

Prob lem 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 2×2 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 $\Theta(1)$ time

complexity:

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

classic case for the Master theorem

$a=1$ (one subproblem)

$b=2$ (size divided by 2)

$f(n) = \Theta(1)$ (work outside recursion)

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

$$f(n) = \Theta(n^0) \rightarrow \text{case 2.}$$

$$\text{so: } T(n) = \Theta(n^{\log_2 1} \times \log n) = \Theta(1 \times \log n) = \Theta(\log n)$$

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

Problem 2.

Let: $S_1 = "SATURDAY"$ (len 8)

$S_2 = "SUNDAY"$ (len 6)

build a (9×7) table include empty prefixes.

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

Initialize

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

edit distance = 3

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

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

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

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

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

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

$$dp[i-1][j-1] + 0 \text{ if } S_1[i-1] == S_2[j-1] \text{ else } 1 \quad // \text{match or sub}$$

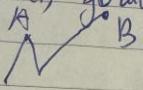
}

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 ex.

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

$$dp[i][w] =$$

$$\{ dp[i-1][w] \text{ if weight } i > w$$

$$\max(dp[i-1][w], \text{value}_i + dp[i-1][w - \text{weight}_i]) \text{ otherwise}$$

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

Item 1 (w=1) 0 0 0 0 1

Item 2 (w=2) 0 0 0 0 1

Item 3 (w=3) 0 3 3 3 3

answer: $dp[3][4] = 3$

3.3.

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

$dp = [0] * (w+1)$ # $dp[i] = \max$ 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 $w=4$: $\max(0, 1 + dp[0]) = 1$
 $\rightarrow dp = [0, 0, 0, 1, 1]$

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

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

Result: $dp[4] = 3$

Complexity:

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

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