Activity 1. Validation results

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

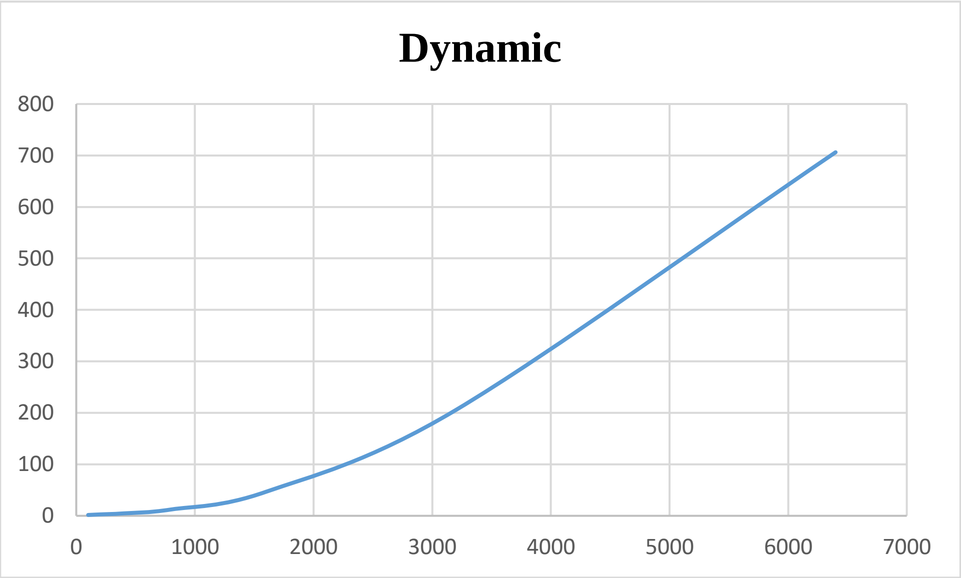
```
DYNAMIC PROGRAMMING:
String1: *GCCCTAGCG
String2: *GCGCAATG
Initializing table...
Filling table...
Print table...
         *          *          G          C          C          C          T          A          G          C          G
         *   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)   0( 0, 0)
         G   0( 0, 0)   1( 0, 0)   1( 1, 1)   1( 2, 1)   1( 3, 1)   1( 4, 1)   1( 5, 1)   1( 6, 0)   1( 7, 1)   1( 8, 0)
         C   0( 0, 0)   1( 1, 1)   2( 1, 1)   2( 2, 1)   2( 3, 1)   2( 4, 2)   2( 5, 2)   2( 6, 2)   2( 7, 1)   2( 8, 2)
         G   0( 0, 0)   1( 1, 2)   2( 2, 2)   2( 3, 2)   2( 4, 2)   2( 5, 2)   2( 6, 2)   3( 6, 2)   3( 7, 3)   3( 8, 2)
         C   0( 0, 0)   1( 1, 3)   2( 2, 3)   3( 2, 3)   3( 3, 3)   3( 4, 4)   3( 5, 4)   3( 7, 3)   4( 7, 3)   4( 8, 4)
         A   0( 0, 0)   1( 1, 4)   2( 2, 4)   3( 3, 4)   3( 4, 4)   3( 5, 4)   4( 5, 4)   4( 6, 5)   4( 8, 4)   4( 9, 4)
         A   0( 0, 0)   1( 1, 5)   2( 2, 5)   3( 3, 5)   3( 4, 5)   3( 5, 5)   4( 6, 5)   4( 7, 5)   4( 8, 5)   4( 9, 5)
         T   0( 0, 0)   1( 1, 6)   2( 2, 6)   3( 3, 6)   3( 4, 6)   4( 4, 6)   4( 6, 6)   4( 7, 6)   4( 8, 6)   4( 9, 6)
         G   0( 0, 0)   1( 1, 7)   2( 2, 7)   3( 3, 7)   3( 4, 7)   4( 5, 7)   4( 6, 7)   5( 6, 7)   5( 7, 8)   5( 8, 7)
Finding longest subsequence...
GCGCG
Printing longest subsequence...
GCGCG

/****************/

RECURSIVE:
Finding longest subsequence...
GCGCG
Program terminated.
```

Algorithmics	Student information	Date	Number of session
	UO: 277921	23/3/2021	5
	Surname: García López		
	Name: Rosa		

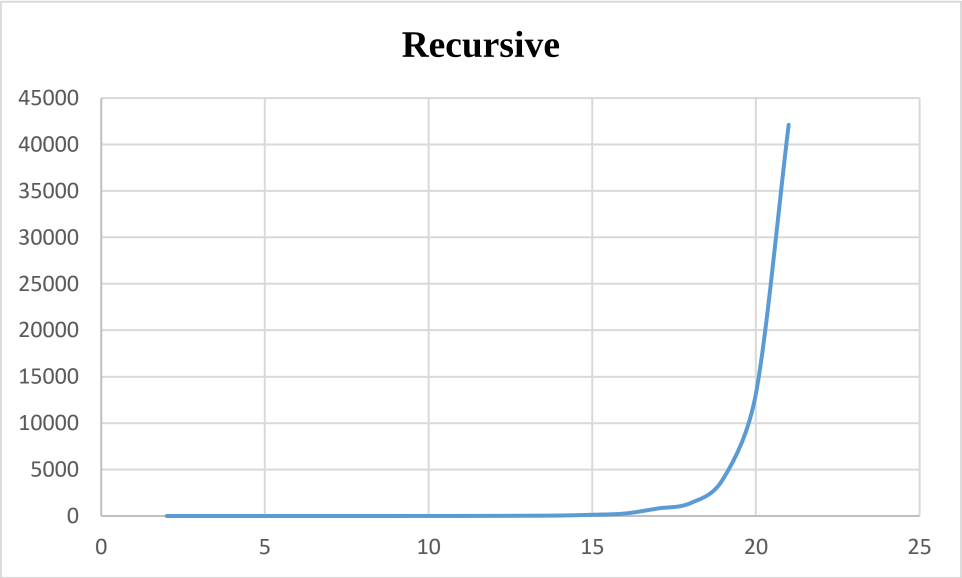
Activity 2. Experimental time measurements

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



Dynamic
800
700
600
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
6000
7000



Recursive
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
5
10
15
20
25

Activity 3. Questions

A. 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 $O(2^n)$, that we can deduce from the values of the table and graph. The waste of stack is $2 * f(n)$ because the method can execute 2 recursive calls.

The dynamic method time complexity and memory space is $O(n * m)$; 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.

B. Compute theoretical times and compare them with the experimental measurements.

Recursive:

t1 = 3.9 ms, n1 = 10 and n2 = 11; then, $t2 = \frac{2^{n2}}{2^{n1}} * t1$ = 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, $t2 = \frac{n2^2}{n1^2} * t1$ = 18.8 ms, the calculated value is greater than the obtained experimentally (12.3 ms).

<table>
<tr><td rowspan="4">Algorithmics</td><td>Student information</td><td>Date</td><td>Number of session</td></tr>
<tr><td>UO: 277921</td><td>23/3/2021</td><td>5</td></tr>
<tr><td>Surname: García López</td><td></td><td></td></tr>
<tr><td>Name: Rosa</td><td></td><td></td></tr>
</table>

C. 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.

D. 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.