Algorithmics	Student information	Date	Number of session
	UO: 293615		
	Surname: Lavelle		
	Name: Gersan		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Measure the loops

Repetitions tLoop1: 10000

Repetitions tLoop2: 1000

Repetitions tLoop3: 100

Repetitions tLoop4: 100

n	tLoop1	tLoop2	tLoop3	tLoop4
100	115	352	126	145
200	243	1608	593	1256
400	564	7282	2729	10014
800	1290	34457	11081	OoT
1600	2839	OoT	49729	OoT
3200	6148	OoT	OoT	OoT
6400	12463	OoT	OoT	OoT
12800	27438	OoT	OoT	OoT
25600	58400	OoT	OoT	OoT
51200	OoT	OoT	OoT	OoT

In **tLoop1**, the inner for loop has a complexity of O(n), while the outer while loop has a complexity of log(n^2). When multiplying them, we get that the program has a total complexity of O(n*log(n^2)).

In **tLoop2**, the for loop that uses the integer j follows an O(n) time complexity. The for loop that uses the integer i also follows an O(n) complexity. On the other hand, the while loop follows a complexity of O(log(n)). When we multiply to get the full complexity we get O(n^{2*}log(n)).

Algorithmics	Student information	Date	Number of session
	UO: 293615		
	Surname: Lavelle		
	Name: Gersan		

In **tloop3**, the for loop using the variable k follows an O(log(n)) complexity. The for loop using the j variable follows a complexity of O(n). This time, the while loop follows a complexity of O(n). The full complexity after doing the multiplication is O(n^2*log(n)).

Finally, in **tLoop4,** all three loops follow a complexity of O(n), so the total complexity will be O(n^3).

Activity 2. Create loops with a given time complexity

Repetitions tLoop5: 100	Time complexity: O(n^2*$\log^2$n)
Repetitions tLoop6: 10	Time complexity: O(n3*logn)
Repetitions tLoop7:	Time complexity: 10 O(n^4)

n	tLoop5	tLoop6	tLoop7
100	298	373	446
200	1369	2934	5731
400	7126	28641	OoT
800	35153	OoT	OoT
1600	OoT	OoT	OoT
3200	OoT	OoT	OoT
6400	OoT	OoT	OoT

After making the measurements in each of the loops, we can conclude that each of the loops' increase is roughly proportional to the time complexity I specified above the chart.

Algorithmics	Student information	Date	Number of session
	UO: 293615		
	Surname: Lavelle		
	Name: Gersan		

Activity 3. Comparison of loop1 and loop2

The measurements in both cases are the same ones from earlier on, also with the same number of repetitions.

n	tLoop1	tLoop2	t1/t2
100	115	352	0.3267
200	243	1608	0.15112
400	564	7282	0.07745
800	1290	34457	0.03744
1600	2839	OoT	OoT
3200	6148	OoT	OoT
6400	12463	OoT	OoT
12800	27438	OoT	OoT
25600	58400	OoT	OoT
51200	OoT	OoT	OoT

In this case, the ratio tends to 0, which happens when the algorithm associated to the denominator is the most complex one. Loop1 has a complexity of O(n*log(n^2)) while loop2 to has a complexity of O(n^{2*}log(n)). So the results are correct.

Algorithmics	Student information	Date	Number of session
	UO: 293615		
	Surname: Lavelle		
	Name: Gersan		

Activity 4. Comparison of loop2 and loop3

n	tLoop3	tLoop2	t3/t2
100	126	352	0.35795
200	593	1608	0.36878
400	2729	7282	0.37476
800	11081	34457	0.32159
1600	49729	OoT	OoT
3200	OoT	OoT	OoT
6400	OoT	OoT	OoT
12800	OoT	OoT	OoT
25600	OoT	OoT	OoT
51200	OoT	OoT	OoT

In this case we can see that the ratio is more or less constant. All values are < 1, so we can consider that the Loop3 has a better complexity than loop2. Loop2 has a complexity of O(n^{2*}log(n)) while loop3 has a complexity of O(n^3).

Algorithmics	Student information	Date	Number of session
	UO: 293615		
	Surname: Lavelle		
	Name: Gersan		

Activity 5. Comparing java and python

Now lets work with loop4 in python and java with and without optimization.

Num of repetitions for python: 100

Num of repetitions for java without optimization: 100

Num of repetitions for java with optimization: 10000

n	tLoop4 python	tLoop4 java (no opt.)	tLoop4 java (opt.)	no opt./python	opt./no opt
100	435	145	171	0.333333333	1.17931034
200	3313	1256	756	0.379112587	0.60191083
400	25693	10014	3938	0.389755965	0.39324945
800	OoT	OoT	23510	OoT	OoT
1600	OoT	OoT	OoT	OoT	OoT
3200	OoT	OoT	OoT	OoT	OoT
6400	OoT	OoT	OoT	OoT	OoT
12800	OoT	OoT	OoT	OoT	OoT
25600	OoT	OoT	OoT	OoT	OoT
51200	OoT	OoT	OoT	OoT	OoT

When comparing the algorithm in python with the non-optimized java one, we get a more or less constant ratio, as expected. The ratio is smaller than one, so the java version is better.

Also as expected, the ratio for the two java cases is very unstable, due to the actions of the java omtimizer.