Subscribe to DeepL Pro to edit this document.  
Visit www.DeepL.com/Pro for more information.

**Laboratory practice No. 1  
Recursion**

|  |  |
| --- | --- |
| **Miguel Ángel Martínez Flórez**  Eafit University  Medellín, Colombia  mamartinef@eafit.edu.co | **Pablo Maya Villegas**  Eafit University  Medellín, Colombia  pmayav@eafit.edu.co |

**3) Practice for final project defense presentation**

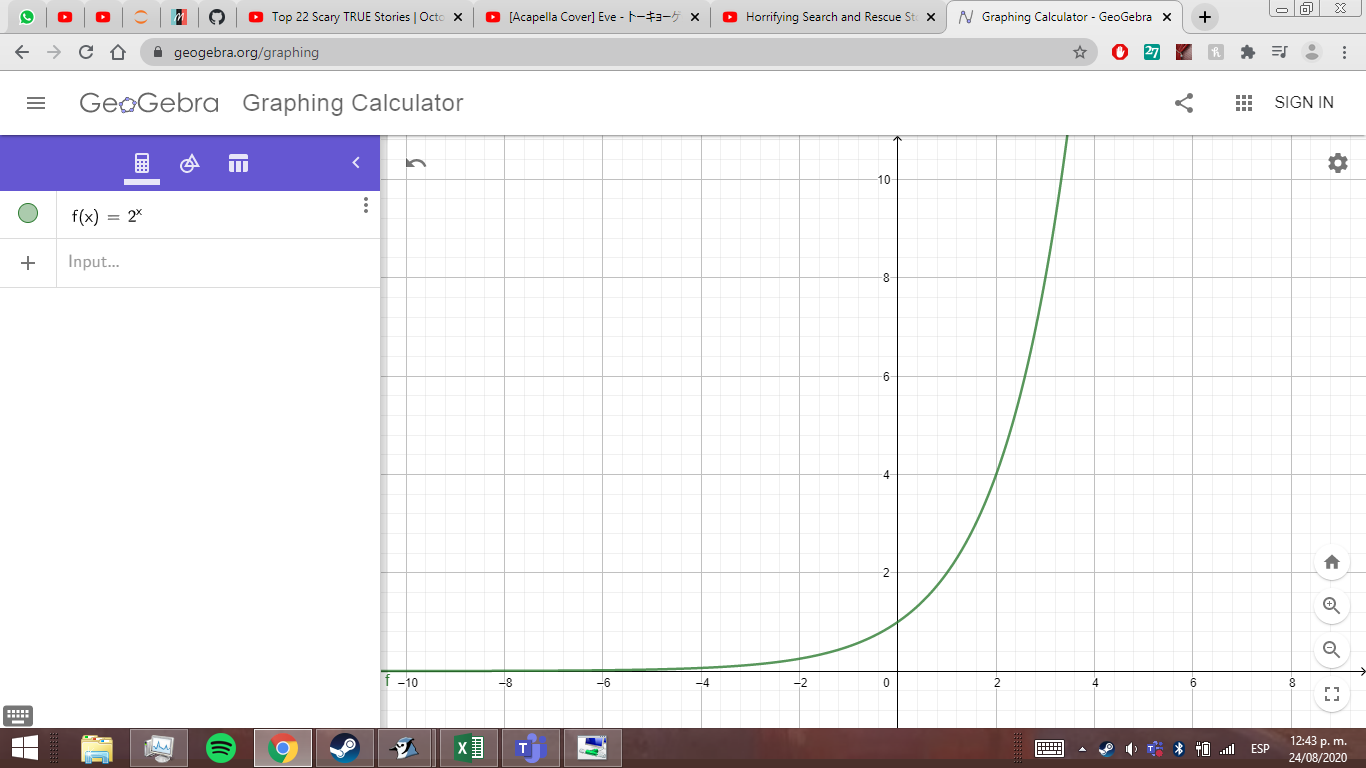
**3.1.** Calculate the worst-case asymptotic complexity of exercise 1.1

The recursive algorithm has a change rate of 2^X, this means that the larger the amount that is evaluated, it will take more time and high numbers can take up to years to finish processing, a single increase of X potentially increases the time and cycles it performs**.**

For exercise 1.1, take times for 20 different problem sizes, generate a graph and analyze the results. Estimate how long this algorithm will take in the longest common subsequence between two mitochondrial DNAs (which have about 300,000 characters each)

