Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 4 PART 1-4

HAMZA YOĞURTCUOĞLU
171044086

Course Assistant:Ayşe Şerbetçi TURAN

PART – 1

PART-1-a)

```
1)  public  linkList<Integer> maximumLengthSublist ( linkList<Integer> List) {
2)        if (List.head == null){
3)                return null;
4)        }else{
5)                linkList<Integer>ReturnSubList = new linkList<Integer>();
6)                Node <Integer> current = List.head;
7)                Integer firstListTail = new Integer(0);
8)                Integer secondListTail = new Integer(0);
9)                Integer firstListSize = new Integer(0);
10)               Integer secondSize = new Integer(0);
11)               Integer currentIndex = new Integer(0);
12)               while(current != null){
13)                    boolean tempTail = False;
14)                    while((firstListSize == 0 || firstListSize < secondListSize)&&current != null){
15)                         if( tempTail == True){
16)                                tempTail++;
17)                                firstListTail = currentIndex;
18)                                currentIndex ++ ;
19)                                firstSize++;
20)                                current = current.next;
21)                         }else{
22)                                if(current.next != &&current.next > current){
23)                                       currentIndex++;
24)                                       current = current.next;
25)                                       firstSize++;
26)                                }else{
27)                                       break;
28)                                }
29)                          }
30)                    }
31)                    tempTail = False;
32)                    while((seconListSize == 0 || firstListSize > secondListSize)&&current != null){
33)                         if( tempTail == True){
34)                                tempTail++;
35)                                secondListTail = currentIndex;
36)                                currentIndex ++ ;
37)                                current = current.next;
38)                                secondSize++;
39)                         }else{
40)                                if(current.next != &&current.next > current){
41)                                       currentIndex++;
42)                                       current = current.next;
43)                                       secondSize++;
44)                                }else{
45)                                       break;
46)                                }
47)                          }
48)                    }
```

```
49)             }
50)             if (firstListSize >= secondListSize){
51)                     for (int i = 0; i<firstListTail;i++)
52)                             current = current.next;
53)                     ReturnSubList.head = current ;
54)                     Node<Integer> temp = ReturnSubList.head;
55)                     current = current.next;
56)                     for(int i = 0; i<firstListSize;i++){
57)                         temp.next = current;
58)                         current = current.next;
59)                         temp = temp.next;
60)                     }
61)             }else{
62)                     for (int i = 0; i<secondListTail;i++)
63)                             current = current.next;
64)                     ReturnSubList.head = current ;
65)                     Node<Integer> temp = ReturnSubList.head;
66)                     current = current.next;
67)                     for(int i = 0; i<secondListSize;i++){
68)                         temp.next = current;
69)                         current = current.next;
70)                         temp = temp.next;
                        }
                }
        }
71)     return ReturnSubList;
}
```

```
S = 2 7 20

Process finished with exit code 0
```

// Problem Solution :

// Traversing On the Parameter List then firstList's tail (index) and size are kept
// if next node smallest than previous node.
// Then SecondList's tail(index) and size are kept.
// If paramater List is not null.
// checked firstListSize and secondListSize whichone is smallest than other that is new list will be
// When end of Parameter List , which size of lists is bigger is checked.
// start of Paramater list is gone to tail of big list.
// neccessary all nodes put in ReturnSubList and is returned.

Time Complexity

Note : x is firstListSize , y is secondListSize n = x + y

T(n) = 3) + 5-12) + 12-49) + 49-60) + 60-70) + 71)
= 1 + 7 + n + n-x + n-y + 1
max(9,n) = n

T(n) = O(n)

PART-1-b)

```
1)  public  linkList<Integer> maximumLengthSublist ( linkList<Integer> List) {
2)
3)      linkList<Integer>FirstSubList = new linkList<Integer>();
4)      if (List.head == null){
5)              return List;
6)      }
7)      FirstSubList.head = List.head;
8)      int Size = 1;
9)      Node<Integer> temp = FirstSubList.head;
10)     while (List.head !=null && List.head.next !=null && List.head < List.head.next){
11)             List.head = List.head.next ;
12)             temp.next = List.head;
13)             temp = temp.next;
14)             Size++;
15)     }
16)     temp.next = null;
17)     linkList<Integer>ReturnList = maximumLengthSublist (List);
18)     Node<Integer> temp2 = ReturnList.head;
19)     int returnSize = 0;
20)     while (temp2 != null){
21)             temp2 = temp2.next;
22)             returnSize++;
23)     }
24)     if(returnSize > Size)
25)         return ReturnList;
26)     else
27)         return FirstSubList;
28) }
//  Creating a FirstSubList(LinkList) for sub list
//  Put the nodes of sublist
//  if next node is bigger than current node , called recursive function
//  If Head of General List is null . Returned null list
//  Checked according to size of Lists
//  Which List is bigger than other one , bigger one is returned.
```

```
S = 2 7 20

Process finished with exit code 0
```

