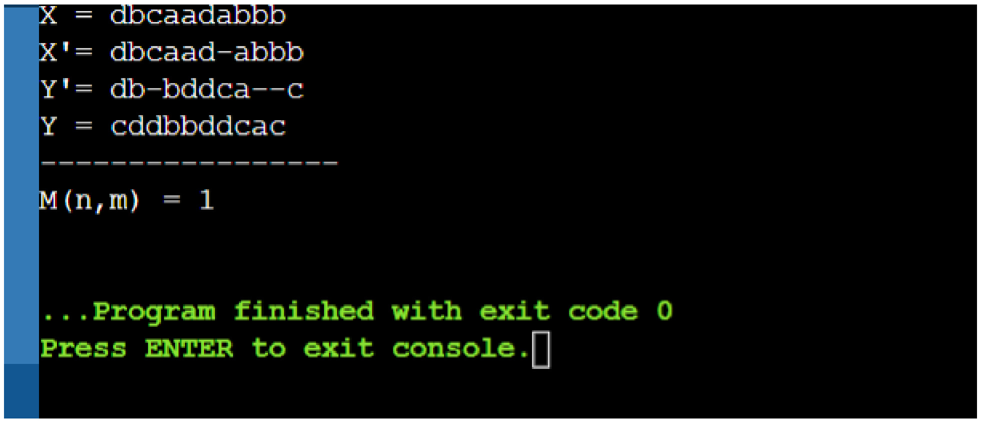
CS: 590 Algorithms

Assignment 4: Dynamic Programming, Greedy Algorithms

Poorvi Raut: 20009560

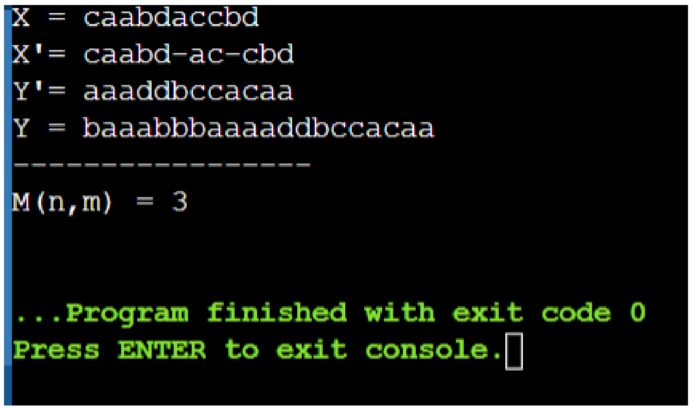
1. Implement the bottom-up version of the Smith-Waterman algorithm given by the recursive definition of the function M (as seen on the slides).

Solution: Here below is the screenshot of the output for input m=n=10



```
X = dbcaadabbb
X'= dbcaad-abbb
Y'= db-bddca--c
Y = cddbbddcac
-----------------
M(n,m) = 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Here below is screenshot of output for input m=10 and n=20



```
X = caabdaccbd
X'= caabd-ac-cbd
Y'= aaaddbccacaa
Y = baaabbbaaaaddbccacaa
-----------------
M(n,m) = 3


