ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 1: Recursion and Complexity

Ana Sofía Gutiérrez Tejada
Universidad Eafit
Medellín, Colombia
asgutierrt@eafit.edu.co

Santiago Hidalgo Ocampo
Universidad Eafit
Medellín, Colombia
shidalgoo1@eafit.edu.co

3) *Practice for final project defense presentation*

3.1 Complexity exercise 1.2

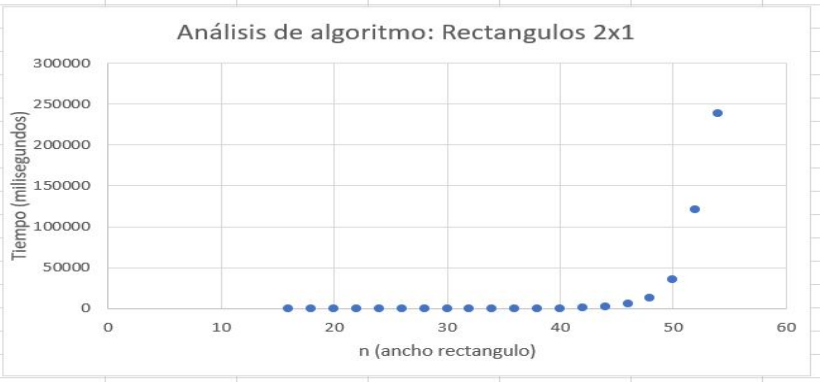
T(n) = T(n-1) + T(n-2) +C_2
T(n) = 2*(T(n-1)) +C_2

Solving in Wolfram: $T(n) = c_1\, 2^{n-1} + c_2\,(2^n - 1)$

T(n) is O((c1*2^(n-1)) + c2*((2^n)-1), by O definition
T(n) is O(2^(n-1) + ((2^n)-1)), by multiplication rule
T(n) is O(2^(n-1)+ 2^n), by addition rule
T(n) is O((2^n * 2^(-1)) + 2^n)
T(n) is O(2^n * (2^(-1) + 1)), by factorization
T(n) is O(2^n), by multiplication rule

3.2 times graph for exercise 1.2

n	Tiempo (milisegundos)
16	8
18	1
20	0
22	0
24	0
26	1
28	2
30	2
32	16
34	35
36	33
38	121
40	294
42	764
44	1998
46	5141
48	13369
50	35077
52	121166
54	238485



The time it takes for this algorithm to calculate the ways of filling a rectangle of 50x2 cm² with rectangles of 1x2 cm² is approximately 35,077 seconds.

3.3 is it useful for thousand n? (exer. 1.2)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

It is not viable, because algorithms with exponential asymptotic complexity are time consuming, thus leading to significant delays in the packaging process.

3.4 Explain GrupSum5

Given an array of ints, it must return true if it exist certain subset that sums to an specific target, additionally, any multiples of 5 must be included and if there is a 1 following one of this it must not be chosen in the subset

3.5 Complexity for Codingbat recursion exercises

- **Recursion 1**

3.5.1. `// T(n)= c_2 + T(n/10)`

Solving in Wolfram: $T(n) = \frac{c_2 \log(n)}{\log(10)} + c_1$ where log(x) is natural logarithm
T(n) is O ((c2/ln10) * ln(n) + c1) , by O definition
T(n) is O((c2/ln10) * ln(n)) , by addition rule
T(n) is O(ln(n)) , by multiplication rule

3.5.2. `//T(n)= c_2 + T(n-1)`

Solving in Wolfram: $T(n) = c_2\, n + c_1$
T(n) is O (c2*n + c1) , by O definition
T(n) is O(c2*n) , by addition rule
T(n) is O(n)

3.5.3. `//T(n)= c_2 + T(n-1)`

Solving in Wolfram: $T(n) = c_2\, n + c_1$
T(n) is O (c2*n + c1) , by O definition
T(n) is O(c2*n) , by addition rule
T(n) is O(n)

3.5.4. `//T(n)= c_2 + T(n/10)`

Solving in Wolfram: $T(n) = \frac{c_2 \log(n)}{\log(10)} + c_1$ where log(x) is natural logarithm
T(n) is O ((c2/ln10) * ln(n) + c1) , by O definition
T(n) is O((c2/ln10) * ln(n)) , by addition rule
T(n) is O(ln(n)) , by multiplication rule

3.5.5. `//T(n)= c_2 + T(n/10)`

Solving in Wolfram: $T(n) = \frac{c_2 \log(n)}{\log(10)} + c_1$ where log(x) is natural logarithm
T(n) is O ((c2/ln10) * ln(n) + c1) , by O definition
T(n) is O((c2/ln10) * ln(n)) , by addition rule
T(n) is O(ln(n)) , by multiplication rule

