Data Structure and Algorithm Analysis(H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Work Sheet 11

Mengxuan Wu

Question 11.1

```
BUTTOM-UP-CUT-ROD'(p, n)
let r[0..n] and s[0..n] be new arrays
r[0] = 0
for j = 1 to n do
    q = p[j]
    s[j] = j
    for i = 1 to j − 1 do
        if q < p[i] + r[j − i] − c then
            q = p[i] + r[j − i] − c
            s[j] = i
    r[j] = q
return r and s
```

Question 11.2

```
MEMOIZED-CUT-ROD'(p, n)
let r[0..n] and s[0..n] be new arrays
for i = 0 to n do
    r[i] = −∞
return MEMOIZED-CUT-ROD-AUX'(p, n, r, s)
```

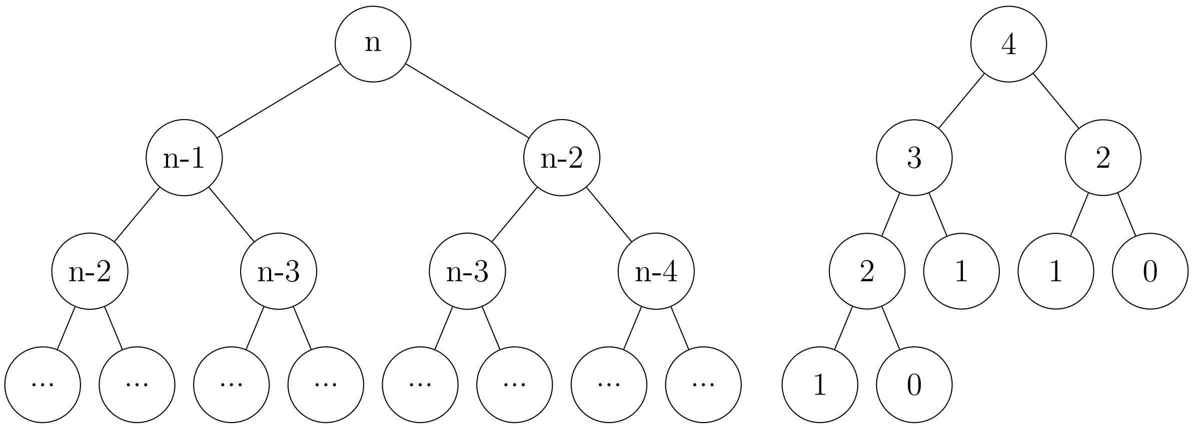
```
MEMOIZED-CUT-ROD-AUX'(p, n, r, s)
if r[n] ≥ 0 then
    return (r[n], s)
if n == 0 then
    q = 0
else
    q = −∞
    for i = 1 to n do
        t, − = MEMOIZED-CUT-ROD-AUX'(p, n − i, r, s)
        if q < p[i] + t then
            q = p[i] + t
            s[n] = i
r[n] = q
return (r[n], s)
```

Question 11.3

FIBONACCI-NUMBER(n)

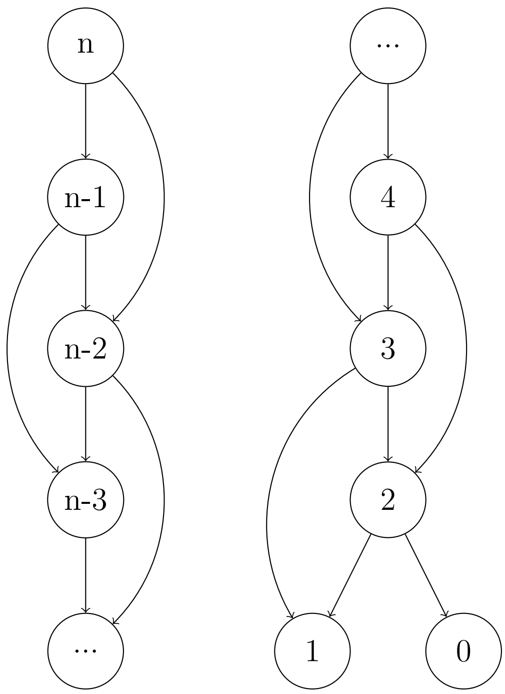
```
let f[0..n] be a new array
f[0] = 0
f[1] = 1
for i = 2 to n do
    f[i] = f[i − 1] + f[i − 2]
return f[n]
```

A non-optimal solution:



As it is shown in the graph, if we denote the number of nodes in the tree as $N(n)$, then we have $N(n) = N(n-1) + N(n-2) + 1$. And we know that $N(0) = 1$ and $N(1) = 1$. By characteristic equation, we can prove that $N(n) = \frac{2}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^{n+1} - \frac{2}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^{n+1} - 1$. The number of edges is $E(n) = N(n) - 1 = \frac{2}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^{n+1} - \frac{2}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^{n+1} - 2$.

An optimal solution:



For an optimal solution, we have $N(n) = n + 1$ with an exception that $N(1) = 1$ The number of edges is $E(n) = 2n - 2$ with an exception that $E(0) = 0$.

Question 11.4

The Bellman equation for share price is $A_k = \max\{B_i | 1 \leq i \leq k\}$ and $B_k = \max\{a_k, B_{k-1} + a_k\}$.

(a)

```
MAX-SHARE-PRICE(a, n)
A = 0
B = 0
for i = 1 to n do
        B = max(a_i, B + a_i)
        A = max(A, B)
return A
```

(b)

The time complexity is $O(n)$. This the loop is executed n times and each time it takes constant time, and the other operations take constant time. Hence, $f(n) = c_1 n + c_2 = O(n)$.