...Program finished with exit code 0
Press ENTER to exit console.
```

2. Implement the top-down with memoization version of the Smith-Waterman algorithm given by the recursive definition of the function M.

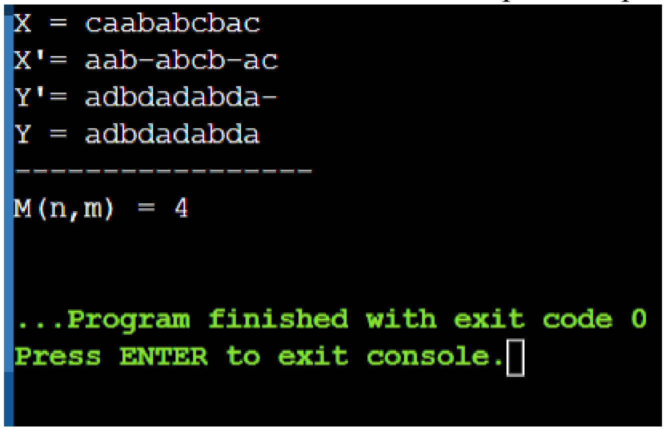
   Notes:

   - How do you initialize the necessary tables given the definition of M. Keep in mind that you have to able to determine whether or not you already computed a table value (memoization).
   - Values could be negative, but is there a limit for how small they can get?

Values can be negative, limit is till the length of the strings.

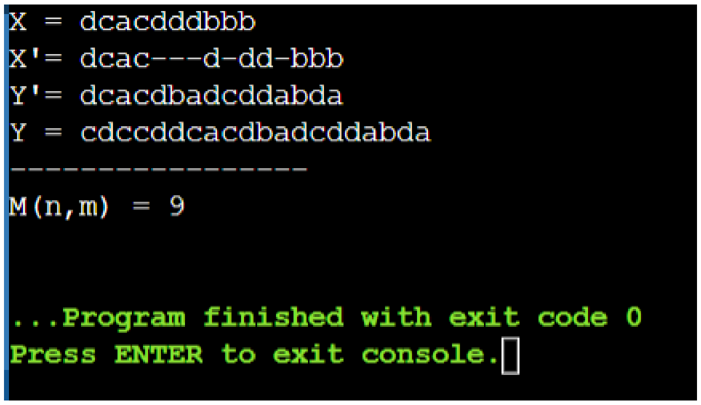
If both the strings are different, with no overlapping common elements, then the the mismatch gappenalty will take -1 each time till the length of the string is reached.

Here below is the screenshot of the output for input m=n=10

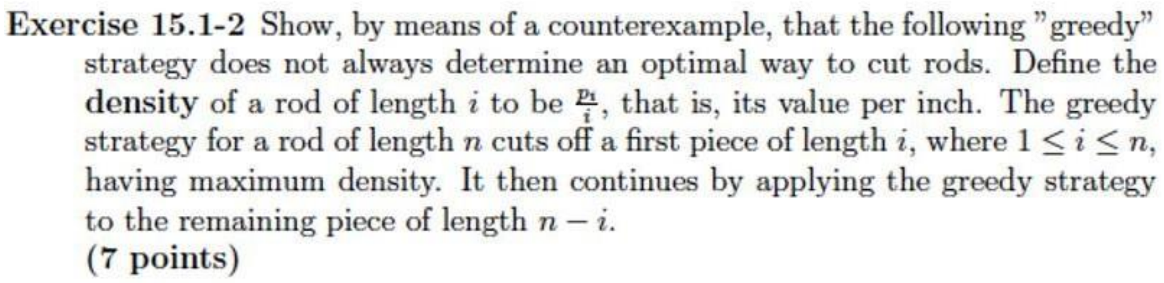


```
X = caababcbcbac
X'= aab-abcb-ac
Y'= adbdadabda-
Y = adbdadabda
-----------------
M(n,m) = 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Here below is screenshot of output for input m=10 and n=20



```
X = dcacdddbbb
X'= dcac---d-dd-bbb
Y'= dcacdbadcddabda
Y = cdccddcacdbadcddabda
-----------------
M(n,m) = 9


...Program finished with exit code 0
Press ENTER to exit console.
```


Exercise 15.1-2 Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the **density** of a rod of length i to be $\frac{p_i}{i}$, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.
(7 points)

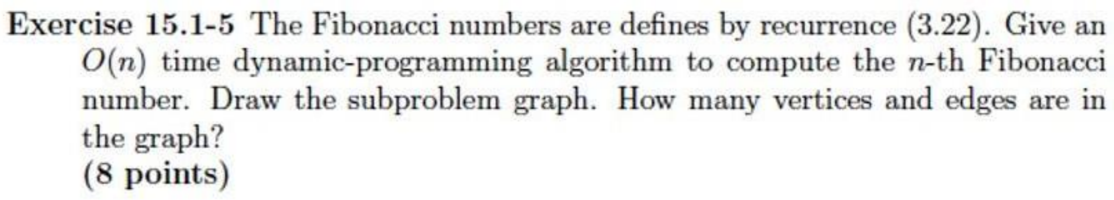
Solution: counterexample of "greedy" strategy:

Length (i)	1	2	3	4
Price(p_i) (dollars)	1	20	33	36
Density (p_i/i)	1	10	11	9

Given: A rod of length 4
As per the greedy algorithm we make the best optimal choice and then solve the sub - problem.
The maximum profit for price per inch (density) is given by:
Maximum density (pi/i) =11 for rod of length 3

Greedy choice: We cut the rod of length 3 and remaining rod length (n-i=1) remains with price 1.
Profit made by greedy approach: 33+1 = 34
The optimal way to cut the rod in length of 2.
Profit by optimal solution: 20+20 =40

Observation: The greedy approach does not provide an optimal way to cut the rods. The profit made is less as per the example provided.