**R\**

|  |  |  |
| --- | --- | --- |
| Size: | Recursions: | Time: |
| 11 | 26667 | 20 |
| 12 | 106006 | 34 |
| 13 | 318032 | 46 |
| 14 | 582732 | 67 |
| 15 | 794759 | 62 |
| 16 | 3438617 | 137 |
| 17 | 12432260 | 363 |
| 18 | 35495254 | 1037 |
| 19 | 85327171 | 1920 |
| 20 | 277044648 | 6717 |
| 21 | 568425981 | 12895 |
| 22 | 760143459 | 12787 |
| 23 | 2872908681 | 63002 |
| 24 | 3064626161 | 59929 |
| 25 | 5177391384 | 61909 |
| 26 | 9494975486 | 177372 |
| 27 | 11607740710 | 173931 |
| 28 | 15925324813 | 171988 |
| 29 | 44919490555 | 887665 |
| 30 | 1,60823E+11 | 3879457 |



This is the 2^X graph confirming the similarity of relationship.

**3.3.** Is the complexity of the algorithm of exercise 1.1? appropriate to find the longest common substring among mitochondrial DNAs such as those in the datasets?

**R\** This algorithm is not suitable for calculating 300 thousand values, because it is potential, that value takes an infinite time, another algorithm, although it is slower in the short term may have a greater potential to calculate that magnitude

**3.4.** Optional exercise] Explain in your own words how GroupSum5 works

**R\**. For this point you must take all multiples of 5 in the sum, and if a 1 follows directly this can not be counted.

To achieve this we are going to process all the numbers of the array, and if one is multiple of 5 then its value is subtracted from the final goal and it is transformed into 0. Just after a multiple of 5 is discovered it is observed if the next number of the array is a 1, take into account that if we are in the last value this step is not done and if it is confirmed that the next one is a 1, then we turn it into 0.

When all the array is already ordered and the reduced goal we are going to verify that this is not negative, otherwise it is impossible to add, and we only call the recursive method groupSum to check that with the rest of the numbers it is possible to reach the goal.

**3.5.** Calculate the complexity of the Online Exercises of numerals 2.1 and 2.2

**R\**

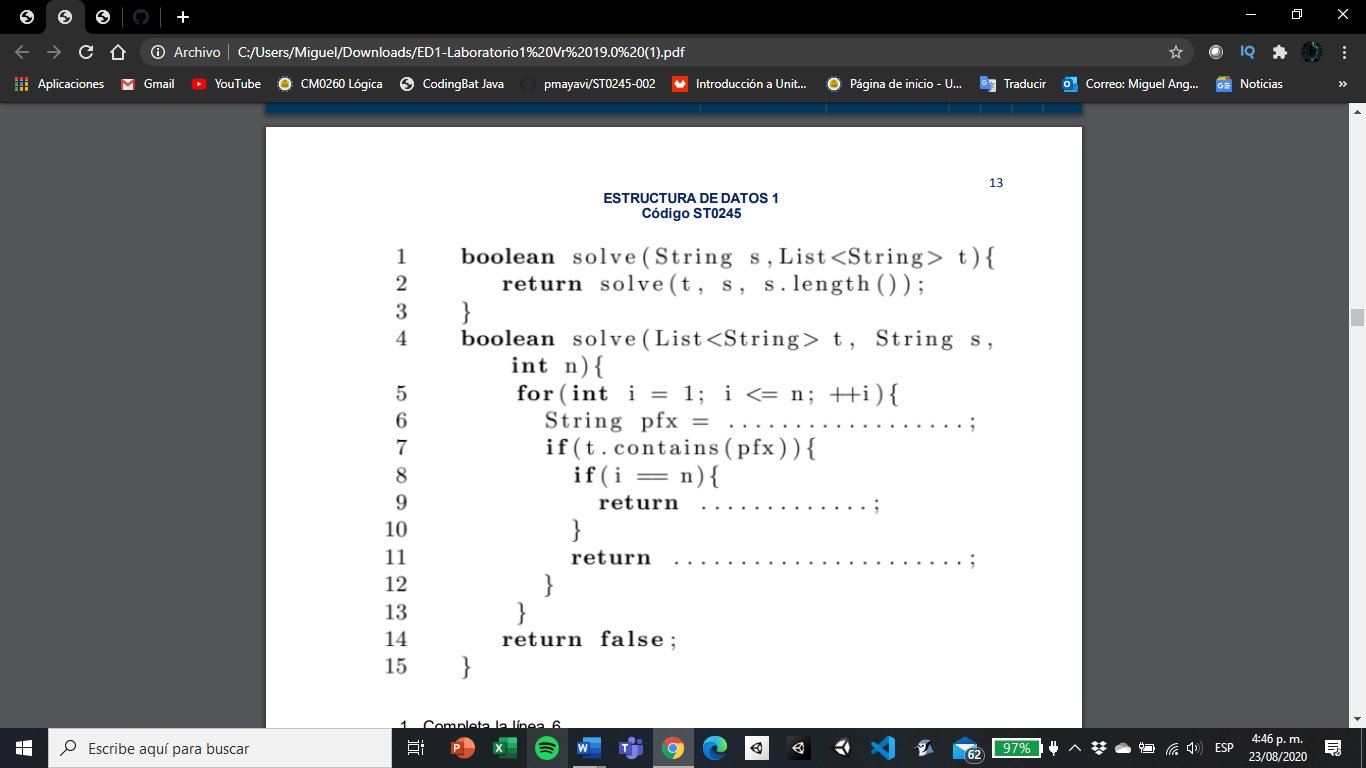
* groupNoAdj = T(5n+3n^2)
* fibonacci = T(2^n)
* countX = T(3+n)
* groupSumClump = T(2+12n+3n^2)
* array11 = T(3+n)
* splitOdd10 = T(n+2^n)
* noX = T(3+n)
* split53 = T(3n+2^n)
* changeYX = T(3+n)
* groupSum5 = T(1+6n+2^n)

