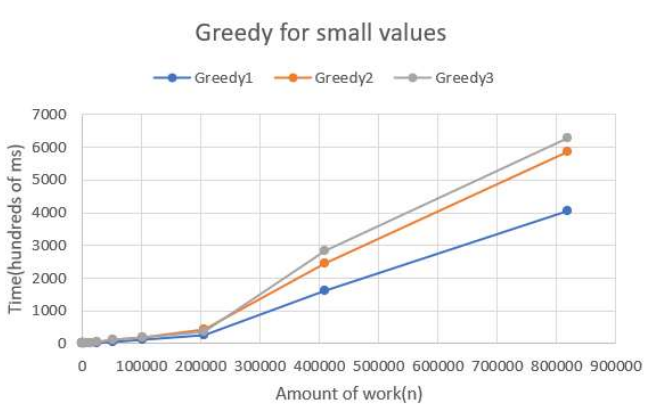
Algorithmics	Student information	Date	Number of session
	UO: 276903	18/03/2021	4
	Surname: Garriga Suárez		
	Name: Carlos		

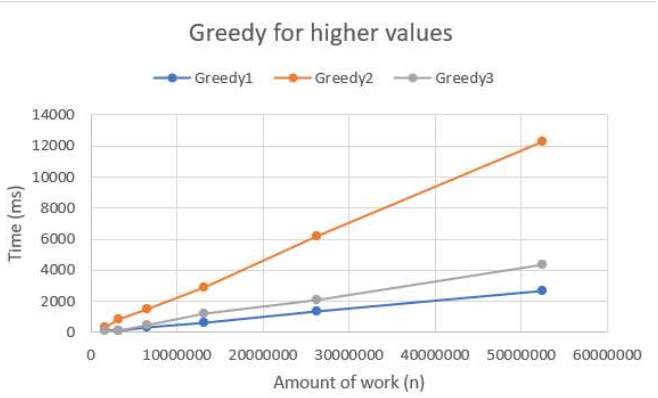
Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Execution Times

n	Greedy1	Greedy2	Greedy3
100	3	5	3
200	2	4	3
400	3	6	5
800	4	11	11
1600	17	12	9
3200	5	18	10
6400	9	24	26
12800	29	32	23
25600	31	50	62
51200	57	110	107
102400	131	185	204
204800	247	433	355
409600	1624	2453	2842
819200	4061	5874	6277
1638400	215	351	120
3276800	92	813	135
6553600	353	1497	487
13107200	632	2899	1210
26214400	1354	6191	2057
52428800	2658	12309	4351





The execution times for my different implementations of the algorithms were the ones that are on the table. I have to clarify that for amounts of work lower than 819200, I had the necessity of using nTimes as the times obtained were less than 50 milliseconds. That is why some values of time before that quantity are higher than the following ones.

Activity 2. Answer the following questions.

- **Explain if any of the greedy algorithms involves the optimal solution from the point of view of the company, which is interested in maximizing the number of "pufosos".**

Yes, in the case for greedy2 the algorithm is created in such a way to make the client paying more money. First you place the bigger segments, so for each iteration you will have to pay the segments already placed which at the end, result in a bigger amount of pufosos to be paid by the client.

Algorithmics	Student information	Date	Number of session
	UO: 276903	18/03/2021	4
	Surname: Garriga Suárez		
	Name: Carlos		

- **Explain if any of the greedy algorithms involves the optimal solution from the point of view of the player, who is interested in minimizing the number of "pufosos".**

Yes, in the case for greedy3 the algorithm is created in such a way to make the client paying less money. First you place the smaller segments, the amount of costs will result less so the accumulated cost will result also less too.

- **Explain the theoretical time complexities of the three greedy algorithms, according to the implementation made by each student, depending on the size of the problem n.**

Regarding my implementations of the different algorithms, with the first algorithm I just needed a two for loops. The first one was iterating all over the segments calculating the prices for each placement and storing them in another list and the second loop was to calculate the total cost for all the costs calculated previously in order to place the segments.
This algorithm would result in a O(n) complexity.

The second algorithm as I needed to place them from longest to shortest length, what I did was to use the method 'sort', from the Collections class of Java and then I used the 'reverse' method from the same class in order to have the segments ordered from longest to shortest length. The methods had O(nlogn) and O(n) complexity, respectively.
And then I applied the same pattern than greedy1.

The method 'sort' from collections is a kind of implementation of mergesort algorithm.

The last algorithm is just the same as greedy2 but without applying the method 'reverse' in order to have the segments ordered from shortest to longest length.