Exercise 15.1-5 The Fibonacci numbers are defines by recurrence (3.22). Give an $O(n)$ time dynamic-programming algorithm to compute the n-th Fibonacci number. Draw the subproblem graph. How many vertices and edges are in the graph?
(8 points)

Solution: Fibonacci numbers are a sequence of numbers where each number is the sum of the two preceding ones, starting from 0 and 1.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

Pseudo code:

```
1. Initialize two variables, a and b, to 0 and 1 respectively.
2. Ask the user for the number of terms they want to generate.
3. Print the value of a.
4. Print the value of b.
5. For i = 2 to n-1:
6.     Set c = a + b
7.     Print the value of c
8.     Set a = b
9.     Set b = c
10.  End For
```

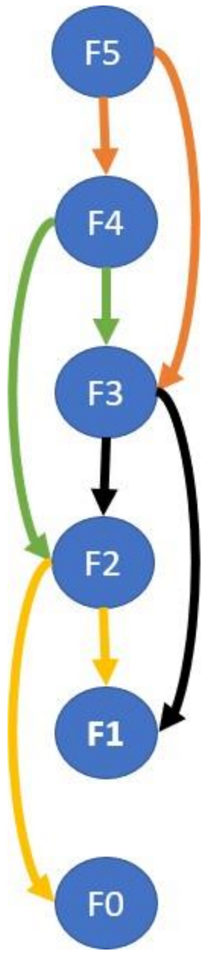
There are (n+1) vertices in the graph subproblem, i.e., F0, F1, F2….Fn.

- F0 and F1 – has 0 leaving edge.
- Reason: they have fixed value and are not dependent on other vertices.

For F2, F3....Fn each has 2 leaving edges.
Reason: they are dependent on two vertices = fibArray[i - 1] + fibArray[i - 2].

Thus, there are 2n – 2 edges in the subproblem graph.

Subproblem graph for n=5



For n=5

Number of edges in subproblem : 2n-2= 2*5-2= 8

Exercise 15.4-1 Determine an LCS of $\langle 1,0,0,1,0,1,0,1\rangle$ and $\langle 0,1,0,1,1,0,1,1,0\rangle$.
(5 points)

	-	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**
-	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1
1	0	1	1	1	2	2	2	2	2
0	0	1	2	2	2	3	3	3	3
1	0	1	2	2	3	3	4	4	4
1	0	1	2	2	3	3	4	4	5
0	0	1	2	3	3	4	4	5	5
1	0	1	2	3	4	4	5	5	6
1	0	1	2	3	4	4	5	5	6
0	0	1	2	3	4	5	5	6	6

Answer:

LCS = 010101

LCS length = 6

4. Find the maximum alignment for X = dcdcbacbbb and Y = acdccabdbb by using the Smith-Waterman algorithm (see slides). Execute the pseudocode algorithm and fill the necessary tables H and P in a bottom-up fassion. Reconstruct the strings X' and Y' using the tables H and P.

H Table:

X\ Y	**-**	**a**	**c**	**d**	**c**	**c**	**a**	**b**	**d**	**b**	**b**
-	0	0	0	0	0	0	0	0	0	0	0
d	0	-1	-1	2	1	0	-1	-1	2	1	0
c	0	-1	1	1	4	3	2	1	1	1	0

d	0	-1	0	↖ 3	3	3	2	1	3	2	1
c	0	-1	1	2	↖ 5	5	4	3	2	2	1
b	0	-1	0	1	4	↖ 4	4	6	5	4	4
a	0	2	1	0	3	3	↖ 6	5	5	4	3
c	0	1	4	3	2	5	5 ↑	5	4	4	3
b	0	0	3	3	2	4	4	↖ 7	← 6	6	6
b	0	-1	2	2	2	3	3	6	6	↖ 8	8
b	0	-1	1	1	1	2	2	5	5	8	↖ 10

P Table:

X\ Y	**-**	**a**	**c**	**d**	**c**	**c**	**a**	**b .**	**d**	**b**	**b**
-	0	0	0	0	0	0	0	0	0	0	0
d	0	↖	↖	↖	←	←	↖	↖	↖	←	←
c	0	↖	↖	↑	↖	↖	←	←	↑	↖	↖
d	0		↑	↖	↑	↖	↖	↖		←	←
c	0		↖			↖	←	←		↖	↖
b	0	↖	↑				↖	↖	←	↖	↖
a	0		←	↑					↖	↖	↖
c	0		↖	←				↖	↖	↖	↖
b	0			↖	↖			↖	←	↖	↖
b	0			↖	↖			↖	↖	↖	↖
b	0	↖		↖	↖		↑	↖	↖	↖	↖

X = dcdcbacbbb

X'= dcdcbacb-bb

Y'= acdcca-bdbb Y= acdccabdbb

M(n,m) = 10 = Maximum alignment