

2 Saturday
FEBRUARY

033-332 Week 5

DYNAMIC PROGRAMMING

Two ways :

- ✓ Tabulation → Bottom Up approach (usually from start to end)
- ✓ Memoization → Top-down approach (often from end to start)

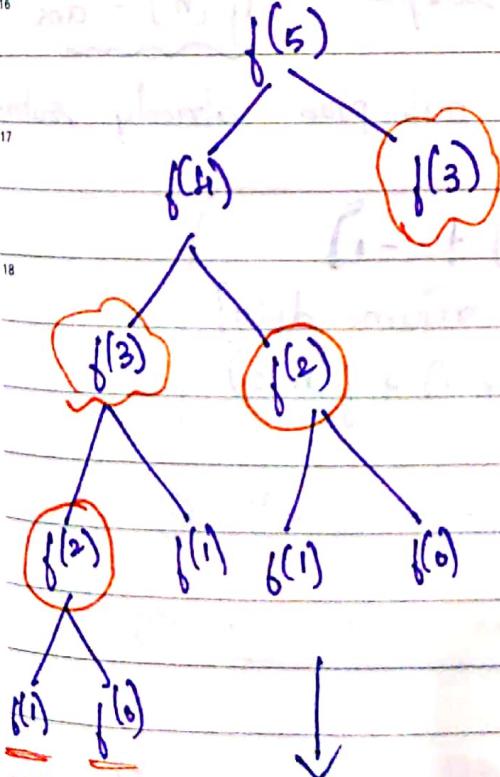
11 → Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

(addition of previous 2 nos)

$$f(n) = f(n-1) + f(n-2)$$

Recursion Tree (n=5)



overlapping sub-problems
happen here

How to avoid? ⇒ MEMOIZATION

(store values of sub-probs
in memo map/table)

3 Sunday

034-331 Week 5

⇒ If your sub-problem returns a single value, then store in

1D - Array

The sub-problems here are $f(6), f(5), f(4), f(3), f(2), f(1), f(0)$

create table of $n+1$ size

2019

MARCH						
W	M	T	W	T	F	S
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

4 Monday
FEBRUARY

1 2 3 4 5 Week 6 035-330

Create 1D array \Rightarrow $dp[7] =$

✓	✓	✓	✓	✓	✓	✓
0	1	2	3	4	5	

Initially store with
 -1 as

update the values
post each recursion call

$f(0) \rightarrow$ returns 0 (store in dp)

$f(1) \rightarrow$ returns 1 (store in dp)

$\checkmark f(2) \rightarrow$ returns 1 (store in dp)

$\checkmark f(3) \rightarrow$ returns 2 (store in dp)

.....

Recursive Code

```
f(n) {
    if (n <= 1)
        return n;
```

```
return 
$$\boxed{f(n-1) + f(n-2)}$$
;
```

}

Sub-problem
(return to dp)
array

MEMOIZATION

① declare $dp[n+1]$

② $\boxed{f(n-1) + f(n-2)}$ \rightarrow store in dp
value of subprob $dp[n] = \text{ans}$

③ check if sub-prob already solved

\checkmark
$$\begin{cases} \text{if } (dp[n] \neq -1) \\ \quad \text{return } dp[n]; \\ \text{return } f(n-1) + f(n-2) \end{cases}$$

MEMOIZATION CODE

Main() {

int[] dp = new int[n+1];

 Arrays.fill(dp, -1); \hookrightarrow declare Arr & fill (-1)
 fib(n, dp);

}

int fib(int n, int[] dp) {

if (n <= 1)

return n;

 \rightarrow Base condn

if (dp[n] != -1)

 return dp[n]; } \rightarrow check if val is already calculated,

return dp[n] = fib(n-1, dp) + fib(n-2, dp);

}

 \downarrow store in dp array & then returnRecursion \rightarrow Tabulation

(Bottom-up)

Base Condition to the Required

① Declan dp[n+1]

② Fill Bas. Case

$$\begin{cases} dp[0] = 0 \\ dp[1] = 1 \end{cases}$$

Tc: O(N)

