DP by Kumar K

DP Part 1 KK (YT)

① Breaking the problem into smaller parts

$1 + 2 + 3 + 4 + 5$

$(1+2) + (3+4) + (5)$

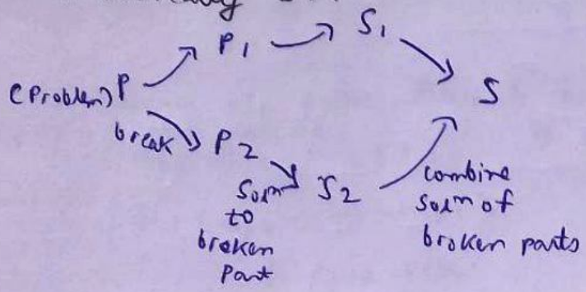
② Solving smaller parts

$(3) + (7) + (5)$

③ Grouping the smaller parts

$3 + 7 + 5 = 15$

This is basically DP.



Q

2	3	2	2	1	1	2
1	2	3	4	5	6	7

(1-based indexing)

q → queries in which index i is given

we need to tell Sum from (index 1 → index i)

q → 4 → (2+3+2+2)

q → 3 → (2+3+2)

q → 6 → (2+3+2+2+1+1)

q → 7 = (2+3+2+2+1+1+2)

(method 1) for loop and calculate sum from index 1 to i

```
int s=0;    for(j=1 ; j<=i ; j++)   // TC= O(n)
            {   s += a[j];
            }
            print(s);
```

for 1 Query it takes O(n) time

for q Query it takes O(n*Q) time (not efficient)

(method 2) DP method, Use following steps to solve any DP problem

① dp → [ ] // Declare empty dp array of size n

② dp[i] → meaning best answer to Question till index i

③ Calculate dp[1], dp[2], ... dp[N] → loop and formula i.e Recurrence relation.

(4) $dp[N]$ is our final answer.

1, 2, 1, 1, 3, 5, size = 6

(1) Make $dp[6]$

(2) $dp[1] = 1$ // the first element

$dp[2] = 1+2=3$ // sum of first two element

$dp[3] = 1+2+1=4$ // sum of first 3 element

$dp[i]$ = sum of first i numbers. so we can write $dp[4] = dp[3] + a[4]$

we already know $dp[3]$

we have made a formula now.

$dp[5] = dp[4] + a[5]$

so $\boxed{dp[i] = dp[i-1] + a[i]}$

now we can find sum according query $= i$ because our if $Q = i$ then it means $dp[i]$. as $Q = i$ means sum from $1 \rightarrow i$

$$\boxed{\text{// TC} = \underbrace{O(Q)}_{\text{for giving sum to query}} + \underbrace{O(N)}_{\text{for calculating } dp[1-N]}}$$

so TC $O(N*Q) > O(N) + O(Q)$ as DP is more efficient.