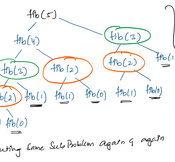
[Solving problem using Subproblems] → Recursion:

$fib(N) = fib(N-1) + fib(N-2)$

1) optimal Substructure

Dynamic Programming

1) Every unique Subproblem should be solved only 1



2) Overlapping Subproblem

Evaluating same SubProblem again & again

```
int dp[N+1] = -1; // All fib are -ve
// Nth fib number  TC: 2^n
int fib(int n){
    if (n==0) return 0;
    if (n==1) return 1;
    if (dp[n] != -1)
        return dp[n];
    dp[n] = fib(n-1) + fib(n-2)
    return dp[n];
}
```

$N \to N-1 \to N-2 \to N-3 \to \cdots 1$

Top Down approach

1) Recursion + (Extra Memory to make sure they are not repeating)

Memoization

Dp Table

Inv: Fib Code

```
int fib1(int n){
    int fib[N+1];
    fib[0] = 0, fib[1] = 1;
    for(int i=2; i<=N; i++){
        fib[i] = fib[i-1] + fib[i-2]
    }
    return fib[N];
}
```

fib[i] = i^{th} fibonacci number — Dp state

Base Conditions

TC: O(N) — $0-1-2-3-4-5 \cdots N$

Bottom up: (Iterative approach) : [Tabulation]

Dp expression

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

current state — 2 previous state

at any point, we need 2 states

We can do this using 3 variables

Todo: H/W [DIY after section]

SC: O(2)

Dp

1) Dp State & Dp[i] : i^{th} fibonacci number : [What need to be calculated]

2) optimal Substructure — Solving Dp[i] using Subproblems

Dp expression → $dp[i] = dp[i-1] + dp[i-2]$

3) Check overlapping Subproblems

4) Dp Table[]

5) Base Conditions : (For which ever valid repeats, your dp expression fail for)

Valid Input $i \geq 0$

→ $i=0, i=1$, expression will fail

Code

6) Find ans in dp table

7) TC & SC → (Table Size)

↳ (No: of Dp states) × [Time taken to calculate a single dp state based dp expression]

8) Space optimization is possible or not?

2(a)

Start $i \to i+1$, $i+2$ } # How many ways we can reach from start 0 → state N

1) [dp[i] : # number of ways to reach i^{th} step] SO:

Total ways: $x + y$ *

$n + y$

N=4
1 1 1 1
1 1 2
1 2 1
2 1 1
2 2

2) $i-1$, $i-2$

x // Number of ways to reach $i-1$ is dp[i-1]

y : dp[i-2]

$dp[i] = dp[i-1] + 2^* dp[i-2]$ *

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

Zoo:

Hayden — 5 — and — Bang — 10 — to 50

Timp + (AN) → all ways

3 — when — 7

ways

3(a) Amazon / Goldman Sachs / Cliff-fit

Given N persons, a person wants to stay alone or get involved with another person [How many ways we can pair all persons]

N = 1, 2, 3

way1 (1, 2, 3)

way2 (1, 2), 3

way3 1, (2, 3)

way4 (1, 3), 2

4 # ways

1) Dp State : Dp[i] = Number of ways we can pair i persons

2) Solve with Subproblem — Dp Expression

Hint: Ben way to get expression is to understand data of ith person

People: 1 2 3 4 5 … i-2 i-1 i

data

He can left alone: i-1, people remaining } dp[i-1]

He can paired — remaining

Pairing 1 & i : i-2 } dp[i-2]

Pairing: 2 & i : i-2 = dp[i-2]

Pairing: 3 & i : i-2 = dp[i-2]

Pairing: i-1 & i : i-2 = dp[i-2]

dp expression: $dp[i] = dp[i-1] + (i-1)\,dp[i-2]$

3) overlapping

4) Dp table = Dp[N+1]

5) Base conditions =

1) Valid Input $n \geq 0$

[n = 0 & n = 1, expression fail]

dp[0] = 0, dp[1] = 1, dp[2] → 2

↳ dp[0] = 1, 0 # people, we can pair in 1 way.

6) for(int i = 3; i <= N; i++) {

dp[i] = dp[i-1] + (i-1)dp[i-2]

3 current — previous state

return dp[n]

dp[2] as per code: 1 } 1, 2 / (1, 2)

Actual example : 2

Note: After filling base check if next state are correct or not,

dp[2] = 2 : Include

dp[2] = dp[2-1] + (2-1)dp[2-2]

= dp[1] + dp[0]

7) TC : (N+1) → O(i) , SC: O(N)

States — i times

8) Space option: Can be done in 3 variables

↳ Those kind of 10, can only done in Tabulation code

9) : Top down / Memoization

dp[5] → dp[N-1] → dp[N-2] → dp[N-3] → dp[3] → dp[2] → dp[1] → dp[0]

We also need to consider valid or not

4(a) Given Dice, find number of ways you can get required sum

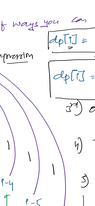
N = 4 :

1 1 1 1
1 1 2
1 2 1
2 1 1
1 3
3 1
2 2
4

State: dp[i] = Number of ways you can get i

Hint:

dp expression



ways to reach i-1

$dp[i] = dp[i-1] + dp[i-2] + dp[i-3] + dp[i-4] + dp[i-5] + dp[i-6]$

$dp[i] = \sum_{j=1}^{6} dp[i-j]$ — 3) overlapping ✓

$dp[i] = \sum_{j=1, i>=j}^{6} dp[i-j]$

Final Dp expression

$dp[0] = dp[0]$ ✓

$dp[1] = dp[0] + dp[1]$ ✓

$dp[2], dp[3], dp[4], dp[5]$

✓ No longer Base condition

Only state for which it what work

$dp[0] = 1$ } We can get sum = 0 by not throwing dice

4) Table : [N+1]

5) Base Conditions:

Valid Input : $N \geq 0$

N = 0, 1, 2, 3, 4, 5 } # expression fails

$dp[5] = dp[4] + dp[3] + dp[2] + dp[1] + dp[0]$

$dp[4] = dp[3] + dp[2] + dp[1] + dp[0]$

6) Find ans: Dp[N]

7) TC → (N+1) × 6 | SC: O(N)

O(N)

8) Space optimization

1) How many state we are considering

7 states

2) That means you can optimize in 7 variable.

SC: O(7)