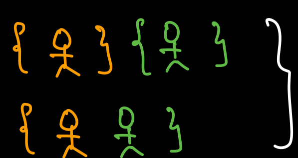
Q. Party pairs.

* Given N persons, How many ways we can pair these N people.

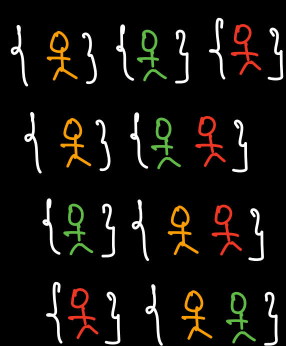
⇒ In a party, a person either wants to stay alone or get paired.

$N=1$  ⇒ 1

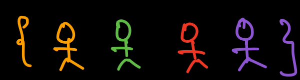
$N=2$  → (2) ways.

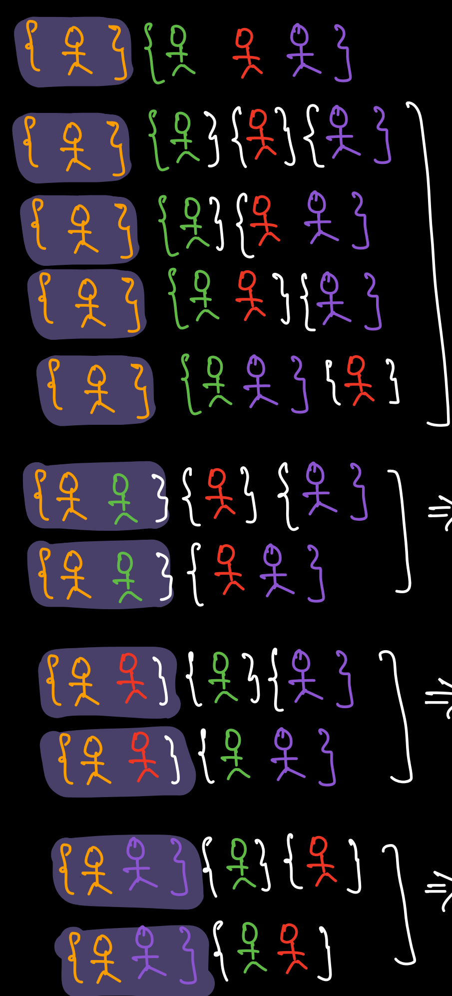


$N=3$  → 4 ways.



