6.8

Initialization

Let Longest Common Substring be denoted LCS

\[ LCS[0, j] = 0 \text{ for } j = 1 \text{ to } m \]
\[ LCS[i, 0] = 0 \text{ for } i = 1 \text{ to } n \]

Recurrence

\[ LCS[X, \ldots X_i, Y, \ldots Y_j] = LCS[X, \ldots X_{i-1}, Y, \ldots Y_{j-1}] + 1 \text{ if } X_i = Y_j \]
\[ = 0 \text{ otherwise} \]

\[ \max LCS = \max_{i=1}^{n} \max_{j=1}^{m} LCS[i, j] \]

Code

\[ \text{for } i = 0 \text{ to } n \]
\[ \text{LCS}[i, 0] = 0 \]
\[ \text{for } j = 0 \text{ to } m \]
\[ \text{LCS}[0, j] = 0 \]
\[ \text{for } i = 1 \text{ to } n \]
\[ \text{for } j = 1 \text{ to } m \]
\[ \text{if } X_i = Y_j \]
\[ \text{LCS}[i, j] = \text{LCS}[i-1, j-1] + 1 \]
\[ \text{else} \]
\[ \text{LCS}[i, j] = 0 \]
\[ \text{max} = 0 \]
\[ \text{for } i = 1 \text{ to } n \]
\[ \text{for } j = 1 \text{ to } m \]
\[ \text{if } \max < \text{LCS}[i, j] \]
\[ \text{max} = \text{LCS}[i, j] \]
Let $P(i,j)$ be the probability of getting exactly $j$ heads in $i$ tosses.

**Initialization**

$P(0,0) = 1$

for $i = 1$ to $n$

$P(i,0) = 0$

for $j = 1$ to $i$

$P(i,j) = 0$ if $j > i$

**Recurrence**

$P(i,j) = P(i-1,j) (1-P_c) + P(i-1,j-1) P_c$

**Code:**

```plaintext
P[0,0] = 1
for i = 1 to n
    P[i,0] = 0 ; P[i,i] = 0
for i = 1 to n
    for j = 1 to i
        P[i,j] = P[i-1,j-1] P_c + P[i-1,j] (1-P_c)
    return P[n,k]
```
Let \( P[i, j] \) be the probability that A wins \( i \) games and B wins \( j \) games.

**Initialization**

\[
P[0, 0] = 1
\]

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

\[
P[0, j] = \frac{1}{2^j} \quad \text{for } j = 1 \text{ to } n
\]

**Recurrence**

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

**Code**

```plaintext
P[0, 0] = 1

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

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

for i = 1 to n
    for j = 1 to n-1
        P[i, j] = \frac{1}{2} P[i-1, j] + \frac{1}{2} P[i, j-1]
    end
end

return P[n, j] for all j
```
Let $x[i]$ be true if $i$ could be charged using diminishing

**Initialization**

$x[0] = true$, $x[i] = false$ for $i = 1$ to $n$

**Recurrence**

$x[c] = x[i]$ or $x[c - x[j]]$ for $j = 1$ to $n$

and $x[j] \notin \mathcal{E}$

code

```plaintext
x[0] = true
for i = 1 to n
    x[i] = false
for i = 1 to n
    for j = 1 to n
        if x[j] \leq i
            x[c] = x[c - x[j]] v x[c]
    x[c] = x[c]
return x[n]
```
\[ x[i,j] \text{ is whether } i \text{ could be changed using the first } j \text{ coins.} \]

**Initialization**

\[ x[0,0] = \text{true} \]
\[ \text{for } i = 1 \text{ to } n \]
\[ x[0, i] = \text{true} \]
\[ \text{for } j = 1 \text{ to } v \]
\[ x[i, 0] = \text{false} \]

**Recurrence**

\[ x[i,j] = x[i,j-1] \text{ or } x[i-x_j,j-1] \quad \text{if } x_j \leq i \]

**Code**

\[ x[0,0] = \text{true} \]
\[ \text{for } i = 1 \text{ to } n \]
\[ x[0, i] = \text{true} \]
\[ \text{for } j = 1 \text{ to } v \]
\[ x[i, 0] = \text{false} \]
\[ \text{for } i = 1 \text{ to } v \]
\[ \text{for } j = 1 \text{ to } n \]
\[ \text{if } x_j \leq i \]
\[ x[i,j] = x[i,j-1] \text{ or } x[i-x_j,j-1] \]

\[ \text{return } x[v,n] \]
function C[i, j] = i can be changed using j wins

Initialization

\[ x[i, 0] = \text{true} \]
\[ x[i, i] = \text{false} \quad i \geq 1 \]
\[ x[0, j] = \text{true} \quad \text{for } j = 1 \text{ to } k \]

Recurrence

\[ x[i, j] = x[i - x_j, j - 1] \quad \text{if } x_j \leq i \]

for \( i = 1 \) to \( n \)
    for \( j = 1 \) to \( k \)
        for \( e = 1 \) to \( n \)
            if \( x_e \leq i \)
                \[ x[i, j] = x[i - x_e, j - 1] \]

return \( x[0, k] \)
Let $\text{cost}(i,j)$ be the cost of the binary tree with nodes from $i$ to $j$.

Initialization:

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

Recurrence:

$$\text{cost}(i,j) = \min_{i < k < j} \left( \text{cost}(i,k) + \text{cost}(k+1,j) + \sum_{z=k+1}^{j-1} P_z \right).$$

Code:

```plaintext
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) = \min_{i < k < j} \left( cost(i,k) + cost(k+1,j) + \sum_{z=k+1}^{j-1} P_z \right)

return cost(1,n)
```
6.26

Let $X[0,j]$ = number of dashes.

Initialization

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

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

Recurrence

$X[i,j] = \min \left[ X[i-1,j-1] + 1, X[i-1,j] + 1, X[i,j-1] + 1 \right]$

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 $X[m,m]$
only change is the optimal substructure (recurrence equality)

\[
\text{cutRod}(n) = \max \left( \text{price}[i] + \text{cutRod}(n-i-j) - j \cdot c \right)
\]

for \( i = 1 \) to \( n \)

Initialization

\[
\text{cutRod}(0) = 0
\]

Recurrence

\[
\text{cutRod}(i) = \max_{1 \leq j \leq i} \left( \text{price}(j) + \text{cutRod}(i-j) - c \right)
\]

Return \( \text{cutRod}(n) \)

Code:

\[
\text{cutRod}[0] = 0 \quad \text{cutRod}[i] = \text{price}[i] - c
\]

for \( i = 1 \) to \( n \)

for \( j = 1 \) to \( i \)

\[
\text{cutRod}[i] \text{ if } (\text{price}[j] + \text{cutRod}[i-j] - c) > \text{cutRod}(i)
\]

\[
\text{cutRod}[i] = \text{price}[j] + \text{cutRod}[i-j] - c
\]

return \( \text{cutRod}[n] \)

Viterbi algorithm is like Shortest Path Algorithm.