Sc: O(N)

only Array

③ Implement Recursion Equation (if n > 1; from 2)

for(i=2; i<=n; i++) {

$$dp[i] = dp[i-1] + dp[i-2];$$

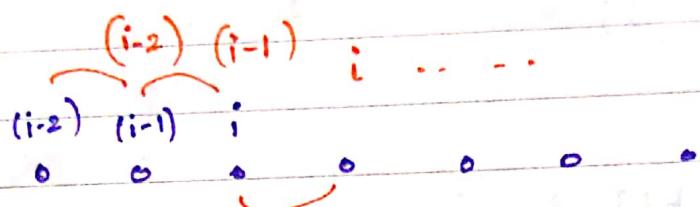
MARCH						
W	M	T	W	T	F	S
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

2019

Tc: O(N)
 Sc: O(N) + O(N)
 ↴
 Recursion stack
 ↓
 Array

Space Optimization in Tabulation

What we require
to solve i ? } = $i-1 \quad a_i \quad i-2$



Each time, 2
variables will be
required for calculating
 i

① ~~at~~ $\text{prev}_2 = 0$ $\text{prev} = 1$

② $\text{for } (i=2; i=n; i++) \{$

$\text{cur} = \text{prev} + \text{prev}_2;$

$\text{prev}_2 = \text{prev}$

$\text{prev} = \text{cur}$

TC: $O(N)$

SC: $O(1)$

}
print prev \rightarrow end of for loop at $(n+1)$,
so cur points at $n+1$
hence return prev

FEBRUARY 2019						
W	M	T	W	T	F	S
5			1	2	3	
6	4	5	6	7	8	9
7	11	12	13	14	15	16
8	18	19	20	21	22	23
9	25	26	27	28	-	-

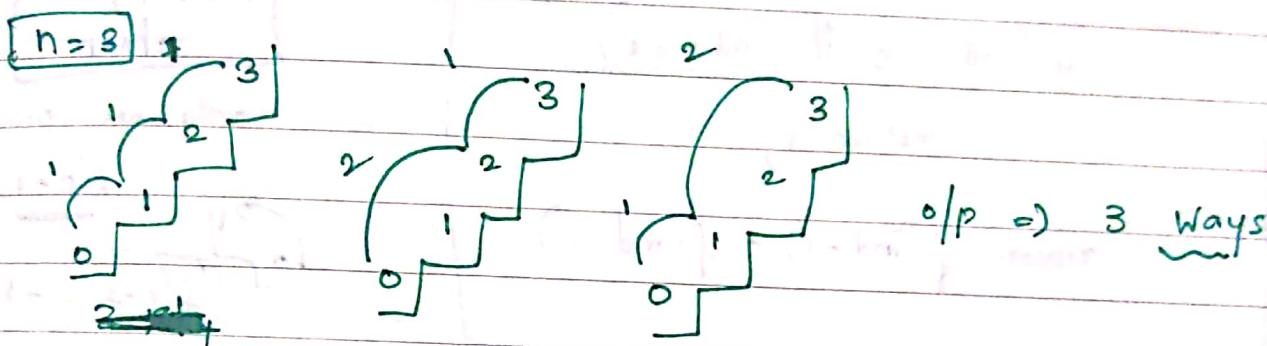
Thursday
FEBRUARY

L2

CLIMBING STAIRS

Given: n (no. of stairs), initially at 0th step & have to reach nth step

Each time \rightarrow climb 1/2 steps, return no. of distinct ways to climb steps



① Count total no. of ways // all possible ways

② Multipl. ways possible:
return min // max

→ Think as Recurs
followed by DP

Soln Recursion using DP

① Try to represent in terms of index.

