Shubham Sanjay Jain

CWID: 10456815

Stevens Institute of Technology

Master of Science in Computer Science

Submitted to: Prof. Iraklis Tsekourakis

CS 590 Algorithms

Solution: Assignment 4

4. **P Table:**

Table used for printing the string.

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	d	d	d	l	l	d	d	d	l	l
2	0	d	d	u	d	d	l	l	u	d	d
3	0	d	u	d	u	d	d	d	d	l	l
4	0	d	d	u	d	d	l	l	u	d	d
5	0	d	u	u	u	d	d	d	l	d	d
6	0	d	l	u	u	d	d	u	d	d	d
7	0	u	d	l	d	d	u	d	d	d	d
8	0	u	u	d	d	u	d	d	l	d	d
9	0	d	u	d	d	u	d	d	d	d	d
10	0	d	u	d	d	u	d	d	d	d	d

H/M Table:

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	-1	-1	2	1	0	-1	-1	2	1	0
2	0	-1	1	1	4	3	2	1	1	1	0
3	0	-1	0	3	3	3	2	1	3	2	1
4	0	-1	1	2	5	5	4	3	2	2	1
5	0	-1	0	1	4	4	4	6	5	4	4
6	0	2	1	0	3	3	6	5	5	4	3
7	0	1	4	3	2	5	5	5	4	4	3
8	0	0	3	3	2	4	4	7	6	6	6
9	0	-1	2	2	2	3	3	6	6	8	8
10	0	-1	1	1	1	2	2	5	5	8	10

X = dcdcbacbbb
X'= dcdcbacb-bb
Y'= acdcca-bdbb
Y = acdccabdbb

15.1-2

Solution:

i	**1**	**2**	**3**
length	**1**	**2**	**3**
Price	**6**	**10**	**12**
Price/Length	**6**	**5**	**4**

So, when working with greedy approach, we try to choose the locally best solution.
The example is we need the rod of length 5.
So, In greedy approach, we will choose the one with most Price/weight Item.
Hence, we end up choosing Item 1 which can be maximum of length 1 and item 2 of length 2 and total we can get the rod is of length 3.
And the Price we can get is **6+10 = 16.**
This can be locally good optimal but not the best optimal solution.

Hence, we use dynamic programming which end up with best cuts.

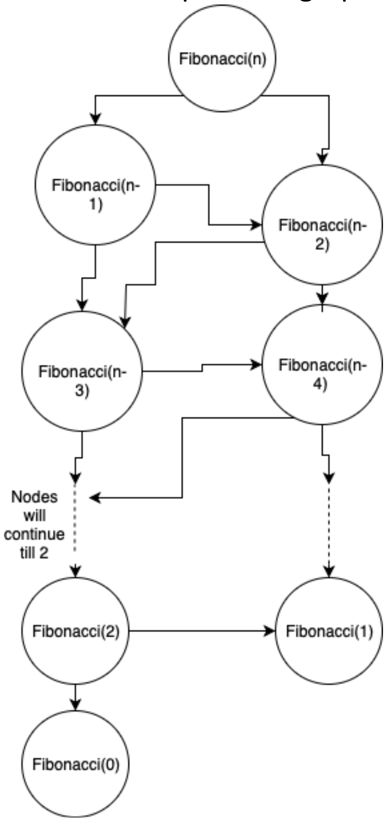
As in the greedy, the length of 2 was wasted, but in Dynamic Programming, we are not be left wasted with any length in this case.
Here,by dynamic programming, we will choose length of 2 and 3 which will total be 4 and price gain will be **10+12 = 22.**
Therefore,
When we need to cut rod of length n, we cut the rod with height P_i/W_i ratio which can be best but not because we can't use fractional value of length. Here, we choose for current and then look best local optimal for later sub problem. But, dynamic programming is used where we solve subproblem first and then choose for present.

15.1-5

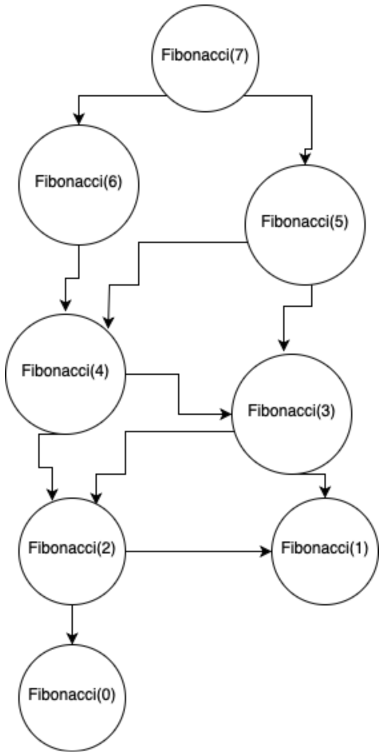
Algorithm for Fibonacci using dynamic programming:

Algorithm (FIBONACCI (x))
(1) *let F* [0…..*x*+1] *be new array*
(2) *F*[0] = 0 and *F*[1] =1
(3) **for** (2 ≤ i ≤ x) **do**
(4) *F*[i] = *F*[i-1] +*F*[i-2]
(5) return F[x]

Fibonacci Subproblem graph for n



Fibonacci Subproblem graph for n = 7.



There are n+1 vertices and 2*(n-1) edges.

15.4-1

LCS of { 1,0,0,1,0,1,0,1} and {0,1,0,1,1,0,1,1,0} is **6**.

		1	**0**	**0**	**1**	**0**	**1**	**0**	**1**
	0	**0**	0	0	0	0	0	0	0
0	0	0	**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: 0 1 0 1 0 1