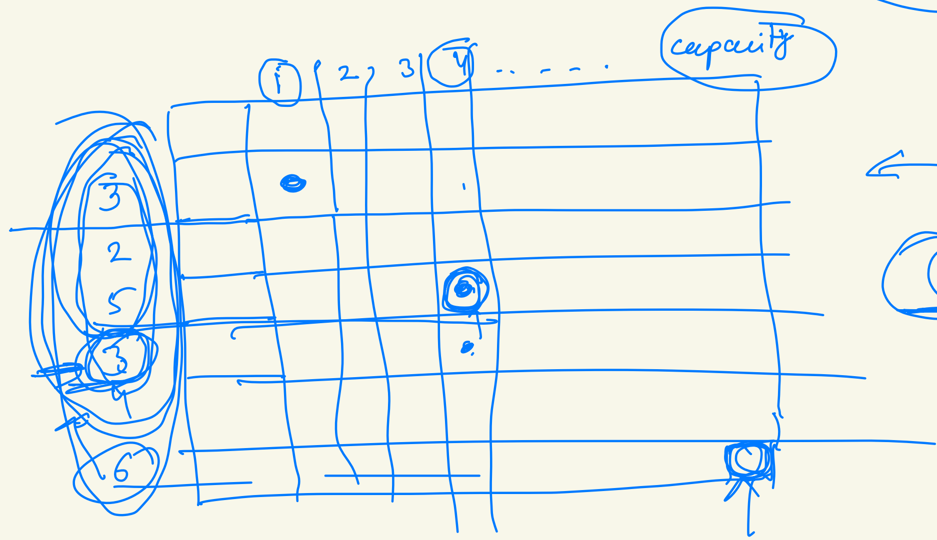
0/1 Knapsack

weights	3	2	5	3	4	6
values	5	7	10	6	10	8

capacity of Bag

capacity



3 + 1

15

15 - 6

9

dp[i-1][9]

DP state

dp[i][j] represents max sum of values we get by using items from 0 to i and a bag with capacity j.

$$dp[i][j] = \begin{cases} dp[i-1][j] & \text{if } weight[i] > j \\ \max\left(dp[i-1][j],\ value[i] + dp[i-1][j - weight[i]]\right) & \end{cases}$$

i = 0 to n

j = 0 to capacity.

$w = [4, 5, 1]$

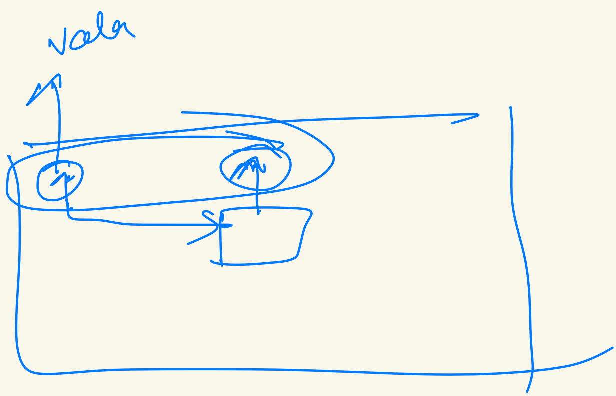
$v = [1, 2, 3]$

$C = 4$

	0	1	2	3	4
	0	0	0	0	0
4	0	0	0	0	1
5	0	0	0	0	1
1	0	3	3	3	3
	0				

$1 + dp[0][0]$

$3 + dp[1][1]$



LCS

longest common subsequence

S_1 a x b f c q d e r f

S_2 a b c d l k e v f

abcdef

s_1 and s_2

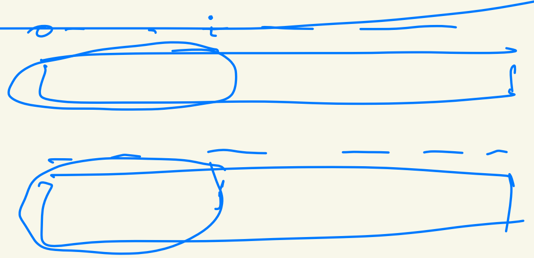
a, b, c, . . . -

a, b, c, . - - .

Dp state

dp[i][j] represents LCS of (0, i] substring in s_1 and (0, j] substring in s_2

s_1.substring(0, i) and s_2.substring(0, j)



$dp[i][0] = 0$

$dp[0][j] = 0$

$dp[i][j]$

$dp[i][0] = 0$

" "

$dp[i][j] =$

$dp[i-1][j-1]$ if $s_1[i-1] == s_2[j-1]$

	0	1					
0	0	0	0	0	0	0	0
1	0						
	0						
	0						
	0						

$i = 1$ to $s_1.\text{length}$

$j = 1$ to $s_2.\text{length}$

$$dp[i][j] = \begin{cases} 1 + dp[i-1][j-1] & \text{if } \underline{s_1[i-1] == s_2[j-1]} \\ \max(dp[i-1][j],\ dp[i][j-1]) & \end{cases}$$

```
lcs ( s1 , s2 , i , j )
{
    if( s1[i] == s2[j] )
        return 1+ lcs ( s1, s2, i+1, j+1 )

    max ( lcs ( s1, s2 i+1, j ) , lcs ( s1, s2, i, j+1))
}
```

