Data Structures and Algorithms

Assignment 1(PART 1)

Jan 1 Semester

Name: Jeslyn Ho Ka Yan:

ID: 8535383

Date of Submission: 29 Jan 2024

Table of Content

1 QUESTION 1 3
2 QUESTION 2 4
3 QUESTION 3 5
4 QUESTION 5 10
4.1 Question 5 (i) 10
4.2 Question 5 (ii) 11

1 Question 1

Constant	$6+n^0$
Log-logarithmic	lg($2^{\lg\lg n}$)= log(log n), $3^{log3\lg n}+n^{2/3}$ = log n = $n^{2/3}$
Logarithm	log(n!),
Polylogarithmic	$\frac{1}{3}\,(log_3 n)^3$
Radicals	-
Linear	-
Linearithmic	lg$(n)^n$
Polynomial- Quadratic	$(1+2+3+\cdots+n)$, $16^{\lg n}=(2^4)^{\lg n}=2^{2\lg n(n^2)}=(2^{\log n^n})^2$ $=(n^2)^2=n^4$
Polynomial- Cube	$2^{3\lg n}$
Exponentiation	$16^{\lg n}$, $2^{\lg n}\times(log_2 n)^0$, 2^{3n}
Factorial	$(n^2+3)!$

2 Question 2

```
Static void doIt (int n) {
      int i
      int j ← (2 × n)
      loop while ( j > 0 ) {
            i ← n
            loop while ( i >= 1 ) {
                  i ← i / 2
            }
            j ← j - 1
      }
}
```

1
2n
2n
$Log_2(n)$
2n

```
Static int myMethod (int n) {
      sum ← 0
      for i ← 1 to n {
            sum = sum + doIt(i)
      }
      return 1
}
```

1
n
$n*(1+6n+ Log_2(n))$
1

DoIt(It) $= 1+2n+2n+Log_2(n)+2n$
$= 1+6n+ Log_2(n)$

$T(n) = n*(1+6n+ Log_2(n))+1+ n+ 1$
$= n*(1+6n+ Log_2(n))+2+ n$

Ans: O n(n log(n)) = **O (n^2 log(n))**

3 Question 3

3.1 Pseudocode

```
function findMax(unsortedList, max=NEGATIVE_INFINITY):
    if unsortedList is empty:
        return max

    // Randomly select an index from the unsortedList
    index = random.nextInt(size of unsortedList)

    // Retrieve the element at the randomly selected index
    element = unsortedList[index]

    // Update the max value if the current element is greater
    max = maximum(max, element)

    // Create a new list without the selected element
    updatedList = removeFromList(index, unsortedList)

    return findMax(updatedList, max)

function removeFromList(indexToBeRemove, unsortedList):
    newList = []

    for i = 0 to size of unsortedList - 1:        // Iterate through the original list
        if i is != to indexToBeRemove:
            newList.add(unsortedList[i])

    return newList
```

Extra (java)

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class FindMaxInt {

    static Random rand = new Random();

    public static void main(String[] args) {
        List<Integer> unsortedList = new ArrayList<>(); // Your list initialization here
        Integer max = Integer.MIN_VALUE;

        System.out.println(findMax(unsortedList, max));
    }
```

```java
private static Integer findMax(List<Integer> unsortedList, Integer max) {
    if (unsortedList.isEmpty())
        return max;

    // Randomly select an index from the unsortedList
    int index = rand.nextInt(unsortedList.size());

    // Retrieve the element at the randomly selected index
    Integer element = unsortedList.get(index);

    // Update the max value if the current element is greater
    max = Math.max(max, element);

    // Remove the selected element from the unsortedList
    unsortedList = removeFromList(index, unsortedList);

    return findMax(unsortedList, max);
}
```

```java
private static List<Integer> removeFromList(int indexToBeRemove, List<Integer> unsortedList) {
    List<Integer> newList = new ArrayList<>();

    for (int i = 0; i < unsortedList.size(); i++) {
        if (i != indexToBeRemove) {
            newList.add(unsortedList.get(i));
        }
    }

