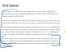


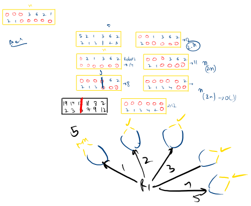
Grid Games

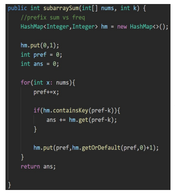
You are given a 0-indexed 2D array grid of size 2 x n, where grid[r][c] represents the number of points at position (r, c) on the matrix. Two robots are playing a game on this matrix.

Both robots initially start at (0, 0) and want to reach (1, n-1). Each robot may only move to the right ((r, c) to (r, c + 1)) or down ((r, c) to (r + 1, c)).

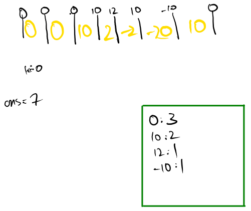
At the start of the game, the first robot moves from (0, 0) to (1, n-1), collecting all the points from the cells on its path. For all cells (r, c) traversed on the path, grid[r][c] is set to 0. Then, the second robot moves from (0, 0) to (1, n-1), collecting the points on its path. Note that their paths may intersect with one another.

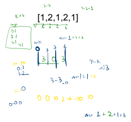
The first robot wants to minimize the number of points collected by the second robot. In contrast, the second robot wants to maximize the number of points it collects. If both robots play optimally, return the number of points collected by the second robot.

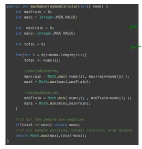




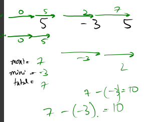
```java
public int subarraySum(int[] nums, int k) {
    //prefix sum vs freq
    HashMap<Integer,Integer> hm = new HashMap<>();

    hm.put(0,1);
    int pref = 0;
    int ans = 0;

    for(int x: nums){
        pref+=x;

        if(hm.containsKey(pref-k)){
            ans += hm.get(pref-k);
        }

        hm.put(pref,hm.getOrDefault(pref,0)+1);
    }
    return ans;

}
```

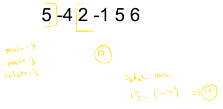


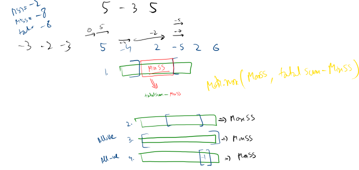




```java
public int maxSubarraySumCircular(int[] nums) {
    int maxTrain = 0;
    int maxi = Integer.MIN_VALUE;

    int  minTrain = 0;
    int mini= Integer.MAX_VALUE;

    int total = 0;

    for(int i = 0;i<nums.length;i++){
        total += nums[i];

        //maxSumSubarray
        maxTrain = Math.max( nums[i], maxTrain+nums[i] );
        maxi = Math.max(maxi,maxTrain);

        //minSumSubarray
        minTrain = Math.min( nums[i] , minTrain+nums[i] );
        mini = Math.min(mini,minTrain);
    }

    //if all the people are negative
    if(total == mini) return maxi;
    //if all people positive, normal solution, wrap around
    return Math.max(maxi,total-mini);
}
```

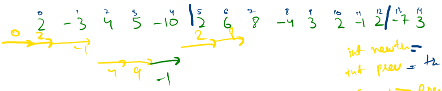






Kadane's

Sum subarray having maximum sum



maxi = 9

train 8

OS = 2

OE = 3

CS = 5

CE = 6

```
int newtrain =
int prev = train + arr[i]
if(newt > prev){
    CS = i
    train = newtrain
} else {
    train = Prevtrain
}
if(train > maxi){
    maxi = train
    OS = CS , OE = CE
}
CE++
```

```
int train = 0
int maxi = -∞

for(int i=0; i<n; i++){
    int newtrain = arr[i];
    int prevtrain = train + arr[i]
    train = Math.max(newtrain, Prevtrain);
    maxi = Math.max(Maxi, train)
}
return result.
```