Luis Enrique Franco Marín
410921353

Forecast Week 7: Sequence Alignment

1. In normally implementation, f(m, n) can find one of optimal alignment. How to find out all of optimal alignments?

   Via dynamic programming. By traversing the 2d array, like the one below (note that the values have been changed by backtracking)

		R	A	M	O	N
	0	0	0	0	0	0
P	0	↖↑← 0	↖↑← 0	↖↑← 0	↖↑← 0	↖↑← 0
E	0	↖↑← 0	↖↑← 0	↖↑← 0	↖↑← 0	↖↑← 0
R	0	↖ 1	← 1	← 1	← 1	← 1
R	0	↖↑ 1	↖↑← 1	↖↑← 1	↖↑← 1	↖↑← 1
O	0	↑ 1	↑← 1	↑← 1	↖ 2	← 2

```
- - R - A M O N
P E R R - - O -
Score = 2
```

   The optimal path goes from the bottom right to the right left. By saving the path taken and traversing again gaining the same score we would get all the optimal alignments.

2. How many optimal alignments may exist? Please construct a set of input to explain your answer.

   for the extreme example of two sequences with no characters in common the optimal alignments can be as much as all the possible ordered combinations

		D	E	F
	0	0	0	0
A	0	↖↑←0	↖↑←0	↖↑←0
B	0	↖↑←0	↖↑←0	↖↑←0
C	0	↖↑←0	↖↑←0	↖↑←0

For the example given in question one another optimal path can be found in this:

		R	A	M	O	N
	0	0	0	0	0	0
P	0	↖↑←0	↖↑←0	↖↑←0	↖↑←0	↖↑←0
E	0	↖↑←0	↖↑←0	↖↑←0	↖↑←0	↖↑←0
R	0	↖1	←1	←1	←1	←1
R	0	↖↑1	↖←1	↖↑←1	↖↑←1	↖↑←1
O	0	↑1	↖←1	↖←1	↖2	←2

- - R A M - O N
P E R - - R O -
Score = 2

3. Suppose both A and B are very long, that we can't maintain all m × n scores in memory. Please find the way which only cache n values.

Taking n as the bigger value the array is minimized to a 1 d array of size n, the criterion for filling the array then goes as follows for a string c of size m and a string d of size n:

```
 for(int i=0;i<=m;i++){
   for(int j=0;i<=n;j++)
      if(i==0||j==0)current[j]=0
      else if(c[i-1]==d[j-1])current[j]=previous[j-1]+1
      else current[j]=max(previous[j-1]-current[j-1])
}
```

4. Analyze space complexity, time complexity in best case and worst case in Q1 and Q2.

   For Q1:

   Memory O(n*m)

   Time:

   Worst case: O(m*n)

   Best case: O(min(m,n))

   For Q2 the same 2d array is used, but for finding all the possible solutions backtracking is necessary and has to be done the same amount of times for both worst and best cases:

   Time $O(2^{n+m})$