    return newList;
}
```

3.2 Caculate the Running times in terms of Θ notation

function findMax(unsortedList, max=NEGATIVE_INFINITY):	1
if unsortedList is empty:	a
return max	
index = random.nextInt(size of unsortedList)	n
element = unsortedList[index]	n
max = max(max, element)	n
unsortedList = removeFromList(index, unsortedList)	1+ (2n+c+1)
return findMax(unsortedList, max)	b

function removeFromList(indexToBeRemove, unsortedList):	1
newList = []	
for i = 0 to size of unsortedList - 1:	n-1
if i is != to indexToBeRemove:	n
newList.add(unsortedList[i])	c
return newList	1

---

removeFromList(indexToBeRemove, unsortedList):
T(n) = 1+(n-1)+n+c+1
= 2n+c+1
findMax(unsortedList, max):
T(n) =1+a+n+n+n+1+(2n+c+1)+b
= 5n+3+a+b+c

Where:
a= 0 or 1 or 0.5 (best case is 1, worst case is 0, average case is 0.5)
b= 0 or 1 or 0.5 (**best case is 0**, **worst case is 1**, average case is 0.5)
c= 0 or 1 or 0.5 (best case is 1 worst case is 0, average case is 0.5)

Worst Case Scenario 0(n)
T(n) =5n+3+(0)+(1)+(0) = 5n+ 4

Average Case Scenario 0(n)
T(n) =5n+3+(0.5)+(0.5)+(0.5) = 5n+4.5

Best Case Scenario 0(n)
T(n) =5n+3+(1)+(0)+(1) = 5n+ 5

4 Question 4

1. $\boldsymbol{T(n)=4T(\frac{n}{2})+n^2+n,\ \ and\ T(1)=1}$

$T(n)=\mathrm{aT}(\frac{n}{b}) + \mathrm{f}(n)$
a =4, b =2, f(n)=n^2+n

Lets compare f(n) with $n^{\log_b a}= n^{\log_2 4}=n^2$
f(n)= n^2+n

Since f(n) and $n^{\log_2 4}$ are in the same order, we will use case 2 of Master Theorem
Therefore, $4T(\frac{n}{2})+n^2+n$ is $O(n^2 \log n)$.

2. $\boldsymbol{T(n)=2T(\frac{n}{2})+n\ lg^3 n, and\ T(1)=1}$

a=2, b=2, c=1, p=3
$\frac{a}{b^c} = \frac{2}{2^1} = 1$, use Case 2 from the Master Theorem

Since, p=3, use.
T(n) $= O(n^{\log_b a} \times \log_b^{p+1} n)$

$= O(n^{\log_2 2} \times \log_2^{3+1} n)$

$= O(n \log_2^4 n)$

3. $\boldsymbol{T(n)=3T(\frac{n}{2})+n\ lg\ n, and\ T(1)=1}$

a=3, b=2, c=1, p=1
c

use,
T(n) $= O(n^{\log_b a})$
$= O(n^{\log_2 3})$

4. $T(n)=4T(\frac{n}{4})+n\ lg\ n, and\ T(1)=1$

$T(n)=4T(\frac{n}{4})+n\ lg\ n$
$T(n)=4T\ [4\text{T}\ (\frac{n}{4^2})+(\frac{n}{4})\text{lg}(\frac{n}{4})]+n\ lg\ n$
$T(n)=4^2T\ (\frac{n}{4^2})+\text{n(lg n}-2)+n\ lg\ n$
$T(n)=4^2T\ (\frac{n}{4^2})+\text{2n lg n}-\text{2n}$

$T(n)=4^2T\ [4\text{T}\ (\frac{n}{4^3})+(\frac{n}{4^2})\text{lg}(\frac{n}{4^2})]+\text{2n lg n}-\text{2n}$
$T(n)=4^3T\ (\frac{n}{4^3})+\text{n(lg n}-4)+\text{2n lg n}-\text{2n}$
$T(n)=4^3T\ (\frac{n}{4^3})+\text{3n lg n}-\text{6n}$

$T(n)=4^3T\ [4\text{T}\ (\frac{n}{4^4}))+(\frac{n}{4^3})\text{lg}(\frac{n}{4^3})]+\text{3n lg n}-\text{6n}$
$T(n)=4^4T\ (\frac{n}{4^4})+\text{n(lg n}-6)+\text{3n lg n}-\text{6n}$
$T(n)=4^4T\ (\frac{n}{4^4})+\text{4n lg n}-\text{12n}$

Genralise the Expansion → $4^k\text{T}\ (\frac{n}{4^k})+\text{klg n}-(\text{k (k}-1))\text{n}$
The recursive call will stop when $(\frac{n}{4^k})=1$, we have $n = 4^k$, hence $k=\log_4 n$

Substitute k into the genralise Expansion equation,
$4^{\log_4 n}\ \text{T}\ (\frac{n}{4^{\log 4n}})+(\log_4 n)\ \text{lg n}-(\ \log_4 n\ (\log_4 n-1))\text{n}$
$= \text{nT}\ (\frac{n}{n})+\text{n}\times\text{lg n}-(\ \log_4 n\ (\log_4 n-1))\text{n}$
$= \text{nT}\ (1)+\text{n}\times\text{lg n}-\text{n}(\log_4 n-1)\text{n}$

The running time complexity is $O=(\text{n log n})$

5. $\boldsymbol{T(n)=T(n-1)+n^2, and\ T(0)=1}$

$T(n)= T(n-1)+n^2$
$T(n)= T\ [((n-1)\ -1)+\ (n-1)^2]\ +n^2$
$T(n)= T(n-2)+\ (n-1)^2+n^2$
$T(n)= T\ [((n-2)\ -1)+\ (n-2)^2]\ +\ (n-1)^2+n^2$
$T(n)= T(n-3)+\ (n-2)^2\ +(n-1)^2+n^2$
$T(n)=$ .....
$T(n)=1^2+2^2+......+(n-1)^2+n^2$ ← Sum of Squares= $\frac{n(n+1)(2n+1)}{6}$

$T(n)= T(0)\ +\frac{n(n+1)(2n+1)}{6}$
$T(n)= 1+\ \frac{n(2n^2+2n+n+1)}{6}$
$T(n)= 1+\ \frac{2n^3+3n^2+n}{6}$

The running time complexity is O=(n^3)

s

5 Question 5

5.1 Question 5 (i)

