**Qn Link :** <https://www.desiqna.in/17037/google-girl-hackathon-oa-2024-april-2024-set-1>

**Question Summary :**

* You are given an array of size “N” you can split the array into K parts such that the sum is maximum.
* The function F(l , r) = a[i] - a[i + 1] + a[i + 2] - a[i + 3] + …… + a[r]
* You need to calculate the each part sum by the function.

**Sample TC :**

[ 3 , 6 , -8]

K = 1 → 3 + 6 - 8 → 1

K = 2 → [3 , 6] [- 8] → (3 - 6) - 8 → -11

→ [3] [6 , -8] → 3 + (6 - - 8) → 17

K = 3 → (3 - 6 + (-8)) = -11

So the maximum sum is 17 and it is obtained when dividing the array into 1 , 2 parts

**Observation :**

There is no need to consider K focus only on maximising the sum .

* If all the numbers in the array is positive , then k = 1 , i.e sum of array is the answer.

**Array → [ + + + + + - - - - - + - - + + - + - - ]**

**Signs representing the sign of the number .**

**Case 1 :**

Whenever the positive number come , sum up to the answer. (k = 1)

We only need to focus on the negative number.

**Case 2 :**

When there is continuous negative number followed after a positive number , then the possibilities are ,

Case 2 (i) : + - - - - -

Case 2 (ii) :- - - - - -( taking all 6 into account)

Case2 (iii) : - - - - - - ( will split the 6 into further parts)

Either we can include it in the sum or exclude it

Example : [6 , -2 , -8]

Include → 6 - -2 - 8 = 0

Exclude → -2 + 8 = 6

So we hereby know that , we need to maintain two numbers , what if the number is positive and what if the number is negative.

Dp[i][+] = store when the number is positive

Dp[i][-] = store when the number is negative

**Sample Tc :**

[5 , 10 , -2 , -8 , -1]

, dp[0][+] = 5

, dp[0][-] = - infinity ( because we can’t make the first element as negative always)

, dp[1][+] = 5 + 10 → 15

, dp[1][-] = -10 + 5 → -5

, dp[2][+] = -2 + 15 → 13

, dp[2][-] = -(-2) + 15 → 17

, dp[3][+] = -8 + 17 → 11

, dp[3][-] = - (-8) + 13 → 21

, dp[4][+] = -1 + 21 → 23

, dp[4] [-] = -(-1) + 11 → 12

Sum =23

**MUST REMEMBER :**

When we choose an element as positive

* We can select the maximum of previous number

When we choose an element as negative then previous number must be positive

* Only include dp[i - 1][+]

**Recurrence Relation :**

* dp[i][+] = a[i] + Math.max(dp[i - 1][+], dp[i - 1][-])
* dp[i][-] = -a[i] + dp[i - 1][+]

**Base Case :**

, dp[1][0] = nums[0]

, dp[1][1] = Integer.Min\_VALUE]

, dp[2][0] = nums[1] + nums[0]

, dp[2][1] = nums[0] - nums[1]

**Code :**

class Solution {

public int maximumSum(int[] nums , int n) {

int [][] dp = new int[n + 1][2];

/\*

0 --> Positive sign

1 --> Negative sign

\*/

dp[1][0] = nums[0];

dp[1][1] = Integer.MIN\_VALUE;

dp[2][0] = nums[0] + nums[1];

dp[2][1] = nums[0] - nums[1];

for(int i = 3 ; i < n ; i++){

dp[i][0] = nums[i] + Math.max(dp[i - 1][0] , dp[i - 1][1]);

dp[i][1] = -1 \* nums[i] + dp[i - 1][0];

}

return Math.max(dp[n][0] , dp[n][1]);

}

}