

Problem 1

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

We use exponentiation by squaring on the  $2 \times 2$  matrix recursive:

if  $n=0 \rightarrow$  return identity matrix

if  $n$  even  $\rightarrow M^n = (M^{\frac{n}{2}})^2$

if  $n$  odd  $\rightarrow M^n = M \times (M^{\frac{n-1}{2}})^2$

each multiplication is  $O(1)$  time

complexity:

$$T(n) = T\left(\frac{n}{2}\right) + O(1) \quad (\text{with } T(1) = O(1), T(0) = O(1))$$

classic case for the Master Theorem

$a=1$  (one subproblem)

$b=2$  (size divided by 2)

$f(n) = O(1)$  (work outside recursion)

compute  $f(n)$  with  $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$f(n) = O(n^0) \rightarrow$  case 2.

$$\text{So: } T(n) = O(n^{\log_b a} \times \log n) = O(1 \times \log n) = O(\log n)$$

The time complexity is  $O(\log_2 n)$  or  $O(\log n)$

No.

Date

## Problem 2.

Let:  $S_1 = \text{"SATURDAY"} \text{ (len 8)}$  $S_2 = \text{"SUNDAY"} \text{ (len 6)}$ build a  $(9 \times 7)$  table include empty prefixes.

	$\epsilon$	S	U	N	D	A	Y
$\epsilon$	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
A	2	1	1	2	3	3	4
T	3	2	2	2	3	4	4
U	4	3	2	3	3	4	5
R	5	4	3	3	4	4	5
D	6	5	4	4	3	4	5
A	7	6	5	5	4	3	4
Y	8	7	6	6	5	4	3

Initialize

$$dp[i][j] = i$$

$$dp[0][j] = j$$

Then  $i=1 \rightarrow 8, j=1 \rightarrow 6$ 

$$dp[i][j] = \min \{$$

$$dp[i-1][j] + 1 \text{ // delete}$$

$$dp[i][j-1] + 1 \text{ // insert}$$

$$dp[i-1][j-1] + (\text{if } S_1[i-1] == S_2[j-1] \text{ else } 1) \text{ // match or sub}$$

)

the final answer is  $dp[8][6] = 3$ 

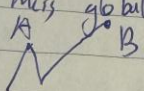
edit distance = 3

### Problem 3

3.1

Greedy fails because it makes locally optimal choice but can miss globally optimal.

e.g.



point A is locally optimal and there is no better point close to A. So greedy would stop there even though there is point B better than A.

I am sorry I don't remember the true course e.g.

3.2.  $n=3$ , value = [1, 2, 3], weight = [4, 5, 1],  $w=4$

$dp[i][w] =$

$\{dp[i-1][w]$  if  $weight_i > w$

$\max(dp[i-1][w], value_i + dp[i-1][w - weight_i])$  otherwise

Initialization:  $dp[0][w] = 0$  for all  $w$  (no items  $\rightarrow$  value 0)

capacity  $w$     0    1    2    3    4

Item 0        0    0    0    0    0

Item 1 ( $v=1$ )    0    0    0    0    1

Item 2 ( $v=2$ )    0    0    0    0    1

Item 3 ( $v=3$ )    0    3    3    3    3

Item 4    answer:  $dp[3][4] = 3$

3.3.

Use a single array  $dp[0..w]$  instead of a 2D table

$dp = [0] * (w+1)$  #  $dp[i] = \text{max value with capacity } w$   
for each item  $i$  in  $1$  to  $n$ :

for  $w = w$  down to  $\text{weight}[i]$

$dp[w] = \max(dp[w], \text{value}[i] + dp[w - \text{weight}[i]])$

initial:  $dp = [0, 0, 0, 0, 0]$

1. Item 1 ( $v=1, wt=4$ ). Update only  $w=4$ :  $\max(0, 1 + dp[0]) = 1$   
 $\rightarrow dp = [0, 0, 0, 0, 1]$

2. Item 2 ( $v=2, wt=5$ ),  $5 > 4 \rightarrow$  ~~no update~~ no update  
 $\rightarrow dp$  remains  $[0, 0, 0, 0, 1]$

3. Item 3 ( $v=3, wt=1$ ) find  $dp = [0, 3, 3, 3, 3]$

result:  $dp[4] = 3$

Complexity:

time: still  $O(n \times w)$

space:  $O(w)$  (only one array of size  $w+1$ )