


Algorithmics	Student information	Date	Number of session
	UO: 276824	7/04/2021	5
	Surname: García Fernández	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Pablo		



Activity 1. Validation Results

GCCCTAGCG GCGCAATG

In the dynamic programming I got the correct result:

```
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( 0, 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( 1, 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( 5, 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( 0, 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...
Printing longest subsequence...
GCCTG
```

That is a different subsequence, but it is also a longest one, not as the supposed one "GCGCG".

And in the recursive one I got also this:

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

GCATGCAT GAATTCAG

In the dynamic programming I got the correct result:

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

And in the recursive one I got also this:

Algorithmics	Student information	Date	Number of session
	UO: 276824	7/04/2021	5
	Surname: García Fernández		
	Name: Pablo		

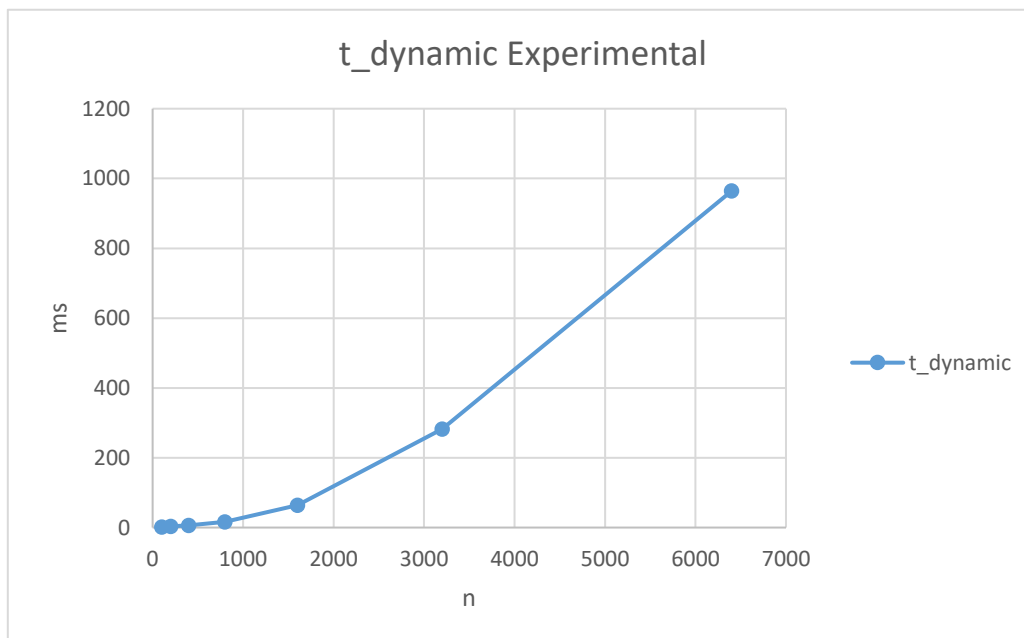
```

RECURSIVE:
Finding longest subsequence...
GATCA
Program terminated.

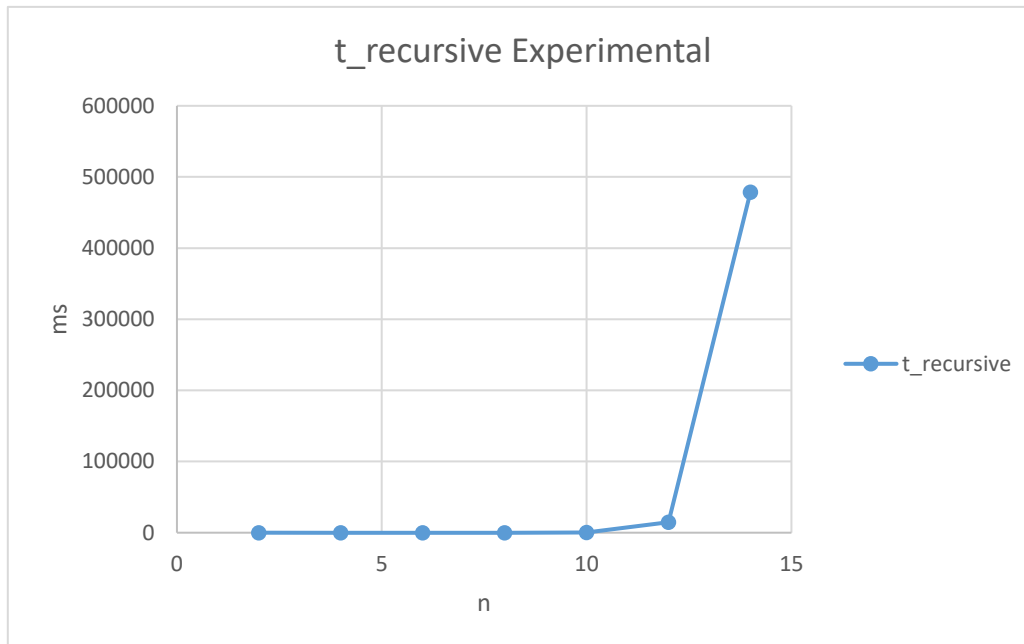
```

