

**Lebanese International University - Beirut Campus**

**School of Arts and Sciences**

**Department of Computer Science**

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name** | : Algorithms | **Course Code** | : CSCI440 |
| **Date** | : 28 May 2018 | **Section** | : A |
| **Instructor** | : Dr. M. Chamseddine | **Time** | : 12:30 – 13:45 |
| **Auditorium** | : | **Seat Number** | : |

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of pages** | : 08 | **Allowed Time** | : 75 minutes |
| **Documents** | : Not Allowed | **Calculators** | : Allowed |

|  |  |  |
| --- | --- | --- |
| **Problem#** | **Grade** | **Total Grade** |
| Part I |  | /100 |
| Question 1 | /20 |
|  |  |
| Part II |  |
| Question 1 | /20 |
| Question 2 | /25 |
| Question 3 | /30 |
|  |  |
| Part III |  |
| Question 1 | /05 |
|  |  |
|  |  |

**Good Luck**

**Question 1 (20 points)**

Show the result of the following sequence of instructions:

join(8,4)

join(3,7)

join(9,5)

join(5,3)

join(9,1)

join(8,2)

join(4,6)

join(7,5)

join(8,9)

join(0,3)

join(9,6)

when the join operations are performed according to: (5 pts. Each)

1. Quick-Find (7 points)
2. Quick-Union (7 points)
3. Weighted quick union (6 points)

**Question 2 (20 points – 10 points each)**

Consider the algorithm shown below. Find the complexity of this algorithm. (show details)

a.

**public static int mystery(int a, int n) {**

**if (n == 0)**

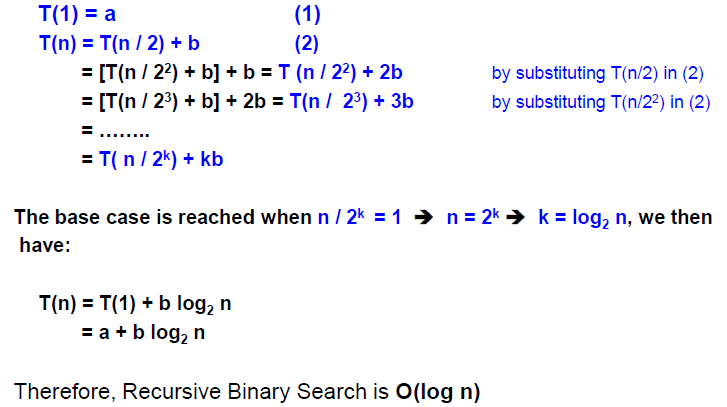
**return 0;**

**return mystery(a \* 2, n / 2) + a;**

**}**

T(0) = 0

T(N) = T(N/2) + b



**b.**

public static int method(int n){

if(n ==0 )

return 0;

int x=n%10;

return x+method(n/10);

}

T(0) = 0

T(N) = T(N/10) + b

**.**

**.**

**.**

**.**

**.**

**O(log10N)**

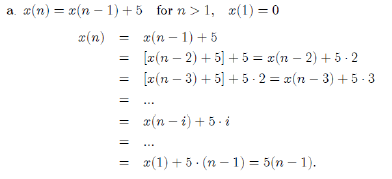
**Question 3 (25 points – 5 points each)**

What is the computational complexity of each of the following pieces of code?

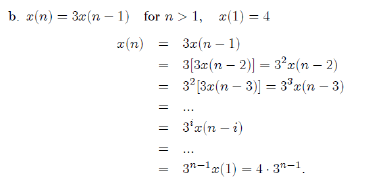
|  |  |
| --- | --- |
|  | O(n(logn)2) |
| The outer for-loop goes round n times. For each i, the next loop goes round m = log2 n times, because of doubling the variable j. For each j, the innermost loop by k goes round j times, so that the two inner loops together go round 1 + 2 + 4 + . . . + 2m−1 = 2m − 1 ≈ n times. Loops are nested, so the bounds may be multiplied to give that the algorithm is O(n2). | O(n2) |
| sum = 0 ;  for(i=1; i<=2\*n; i++)              sum = sum + 1; | O(n) |
| sum = 0 ;  for(i=1; i<=n; i++)         for(j=1; j<=n; j\*=2)                      sum = sum + 1 ; | O(nlog n) |
| for(int i=1; i<=n; i\*=2)  while(i>n){  sum+=i;  i++;  } | O(log n) |

**Question 4 (30 points – 10 points each)**

1. T(n) = T(n − 1) + 5 for n > 1, T(1) = 0



1. T(n) = 3T(n − 1) for n > 1, T(1) = 4



1. T(N) = T(N/4) + 1 for n > 1, T(1) = 1

T(1) = 1 (1)

T(n) = T(n / 4) + 1 (2)

= [T(n / 42) + b] + 1 = T (n / 42) + 2 by substituting T(n/4) in (2)

= [T(n / 43) + b] + 2 = T(n / 43) + 3 by substituting T(n/42) in (3)

= ……..

= T( n / 4k) + k

The base case is reached when n / 4k = 1 n = 4k k = log4 n, we then

have:

T(n) = T(1) + log4 n

= 1 + log4 n

Therefore, the Recursive function complexity is **O(log4 n)**

**Question 5 (5 points - 10 minutes)**

Suppose you have a pointer to the head of singly linked list *(each node has a pointer to the next element, and the last node’s pointer is NULL)*

null

Unfortunately, a virus might corrupt the list, so that the last node has a pointer back to some other node in the linked list.

If you know the number of nodes in the linked list, how many steps, in its worst [5 points] and best case[5 points], does it require to determine if the linked list is corrupted or not? What is the time complexity [5 points]?

**Solution**

We traverse the list and increase counter each time we go over a node. Once the counter exceeds N, than the list is corrupted and that requires N+1steps => O(N). Once we reach null, then the list is not corrupted and that requires N steps => Θ(N).