Stevens Institute of Technology

Department of Computer Science

CS590: Algorithms

Prof. Iraklis Tsekourakis

Homework Assignment 4

Submitted by: Malav Shastri, CWID:10456244

4. Find the maximum alignment for X = dcdcbacbbb Y = acdccabdbb by using the smith waterman algorithm.

Solution:

H[n][m]

	-	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[n][m]

	-	a	c	d	c	c	a	b	d	b	b
-	0	0	0	0	0	0	0	0	0	0	0
d	0	d	d	d	l	l	d	d	d	l	l
c	0	d	d	u	d	d	l	l	u	d	d
d	0	d	u	d	u	d	d	d	d	l	d
c	0	d	d	u	d	d	l	l	u	d	d
b	0	d	u	u	u	d	d	d	l	d	d
a	0	d	l	u	u	d	d	u	d	d	d
c	0	u	d	l	d	d	u	d	d	d	d
b	0	u	u	d	d	u	d	d	l	d	d

	-	a	c	d	c	c	a	b	d	b	b
b	0	d	u	d	d	u	d	d	d	d	d
b	0	d	u	d	d	u	d	d	d	d	d

X = dcdcbacbbb
X' = dcdcbacb-bb
Y' = acdcca-bdbb
Y = acdccabdbb
M(n,m) = 10

Exercise 15.1-5 Show by means of counter example, that the following greedy strategy does not always determine an optimal way to cut rods. Define a density of a rod of length i to be 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<=i<=n, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length n-i.

Solution:
Let length of the rod be 10; n = 10.
Below is the example table representing prices according to length and defined density.

Length (i)	1	2	3	4	5	6
Price (P)	5	12	20	28	37	45
Density (P/i)	5	6	6.66	7	7.4	7.5

According to greedy strategy we cut first piece to be length of 6 as it has maximum density and it yields price 45 after that the maximum density to choose would be 7.4 but as the total length of road is 10 we can't select that as that would make the rod length 11. So we would select the second best which would be a length 4 which has a density of 7. So now after second selection total price would be 45+28 = 73.

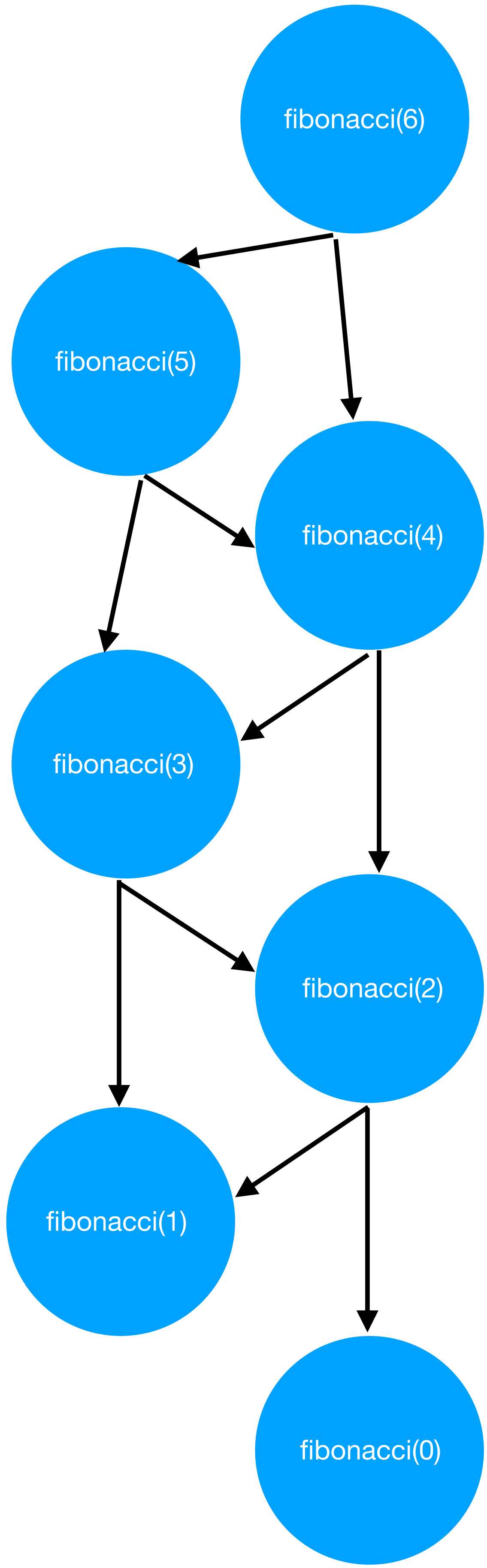
Now we can see that this is not at all an optimal solution we can get the the highest price of 37+37=74 by selecting an option of length 5 twice. Which will also preserve the length 10.

Exercise 15.1-5 The Fibonacci numbers are defined by recurrence. Give an O(n) time dynamic programming algorithm to compute an n^{th} Fibonacci number. Draw the sub-problem graph. How many vertices and edges are in the graph.

Solution:

```
int fibonacci (int n){
	int fib[n]		//array of size n
	if( n = 0){
		return 0
	}
	if(n = 1){
		return 1
	}
	fib[0] = 0
	fib[1] = 1
	for(i = 2; i<n; i++){
		fib[i] = fib[i-1]+fib[i-2]
	}
	return fib[n-1]
```

Below is the subproblem graph.



fibonacci(6)
fibonacci(5)
fibonacci(4)
fibonacci(3)
fibonacci(2)
fibonacci(1)
fibonacci(0)

Total Vertices (Total Number of Sub problems): **n+1** (where n is nth Fibonacci number which we need to compute.)
Total Edges: **2(n-1)** (Where n is nth Fibonacci number which we need to compute.)
Running time: O(n)

Exercise 15.4-1 Determine an LCS of <1,0,0,1,0,1,0,1> and <0,1,0,1,1,0,1,1,0>.
Solution :

LCS: <1,0,0,1,1,0>

		0	1	0	1	1	0	1	1	0
	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1
0	0	1	1	2	2	2	2	2	2	2
0	0	1	1	2	2	2	3	3	3	3
1	0	1	2	2	3	3	3	4	4	4
0	0	1	2	3	3	3	4	4	4	5
1	0	1	2	3	4	4	4	5	5	5
0	0	1	2	3	4	4	5	5	5	6
1	0	1	2	3	4	5	5	6	6	6

		0	1	0	1	1	0	1	1	0
	0	0	0	0	0	0	0	0	0	0
1	0	u	d	l	d	d	l	d	d	l
0	0	d	u	d	l	l	d	l	l	d
0	0	d	u	d	u	u	d	l	l	d
1	0	u	d	u	d	d	u	d	d	l

		0	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**
0	0	d	u	d	u	u	d	u	u	d
1	0	u	d	u	d	d	u	d	d	u
0	0	d	u	d	u	u	d	u	u	d
1	0	u	d	u	d	d	u	d	d	u