Q.1 Party Pairs

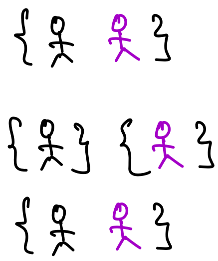
GS: Given **N** persons, how many ways we can pair these **N** persons.

Note:-
- A person either wants to stay alone or get paired
- All the N people need to be present in party.

$N=1$

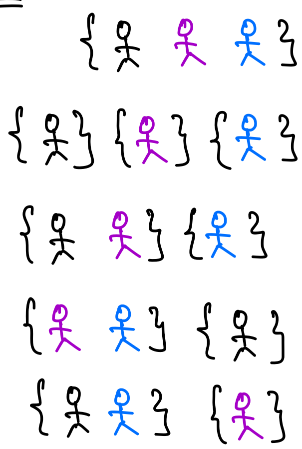
→ 1 way.

$N=2$



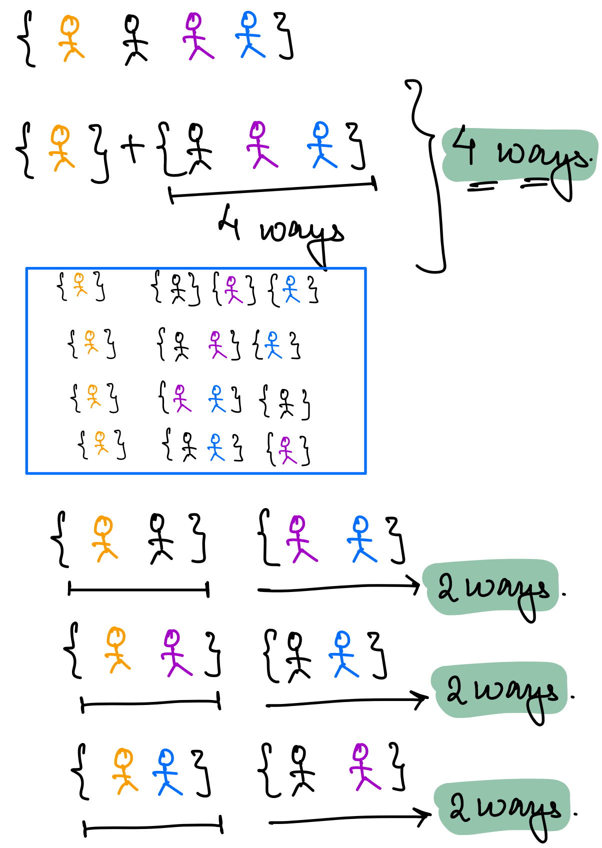
→ 2 ways

$N=3$

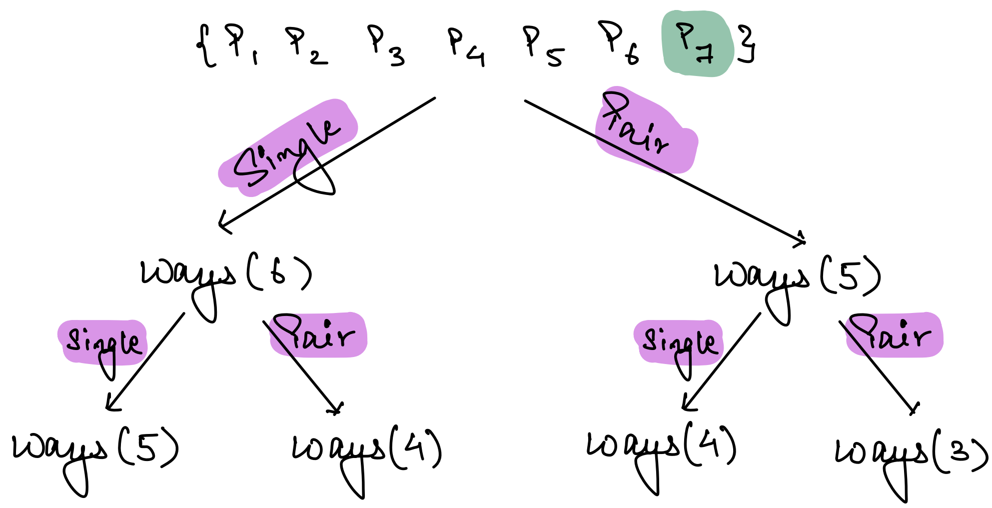


→ 4 ways.