- **Recursion 2**

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.5.6. splitOdd10

T(n) = T(n-1) + T(n-2) +C_2
T(n) = 2*(T(n-1)) +C_2

Solving in Wolfram: $T(n) = c_1\, 2^{n-1} + c_2\,(2^n - 1)$

T(n) is O((c1*2^(n-1)) + c2*((2^n)-1), by O definition
T(n) is O(2^(n-1) + ((2^n)-1)), by multiplication rule
T(n) is O(2^(n-1)+ 2^n), by addition rule
T(n) is O((2^n * 2^(-1)) + 2^n)
T(n) is O(2^n * (2^(-1) + 1)), by factorization
T(n) is O(2^n), by multiplication rule

3.5.7. groupSum5

T(n) = T(n-1) + T(n-2) +C_2
T(n) = 2*(T(n-1)) +C_2

Solving in Wolfram: $T(n) = c_1\, 2^{n-1} + c_2\,(2^n - 1)$

T(n) is O((c1*2^(n-1)) + c2*((2^n)-1), by O definition
T(n) is O(2^(n-1) + ((2^n)-1)), by multiplication rule
T(n) is O(2^(n-1)+ 2^n), by addition rule
T(n) is O((2^n * 2^(-1)) + 2^n)
T(n) is O(2^n * (2^(-1) + 1)), by factorization
T(n) is O(2^n), by multiplication rule

3.5.8. splitArray

T(n) = T(n-1) + T(n-2) +C_2
T(n) = 2*(T(n-1)) +C_2

Solving in Wolfram: $T(n) = c_1\, 2^{n-1} + c_2\,(2^n - 1)$

T(n) is O((c1*2^(n-1)) + c2*((2^n)-1), by O definition
T(n) is O(2^(n-1) + ((2^n)-1)), by multiplication rule
T(n) is O(2^(n-1)+ 2^n), by addition rule
T(n) is O((2^n * 2^(-1)) + 2^n)
T(n) is O(2^n * (2^(-1) + 1)), by factorization
T(n) is O(2^n), by multiplication rule

3.5.9. groupNoAdj

T(n) = T(n-1) + T(n-2) +C_2
T(n) = 2*(T(n-1)) +C_2

Solving in Wolfram: $T(n) = c_1\, 2^{n-1} + c_2\,(2^n - 1)$

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

T(n) is O((c1*2^(n-1)) + c2*((2^n)-1), by O definition
T(n) is O(2^(n-1) + ((2^n)-1)), by multiplication rule
T(n) is O(2^(n-1)+ 2^n), by addition rule
T(n) is O((2^n * 2^(-1)) + 2^n)
T(n) is O(2^n * (2^(-1) + 1)), by factorization
T(n) is O(2^n), by multiplication rule

3.5.10. split53

T(n) = T(n-1) + T(n-2) +C_2
T(n) = 2*(T(n-1)) +C_2

Solving in Wolfram: $T(n) = c_1\, 2^{n-1} + c_2\,(2^n - 1)$

T(n) is O((c1*2^(n-1)) + c2*((2^n)-1), by O definition
T(n) is O(2^(n-1) + ((2^n)-1)), by multiplication rule
T(n) is O(2^(n-1)+ 2^n), by addition rule
T(n) is O((2^n * 2^(-1)) + 2^n)
T(n) is O(2^n * (2^(-1) + 1)), by factorization
T(n) is O(2^n), by multiplication rule

3.6 Explain the used variables in complexity for codingbat exercises

- **Recursion 1**

3.6.1. count7

n means the number of digits of the given number

3.6.2. countX

n means the length of the String

3.6.3. Triangle

n means the length of the base of the triangle (number of rows)

3.6.4. count8

n means the number of digits of the given number

3.6.5. sumDigits

n means the number of digits of the given number

- **Recursion 2**

3.6.6. splitOdd10

n means the length of the given array(nums)

3.6.7. groupSum

n means the length of the given array(nums)

3.6.8. splitArray

n means the length of the given array(nums)

3.6.9. groupNoAdj

n means the length of the given array(nums)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.6.10. split53

n means the length of the given array(nums)

4) Practice for midterms

4.1 *SumaGrupo*(start+1, nums, target);

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

4.3

4.3.1 int res = solucionar(n-a,a,b,c) + 1;

4.3.2 res = Math.max(res,solucionar(n-b,a,b,c) + 1);

4.3.3 res = Math.*max*(res,*solucionar*(n-c,a,b,c) + 1);

4.4 e) the sum of the elements of the array a and it is O(n)

4.5

4.5.1 return n ==1 ? 1: 2;

