Activity 1. [Iterative Models]

Table 1. In milliseconds and without optimization. CHANGE THIS USING THE CORRECT THING

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | tLoop1 | tLoop2 | tLoop3 | tLoop4 |
| 100 | 61\*10^-4 | 176\*10^3 | 120\*10^-2 | 123\*10^-2 |
| 200 | 125\*10^-4 | 588\*10^-3 | 399\*10^-2 | 760\*10^-2 |
| 400 | 203\*10^-4 | 2862\*10^-3 | 1514\*10^-2 | 5534\*10^-2 |
| 800 | 588\*10^-4 | 166\*10^-1 | 6906\*10^-2 | 451 |
| 1600 | 1159\*10-4 | 505\*10^-1 | 302 | 3408 |
| 3200 | 264\*10^-3 | 2251\*10^-1 | 1124 | 28931 |
| 6400 | 422\*10^-3 | 915 | 4964 | OoT |
| 12800 | 1042\*10^-3 | 4153 | 20107 | OoT |
| 25600 | 231\*10^-2 | 18902 | OoT | OoT |
| 51200 | 461\*10^-2 | OoT | OoT | OoT |

For Loop1 it has a complexity of O(n\*log(n)), and it does full fill it.

For Loop2 it has a complexity of O(n^2\*log(n)), it follows it.

Loop 3 has a complexity of O(n^2\*log(n)), it follows the complexity.

Loop 4 has a complexity of O(n^3), it follows it as it increases very quickly.

Activity 2. [Create models of given complexity]

|  |  |  |  |
| --- | --- | --- | --- |
| N | tLoop5 | tLoop6 | tLoop7 |
| 100 | 507\*10^-3 | 275\*10^-1 | 794 |
| 200 | 2042\*10^-3 | 256 | 11390 |
| 400 | 144\*10^-1 | 1660 | OoT |
| 800 | 575\*10^-1 | 15562 | OoT |
| 1600 | 340 | 4777 | OoT |
| 3200 | 1206 | 42739 | OoT |
| 6400 | 5933 | OoT | OoT |

Loop5 complexity is O(n^2\*log^2(n))

Loop6 complexity is O(n^3\*log(n))

Loop7 complexity is O(n^4)

They all follow the expected complexity.

Activity 3. [Comparison of two algorithms]

|  |  |  |  |
| --- | --- | --- | --- |
| n | tLoop1 | tLoop2 | t1/t2 |
| 100 | 61\*10^-4 | 176\*10^3 | 0,035 |
| 200 | 125\*10^-4 | 588\*10^-3 | 0,021 |
| 400 | 203\*10^-4 | 2862\*10^-3 | 0,007 |
| 800 | 588\*10^-4 | 166\*10^-1 | 0,003 |
| 1600 | 1159\*10-4 | 505\*10^-1 | 0,0023 |
| 3200 | 264\*10^-3 | 2251\*10^-1 | 1,17\*10^-3 |
| 6400 | 422\*10^-3 | 915 | 0,0004 |
| 12800 | 1042\*10^-3 | 4153 | 0,0011 |
| 25600 | 231\*10^-2 | 18902 | 1,22\*10^-4 |
| 51200 | 461\*10^-2 | OoT | OoT |

The algorithm used in Loop1 is clearly better, as the ratio tends to 0. And this is correct as the complexity of O(n\*log(n)) is better than O(n^2\*log(n)).

|  |  |  |  |
| --- | --- | --- | --- |
| n | tLoop3 | tLoop2 | t3/t2 |
| 100 | 120\*10^-2 | 176\*10^3 | 6,81 |
| 200 | 399\*10^-2 | 588\*10^-3 | 6,78 |
| 400 | 1514\*10^-2 | 2862\*10^-3 | 5,29 |
| 800 | 6906\*10^-2 | 166\*10^-1 | 4,16 |
| 1600 | 302 | 505\*10^-1 | 5,98 |
| 3200 | 1124 | 2251\*10^-1 | 4,99 |
| 6400 | 4964 | 915 | 5,42 |
| 12800 | 20107 | 4153 | 4,84 |
| 25600 | OoT | 18902 | OoT |
| 51200 | OoT | OoT | OoT |

This time we can see that Loop2 is a better algorithm than Loop3 even though they have the same complexity.

TABLE 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | tLoop4 (Python)-t41 | tLoop4 (Java without optimization) – t42 | tLoop4 (Java with optimization) – t43 | t42/t41 | t43/t42 |
| 100 | 3 | 123\*10^-2 | 173\*10^-4 | 0,41 | 0,01 |
| 200 | 24 | 760\*10^-2 | 789\*10^-4 | 0,316 | 0,01 |
| 400 | 189 | 5534\*10^-2 | 3980\*10^-4 | 0,29 | 7,19\*10^-3 |
| 800 | 1557 | 451 | 232\*10^-2 | 0,289 | 0,005 |
| 1600 | 12843 | 3408 | 1603\*10^-2 | 0,26 | 4,7\*10^-3 |
| 3200 | OoT | 28931 | 10909\*10^-2 | OoT | 3,77\*10^-3 |
| 6400 | OoT | OoT | 111 | OoT | OoT |

Java is better than python, as we saw it is a compiler language. Java with optimization is better as it takes advantage of the aspect of java being a compiler language.