Activity 1. [Loop1.java, Loop2.java, Loop3.java and Loop4.java times in milliseconds and WITHOUT OPTIMIZATION):]

N	time_Loop1	time_Loop2	time_Loop3	time_Loop4
100	0.01	0.1	1	1
200	0.01	0.6	4	5
400	0.02	2.9	15	37
800	0.05	12.4	62	283
1600	0.1	47.8	260	2152
3200	0.22	217.9	1081	16856
6400	0.44	865.4	4488	Oot
12800	1	3918.6	19123	Oot
25600	2.03	1757	Oot	Oot
51200	4.36	7000.8	Oot	Oot

Loop1 = O(log^2(n))

Loop2 = O(n*log^2(n))

Loop3 = O(n*log^2(n))

Loop4 = O(n^3)

Activity 2. [Loop5.java, Loop6.java, Loop7.java times in milliseconds and WITHOUT OPTIMIZATION

N	Loop5	Loop6	Loop7
100	5	73	526
200	27	602	8088
400	131	5078	OoT
800	607	43995	OoT
1600	2803	OoT	OoT
3200	12953	OoT	OoT
6400	58494	OoT	OoT

Activity 3. [TABLE3 (times in milliseconds and WITHOUT OPTIMIZATION)]

N	time_loop1 (s)	time_loop2 (s)	T1/T2 Ratio
100	0.01	0.1	0.10
200	0.01	0.6	0.0167
400	0.02	2.9	0.0069
800	0.05	12.4	0.0040
1600	0.1	47.8	0.0021
3200	0.22	217.9	0.0010
6400	0.44	865.4	0.0005
12800	1.00	3918.6	0.00026

25600	2.03	1757	0.00116
51200	4.36	7000.8	0.00062

As the ratio goes to 0, T1 is better than T2

Activity 4. [TABLE4 (times in milliseconds and WITHOUT OPTIMIZATION)]

N	time_loop3 (s)	time_loop2 (s)	T3/T2 Ratio
100	1.0	0.1	10.000
200	4.0	0.6	6.667
400	15.0	2.9	5.172
800	62.0	12.4	5.000
1600	260.0	47.8	5.439
3200	1081.0	217.9	4.961
6400	4488.0	865.4	5.186
12800	19123.0	3918.6	4.880
25600	Oot	1757.0	Oot
51200	Oot	7000.8	Oot

As ratio keeps constant (5) T3 has the same complexity than T2, although T3 is 5 times worse

Activity 5. ([Same algorithm in different development environments])

N	Python(T41)	JAVA OPT(T43)	JAVA NO OPT(T42)	T42/T41	T43/T42
100	27	2	4	0.148	0.5
200	130	3	15	0.115	0.2
400	578	6	106	0.18	0.05
800	6009	4	708	0.117	0.005
1600	36091	30	5444	0.15	0.005
3200	Oot	184	42798		0.004
6400	Oot	1180	Oot		
12800	Oot	8062	Oot		
25600	Oot	61144	Oot		

Taking the ratios into account, T42 is better than T41 and T43 is better than T42