Algorithmics	Student information	Date	Number of session
	UO: 294067	14/02/24	1.2
	Surname: Díaz Álvarez		
	Name: Paula		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Times for iterative methods

$f(n_1)$ -> t_1 $\qquad$ $t_2 = \frac{f(n_2)}{f(n_1)} \cdot t_1 = k \cdot t_1$

$f(n_2)$ -> t_2?

TABLE1 (times in milliseconds and WITHOUT OPTIMIZATION):

n	T Loop 1	T Loop 2	T Loop 3	T Loop 4
100	0.0062	0.21	1.9	1.5
200	0.0114	0.80	7.8	6.4
400	0.0288	3.71	28.9	49.8
800	0.0638	17.65	89.2	405.3
1600	0.1394	71.41	302.2	4047.8
3200	0.2943	466.36	1457.3	OoT
6400	0.6318	OoT	OoT	OoT
12800	1.4708	OoT	OoT	OoT
25600	3.2846	OoT	OoT	OoT
51200	5.8693	OoT	OoT	OoT

- **Loop1**:
  - Complexity: O(n log(n))
  - We see if the complexity is the one expected with the more representative values (the biggest times obtained):

$t_{2,1} = \frac{2*1600\ log\ (2*1600)}{1600\ log\ (1600)} \cdot 0.1394 = 0.305$ Real=0.2943

$t_{2,2} = 2 \cdot \frac{log\ (2*3200)}{log\ (3200)} \cdot 0.2943 = 0.6392$ Real=0.6318

$t_{2,3} = 2 \cdot \frac{log\ (2*6400)}{log\ (6400)} \cdot 0.6318 = 1.3638$ Real=1.4708

$t_{2,4} = 2 \cdot \frac{log\,(2*12800)}{log\,(12800)} \cdot 1.4708 = 3.1572$ Real=3.2846

$t_{2,5} = 2 \cdot \frac{log\,(2*25600)}{log\,(25600)} \cdot 3.2846 = 7.0178$ Real=5.8693

- **Loop2**:
  - Complexity: $O(n^2 \log(n))$

$t_{2,1} = \frac{(2\cdot 100)^2\, log\,(2\cdot 100)}{100^2\, log\,(100)} \cdot 0.21 = 4 \cdot \frac{log\,(2\cdot 100)}{log\,(100)} \cdot 0.21 = 0.9664$ Real = 0.80

$t_{2,2} = 4 \cdot \frac{log\,(2\cdot 200)}{log\,(200)} \cdot 0.80 = 3.6186$ Real=3.71

$t_{2,3} = 4 \cdot \frac{log\,(2\cdot 400)}{log\,(400)} \cdot 3.71 = 16.56$ Real=17.65

$t_{2,4} = 4 \cdot \frac{log\,(2\cdot 800)}{log\,(800)} \cdot 17.65 = 77.92$ Real=71.41

$t_{2,5} = 4 \cdot \frac{log\,(2\cdot 1600)}{log\,(1600)} \cdot 71.41 = 312.48$ Real=466.36

- **Loop3**:
  - Complexity: $O(n^2 \log(n))$

$t_{2,1} = 4 \cdot \frac{log\,(2\cdot 100)}{log\,(100)} \cdot 1.9 = 8.7439$ Real = 7.8

$t_{2,2} = 4 \cdot \frac{log\,(2\cdot 200)}{log\,(200)} \cdot 7.8 = 35.2817$ Real=28.9

$t_{2,3} = 4 \cdot \frac{log\,(2\cdot 400)}{log\,(400)} \cdot 28.9 = 128.97$ Real=89.2

$t_{2,4} = 4 \cdot \frac{log\,(2\cdot 800)}{log\,(800)} \cdot 89.2 = 393.8$ Real=302.2

$t_{2,5} = 4 \cdot \frac{log\,(2\cdot 1600)}{log\,(1600)} \cdot 302.2 = 1322.37$ Real=1457.3

- **Loop4**:
  - Complexity: $O(n^3)$

$t_{2,1} = \frac{(2\cdot 100)^3}{100^3} \cdot 1.5 = 8 \cdot 1.5 = 12$ Real=6.4

$t_{2,2} = 8 \cdot 6.4 = 51.2$ Real=49.8

$t_{2,3} = 8 \cdot 49.8 = 398.4$ Real=405.3

$t_{2,4} = 8 \cdot 405.3 = 3242.4$ Real=4047.8

Activity 2. Times for iterative methods given a complexity

TABLE2 (times in milliseconds and WITHOUT OPTIMIZATION):

n	T Loop 5	T Loop 6	T Loop 7
100	1.3	8.7	65
200	5.8	85.7	832
400	30.5	808.4	14134
800	153.2	OoT	OoT
1600	726.1	OoT	OoT
3200	4656.5	OoT	OoT
6400	OoT	OoT	OoT