$N=4$

 ⇒ 10

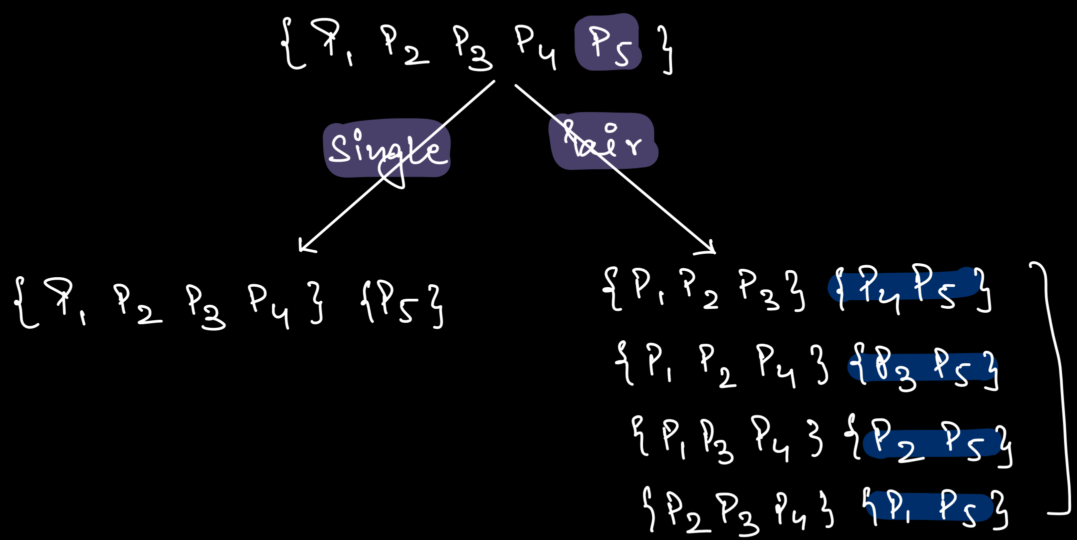


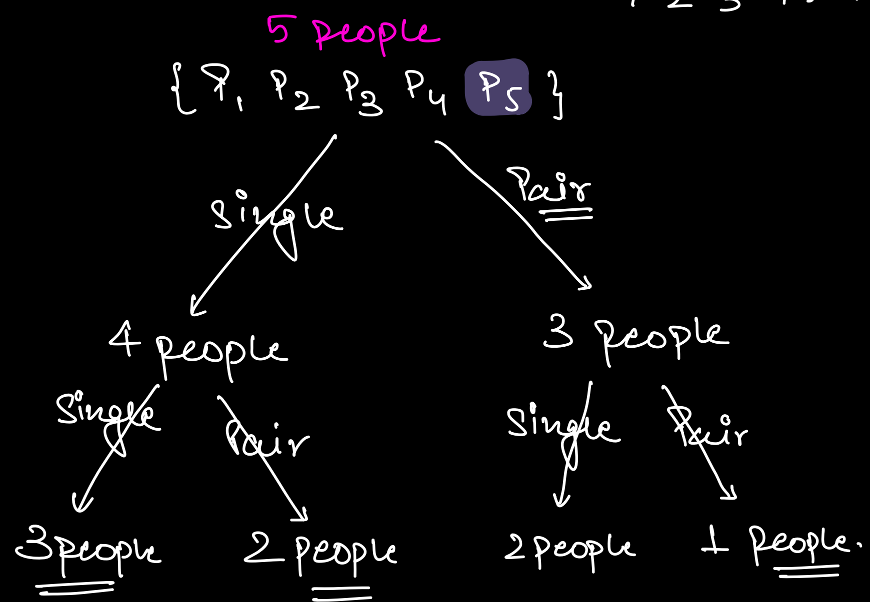
⇒ (4)

⇒ (2)

⇒ (2)

⇒ (2)





→ Optimal Substructure
→ Overlapping subproblems. } DP

dp State

dp[i] = # of ways in which i persons can party.

P_1 P_2 P_3 P_4 P_5 ⇒ ways(5)

P_5 → Single

⇒ ways(4) : { P_1 P_2 P_3 P_4 }

P_5 → Pair ⇒ 4x ways(3)

{P_1 P_5} { P_2 P_3 P_4 } ways(3)

{P_2 P_5} { P_1 P_3 P_4 } ways(3)

{P_3 P_5} { P_1 P_2 P_4 } ways(3)

{P_4 P_5} { P_1 P_2 P_3 } ways(3)

ways(5) = ways(4) + 4x ways(3)

P_1 P_2 P_3 P_4 ------ P_{i-1} P_i

dp Expression :

$$\boxed{dp[i] = dp[i-1] + (i-1) * dp[i-2]}$$

* Base Case ⇒ i=0|1

dp[0] = 0
dp[1] = 1
dp[2] = dp[1] + (1) * dp[0]
= 1

$\boxed{\begin{array}{l} dp[0] = 1 \\ dp[1] = 1 \end{array}}$

dp[2] =
dp[1] + (1) * dp[0]
⇒ 2

Code

```
int dp[N+1];
dp[0] = 1
dp[1] = 1
for(i=2; i<=N; i++){
    dp[i] = dp[i-1] + (i-1)*dp[i-2];
}
return dp[N];
```

TC: $\underbrace{\text{\# of dp states}}_{N} * \underbrace{\text{TC of each dp state}}_{1}$

: O(N)}

SC: O(N) ⟶ O(1)

Todo.

Note: At max 3 variables are required at any point of time.

Q.2 Min. no. of perfect squares to be added to get the target sum.

Amazon MS.