4.5.2 b) T(n)=T(n−1)+T(n−2)+C

4.6

4.6.1 sumaAux(n,i+=2);

4.6.2 sumaAux(n,i+1);

4.7

4.7.1 S,i+1,t-S[i]

4.7.2 S,i+1,t

4.8

4.8.1 return 0;

4.8.2 ni+nj

4.9 c) 22

4.10 b). 6

4.11

4.11.1 return lucas(n-1) + lucas(n-2);

4.11.2 c) T(n)=T(n−1)+T(n−2)+c, que es O(2^n)

4.12

4.12.1 sat

4.12.2 Math.max(fi, fj);

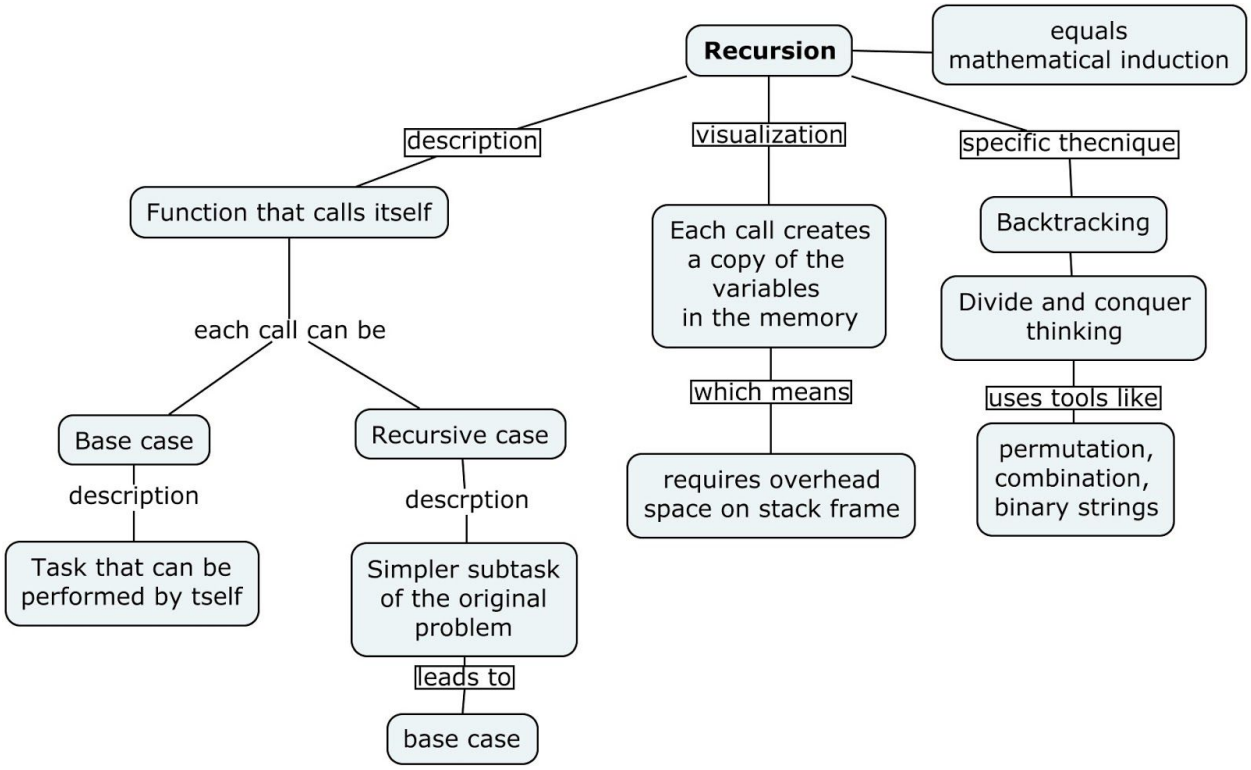
4.12.3 sat

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

5) Data structures and algorithms made easy in java by Narasimha Karumanchi Chapter 2: Recursion and backtracking.

Any function that calls itself is recursive, each call must be a slightly simpler version of the original one until they converge on the base case. it is usually simpler and shorter that iterative code.
The typical structure of a recursive function contains a base case, which is a task that can be performed without calling itself, followed by the recursive case or part that performs a subtask.The visualization of a recursive function works this way: each recursive call makes a copy of the new variables in the memory and once it ends, that copy is removed from the memory. Base on this, recursive functions require overhead space on the stack frame, on the other hand, in iteration, a infinite loop might loop forever because it is not using extra memory, in recursion, this leads to a stack overflow problem.

Backtracking is a method of exhaustive search using divide and conquer, sometimes you have to explore all possibilities, this is an slow alternative but it can be improved using different techniques like basic algorithms for different task like permutations, combinations, general strings and binary strings.



PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473