```
function enchantedForestGame(n):
    adjacencyMatrix = initializeMatrix(n)
    winners = set()
    meetingPairs = set()

    function meet(i, j):
        adjacencyMatrix[i][j] = 1
        adjacencyMatrix[j][i] = 1
        meetingPairs.add((i, j))

    function findWinner():
        for each row in adjacencyMatrix:
            if all elements in the row = 1 and the corresponding player is != in winners:
                winners.add(player)
                if winner.length== n
                    return player

    function initializeMatrix(length):
        matrix = empty n x n matrix filled with zeros
        return matrix
```

In this algortihm I had decided to use Adjacency Matrix data structure to for this multiplayer game. The meet function is the adjacency matrix and meetingPairs will be updated when the two players meet. In the adjacency matrix, it sets the link between players I and J to 1. It also adds the pair (i, j) to the meetingPairs set.

The finding Winner function, checks if a player has met every other player by iterating over every row in the adjacency matrix. The player will then be added to the winner set if a row with all elements equal to one is not already in the winner set.Then the function will then check if the winner set is acutally equal to n, meaning that that player have met every player.

The Matrix intialization function, creates an empty set of ‘n $\times$ n’ matixs that is full of zero. It serves to configure the adjacency matrix's initial state.

5.2 Question 5 (ii)

n^2
1
1
n
n
n
n
N*n
a
N*n
b
n^2
c

```
function Game(int n):
    adjacencyMatrix = emptyMatrix (n)
    winners = set()
    meetingPairs = set()

    function meet(i, j):
        adjacencyMatrix[i][j] = 1
        adjacencyMatrix[j][i] = 1
        meetingPairs.add((i, j))

    function findWinner():
        for each row in adjacencyMatrix:
            if all elements in the row = 1 and the corresponding player is != in winners:
                winners.add(player)
                if winner.length== n
                    return player

    function emptyMatrix(length):
        matrix = empty n x n matrix fulled with zeros
        return matrix
```

Where:

a, c= 0 or 1 (best case is 1, worst case is 0)

c= 0 or 1(**best case is 0**, **worst case is 1)**

Worst Case Scenario

T(n) $= 4\,n^2 + 4n+2+a+b+c$

$=4\,n^2 + 4n+4$

Ans: The overall run time complexity is 0(n^2)