$N = 1 \Rightarrow 1^2 \Rightarrow 1$

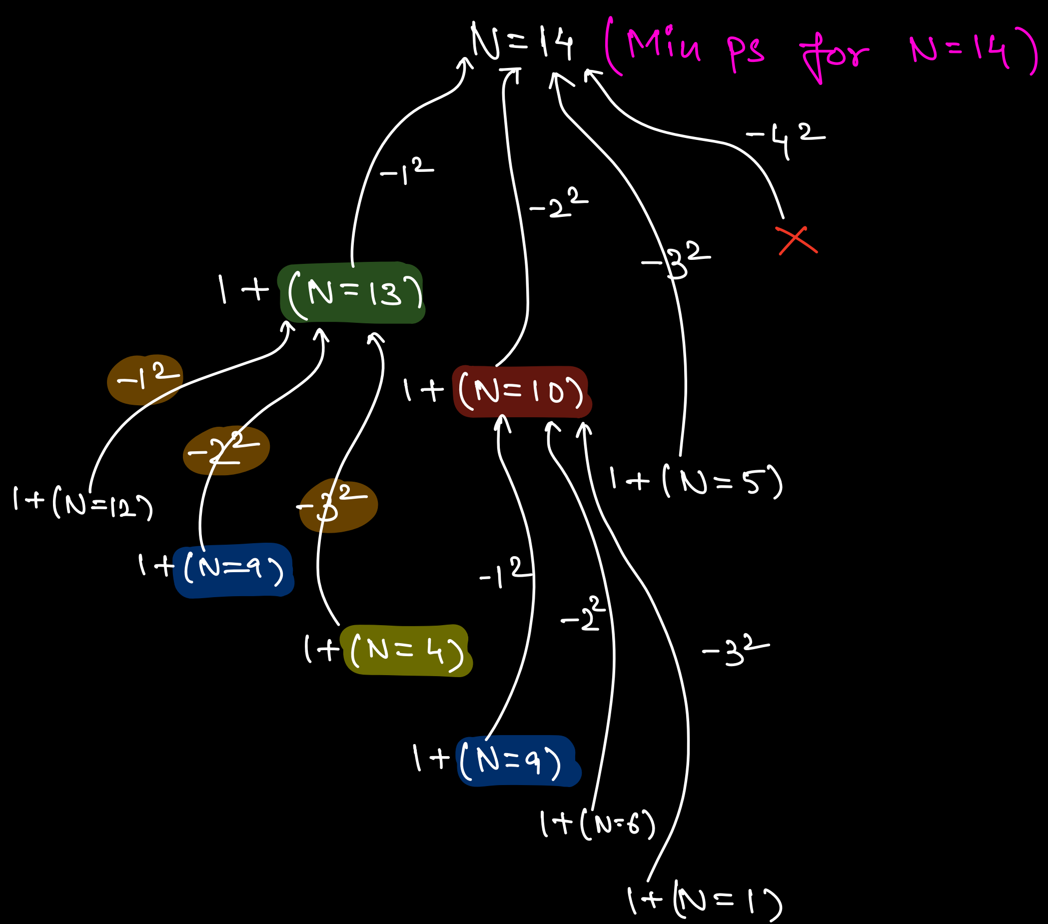
$N = 6 \Rightarrow 1^2 + 1^2 + 2^2 \Rightarrow 3$

$N = 9 \Rightarrow 3^2 \Rightarrow 1$

$N = 10 \Rightarrow 1^2 + 3^2 \Rightarrow 2$

$$N = 12 \Rightarrow \begin{cases} 3^2 + 1^2 + 1^2 + 1^2 \Rightarrow 4 \\ 2^2 + 2^2 + 2^2 \Rightarrow 3 \end{cases}$$

Greedy won't work.



→ Optimal Substructure
→ Overlapping subproblems. } DP

dp state

dp[i] = Min no. of perfect squares required to get sum = i.

dp table

int dp[N+1];

dp Expression

$$dp[i] = \text{Min}\begin{cases} 1 + dp[i-1^2] \\ 1 + dp[i-2^2] \\ 1 + dp[i-3^2] \\ \vdots \\ 1 + dp[i-j^2] \end{cases}$$

$i >= j^2$

$$= \min\left\{ 1 + \forall_{j=1}^{j^2 <= i} dp[i-j^2] \right\}$$

Base Case

$dp[0] = 0$

$dp[1] = 1 + dp[1 - 1^2]$
$= 1 + 0$
$= 1$

$dp[0] = 1$

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

int dp[N+1];

```
int dp[N+1];
dp[0] = 0;
for(i=1; i<=N; i++){
    ans = i;
    for(j=1; j*j<=i; j++){
        ans = min(ans, dp[i-j*j]+1);
    }
    dp[i] = ans;
}
return dp[N];
```

TC : $O(N\sqrt{N}) \Rightarrow$ Worst Case

SC : $O(N) \xrightarrow{\text{optimise}} \times$

$N = 6$

0	1	2	3	4	5	6
0	1	2	3	1	2	3

$$dp[2] = 1 + dp[2-1^2] \Rightarrow 1 + dp[1]$$

$$dp[3] = 1 + dp[3-1^2] \Rightarrow 1 + dp[2]$$

$$dp[4] = \left.\begin{array}{l} 1 + dp[4-1^2] \Rightarrow 1+3 = 4 \\ 1 + dp[4-2^2] \Rightarrow 1+0 = 1 \end{array}\right] \Rightarrow 1$$

$$dp[5] = \begin{array}{l} 1 + dp[5-1^2] = 1 + dp[4] = 2 \\ 1 + dp[5-2^2] = 1 + dp[1] = 2 \end{array}$$

$$dp[6] = \left.\begin{array}{l} 1 + dp[6-1^2] = 1 + dp[5] = 3 \\ 1 + dp[6-2^2] = 1 + dp[2] = 3 \end{array}\right] 3$$

$$12 \begin{array}{l} \nearrow 12 - 3^2 = 3 \Rightarrow 4 \\ \searrow 12 - 2^2 = 8 \Longrightarrow 3 \end{array}$$