LPS   longest palindrome subsequence

aebf cgd chbia

abcdcba

LCS   S + reverse S

$$LPS(S) = LCS(S, \text{Rev}S)$$

DP state

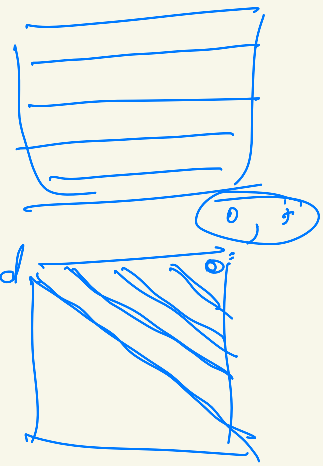
dp[i][j] rep lps length of substring (i, j)

$$S = \text{"}abcde\text{———"}$$

i j

i i+1 j-1 j

a b c b a



a b c d

a	ab	abc	abcd
b	bc	bcd	
c	cd		
d			

i j

i j = 0

j - i = 1

j - i = 2

$dp[i][i] = 1$

Run loop diagonally

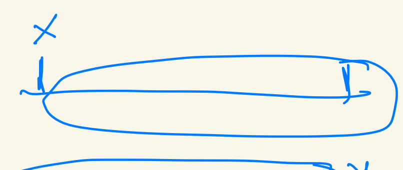
$dp[i][j]$ dependend on $dp[i+1][j-1]$

len = 1 to s.length

i = 0 to s.length - len

j = i + len - 1 ; +2

$$dp[i][j] = \begin{cases} 2 + dp[i+1][j-1] & \text{if } s[i] == s[j] \\ \max(dp[i+1][j],\ dp[i][j+1]) \end{cases}$$



Edit distance

$$dp[i][j] = \begin{cases} dp[i-1][j-1] & \text{if } s_1[i] == s_2[j] \\ 1 + \min(dp[i-1][j-1],\ dp[i][j-1], \\ \quad dp[i-1][j]) & \end{cases}$$

Regular exp matching

String S and String pattern

character

.

*

a b .

a b c

a b f

a b g

*

a b *

a

a b

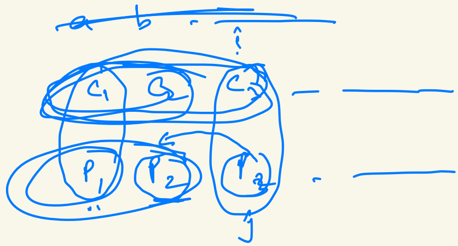
a b b

a b b b

i j

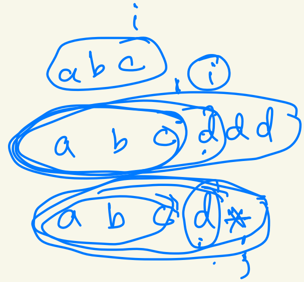
fill ith char in s — fill j char in p

s.substring(0, i) — p.substring(0, j)



$dp[i][j] =$

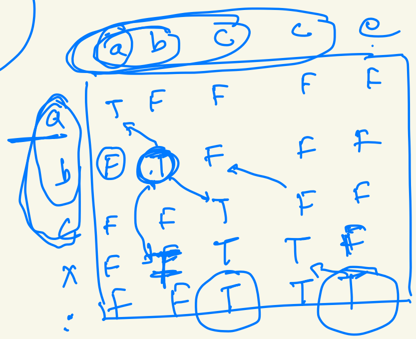
$dp[i][j] = \begin{cases} \text{if } s[i] == p[j] \text{ or } p[j] == \text{'.'} \\ \quad \text{then } dp[i-1][j-1] \\ \text{if } p[j] == \text{'*'} \\ dp[i][j-2] \;||\; ((s[i] == p[j-1] \text{ or } p[j-1] == \text{'.'}) \\ \quad \text{and } dp[i-1][j]) \end{cases}$



abcce

abc*.

abcc.



a ab

abc abc*

abcce

abc*

abc

abc*

*

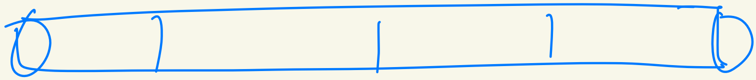
b

ab

ab c*

...

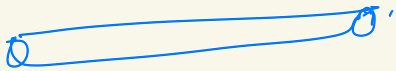
Rod cutting



1	5
2	11
3	13

DP state

dp[i] rep max value we can get from rod of length i



4 + 1

3 + 2

2 + 2 + 1

1 + 1 + 1 + 1 + 1

$i = 1$ to n

$dp[i] =$

for j in 0 to $i-1$

$\max(dp[i], price[j] + dp[i-j-1])$

Optimal Binary Search

matrix chain multiplication

optimal game strategy

N coins are there

P_1

P_2 2, 3 P_1 5 P_2 9 P_1 2 P_2 6 P_1

P_2

P_1 — 6 + 3 + 4 → 13

P_2 → 2 + 5 + 2 → 9

C_1 C_2 C_3 C_4 C_5 C_6

$C_1 + C_3 + C_5 =$

$C_2 + C_4 + C_6 =$

7 5 4 3 8 6 5 7

24 21

$c_0', c_1', \ldots \ldots c_n'$ $\quad (i+1, j)$

$f_A(i, j)$

$$\max\left(c[i] + f_B(i+1, j),\; c[j] + f_B(i, j-1)\right)$$

$$\min\left(f_A(i+2, j),\; f_A(i+1, j-1)\right)$$

$$\min\left(f_A(i+1, j-1),\; f(i, j-2)\right)$$

$f_A(i, j) =$

$$\max\Big(c[i] + \min\left(f_A(i+2, j),\; f_A(i+1, j-1)\right),$$
$$c[j] + \min\left(f_A(i+1, j-1),\; f(i, j-2)\right)\Big)$$

✓ ✓ ✓