**3.6.** Calculate the complexity of the Online Exercises of numerals 2.1 and 2.2

**R\** The n is the magnitude of the input to the method, if it is a string of 8 characters, then n is 8; and m is the times that an increase is made, so to speak, a limitation of how many cycles are allowed.

***4) Drill of Partial***

* 1. One of the crucial technologies for the success of the Fourth Industrial Revolution is cybersecurity, and one of the most important areas of cybersecurity is cryptography. Cryptography is the art and technique of writing secret keys in such a way that what is written is only intelligible to those who know how to decipher it. Let's consider an application of cryptography. Some words of the Spanish alphabet can be written as the consecutive union of the symbols of the universal periodic table. For example, the word "Population" can be written as the union of the symbols {Po, B, La, C, I, O, N}, PoBLaCION. They give you a string of charactersSand a set of symbolsT. The goal is to determine whetherS can be written as the consecutive union of zero or moreT symbols. For example, whetherT={Ti,B,I,O,C} andS="Biotic", the answer is true; forS= "", the answer is true; and forS="Titan", the answer is false. The following code solves the problem, but it is missing some lines; please complete them. Thank you! You can assume that T contains all the possible combinations of accents, upper and lower case of each symbol in the periodic table. In Java, the s.substring(a,b) method returns the substring of s between indexes a and b-1, included. The t.contains(s) method returns true if the string s is inside t; otherwise, false.



1. Complete line 6................................

a. s.substring(0, i)

b. s.substring(0, n)

c. s.substring(i, n)

R\ c. s.substring(i, n)

2. Complete line 9.................................

a. false

b. s.substring(0, i)