② Do all possible stuff on that index acc to prob statement

iii) Count all ways \rightarrow sum of all stuffs

find min \rightarrow min (of all stuffs)

MARCH 2019						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

2019

8 Friday
FEBRUARY

Week 6 039-326

⇒ This prob is same as fibonacci

little change in base case;

$n == 0$;
return 1

only one possibility

$f(\text{ind})$

if ($\text{ind} == 0 \text{ || } \text{ind} == 1$)

return 1;

return $f(\text{ind}-1) + f(\text{ind}-2)$

}

$n == 1$;
return 1

only one possibility

$f(n-1)$

$f(n-2) = -1$ X

* Represent in terms of index $f(3)$

+ Try all possible ways: can jump 1 step / 2 step

$lh = f(\text{ind}-1)$
 $rh = f(\text{ind}-2)$

+ All possible shifts

⇒ return $lh + rh$

FEBRUARY	2019
S	5
W	6
T	7
F	8
S	9
S	10

★

9 Saturday
FEBRUARY

(13)

Frog Jump

Gn → Frog on 1st step of floor, want to reach the nth stair

Height [i] is the height of $(i+1)th$ stair,
energy lost if jump from i to j \rightarrow $ht(i-1) - ht(j-1)$

can jump either +1 / +2 steps. Find min total energy to reach from 1st to nth stair

Eg \Rightarrow 10 20 30 ~~30~~ 10 \rightarrow min energy after reaching from $(0 \rightarrow 3)$

idx 0 1 2 3

idx

Possibilities

$$\textcircled{1} \quad \begin{matrix} \xrightarrow{10} & \xrightarrow{10} & \xrightarrow{20} \\ 10 & 20 & 30 & 10 \end{matrix} = 40$$

$$\textcircled{2} \quad \begin{matrix} \xrightarrow{10} & \xrightarrow{10} & \xrightarrow{10} & \xrightarrow{10} \\ 10 & 20 & 30 & 10 \end{matrix} = \textcircled{20} \rightarrow \text{Min.}$$

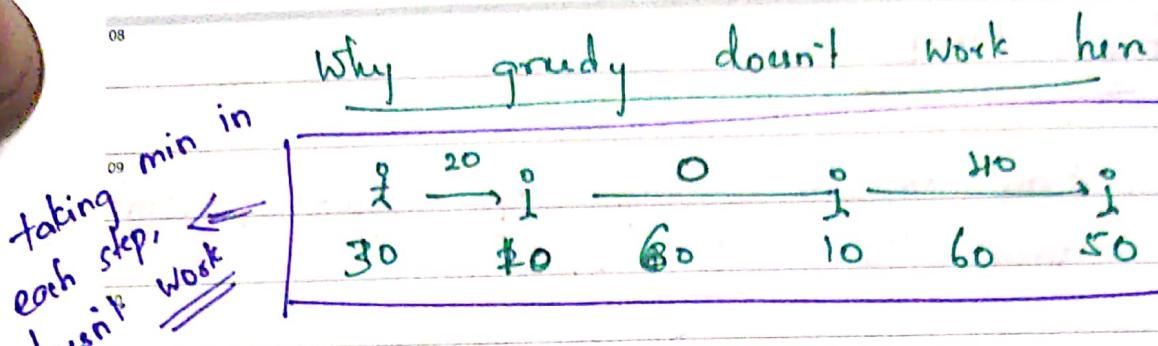
$$\textcircled{3} \quad \begin{matrix} \xrightarrow{20} & \xrightarrow{20} \\ 10 & 20 & 30 & 10 \end{matrix} = 40$$

041-324 Week 6

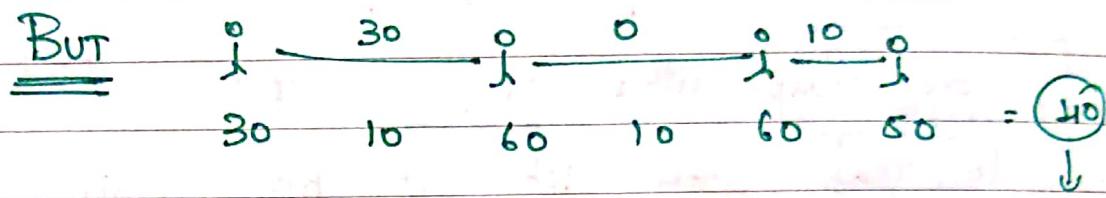
Here, we are trying all possible ways \rightarrow Use Recursion

MARCH 2019						
W	M	T	W	T	F	S
9			1	2	3	4
10	4	5	6	7	8	9
11	11	12	13	14	15	16
12	18	19	20	21	22	23
13	25	26	27	28	29	30
	31					

2019



11 \Rightarrow Moving in min: energy loss in each step.