$N=4$



10 ways.



→ Optimal substructure
→ overlapping subproblems.

} DP.

$\{P_1\ P_2\ P_3\ P_4\ P_5\}$

$P_5 \to$ Single $\quad \{P_1\ P_2\ P_3\ P_4\} \to$ ways(4)

Ways(5) = Way(4) + 4× Ways(3)

$P_5 P_4$: $\{P_1\ P_2\ P_3\} \to$ ways(3)

$P_5 P_3$: $\{P_1\ P_2\ P_4\} \to$ ways(3)

$P_5 P_2$: $\{P_1\ P_3\ P_4\} \to$ ways(3)

$P_5 P_1$: $\{P_2\ P_3\ P_4\} \to$ ways(3)

dp[i] : # of ways i persons can party.

int dp[N+1]

$P_1 \; P_2 \; P_3 \; \ldots\ldots \; P_{i-1} \; P_i$

DP Expression { $dp[i] = dp[i-1] + (i-1) * dp[i-2]$

Base Condition

$dp[0] = 0$
$dp[1] = 1$
$dp[2] = dp[1] + 1 * dp[0]$
$= 1$ ✗

$dp[0] = 1$
$dp[1] = 1$
$dp[2] = 2$

```
int party ( int n) {      → Iterative + Tabulation.
    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]
}
```

* $dp[1] = 1$, $dp[2] = 2$ } (i=3; i<=n...

TC: # of States * TC of each state

$N * 1$

$\Rightarrow O(N)$

SC : $O(N)$

→ TODO: $O(1)$

Q.2 Min. no. of perfect squares to be added to get Target sum (N)