c. true

R\ c. true

3. Complete line 11 .................................

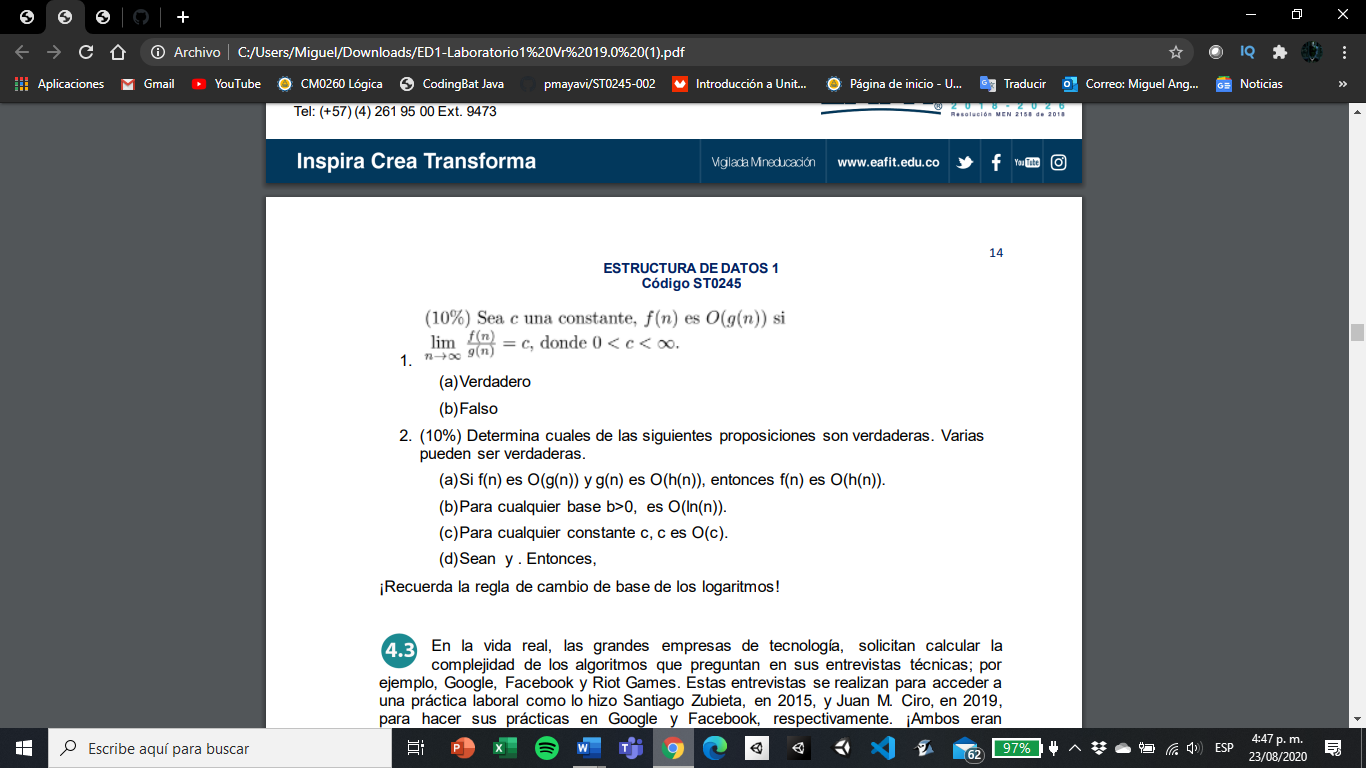
a. solve(t, s.substring(i), n - i)

b. solve(pfx, t), n - i)

c. solve(t, s.substring(n), I - n)

R\ a. solve(t, s.substring(i), n - i)

* 1. In real life, companies like Riot Games make extensive use of databases. In many databases, the search times are O(log n).



