Activity 1. Measuring times of different complexity algorithms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **tLoop1** | **tLoop2** | **tLoop3** | **tLoop4** |
| **100** | 56 | 1813 | 5682 | 5850 |
| **200** | 105 | 6648 | 26855 | 45587 |
| **400** | 228 | 31459 | OoT | OoT |
| **800** | 537 | OoT | OoT | OoT |
| **1600** | 1238 | OoT | OoT | OoT |
| **3200** | 2585 | OoT | OoT | OoT |
| **6400** | 5620 | OoT | OoT | OoT |
| **12800** | 14466 | OoT | OoT | OoT |
| **25600** | 29248 | OoT | OoT | OoT |
| **51200** | OoT | OoT | OoT | OoT |

Loop1: this algorithm consists on two nested loops. The outer one iterates from 1 to n2, multiplying the counter times 3 in each iteration. Therefore, the loop iterates log3(n2) times. Nested inside the outer loop, there is a loop iterating linearly from 1 to 2n with a 3-unit step, so this this inner loop iterates 2n/3 times. The overall number of iterations of these two loops is: (2n/3)\*(log3(n2)), this means Loop1 has complexity O(n\*log(n)).

Loop2: this algorithm has three nested loops. The outermost iterates from n to 1, dividing the counter by three each time. This means the outermost loop iterates log3(n) times. The middle loop iterates from 1 to n with a 1-unit step. Therefore, it iterates n times. Finally, the innermost one iterates from n to 0, with a negative 2-unit step, which means it iterates n/2 times. The overall number of iterations of these two loops is: log3(n)\*n\*n/2, this implies Loop2 has a complexity O(n2log(n)).

Loop3: this algorithm consists on three nested loops. The outermost one iterates from 1 to 2n with a 1-unit step, which means it iterates 2n times. The middle one iterates from i to zero, with a negative 2-unit step. Finally, the last loop iterates from 1 to n, multiplying the counter by two each time, therefore it iterates log2(n) times. Since the middle loop iterates according to the value of i, we can assure that the complexity of the middle loop is also n, since the outermost loop is also dependent on n. Therefore, the overall complexity of Loop3 is O(n2log(n)).

Loop4: following a reasoning like the one made for loop 3, since the outermost loop of this method iterates n times, the inner loop iterates a number of times proportional to i and the innermost one iterates a number of times proportional to j, we can assure the overall complexity of this algorithm is O(n3).

If we analize the times obtained for each algorithm, we can clearly see that the times increase accordingly to the complexity of the different algorithms used. However, it is worth mentioning that, even though the complexity of Loop2 and Loop3 is the same, it looks like there is a great difference among the times measured, but the size of the sample should be taken into account in order to understand these values, since the sample is quite small and the base of the logarithms is different, even though the growth trend of both algorithms is the same.

Activity 2. Creation of iterative models of a given time complexity

|  |  |  |  |
| --- | --- | --- | --- |
| N | tLoop5 | tLoop6 | tLoop7 |
| 100 | 67 | 631 | 10455 |
| 200 | 285 | 5987 | OoT |
| 400 | 1300 | 58812 | OoT |
| 800 | 6335 | OoT | OoT |
| 1600 | 28969 | OoT | OoT |
| 3200 | OoT | OoT | OoT |
| 6400 | OoT | OoT | OoT |

Activity 3. Two algorithms with different complexity

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **tLoop1** | **tLoop2** | **t1/t2** |
| **100** | 56 | 1813 | 0.03088 |
| **200** | 105 | 6648 | 0.01579 |
| **400** | 228 | 31459 | 0.00724 |
| **800** | 537 | OoT | NaN |
| **1600** | 1238 | OoT | NaN |
| **3200** | 2585 | OoT | NaN |
| **6400** | 5620 | OoT | NaN |
| **12800** | 14466 | OoT | NaN |
| **25600** | 29248 | OoT | NaN |
| **51200** | OoT | OoT | NaN |

The quotient makes sense, since the complexity O(nlogn) is lower than the complexity O(n2logn), as n increases, the times for loop 1 increase, and the times for loop 2 increase quite faster, since the complexity is higher. That implies that as n increases, the quotient t1/t2 tends to 0, since tloop2 increases faster than tloop1.

Activity 3. Two algorithms with the same complexity

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **tLoop2** | **tLoop3** | **t3/t2** |
| **100** | 118 | 439 | 3.72036 |
| **200** | 420 | 1840 | 4.38095 |
| **400** | 2048 | 7887 | 3.85107 |
| **800** | 8949 | 32480 | 3.62945 |
| **1600** | 33777 | OoT | NaN |
| **3200** | OoT | OoT | NaN |
| **6400** | OoT | OoT | NaN |
| **12800** | OoT | OoT | NaN |
| **25600** | OoT | OoT | NaN |
| **51200** | OoT | OoT | NaN |

As it can be seen, the proportion of times is ***more or less*** the same for each size of N. Since these two algorithms have the same theoretical complexity, it is just what is supposed to happen, they have directly proportional times.

Activity 4. Same algorithm in different development environments

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | tLoop4  python  t41 | tLoop4  non-optimized java  t42 | tLoop4  optimized  java  t43 | T42/t41 | T43/t42 |
| 200 | 1389 | 64 | 6 | 0.04607 | 0.09375 |
| 400 | 10199 | 411 | 8 | 0.04029 | 0.01946 |
| 800 | OoT | 2989 | 47 | NaN | 0.01572 |
| 1600 | OoT | 21896 | 240 | NaN | 0.01096 |
| 3200 | OoT | OoT | 1316 | NaN | NaN |
| 6400 | OoT | OoT | 7648 | NaN | NaN |

Since the three algorithms are the same, the proportion of times is more or less the same for any time, so t42/t41 is constant and t43/t42 is also constant. This makes absolute sense because the algorithm is the same and therefore the theretical complexity.