A node in a binary tree is defined as follows:

```
struct BTNode {
    int data;
    BTNode *left;
    BTNode *right;
};
typedef BTNode *BTREE;
```

Q9) Write a **Max_BT_It()** function for finding (searching) a maximum element in a binary tree **without recursion (iterative)**. **(1.5 pts)**

```
BTNode* Max_BST(BTREE BTree) {
    if (BTree == NULL) {
        return NULL;
    }
    while (BTree -> right) {
        BTree = BTree -> right;
    }
    return BTree;
}
```

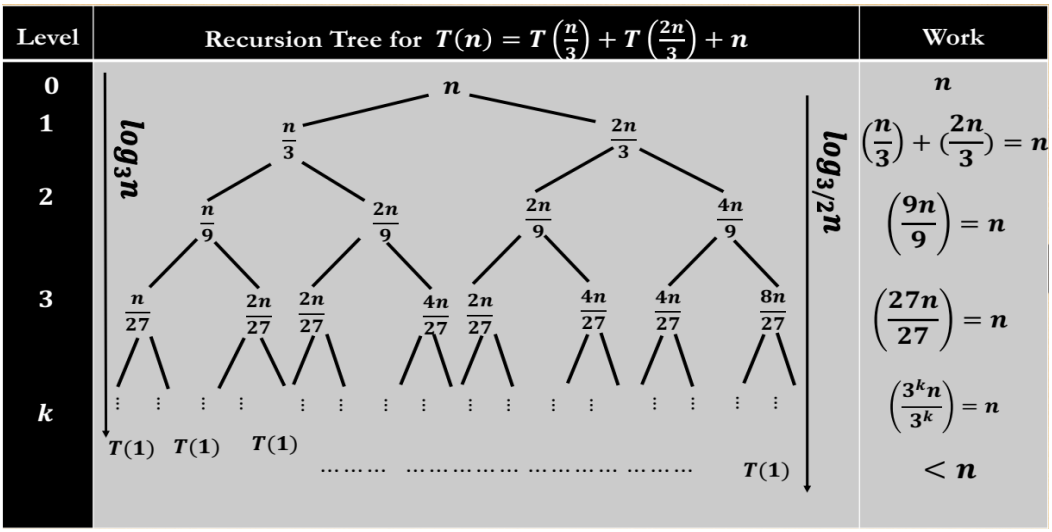
Exercise Bonus : (2 Points / 5 Minutes)

Give the recursive relation $T(n)$ for the running time of the following code snippet and solve its complexity using the recursive substitution method. **(1 + 1 = 2 pts)**

```
int bar(int n){
    if (n<=10){
        return 0;
    }
   int r = 0;
    for(int i=0; i<n; i++){
        r++
    }
    return bar(n/3) + bar(2n/3);
}
```

$$T(n) = \begin{cases} 1 & \text{if } n \leq 10 \\ T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n & \text{otherwise} \end{cases}$$ **(1 pt)**

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$



$$Total = \underbrace{n + n + n + \cdots + n + n}_{log_{3/2} n \text{ Times}} = n\, log_{3/2} n = \mathcal{O}(n \log n)$$ **(1 pt)**

Good Luck