**Exercise 14.1**

**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 pi/i, that is, its value per inch. The greedy strategy for a rod of length cuts off a first piece of length, where 1<i=<=n, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length n-i.**

**Ans)**

Counter example:

Assume a rod of size 4.

P = [1,6,10,1]

Let D denote the densities of each piece of size 0-4

D =[1, 3, 3.3, 4]

Here, the maximum density is the piece 3.

Greedily, we select a piece of size 3 to cut off first.

Revenue = 10

Length of rod left = 1

Only option is left is P[1] = 1

Therefore,

total revenue = 10 + 1 = 11

This is not optimal. The optimal way of breaking this rod would be into 2 pieces of size 2. This would give us 2\* 6 = 12 revenue totally.

**3. Consider a modification of the rod-cutting problem in which, in addition to a price pi for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem.**

**Ans)**

MEMOIZED-CUT-ROD(p,c,n):

let r[0:n] be an array with each value of r denoting revenue of rod of size r

for i = 0 to n:

r[i] = -inf

MEMOIZED-CUT-ROD-AUX(p,c,n,r)

MEMOIZED-CUT-ROD-AUX(p,c,n,r):

if r[n] >= 0:

return r[n]

if n == 0:

return 0

q = -inf

for i = 0 to n:

q = max{p[i] + MEMOIZED-CUT-ROD-AUX(p,c,n-i,r) -c }

r[n] = q

return r[n]

The changes we need to make are to ensure we account for this cost at each step of our recursive call to the MEMOIZED-CUT-ROD-AUX function.

**5.** **Give an O(n) time dynamic programming algorithm in pseudocode to compute the nth Fibonacci number. Draw the subproblem graph. How many vertices and edges does the graph contain?**

Ans)

FIBONACCI(n):

if n ==0 or n==1:

return n

Let f[0:n] be an array where f[i] stores the fibonacci of i

for i = 0 to n:

f[i] = -1

f[0] = 0

f[1] = 1

return FIBONACCI-AUX(n,f)

FIBONACCI-AUX(n,f):

if f[n] < 0:

f[n] = + FIBONACCI-AUX(n-2,f) + FIBONACCI-AUX(n-1,f)

return f[n]

Graph contains n+1 vertices

If n>1:

Number of edges = 2n-2

if n<=1:

Number of edges = 0

Subproblem graph:

\*

\*