$N = 6 : 1^2 + 1^2 + 2^2 = \underline{\underline{6}}. \longrightarrow 3$

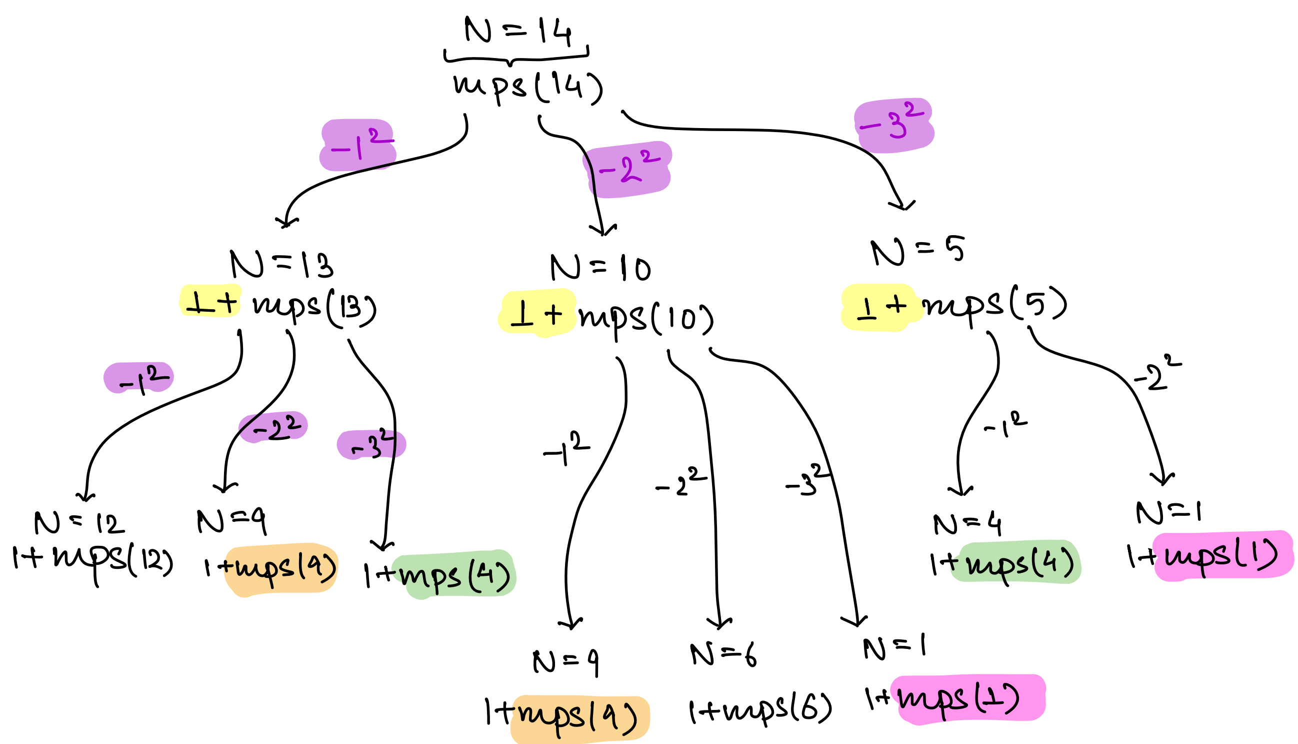
$N = 10 : 1^2 + 3^2 = 10 \longrightarrow 2$

$N = 9 : 3^2 = 9 \longrightarrow 1$

$N = 12 : 1^2 + 1^2 + 1^2 + 3^2 \;\cancel{\longrightarrow}\; 4$

$\quad\quad\; \hookrightarrow 2^2 + 2^2 + 2^2 \longrightarrow 3$

} Greedy won't work.



→ Optimal substructure

→ Overlapping subproblems. } DP.

Steps :-

1) DP state

$dp[i]$: min perfect squares required to get sum = i.

2) DP expression

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

$i - j^2 >= 0$

$j^2 <= i$

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

Base Case :-

$dp[0] = 0$ ✓

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

$= dp[0] + 1$

$= 1$

$dp[0] = 1$ ✗

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

$= dp[0] + 1$

$= 2$ ✗

Code

```
int minPerfectSquares(N) {
    int dp[N+1];
    dp[0] = 0;
    for(i = 1; i <= N; i++) {
        ans = i; // INT_MAX.
        for(j = 1; j*j <= i; j++)
            ans = min(ans, dp[i-j*j] + 1);

        dp[i] = ans;
    }
    return dp[N];
}
```

TC: # of states * TC of 1 state

$\rightarrow N * \sqrt{N}$

$\rightarrow O(N\sqrt{N})$

SC: $O(N)$

N = 6

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

Q: Given N elements, find the max subsequence sum.

Ordered based on indexes.

A: {2, -4, 5, 3, -8, 1} → 2+5+3+1 = 11

A : {2, 6, -1, 4, 3, -5} → 15

A : {-4, -5, -8, -10, -2} → -2

A : {3, 2, 4, 8} → 17

⇒ find the sum of +ve elements, if all elements are -ve then pick max ele.

Q: Given arr[N], find max subsequence sum.

