ATCODER EDUCATIONAL DP Contest — Part 1

Q. A

Frog

10	30	40	20

start — end

a[i] → height of Stone

$N \le 10^5$

$i^{th} \to (i+1)$

$i^{th} \to (i+2)$

How can we solve it recursively ??

↳ Base Case

↳ Recursive Intuition

↳ Self work

Base Case ?? → what if you only had 1 stone → 0

$(i=0) = 0$

$$f(i) = \min\left(\left(|h_i - h_{i-1}| + f(i-1)\right), \left(|h_i - h_{i-2}| + f(i-2)\right) \right)$$

min cost to reach the i^{th} stone from 1^{st} stone

Top Down DP

fibonacci → modifier

TC ← $O(n)$

+1

+2

A

y	10	30	40	90
	i=0	1	2	3
	0	20	30	30

$\rightarrow$ Heights

$\rightarrow$ DP

$\rightarrow$ 30 ans

$0 + |30-10| = 20$

$\min\left(20 + |30-40|,\ 0 + |40-10|\right)$

$20 + |30-20|,\ 30 + |40-20|$

$(30,\ 50)$

$$f(i) = \min\left(|h_i - h_{i-j}| + f(i-j) \right) \quad \forall j \in [1, k]$$

$\hookrightarrow$ loop

recurrence

for $k=2$ problem B is same as problem A

$$f(i) = \min\left(|h_i - h_{i-1}| + f(i-1),\ |h_i - h_{i-2}| + f(i-2) \ldots\ldots \ldots |h_i - h_{i-k}| + f(i-k) \right)$$

TC $\to$ a $O(n)$

b $O(n^2)$

c $O(nk)$

d Nam

S-6

Problem C → Vacation

i^{th} day → what activity can i do on the i^{th} day

→ 2 choices ③

→ We can't choose the activity that we choose on $(i-1)^{th}$ day

any j^{th} activity on $(i-1)^{th}$ day →

Among the 2 choices we have we will pick the one with max points

$$f(i, j) = \begin{cases} p_a + \max(f(i-1, b), f(i-1, c)) & \text{if we choose activity } a \text{ on } (i)^{th} \text{ day} \\ p_b + \max(f(i-1, a), f(i-1, c)) & \text{if we choose activity } b \text{ on } (i)^{th} \text{ day} \\ p_c + \max(f(i-1, a), f(i-1, b)) & \text{if we choose activity } c \text{ on } (i)^{th} \text{ day} \end{cases}$$

i → i^{th} day

j → activity

max points gainable on i^{th} day

Recursion

$f(i-1, a)$

For our curr day

$\max(a, b, c)$

Base case

	A	B	C
i=1	10	40	70
i=2	20	50	80
i=3	30	60	90

90, 120, 120

	A	B	C		
dp i=1	10	40	70	→	max
dp i=2 →	90	120	120	→	120
→	150	180	210	→	210

Problem-0 Knapsack

→ For each item we can either choose the item
→ if the weight permits or we can leave it.

Brute Force

Recurrence Relation

{A B C} W

( ) (AB) (BC)
(A) (AC)
(B) (ABC)
(C)

max profit

$f(i, w')$ =

i : consider the i^{th} elemt, what max profit we can make

w' : wt of knapsack

$$f(i, w') = \begin{cases} f(i-1, w') & \text{if } (w[i] > w') \\ \text{or} & \\ \max\left(\underbrace{f(i-1, w')}_{\text{not chosen } i^{th} \text{ elemt}},\ \underbrace{f(i-1, w'-w[i]) + v[i]}_{\text{choosing the } i^{th} \text{ elemt}}\right) & \end{cases}$$

wt of knapsack is 0

no item present

3   $W = 8$

$i=1$	3	30	→	$i=1$
	4	50		$i=2$
	5	90		$i=3$

	0	1	2	3	4	5	6	7	8
$i=0$	0	0	0	0	0	0	0	0	0
$i=1$	0	0	0	30	30	30	30	30	30
$i=2$	0	0	0	30	50	50	50	80	
$i=3$	0								

n^{th} row

$n \times W$

TC $O(n \times W)$

X X X X X

$(k-1)^{th}$ row

k^{th} row

$(30, 50 + 30)$ 80

$O(1)$

$O(W)$

$dp[i][j] \rightarrow$ max profit with knapsack wt as j and all items till i^{th} idx

$(30, 50 + 0)$

$\rightarrow dp[i][j] = dp[i-1][j]$

$dp[i][j] = \max(dp[i-1][j],\ v[i] + dp[i-1][j - wt[i]])$

$(0, 30 + 0)$

Problem Knapsack 2

$f(i, j)$ → profit to sum

W

function which returns the minimum weight required to get a profit j with first i elements

$$f(i,j) = \begin{cases} f(i-1, j) & \text{if } val[i] > j \\ \min\big(f(i-1, j),\ wt[i] + f(i-1, j - val[i])\big) & \text{else} \end{cases}$$

possible code

→ for (i = n-1)

(0)

wt

if all element has 10^3 profit → max profit $n \times 10^3$

10^2

j to → 10^3

Base Case

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
i=0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x
i=1	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
i=2	0	0		3	∞	4									
i=3	0														

dp[i-1][8-5]

3 8 ← knapsack wt

$\min(\infty,\ 4+\infty)$ → 4

$\min(\infty,\ 4+3)$

8 - 5

i=1 → 3, 4, 5 wt

3, 5, 6 → profit

0	1	2	3	4	5	6	7	8	9	10
0	∞	∞	3	∞	∞	∞	∞	∞	∞	∞

$i=1$

$i=2$

0	∞	∞	3	∞	4	∞	∞	7	

min wt

2 cases → pick
→ donot pick

$$\min\begin{pmatrix} 4+3 \\ \infty \end{pmatrix}$$

$dp[i][8-5] = dp[1][3]$