Algorithmics	Student information	Date	Number of session
	UO: UO269412		3
	Surname: Carrillo		
	Name: Javier		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Basic recursive models

Subtraction 4

This class was supposed to be a recursive method by subtraction with complexity O(3^(n/2)) and as such, we need specific number and size of subproblems, complexity of the method without recursive calls. The wording doesn't state any of these, but the complexity that must be achieved. To obtain it, we must find the case where the number of problems is greater than 1 (in this case, it will be 3 because of the desired complexity). Then we need a specific value for the size of the subproblem and looking at both the formula and expression we want, we get that such value is 3. Since the complexity of the problem without the calls isn't present in the formula, we will give it the value 1, making such problem with a linear complexity. Then, we have that a = 3, b = 2, k = 1 and the wanted subtraction is achieved.

Division 4

This class was supposed to be a recursive method by subtraction with complexity O(n^2) and as such, we need specific number and size of subproblems, complexity of the method without recursive calls. The wording states that the number of subproblems must be 4. Taking this into account, we will try to find values which will make the formula O(n^k), this being achieved by having number of problems (a) smaller than the size of the problem (b) ^ complexity of the problem without calls (k). Since we need the k to be 2 to satisfy the wording, we will need a b which satisfies that b^2 > 4; so, we will get the smaller value possible, b = 3.

Algorithmics	Student information	Date	Number of session
	UO: UO269412		3
	Surname: Carrillo		
	Name: Javier		

Activity 2. Uncle Scrooge

nTimes = 1000	Gilito1		Gilito2		Comparison	
Number of coins	Energy	Time	Energy	Time	Energy	Time
10	1		4		-3	
20	3		4		-1	
40	16		7		9	
80	40		8		32	
160	71		8		63	
320	2		8		-6	
640	190		11		179	
1280	419		10		409	
2560	582		11		571	
5120	103		12		91	
10240	567		15		552	
20480	183		14		169	
40960	8345	45	15	35	8330	10
81920	17280	159	18	70	17262	89
163840	13716	106	19	119	13697	-13
327680	44462	239	19	246	44443	-7
655360	128463	680	19	515	128444	165
1310720	439769	2313	22	1088	439747	1225
2621440	536424	2725	20	2173	536404	552
5242880	2025204	10794	23	4276	2025181	6518
10485760	5078780	25260	25	8412	5078755	16848
20971520	6176029	30839	24	17134	6176005	13705
41943040	18319365	91571	25	34323	18319340	57248

In this table we measured the times and energy it costs Gilito1 and Gilito2 to operate with a n quantity, to later then compare the two values in two columns. The first times weren't present since it took so little time it couldn't be compared.

The energy model is as follows, for example, getting the 10 coins example, we will calculate the balance one time to divide the sample in 5 and 5, then another balance will be made to compare the last two elements, thus reducing the sample to size 4. Then another call will be made to divide the sample in 2. Then, another call to compare such coins. As the sample is being increased by times 2, the energy will just be increasing in a

linear +1 manner, with some variations if the number of coins we get by dividing is being odd. So we can conclude that it follows a +1 trend in this sample, however, in the worst cases, it may be +a little more quantity because of the odd numbers

The time model has a complexity O(n), which makes sense considering a = 1, b = 2 and k = 1, thus O(n^k) = O(n^1) = O(n)

This can be proven with some examples (n1 = 40960, n2 = 81920, t1 = 35, t2 = 70, tht2 = 70; n1 =1310720, n2 = 2621440, t1 = 1088, t2 = 2173, tht2 = 2176)