Algorithms and data structures task 1

# Task 1-2

A screenshot of a computer

Description automatically generated with medium confidence

The complexity of the program is 5n^2 -5n +4. We can see that its upper and lower bound can be expressed by a constant \* n^2.

0 <= C\_1 \* n^2 <= 5n^2-5n+4 <= C\_2 \* n^2

C\_1 can be 3 and n\_0 = 1 and C\_2 = 5, thus proving that n^2 can be a lower and upper bound for our complexity, 5n^2-5n +4. The algorithm here has the time complexity described by Θ(n^2).

# Task 1-3

|  |  |
| --- | --- |
| Total elements - n | Time usage in ms |
| 10 | ≈ 3.86 \* 10^-5 |
| 100 | ≈ 0.014 |
| 1000 | ≈ 0.137 |
| 10000 | ≈ 14.96 |
| 100000 | ≈ 3051 |
| 1000000 | ≈ |

We shall now analyze the results with our predicted time complexity, which can be simplified to n^2.

With 10 elements in the array, we get our result in approximately 3.86\*10^-5 milliseconds.

This means that the proportionality constant k can be based on our first result with 10 elements.

3.86\*10^-5 ms = k \* n^2 , note that on our first try we had 10 elements, therefore we can use n = 10

This gives us k = 3.86\*10^-5/10^2

We will now use this proportionality constant to confirm if our algorithm’s complexity is in fact approximately n^2 (quadratic).

For n = 100, using the constant k we can predict how long it could take. K\*n^2 should give us the approximate time usage, here it would be 3.86\*10^-5/10^2 \* 100^2 milliseconds, time usage should therefore be approximately 0.00386 milliseconds. The difference between our prediction and the actual result from our test is 0.014 - 0.00386 = 0.01014 milliseconds.

For n = 1000, 3.86\*10^-5/10^2 \* 1000^2 = 0.386, The difference between our prediction and the actual result from our test is 0.386 - 0.014 = 0.249 milliseconds.

For n = 10000, 3.86\*10^-5/10^2 \* 10000^2 = 38.6, The difference between our prediction and the actual result from our test is 38.6 - 14.96 = 23.64 milliseconds.

For n = 100000, 3.86\*10^-5/10^2 \* 100000^2 = 3860, The difference between our prediction and the actual result from our test is 3860 – 3051 = 809 milliseconds.

**Conclusion:** we can see that the time usage follows a quadratic pattern and that our estimates scale with the same pattern. The increasing differences between prediction and our test result is because of the small differences at the start making a big impact in the end, since quadratic functions like n^2 start slow and start scaling at an ever-increasing acceleration towards infinity.

Its also important not to focus on small amounts of n because our test data won’t be accurate enough for cross checking.

A good observation that will support our claim is the time usage jump between n = 10000 and n = 100000. If n increases 10-fold, the time usage should increase 100-fold. We can see a similar pattern on both our test data and predicted result that both show that it has increased 100-fold.

14.96 \* 100 = 1496 and our test result was 3051, both our test result and predicted result have 4 digits.