1.

(a)True

(b)False

R\ b) false

2. (10%) Determine which of the following statements are true. Several

can be true.

(a)If f(n) is O(g(n)) and g(n) is O(h(n)), then f(n) is O(h(n)).

(b)For any base b>0, it is O(ln(n)).

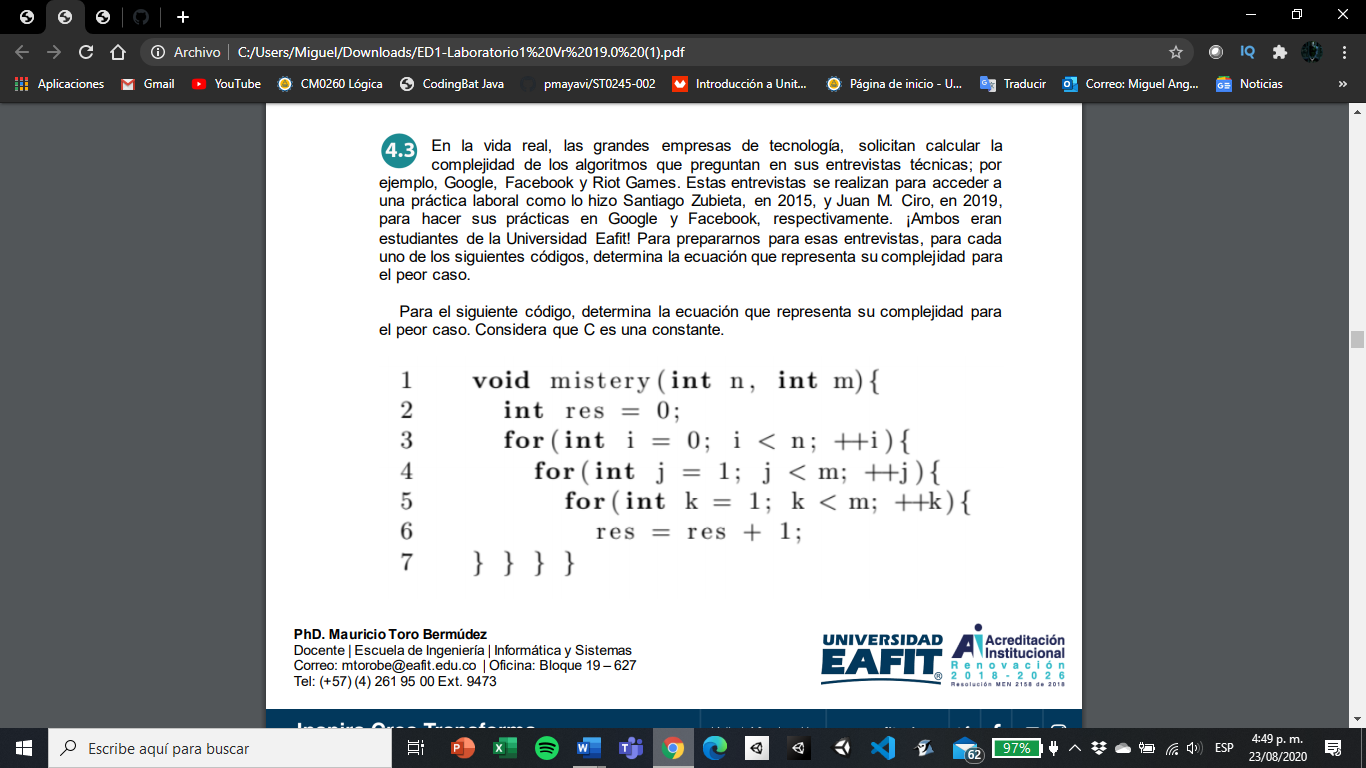
(c)For any c constant, c is O(c).