Q. Given an Array, find the max subsequence **sum.**

A: { -2 4 5 3 -8 1 } ⇒ 13

A: { 2 6 -1 4 3 -5 } ⇒ **15**

A: { -4 -3 -8 -2 } ⇒ -2

A: { 1 2 4 3 } ⇒ **10**

⇒ find the sum of all +ve **no's.**

⇒ If all elements are -ve then return **MAX.**

Q. ☆ Given arr[N], find the max subsequence **sum.**

Note: We can't pick 2 adjacent elements in the subsequence. Empty subseq. isn't **allowed.**

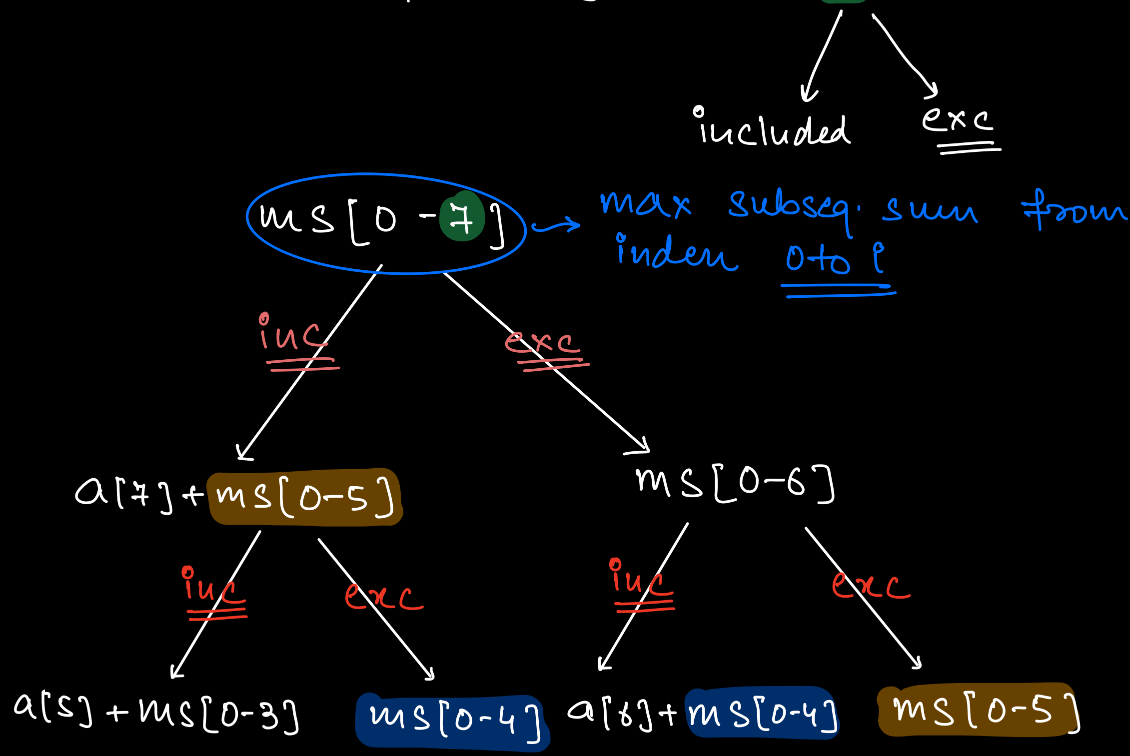
A: 9 14 3 ⇒ **14**

A: 9 4 13 24 ⇒ **33**

A: 13 14 2 ⇒ **15**

Note: ~~Max~~ (Sum of even indices, Sum of odd indices)

A:	0	1	2	3	4	5	6	7
	2	-1	-4	5	3	-1	4	2



→ Optimal Substructure
→ Overlapping subproblems. } DP

dp state :

dp[i] : Max subsequence sum from index [0, i]

dp table.

int dp[N];

dp Expression

$$dp[i] = \text{Max}\begin{cases} A[i] + dp[i-2] & \text{include } i \\ dp[i-1] & \text{exclude } i \end{cases}$$

Base Case

$i = 0/1$

dp[0] : Max subsequence sum from index [0, 0]
= A[0]

dp[1] = max(A[0], A[1])

Code

```
int dp[N];
dp[0] = A[0]
dp[1] = max(A[0], A[1]);
for(i=2; i<N; i++){
    dp[i] = max(A[i]+dp[i-2], dp[i-1]);
}
return dp[N-1];
```

TC: O(N)
SC: O(N) $\xrightarrow{\text{optimise ?}}$ Yes
O(1)

Note: At max we need 3 variables.

A : 9 4 13 24

dp	9	9	22	33
	0	1	2	3

$$dp[2] = \max(\underbrace{A[2] + dp[0]}_{13+9=22}, \underbrace{dp[1]}_{9})$$

$$dp[3] = \max(\underbrace{\underline{A[3]} + dp[1]}_{24+9=33}, \underbrace{dp[2]}_{22})$$

——— * ———