Best Ans

- 15 ✓ Use Recursion \Rightarrow try all possible ways
 ✓ Among poss. ways \Rightarrow find min

17 $f(\text{ind}) \{$

if ($\text{ind} == 0$)

return 0;

int lh = $f(\text{ind}-1) + \text{abs}(\text{arr}[\text{ind}] - \text{arr}[\text{ind}-1]);$

if ($\text{ind} > 1$)

int rh = $f(\text{ind}-2) + \text{abs}(\text{arr}[\text{ind}] - \text{arr}[\text{ind}-2]);$

return Math.min (lh, rh);

}

FEBRUARY					2019
W	M	T	W	F	S
5	6	7	8	9	10
7	11	12	13	14	15
8	18	19	20	21	22
9	25	26	27	28	-

12

Tuesday
FEBRUARY

043-322 Week 7

0 1 2 3 4

5

[30, 10, 60, 10, 60, 50]

n=6

~~f(5) f~~

f(5)

210

30

f(4) + abs(10)

20

f(3) + abs(40)

20

f(3) +

f(2) 30

30

f(2) +

f(1) 20

f(1) final 0+20

f(0) +

left = f(0) + abs(30-10)

right = max

return min(20, max);

f(1) = 20

f(1)
20
(min)f(0)
X (max)
0+abs(20)

f(2) { (20+50=70)

left = f(1) + abs(50)

(0+30)

right = f(0) + abs(30)

f(2) = 30

return min(-70, 30)

f(3) { (30+50=80)

left = f(2) + abs(50)
(20+0=20)

right = f(1) + abs(0)

return min(80, 20)

f(4) {

left = f(3) + abs(50)
right = f(2) + abs(0)

return min(-70, 30)

f(5) {

left = f(4) + abs(10)
right = f(3) + abs(0)

return min(60, 0)

MARCH 2019						
S	M	T	W	T	F	S
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

f(3) = 20

f(4) = 30

f(5) = 110

13 Wednesday
FEBRUARY

Week 7 • 044-321

MEMOIZATION

```
08 main() {  
09     int[] dp = new int[n+1];  
10     Arrays.fill(dp, -1);  
11     jump(n, dp);  
12 }
```

TC: $O(N)$ linear
SC: $O(N) + O(N)$

```
13     int jump(int n, int[] dp) {  
14         if (n == 0)  
15             return dp[n] = 0;  
16         if (dp[n] != -1) return dp[n];  
17         int left = jump(n-1, dp) + abs(ht[n] - ht[n-1]);  
18         int right = INT_MAX;
```

```
19         if (n > 1) right = jump(n-2, dp) + abs(ht[n] - ht[n-2]);  
20         dp[n] = Math.min(left, right);  
21     }
```

FEBRUARY 2019						
W	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24				

TABULATION

Bottom-upDeclare $dp[]$ array

step 1:

Base Case

$$dp[0] = 0$$

step 2:

Iterate for next steps

step 3:

Code

```
int[] dp = new int[n];
```

```
Arrays.fill(dp, -1);
```

```
dp[0] = 0;
```

```
for (int i = 1; i < n; i++) {
```

```
    int l = dp[i-1] + Math.abs(ht[i] - ht[i-1]);
```

```
    int r = Int.MAX;
```

```
    if (i > 1)
```

```
        r = dp[i-2] + Math.abs(ht[i], ht[i-2])
```

```
    dp[i] = Math.min(l, r);
```

```
}
```

```
sop(dp[n-1]);
```

```
}
```

[30, 10, 60, 10, 60, 50]

