Home work # 6

6.8

Initialization

Let Longest Common Substring be denoted LcS

$LcS[0, j] = 0$ for $j = 1$ to m

$LcS[i, 0] = 0$ for $i = 0$ to n

Recurrence

$$LcS[x_1 \cdots x_i, y_1 \cdots y_j] = LcS[x_1 \cdots x_{i-1}, y_1 \cdots y_{j-1}] + 1 \quad \text{if } x_i = y_j$$

$$= 0 \quad \text{otherwise}$$

$$\max LcS = \max_{i=\phi}^{n} \max_{j=1}^{m} LcS[i,j]$$

Code

```
for i = 0 to m
    LcS[i,0] = 0
    for j = 0 to m
        LcS[0,j] = 0

for i = 1 to n
    for j = 1 to m
        if xi == yj
            LcS[i,j] = LcS[i-1,j-1]+1
        else
            0

max = 0
for i = 1 to n
    for j = 1 to n
        if max < LcS[i,j]
            max = LcS[i,j]
```

6.10.

Let $P[i, j]$ in i tosses you get j heads.

Initialization

$P[0, 0] = 1$

for $i = 1$ to n

$P[i, 0] = 0$

~~for i = 1 to n~~ $P[i, j] = 0$ if $j > i$

Recurrence

$$P[i, j] = P[i-1, j]\,(1 - P_i) + P[i-1, j-1]\,P_i$$

Code:

```
P[0,0] = 1
for i = 1 to n
    P[i,0] = 0 ; P[i, i+1] = 0
for i = 1 to n
    for j = 1 to k
        P[i,j] = P[i-1, j-1] P_i + P[i-1, j] (1-P_i)
return P[n,k]
```

6.13.

Let $P[i, j]$ be the probability that A wins i games and B wins j games

Initialization

$P[0,0] = 1$

$P[i, 0] = \frac{1}{2^i}$ for $i = 1$ to n

$P[0, i] = \frac{1}{2^i}$ for $i = 1$ to n

recurrence

$P[i,j] = \frac{1}{2} P[i-1, j] + \frac{1}{2} P[i, j-1]$

code

```
P[0,0] = 1
for i = 1 to n
    P[i,0] = 1/2 P[i-1,0]
for i = 1 to n
    P[0,i] = 1/2 P[0,i-1]

for i = 1 to n
    for j = 1 to n-1
        P[i,j] = 1/2 P[i-1,j] + 1/2 P[i,j-1]

return P[n, j] for all j
```

6.17.

Let $X[i]$ be true if i could be changed using denominations

Initialization

$X[0] = \text{true}$, $X[i] = \text{False}$ for $i = 1$ to v

Recurrence

$X[i] = X[i]$ or $X[i - x_j]$ for $j = 1$ to n
and $x_j \leq i$

Code

$X[0] = \text{true}$
for $i = 1$ to v
$X[i] = \text{false}$

for $i = 1$ to v
for $j = 1$ to n
~~$X[i]$~~ = if $x_j \leq i$
$X[i] = X[i - x_j] \vee X[i]$

return $X[v]$

6.18

$X[i, j]$: whether i could be changed using the first j coins.

Initialization

$x[0,0] =$ true

for $i = 1$ to n

$x[0, i] =$ true

for $j = 1$ to v

$x[j, 0] =$ false

Recurrence

$x[i,j] = x[i,j-1]$ OR $x[i-x_j, j-1]$ if $x_j \leq i$

code

```
X[0,0] = true
for i = 1 to n
    x[0,i] = true
for j = 1 to v
    x[j,0] = false

 for i = 1 to v
    for j = 1 to n
      if x_j ≤ i
          x[i,j] = x[i,j-1] ∨ x[i-x_j, j-1]

return X[v, n]
```

6.19

$X[i, j]$: i can be changed using j coins

Initialization

$x[0, 0]$ = true

$x[i, 0]$ = False $i = 1$ to v

$x[0, j]$ = true for $j = 1$ to k

ϕ

Recurrence

$x[i, j] = x[i - x_j, j-1]$ if $x_j \leq i$

for $i = 1$ to v

for $j = 1$ to k

for $\ell = 1$ to n

~~$x[i,j]$~~ if $x_\ell \leq i$

$X[i, j] = X[i - x_\ell, j-1]$

return $X[v, k]$

6.20.

Let cost(i,j) is the cost of the binary tree with nodes from i to j

Initialization

$$\text{cost}(i,i) = P_i \quad , \quad \text{cost}(i, i+1) = \min(P_i + 2P_{i+1},\ 2P_i + P_{i+1})$$

recurrence

$$\text{cost}(i,j) = \min_{i<k\leq j} \text{cost}(i,k) + \text{cost}(k+1,j) + \sum_{i=1}^{j} P_i$$

code:

```
for i = 1 to n
    cost(i,i) = P_i
for i = 1 to n-1
    cost(i,i+1) = min(P_i + 2P_{i+1}, 2P_i + P_{i+1})
for i = 1 to n-2
    for j = i+2 to n
        ~~cost(i,j)~~ = cost(i,j) = min_{i<k<j} (cost(i,k-1) + cost(k+1,j) + Σ_{l=i}^{j} P_l)

return cost(1,n)
```

6.26

Let $X[i, j]$ = total number of dashes.

Initialization

$X[0, j] = j$ for $j = 0$ to m

$X[i, 0] = i$ for $i = 0$ to n

recurrence

$$X[i, j] = \min \left[X[i-1, j-1] \text{ if } x_i = y_j, \; X[i-1, j] + 1, \; X[i, j-1] + 1 \right]$$

Code

```
for j = 0 to m
   X[0, j] = j
for i = 0 to n
   X[i, 0] = i
for i = 1 to m
   for j = 1 to m
      if x_i = y_j
            then X[i-1, j-1]
         else min ( X[i-1, j] + 1, X[i, j-1] + 1 )
return X[m, m]
```

1.~~5~~.13

only change in the optimal substructure (Recurrence equation)

$$cutRod(n) = \max(Price[i] + cutRod(n-i) - c)$$

for $i = 1$ to n

Initialization

$cutRod(0) = 0$

Recurrence

$$cutRod(i) = \max_{1 \leq j \leq i} (Price(j) + cutRod(i-j) - c)$$

Return cutRod(n)

Code:

```
cutRod[0] = 0 ;  cutRod[i] = Price[i] - C
for i = 1 to n
    for j = 1 to i
        if (Price[j] + cutRod(i-j) - C) > cutRod(i)
                cutRod[i] = Price[j] + cutRod[i-j] - C

return cutRod[n]
```

Vitrebi algorithm is like Shortest Path Algorithm.