- **Loop5**:
  - Complexity: $O(n^2 \log^2(n))$
  - We see if the complexity is the one expected with the more representative values (the biggest times obtained):

$t_{2,1} = \frac{(2*100)^2 \log^2(2*100)}{100^2 \log^2(100)} \cdot 1.3 = 4 \cdot \frac{\log^2(2*100)}{\log^2(100)} \cdot 1.3 = 6.88$ Real=5.8

$t_{2,2} = 4 \cdot \frac{\log^2(2*200)}{\log^2(200)} \cdot 5.8 = 29.667$ Real=30.5

$t_{2,3} = 4 \cdot \frac{\log^2(2*400)}{\log^2(400)} \cdot 30.5 = 151.861$ Real=153.2

$t_{2,4} = 4 \cdot \frac{\log^2(2*800)}{\log^2(800)} \cdot 153.2 = 746.475$ Real=726.1

$t_{2,5} = 4 \cdot \frac{log^2 (2*1600)}{log^2 (1600)} \cdot 726.1 = 3475.779$ Real=4656.5

- **Loop6**:
  - Complexity: O(n^3 log(n))

$t_{2,1} = \frac{(2*100)^3\, log\,(2*100)}{100^3\ log\,(100)} \cdot 8.7 = 8 \cdot \frac{log\,(2*100)}{log\,(100)} \cdot 8.7 = 80.076$ Real=85.7

$t_{2,2} = 8 \cdot \frac{log\,(2*200)}{log\,(200)} \cdot 85.7 = 775.292$ Real=808.4

- **Loop7**:
  - Complexity: O(n^4)

$t_{2,1} = \frac{(2*100)^4}{100^4} \cdot 65 = 16 \cdot 65 = 1040$ Real=832

$t_{2,2} = 16 \cdot 832 = 13312$ Real=14134

Activity 3. Comparison of two algorithms with different complexity

TABLE3 (times in milliseconds and WITHOUT OPTIMIZATION):

n	T Loop 1	T Loop 2	t1 / t2
100	0.0062	0.21	0.0295
200	0.0114	0.80	0.0143
400	0.0288	3.71	0.0078
800	0.0638	1.765	0.0361
1600	0.1394	7.141	0.0195
3200	0.2943	46.636	0.0063
6400	0.6318	OoT	~ 0

12800	1.4708	OoT	~ 0
25600	5.2693	OoT	~ 0
51200	OoT	OoT	~ 0

Loop 1 has O(n log(n)) complexity, and Loop 2 has $O(n^2 \log(n))$

As both algorithms have different complexities, when the size grows, the ratio tends to 0, so Loop 1 has a lower complexity compared with Loop 2 (As predicted before)

Activity 4. Comparison of two algorithms with same complexity

TABLE4 (times in milliseconds and WITHOUT OPTIMIZATION):

n	T Loop 3	T Loop 2	t3 / t2
100	1.9	0.21	9.0476
200	7.8	0.80	9.75
400	28.9	3.71	7.7898
800	89.2	1.765	50.5382
1600	302.2	7.141	42.3190
3200	1457.3	46.636	31.2463
6400	OoT	OoT	-
12800	OoT	OoT	-
25600	OoT	OoT	-
51200	OoT	OoT	-

Loop 3 and 2 have $O(n^2 \log(n))$ complexity.

As both algorithms have the same complexity, the ratio tends to a constant, and as it's greater than 1; Loop 2 has a better complexity

Activity 5. Comparison of two algorithms in different development environments

TABLE5 (times in milliseconds):

n	T Loop 4: t41 (Python)	T Loop 4 : t42 (Java, no optimization)	T Loop 4 : t43 (Java with optimization)	t42/t41	t43/t42
100	0	1.5	0.048	inf	0.032
200	38	6.4	0.152	0.1684	0.0236
400	208	49.8	0.753	0.2394	0.0151
800	1898	405.3	4.683	0.2135	0.0116
1600	18427	4047.8	37.959	0.2197	0.0094
3200	OoT	OoT	OoT	-	-
6400	OoT	OoT	OoT	-	-

They are the same algorithm, so they have the same complexity, so the ratio tends to a constant:

- For t42/t41: the ratio tends to < 1, so the Java version with no optimization is better than the Python version
- For t43/t42: the ratio tends to <1, so the Java version with optimization is better than the Java version with no optimization