Activity 2. Experimental time measurements

n	t_dynamic		n	t_recursive
100	2		2	0
200	3,6		4	0,2
400	6,1		6	2,1
800	16,2		8	32,7
1600	63,8		10	496,7
3200	282		12	14887,7
6400	964,4		14	478734,2



Algorithmics	Student information	Date	Number of session
	UO: 276824	7/04/2021	5
	Surname: García Fernández		
	Name: Pablo		



Activity 3. Answer to questions in Section 3

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

The theoretical complexities are for the dynamic $O(n^2)$ because of the two for loops and the theoretical complexity for the recursive since “a=3” and “b = 1” the complexity is $O(3^n)$.

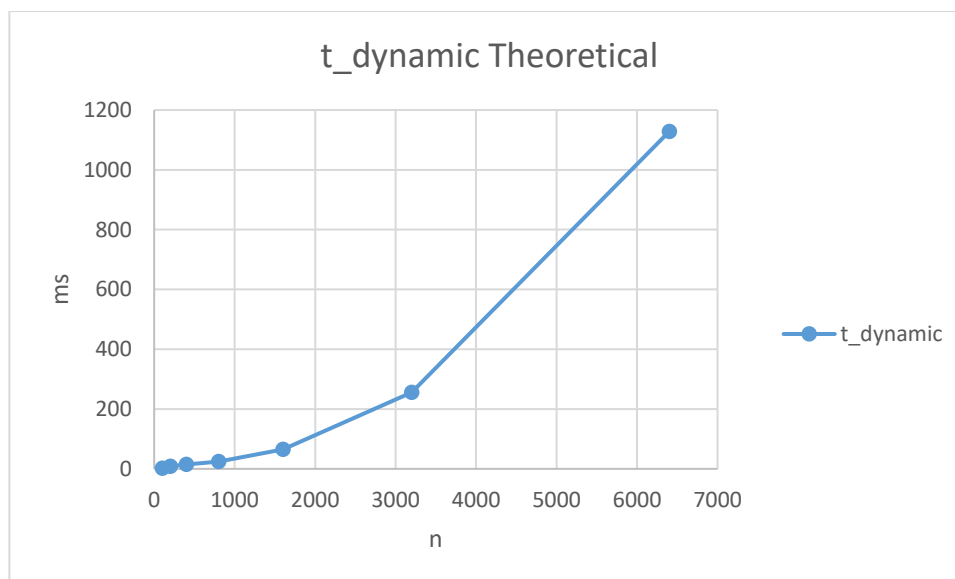
2. Compute theoretical times and compare them with the experimental measurements.

n	t_dynamic	n	t_recursive
100	2	2	0

Algorithmics	Student information	Date	Number of session
	UO: 276824	7/04/2021	5
	Surname: García Fernández		
	Name: Pablo		

200	8	4	0
400	14,4	6	5,31228E+94
800	24,4	8	1,4816E+191
1600	64,8	10	#¡NUM!
3200	255,2	12	#¡NUM!
6400	1128	14	#¡NUM!

In the theoretical results of the dynamic I got ones that are close to the real experiment but in the recursive one I do not know why I am getting that high values, I could be because the complexity that I have achieved is not the real one or so.



It has no sense to plot the theoretical recursive table since it has few values as they are too high.

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

It can not be done because the complexity is $O(3^n)$ and in those cases the computer needs an incredible amount of time to achieve some results, that's why it is not

Algorithmics	Student information	Date	Number of session
	UO: 276824	7/04/2021	5
	Surname: García Fernández		
	Name: Pablo		

good to use that type of complexity. I think that it is raised because of a stack overflow or whether a overflow in some variable while executing those quantities.

4. 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.

I have been trying days this algorithm and trying to vary it and I did not manage to find another way to obtain different subsequences is that so that I managed to find a new subsequence that is not in the example but not to recreate the ones that are the example.

EDIT: I found another way, that now I am sure that might be the correct but in the case of the dynamic, the photos above are from the old code that is not wrong but it has a different result, the new way is this one:

```
int i = size1 - 1;
int j = size2 - 1;

CellTable currentCell = table[i][j];
int length = currentCell.value;
while(length > 0 && i >= 0) {
    if(currentCell.value > table[currentCell.iPrev][currentCell.jPrev].value) {
        result = str1.charAt(i) + result;
        length--;
    }
    i = currentCell.iPrev;
    j = currentCell.jPrev;
    currentCell = table[currentCell.iPrev][currentCell.jPrev];
}
```