(d)Sean and . Then,

R\ (c)For any constant c, c is O(c).

Remember the logarithmic base change rule!

* 1. In real life, large technology companies, ask to calculate the complexity of the algorithms they ask in their technical interviews; for example, Google, Facebook and Riot Games. These interviews are conducted to access a work placement as Santiago Zubieta did in 2015 and Juan M. Ciro did in 2019 to do his internship at Google and Facebook, respectively. Both were students at Eafit University! To prepare for these interviews, for each of the following codes, determine the equation that represents their complexity for the worst case. For the following code, determine the equation that represents its complexity for the worst case. Consider C as a constant.



*The complexity of the mistery function is*

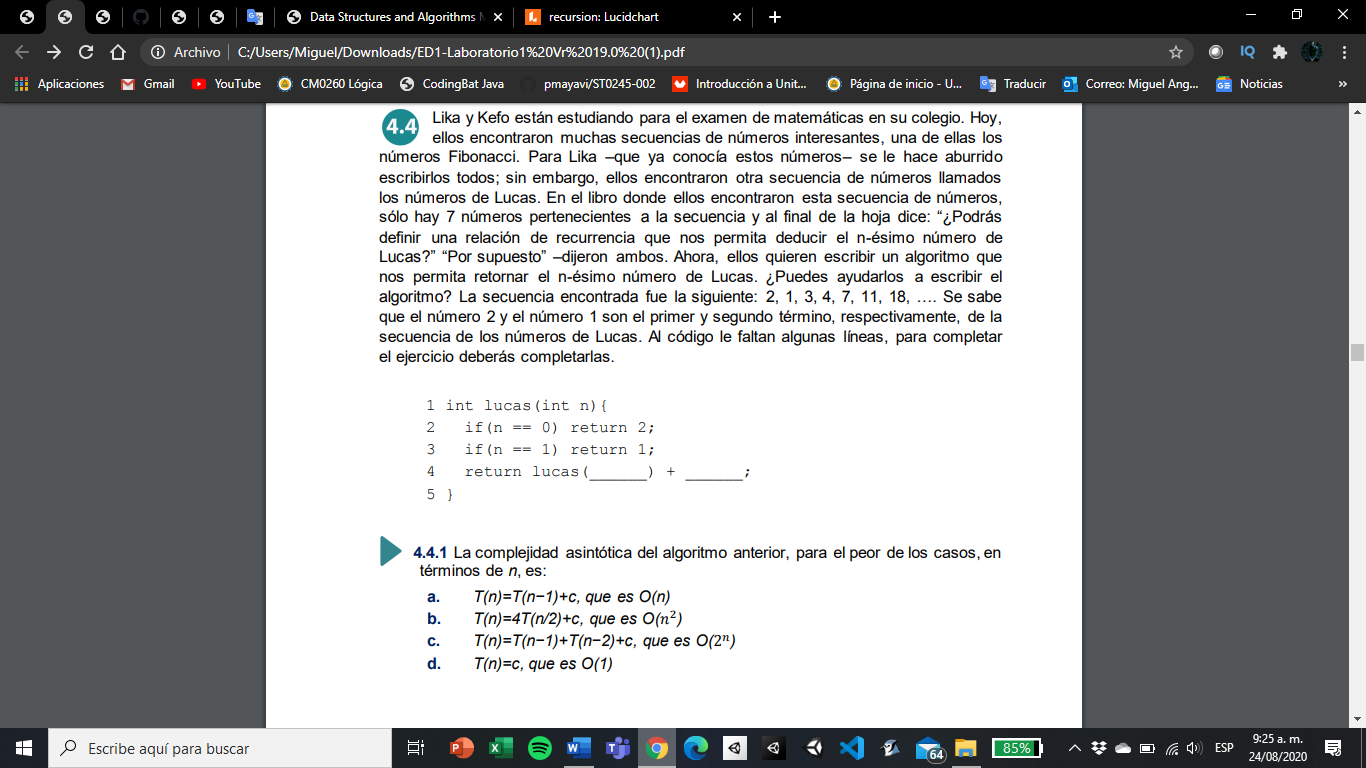
*a. T(n,m) = C x n x m*

*b. T(n,m) = C x n x m^2*

*c. T(n,m) = C x n x mlogm*

*d. T(n,m) = C x n x m^3R\  
 b. T(n,m) = C x n x m^2*

* 1. Lika and Kefo are studying for the math exam at their school. Today, they found many interesting number sequences, one of them being the Fibonacci numbers. For Lika - who already knew these numbers - it gets boring to write them all down; however, they found another sequence of numbers called the Lucas numbers. In the book where they found this sequence of numbers, there are only 7 numbers belonging to the sequence and at the end of the page it says: "Can you define a recurrence relationship that will allow us to deduce the n-th number of Luke" "Of course" - they both said. Now, they want to write an algorithm that will allow us to return the nth number of Luke. Can you help them write the algorithm? The sequence found was the following: 2, 1, 3, 4, 7, 11, 18, ... It is known that the number 2 and the number 1 are the first and second terms, respectively, of the sequence of Luke's numbers. The code is missing some lines, to complete the exercise you will have to complete them.



*4.4.1 The asymptotic complexity of the above algorithm, for the worst case, in*

*terms of n, es:*

*a. T(n)=T(n-1)+c, which is O(n)*

*b. T(n)=4T(n/2)+c, which is O(𝑛2)*

*c. T(n)=T(n-1)+T(n-2)+c, which is O(2𝑛)*

*d. T(n)=c, which is O(1)*

*R\ C. T(n)=T(n-1)+T(n-2)+c, which is O(2𝑛)*

***4.5*** *In real life, palindromes are used to develop algorithms for*

*compression of DNA chains. For this partial, it considers an algorithm capable*

*to say whether a string of characters is a palsy or not. A palsy is*

*a string that reads the same from left to right as from right to left. A*

*Some examples follow:*

*- For "love of Rome", the answer is true*

*- For "mommy", the answer is false*

*- For "cocoococ", the answer is true*

*The isPal algorithm solves the problem, but it is missing a few lines. Complete them, by*

*favor.*

*01 static boolean isPal(String s) {*

*02 if(s.length() == 0 || s.length() == 1)*

*03 return........ ;*

*04 if(......... )*

*05 return isPal(s.substring(1, s.length()-1));*

*06 //else*

*07 return false;*

*08 }*

*In Java, the s.charAt(i) method allows to know which character is in the i position of*

*the string s and s.substring(a,b) returns a substring of s between the indexes a and*

