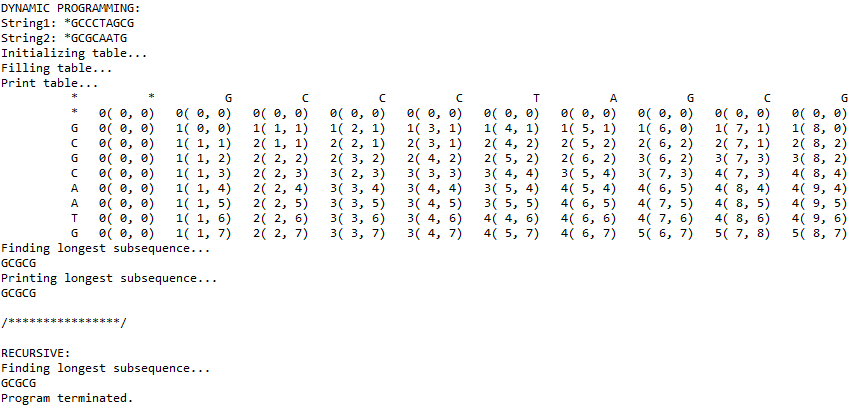
Activity 1. Validation results

The output of the *LCSTest* class with inputs ‘GCCCTAGCG’ and ‘GCGCAATG’ is:

Activity 2. Experimental time measurements

|  |  |
| --- | --- |
| **n** | **t\_dynamic (ms)** |
| 100 | 1.6 |
| 200 | 2.7 |
| 400 | 4.7 |
| 800 | 12.3 |
| 1600 | 46.7 |
| 3200 | 205.6 |
| 6400 | 706.2 |

|  |  |
| --- | --- |
| **n** | **t\_recursive (ms)** |
| 2 | 0.1 |
| 3 | 0.1 |
| 4 | 0.2 |
| 5 | 0.3 |
| 6 | 0.4 |
| 7 | 0.6 |
| 8 | 1 |
| 9 | 1.6 |
| 10 | 3.9 |
| 11 | 6.7 |
| 12 | 14.1 |
| 13 | 29.5 |
| 14 | 50.4 |
| 15 | 131.6 |
| 16 | 268.9 |
| 17 | 807.8 |
| 18 | 1385.8 |
| 19 | 4007.6 |
| 20 | 13188.3 |
| 21 | 42123.4 |

Activity 3. Questions

1. **Determine theoretically complexities (time, memory space and waste of stack) for both implementations, recursive (approximated) and using programming dynamic.**

The recursive method time complexity and memory space is , that we can deduce from the values of the table and graph. The waste of stack is because the method can execute 2 recursive calls.

The dynamic method time complexity and memory space is ; where *n* and *m* are the lengths of the input strings, that we can deduce from the values of the table and graph. There is no waste of stack because there are no recursive calls.

1. **Compute theoretical times and compare them with the experimental measurements.**

Recursive:

t1 = 3.9 ms, n1 = 10 and n2 = 11; then, = 7.8 ms, the calculated value is greater than the obtained experimentally because the time complexity is an approximation; nevertheless, the difference is not much.

Dynamic:

t1 = 4.7 ms, n1 = 400 and n2 = 800; then, = 18.8 ms, the calculated value is greater than the obtained experimentally (12.3 ms).

1. **Why** **large sequences cannot be processed with the recursive implementation? Explain why dynamic programming implementation raises an exception for large sequences.**

Large sequences cannot be processed with the recursive implementation because there is not enough free memory to store the result of the recursive calls.

1. **The amount of possible LCS can be more than one, e. g. GCCCTAGCG and GCGCAATG has two GCGCG and GCCAG. Find the code section that determines which subsequence is chosen, modify this code to verify that both solutions can be achieved.**