Time Complexity

NOTE:
-> x is sublist length (Constant)
-> y remaining General List (Constant)

	3-9)	+	10-15)	+	17)	+	20-23)	+	24-26)
T(n) =	6	+	n-y	+	T(n/2)	+	x	+	1

T(n) = 7 + n + T(n)

Master Teorem

a = 1 , b = 2 , d = 1

T(n) = Θ(n)

Induction

1-) 7 + n <= c.n n>n_0 => O(n)

c = 8 ,n_0 = 1 => 7<=7 (okey)

n = k => 7 + k <= 8k

n = k +1 => 8 + k <= 8k + 8

0 <= 7k (okey)

2-) 7 + n >= c. n n>n_0 => Ω(n)

c = 1 ,n_0 = 1 => 7>=1 (okey)

n = k => 7 + k >= k

7>=0 (okey)

PART-2

```
1)  void sortedArraySearches(int x)
2){
3)      int[] sortedArray = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23};
4)      int front = 0;
5)      int tail = sortedArray.length()-1;
6)
7)      while(start <= end)
8)      {
9)         if (sortedArray[start] + sortedArray[end] == x)
10)        {
11)           System.out.printf("Two numbers and Sum: %d = %d + %d\n",x
12)                                 , sortedArray[start],sortedArray[end] );
13)           break;
14)        }
15)        else if ( sortedArray[start] + sortedArray[end] > x)
16)        {
17)           --end;
18)        }
19)        else
20)           ++start;
21)     }
22)  }
```

```
Two numbers and Sum: 27 = 4 + 23

Process finished with exit code 0
```

Complexity

x is start of array side that is not traversed
y is end of array side that is not traversed

T(n) = 3-5) + 7-20)

= 3 + n-x-y

T(n) = Θ(n)

PART-3

```
1)    for (i=2*n; i>=1; i=i-1)
2)          for (j=1; j<=i; j=j+1)
3)                for (k=1; k<=j; k=k*3)
4)                      print(“hello”)
```

Time Complexity

$T(n) = (\Sigma_{i=1}^{2n} (\Sigma_{j=1}^{i} j/3 + 1)) = j*(j+1)/6 + j = 1/6\Sigma_{i=1}^{2n} (j^2+7j)$

$= (2*n(2*n+1)/2)*(\Sigma_{j=1}^{2*n} \log_3(j))$
$= n(2n+1)(\Sigma_{j=1}^{2n} \log_3(j))$
$= (2n^2+n)(\Sigma_{j=1}^{2n} \log_3(j))$

$= O(n^2 (\Sigma_{j=1}^{2n} \log(j))) = O(n^2 \log(n))$

PART-4

```
1) float aFunc(myArray,n){
2)     if (n==1){
3)        return myArray[0];
4)     }
5)
6)     for (i=0; i <= (n/2)-1; i++){
```

```
7)      for (j=0; j <= (n/2)-1; j++){
8)        myArray1[i]  =  myArray[i];
9)        myArray2[i]  =  myArray[i+j];
10)       myArray3[i]  =  myArray[n/2+j];
11)       myArray4[i]  =  myArray[j];
12)     }
13)     }
14)
15)     x1 = aFunc(myArray1,n/2);
16)     x2 = aFunc(myArray2,n/2);
17)     x3 = aFunc(myArray3,n/2);
18)     x4 = aFunc(myArray4,n/2);
19)
20)     return x1*x2*x3*x4;
}
```

Time Complexity

$T(n) = 1$

T (n) = 2-3) + 6-11) + 15-18) + 20)

We are using Master Teorem

$T(n) = 1 + 4.n^2/4 + T(n/2) + 4$

$T(n) = 5 + n^2 + T(n/2)$

a = 4 , b = 2 , d = 2

$a = b^d \Rightarrow 4 = 2^2 \Rightarrow 4 = 4 \Rightarrow n^d \Rightarrow T(n) = \Theta(n^2 \log n)$