*b−1.*

*Please complete the missing lines:*

*1. Complete line 3................................*

*a. true*

*b. false*

*c. s*

*R\ a.True*

*2. Complete line 4.................................*

*a. s.substring(0,s.length()-1).equals(s.substring(s.length()-1, 0))*

*b. s.charAt(0) == (s.charAt(s.length()-1))*

*c. true*

*R\ b. s.charAt(0) == (s.charAt(s.length()-1))*

***4.6*** *Pepito wrote the following code using recursion:*

*private int b(int[ ] a, int x, int low, int high) {*

*if (low > high) return -1;*

*int mid = (low + high)/2;*

*if (a[mid] == x) return mid;*

*else if (a [mid] < x)*

*return b(a, x, mid+1, high);*

*else*

*return b(a, x, low, mid-1);*

*}*

*Which recurrence equation describes the behavior of the above algorithm*

*for the worst case scenario?*

*a) T(n)=T(n/2)+C*

*b) T(n)=2.T(n/2)+C*

*c) T(n)=2.T(n/2)+Cn*

*d) T(n)=T(n-1)+C*

*R\ d) T(n)=T(n-1)+C*

***4.7*** *What does the unknown algorithm calculate and what is the complexity*

*asymptotic in the worst case of the unknown algorithm?*

*01 public int unknown(int[] a){*

*02 return aux(a, a.length-1); }*

*03*

*04 public int aux(int[] a, int n){*

*05 if(n < 1) return a [n];*

*06 else return a [n] + aux(a, n-1); }*

*Choose the answer you think is right:*

*a) The sum of the elements of the arrangement to and is*

*b) Orders the arrangement to and is O(n.log n)*

*c) The sum of the elements of the arrangement a and is O(1)*

*d) The maximum value of an arrangement to y is O(n)*

*(e) The sum of the elements of the arrangement a and is O(n)*

*R\ e) The sum of the elements of the arrangement a and is O(n)*

***4.8*** *In real life, game theory has proven to be very useful in*

*modern artificial intelligence; for example, to transfer*

*style that allows to generate works of art with the style of a certain painter. For this*

*considers a game in which a player can win 3, 5 or 7 points in a single*

*How many ways can a player get a total of T points? A*

*Some examples follow:*

*- For T=10. Answer: 3 which is 5+5, 7+3, 3+7.*

*- For T=2. Answer: 0.*

*- For T=15. Answer: 8.*

*The ways algorithm solves the problem, but it lacks some lines, to be able to solve*

*the exercise must be completed.*

*01 int ways(int T){*

*02 //Case(s) base(s).*

*03 ............*

*04 ............*

*05 int f1 = ways(T - 3);*

*06 int f2 = ways(T - 5);*

*07 int f3 = ways(T - 7);*

*08 return ...........;*

*09 }*

*4.5.1 How many instructions does the algorithm execute in the worst case?*

*a. T(n)=T(n-1)+C*

*b. T(n)=T(n-1)+T(n-2)+C*

*c. T(n)=T(n/2)+C*

*d. T(n)=T(n+1)+C*

*R\ b. T(n)=T(n-1)+T(n-2)+C*

***4.6*** *Alek and Krish are playing Number. Number is a game in which one player 1,*

*gives a number n (1<=n<=10100) to a player 2 and player 2 must determine*

*the sum of all digits of n, except for the case where there are two digits*

*adjacent (i.e., contiguous, in a row) that are the same.*

*If there are two adjacent digits, neither of the two adjacent numbers is added.*

*Between Alek and Krish they wrote a code to make this faster, but it has been deleted*

*Could you help Alek and Krish reconstruct the code?*

*1 public int sum(String n) {*

*2 return sumaAux(n, 0);*

*3 }*

*4*

*5 private int sumaAux(String n, int i){*

*6 if (i >= n.length()) {*

*7 return 0;*

*8 }*

*9 if(i + 1 < n.length() &&*

*n.charAt(i) == n.charAt(i + 2)){*

*10 return sumAux(n,i+1) ;*

*11 }*

*12 return (n.charAt(i) - '0') + sumaAux(n,i+1);*

*13 }*

*Operation n.charAt(i) - '0' converts a character to its integer equivalent,*

*for example, the character '1' transforms it into the number 1.*

*4.6.1 Complete line 10:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4. 6.2 Complete line 12:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

***4.8*** *Consider the following program. What is the output generated by*

*fun(11,5)? As an example: fun(10,3)=20.*

*int fun(int n, int m){*

*if(n % m == 2) return n;*

*return fun(n + m, n - m);*

*}*

*Choose the answer you think is right:*

*a. 11*

*b. 5*

*c. 22*

*d. 2*

*R\ c. 22*

***4.9*** *Consider the following program. What is the output for fun(1,4)?*

*As an example: fun(1,2)=4.*

*int fun(int m,int n){*

*if(m==0){*

*return (n+1);*

*}*

*if(m>0 && n==0){*

*return fun(m-1,1);*

*}*

*int a=fun(m,n-1);*

*return fun(m-1,a);*

*}*

*Choose the answer you think is right:*

*a. 4*

*b. 6*

*c. 5*

*d. 12*

*R\ b. 6*

***4.10*** *Lika and Kefo are studying for the math exam in their*

*school. Today, they found many sequences of numbers*

*interesting, one of which is the Fibonacci numbers. For Lika -who already knew*

*these numbers-it gets boring to write them all down; however, they found another*

*sequence of numbers called Luke's numbers. In the book where they*

*found this sequence of numbers, there are only 7 numbers belonging to*

*sequence and at the end of the sheet it says: "Can you define a recurrence relationship that will*

*allow the nth number of Luke to be deducted?" "Of course," they both said. Now,*

*they want to write an algorithm that will allow us to return the nth number of Luke.*