Note:- In a sequence, 2 adjacent elements can't be picked. Empty subsequence is NOT possible

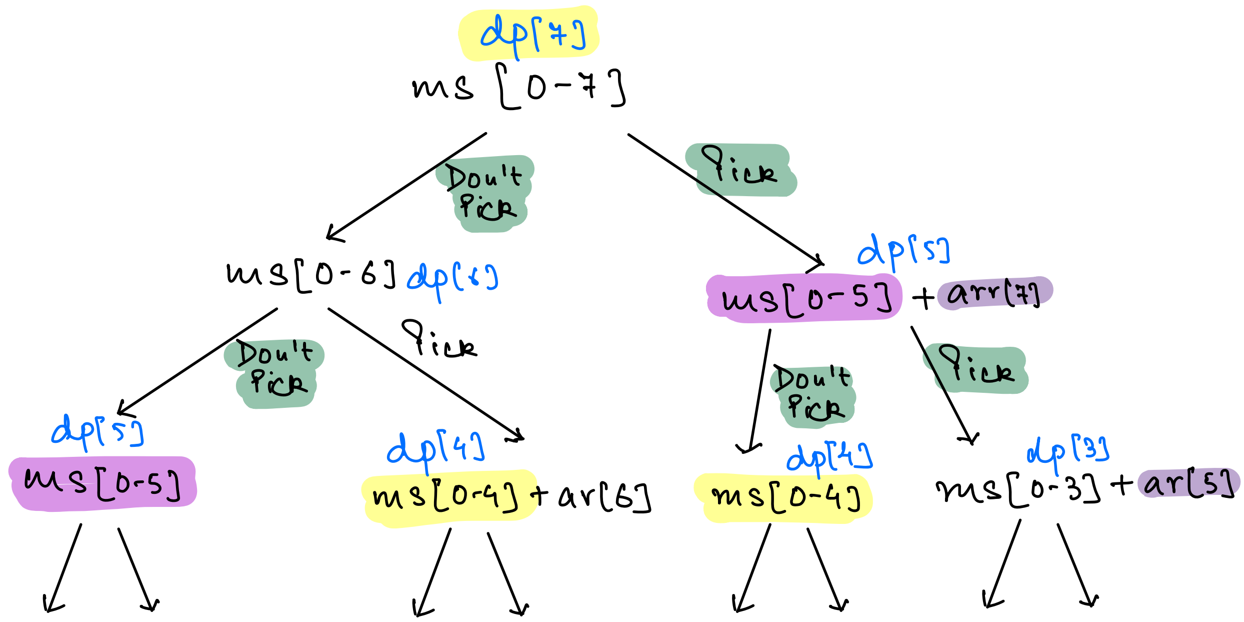
A : {9, 14, 3} → 14.

A : {9, 4, 13, 24} → 33

A : {13, 14, 2} → 15.

A[8] :

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



✓ → Optimal substructure
✓ → Overlapping subproblems.   } DP.

- dp[i] ⇒ Max subsequence sum from [0-i] } DP State.
- DP Expression

$$dp[i] = \max \begin{cases} arr[i] + dp[i-2] \rightarrow \text{Pick } i^{th} \text{ index} \\ dp[i-1] \rightarrow \text{Don't pick } i^{th} \text{ index} \end{cases}$$

- Base Case

$i = 0$: $dp[0] = arr[0]$

$i = 1$: $dp[1] = \max(arr[0], arr[1])$

↳ Max sub.seq. sum from [0-1].

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

if dp[i-2] < 0
then Don't
include.

TC: O(N)

SC: O(N) ⟶ O(1) {Todo}

A: {-2, -4, -1} N=3

0	1	2
-2	-2	-3

dp[2] = max(dp[1], -1+ dp[0])

(-4, -3)

$$dp[i] = \max \begin{cases} arr[i] & (\text{if } dp[i-2] < 0) \\ arr[i] + dp[i-2] \rightarrow \text{Pick } i^{th} \text{ index} \\ dp[i-1] \rightarrow \text{Don't pick } i^{th} \text{ index} \end{cases}$$

N=4 A: [-3, -6, -8, -2]

0	1	2	3
-3	-3	-3	-2

dp[2]: max { -8, -8+(-3), -3 }

: -3

dp[3]: max (-2 , -2+(-3) , -3)