Dynamic Programming

Video – 85

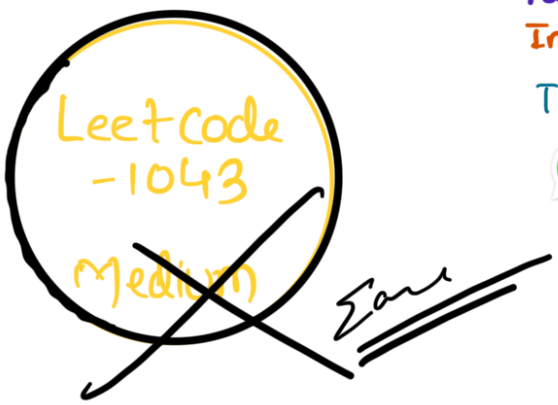
Note :- This playlist is only for explanation of Qns & solutions.

See my "DP Concepts & Qns" playlist for understanding DP from scratch...

Facebook
Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK



1043. Partition Array for Maximum Sum

Medium | Topics | Companies | Hint

Given an integer array `arr`, partition the array into (contiguous) subarrays of length **at most** `k`. After partitioning, each subarray has their values changed to become the maximum value of that subarray.

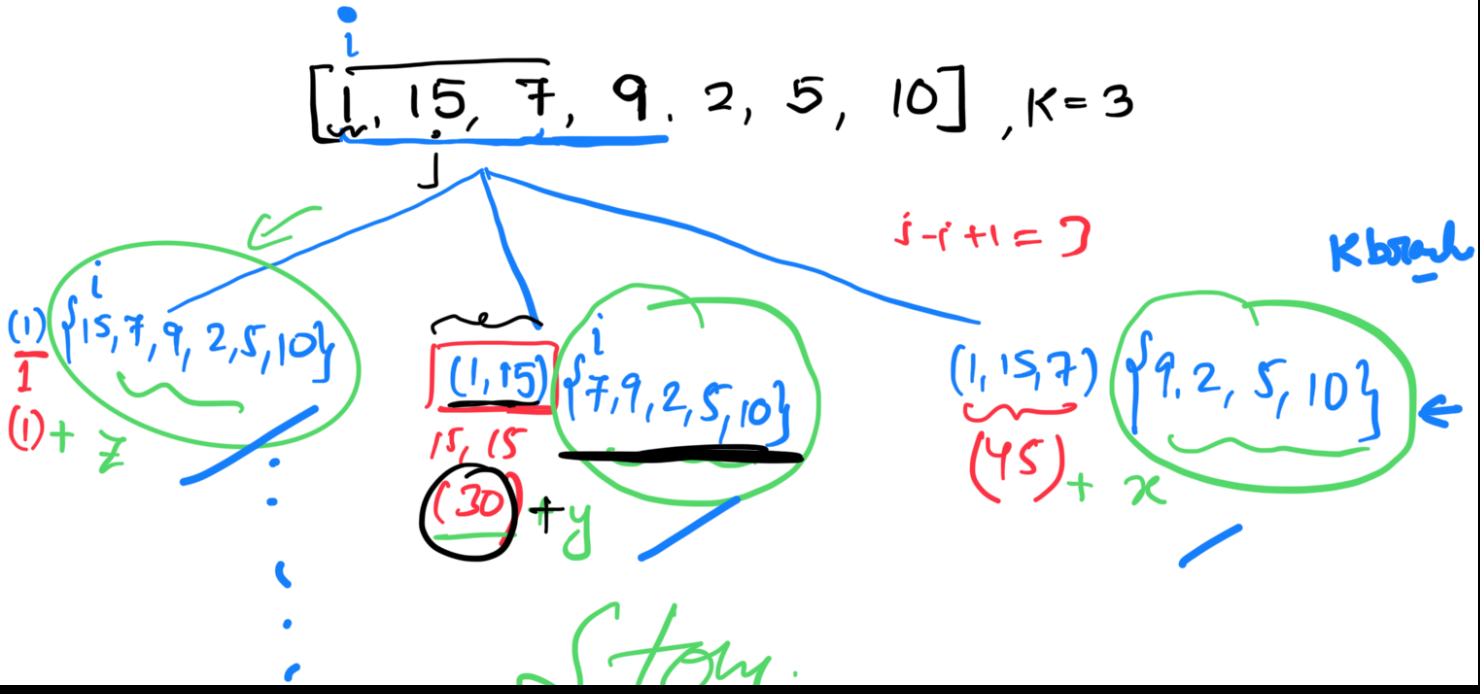
Return *the largest sum of the given array after partitioning. Test cases are generated so that the answer fits in a* ***32-bit*** *integer.*

Example:- [1, 15, 7, 9, 2, 5, 10] , K = 3

Output :- 84

Thought Process

↳ Why ~~DP~~ ???



Story To Code :-

```
int Solve (i, arr, K) {
    if (i >= arr.size()) return 0;

    result = 0;
    curr_max = -1;
    for (j=i ; j<n && j-i+1<=K ; j++) {
        curr_max = max(curr_max, arr[j]);
        result=max(result, ((j-i+1)*curr_max) + Solve(j+1, arr, K));
    }

    return result;
}
```

Time Complexity :-

without Memo :- $O(K^n)$ (Exp).

with Memo :- $O(n * K)$

Bottom up :-

nums = [1, 15, 7, 9, 2, 5, 10] , K=3

// t[i] = Maximum Sum by partitioning the array of size i

return t[n];

// t[n+1] ;

$i = 1$ ← only one element

$i = 2$ ← on the ch

size ← `for (i = 1 ; i<=n ; i++) {`

`cur_max = -1;`

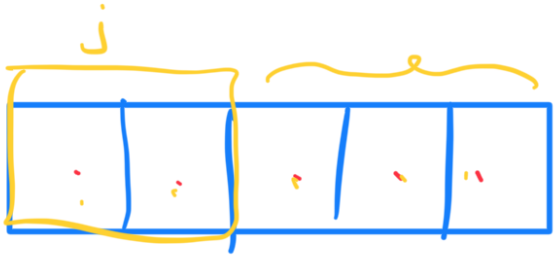
subarray size ← `for (j = 1 ; j<=K && i-j>=0; j++)`

⇒ `cur_max = max(cur_max, arr[i-j]);`

⇒ `t[i] = max(t[i], (cur_max * j) + t[i-j]);`

`}`

`ret t[n];`



size = 6 (i)

← with 2 by