*Can you help them write the algorithm? The sequence found was as follows: 2,*

*1, 3, 4, 7, 11, 18, …. It is known that the number 2 and the number 1 are the first and second*

*term, respectively, of the sequence of Luke's numbers. The code is missing*

*some lines, to complete the exercise you will have to complete them.*

*1 int lucas(int n){*

*2 if(n == 0) return 2;*

*3 if(n == 1) return 1;*

*4 return lucas (return lucas(n-1) + lucas(n-2);*

1. *}*

*4.11.1 The asymptotic complexity of the previous algorithm, for the worst case,*

*in terms of n, es:*

*a. T(n)=T(n-1)+c, which is O(n)*

*b. T(n)=4T(n/2)+c, which is O(𝑛2)*

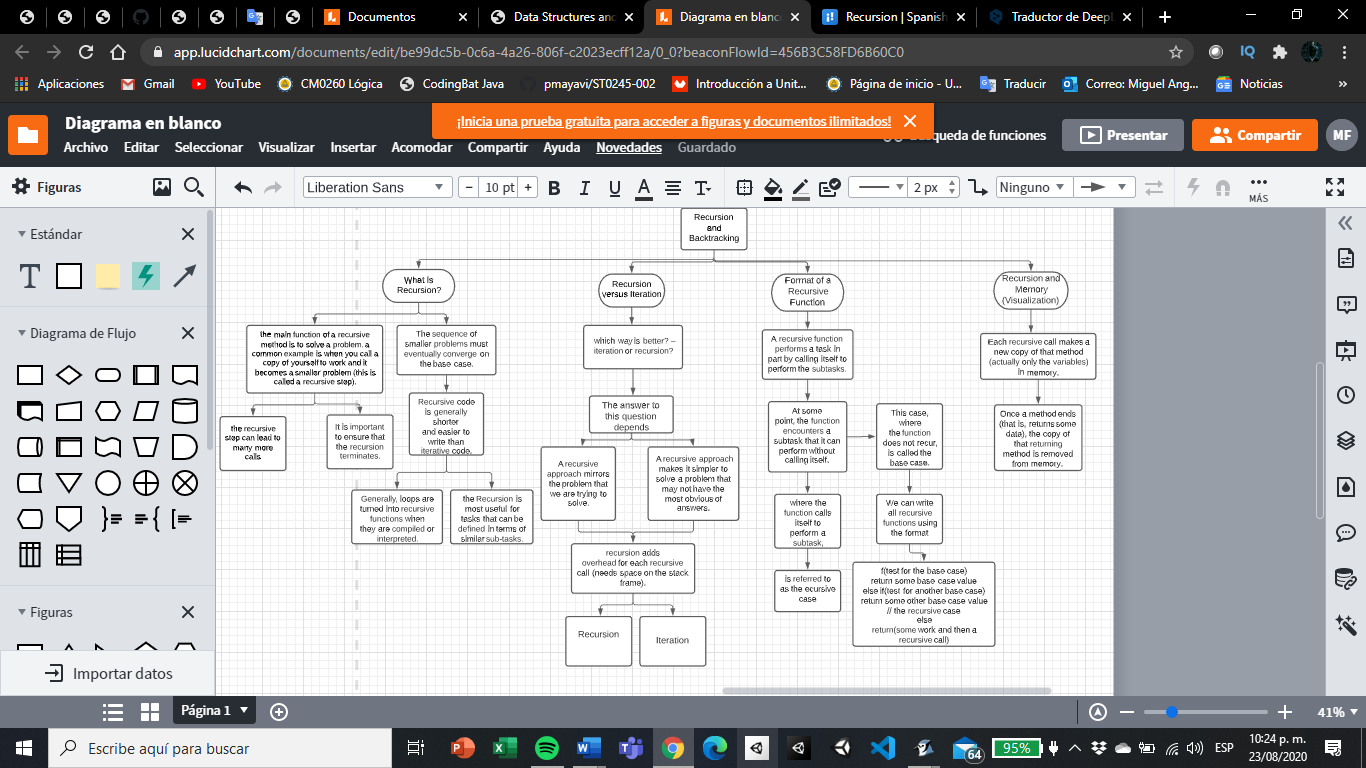
*c. T(n)=T(n-1)+T(n-2)+c, which is O(2𝑛)*

*d. T(n)=c, which is O(1)*

*R\ c. T(n)=T(n-1)+T(n-2)+c, which is O(2𝑛)*

***5) Recommended reading (optional)***

Concept map



https://app.lucidchart.com/invitations/accept/44432a03-c4dd-43cf-901c-70efaaea5545