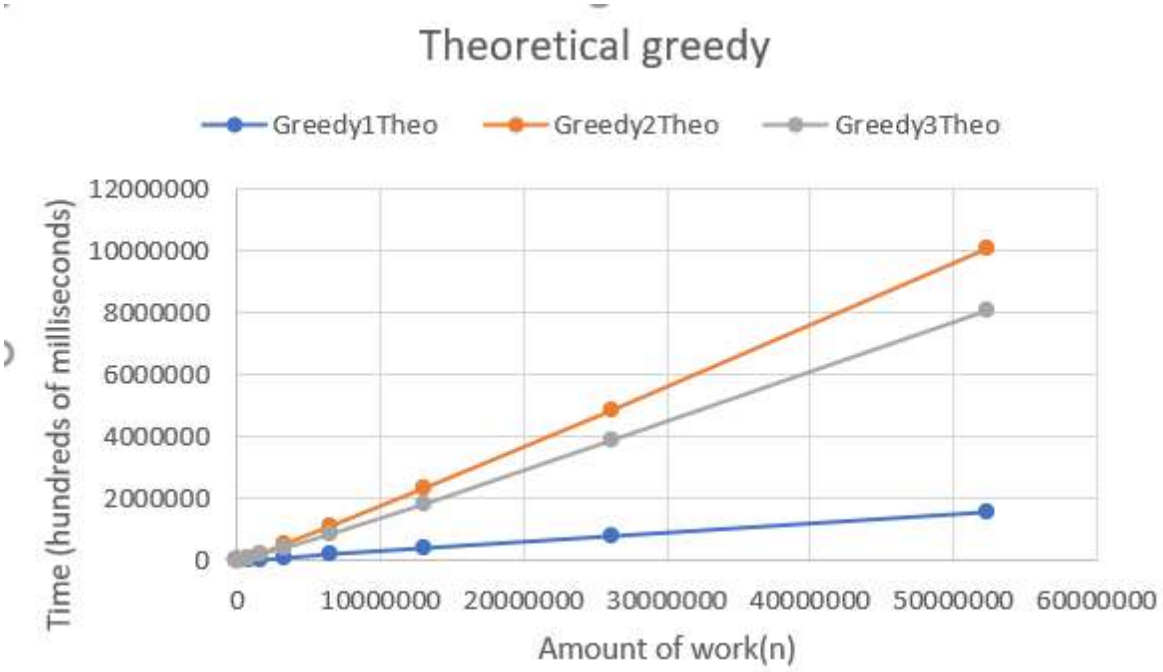
Finally, we have that greedy1, greedy2 and greedy3 have O(n), O(nlogn) and O(nlogn) complexities, respectively.

- **Explain if the times obtained in the table are in tune or not with the complexities set out in the previous section.**

Obviously the one that is going to last less is greedy1 as it has linear complexity.

Despite greedy2 and greedy3 have the same complexity we can observe that greedy3 have bigger values of time with bigger amounts of work compared to greedy3. This is because greedy3 does not need to use the method 'reverse' which has a linear complexity and in addition results in more time spent in the algorithm.

This can be reflected in the graphs shown above.



Here I have a graph with the theoretical values from the different algorithms. It is very representative compared to the graphs before as greedy2 tends to last more time at bigger values.

	Greedy1Theo	Greedy2Theo	Greedy3Theo
100	3	5	4
200	6	11,50515	9,20411998
400	12	26,0205999	20,8164799
800	24	58,0617997	46,4494398
1600	48	128,164799	102,531839
3200	96	280,411998	224,329599
6400	192	608,988796	487,191037
12800	384	1314,30719	1051,44575
25600	768	2821,27358	2257,01886
51200	1536	6027,86555	4822,29244

Algorithmics	Student information	Date	Number of session
	UO: 276903	18/03/2021	4
	Surname: Garriga Suárez		
	Name: Carlos		

102400	3072	12826,3679	10261,0943
204800	6144	27194,0094	21755,2075
409600	12288	57470,5659	45976,4527
819200	24576	121106,226	96884,9808
1638400	49152	254542,641	203634,113
3276800	98304	533745,659	426996,527
6553600	196608	1116812,07	893449,657
13107200	393216	2332265,65	1865812,52
26214400	786432	4861814,32	3889451,46
52428800	1572864	10118194,7	8094555,75

Here I left the values of the calculated times. I had to apply t2 = k * t1 for greedy1 as it has O(n) complexity and t2 = (log(n2) / log(n1)) * t1 for greedy2 and greedy3 as they have the same theoretical complexity.