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

0	20	-1	-1	-1	-1
0	1	2	3	4	5

0	20	30	-1	-1	-1
0	1	2	3	4	5

0	20	30	20	-1	-1
0	1	2	3	4	5

0	20	30	20	30	-1
0	1	2	3	4	5

0	20	30	30	40
0	1	2	3	4

ans

Further you can space
optimize using 2 var
concept

2019



18 Monday
FEBRUARY

Week 8 04.3.15

(L5)

Max sum of
non-adjacent elements

Given: Array of n integers \rightarrow Return the max sum
of subsequence,

In subsequence, ele's can't be adj.

ilp \Rightarrow 2 1 4 9
olp \Rightarrow 2 9 11 (2, 9)

Approach \rightarrow Try all sub-sequences with gn Constraint
or pick the one with max sum

(RECURSION)

pick not pick

Recursion Approach

- + Every index \Rightarrow two options
- + pick a move to (i+2) element (because no adj)
If picked idx 0 \rightarrow move to idx 2
- + not pick a move to (i+1) element
If not picked idx 0 \rightarrow pick idx 1

RecurrenceRelation:

i) Represent in terms of index:

$f(\text{idx}) = \max \text{ sum of subsequence}$
from idx 0 to idx last

ii) Try all choices . . .

Pick / non-pick

(keeping in mind
non-adj elements)

$$\checkmark \text{ pick} \Rightarrow \text{arr}[\text{idx}] + f(\text{idx} - 2)$$

$$\checkmark \text{ non-pick} \Rightarrow 0 + f(\text{idx} - 1)$$

$\Rightarrow f(\text{idx} - 2)$
↓
because no
adj elts

iii) Return max of all choices

(return max of 2 choices:

Pick / non-pick)

BASE CONDN

if ($\text{idx} == 0$)

return $\text{arr}[\text{idx}]$

if ($\text{idx} < 0$)

return 0

When will we \Rightarrow only if we
reach idx 0? ignore idx 1

we reach here when
we considered idx 1,
so further can be
ignored

So Consider idx 0

MARCH	MON	TUE	WED	THU	FRI	SAT	SUN
2019							
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	

PSEUDO - CODE

08
f(ind, arr[]) {

09
 if (ind == 0)

10
 return arr[ind];

11
 if (ind < 0)

12
 return 0;

13
 int pick = arr[ind] + f(ind-2, arr);

14
 int nonpick = 0 + f(ind-1, arr);

15
 return Math.max (pick, non-pick);

16
[2, 1, 4, 9]

17
⑪

dp

f=3

9+ ②

9+ f=1

max(11, 6)

0+ ⑥

0+ f=2

max(6, 2)

4 + f(0)

0 + 2

num 2

4 + 2

num 2

1 + f(-1)

int 0

0 + f(1)

max(1, 2)

1 + 0

0 + 2

num 2

1 + 2

num 2

21 Thursday
FEBRUARY

052-313 Week 8

Memoization → (overlapping problems)

int main (int n, int arr) {

③ dp declare ← int [] dp = new int [n];

Arrays. fill (dp, -1);

solve (n-1, arr, dp);

}

public int solve (int ind, int [] arr, int [] dp) {

① Represent
in index

if (ind == 0)

return arr [ind];

② Base cond

if (ind < 0)

return 0;

③ check if
already calculated

if (dp [ind] != -1) → return dp [ind];

int pick = arr [ind] + solve (ind-2, arr, dp);

int not pick = 0 + solve (ind-1, arr, dp);

④ Export return,

store it

return dp [ind] = Math . Max (pick, not pick).

TC: O(N)

SC: O(N) + O(N)

MARCH 2019						
W	M	T	W	T	F	S
9				1	2	3
10	4	5	6	7	8	9
11	11	12	13	14	15	16
12	18	19	20	21	22	23
13	25	26	27	28	29	30
	31					

2019

22 Friday
FEBRUARY

Week 8 053-312

TABULATION

int [] dp = new int[n];

Base ← [dp[0] = arr[0];
int neg = 0;

for (int i=1; i<n; i++) {

 take = arr[i];

 if (i>1)

 take += dp[i-2];

 not take = 0 + dp[i-1];

 dp[i] = max (take, not take);

from
max of
both