Arrays – Medium Part 1.2.2

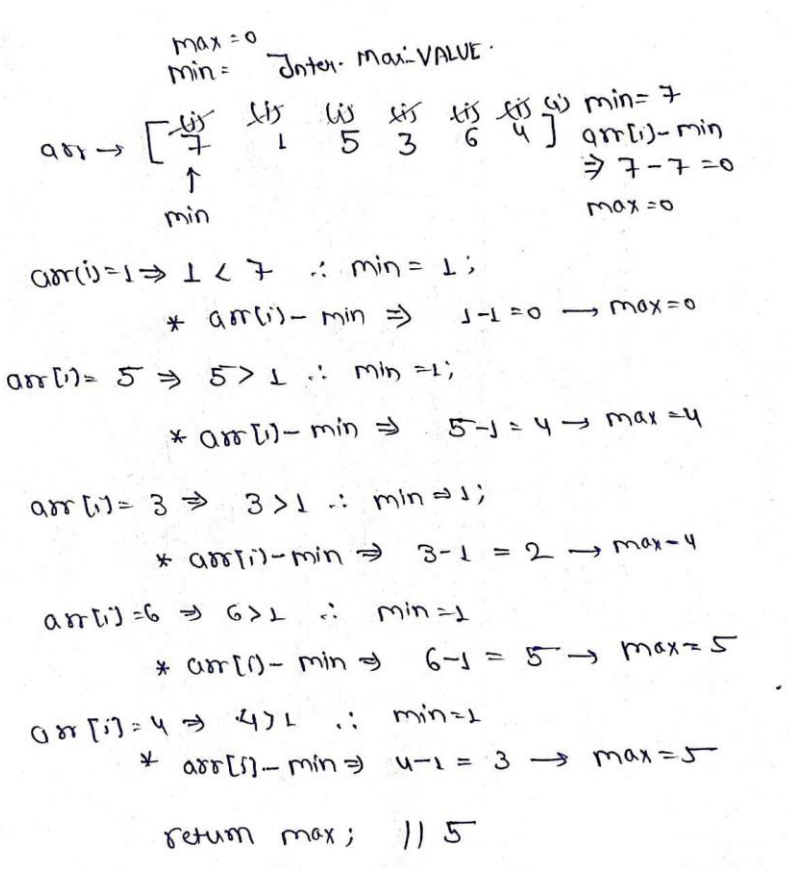
1. **Buy and Sell stock**

Approach 1:- Calculate all possible differences, keep a maxprofit variable keep updating it when you find a great difference. TC:-O(N2) SC:-O(1)

Approach 2:- Keep checking profit, it it becomes negative- update the buyingdate to the lesser no, keep a maxprofit variable and update it when you find a bigger profit. Tc:- O(N) SC:-O(1)

```
class Solution {
    public int maxProfit(int[] prices) {
        int max=0;
        int date=0;
        for(int i=0;i<prices.length;i++){
            int profit= prices[i]-prices[date];
            max=Math.max(profit,max);
            if(profit<0){
                date=i;
            }
        }
        return max;
    }
}
```

Approach 3:- Keep checking the minprice(min ele in array) and check profit for that. Have a maxprofit variable and check for ele after the minprice. TC:- O(N) SC:-O(1)



```
static int maxProfit(int[] arr) {
int maxPro = 0;
int minPrice = Integer.MAX_VALUE;
for (int i = 0; i < arr.length; i++) {
    minPrice = Math.min(minPrice, arr[i]);
    maxPro = Math.max(maxPro, arr[i] - minPrice);
}
return maxPro;
}
```

2. **Rearrange the elements in array by sign**

Approach 1:- make a positive array , make a negative array add ele to them in order. Then create an ans array and put positive negative ele in it one by one. TC:- O(N) SC:- O(N)

Approach 2:- create an ans array, put postitive ele in positive indices in ans array, and put negative ele in negative indices ans array. TC:- O(N) SC:-O(N)

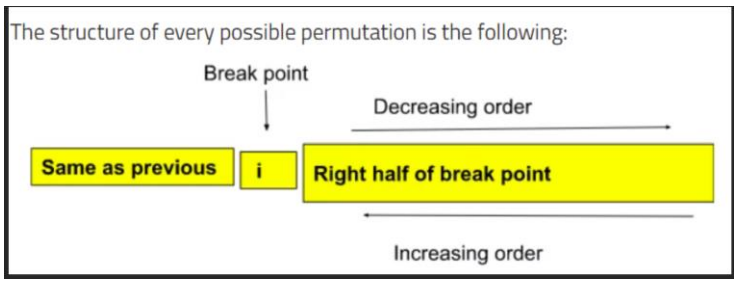
```
public class Solution {
    public int[] rearrangeArray(int[] nums) {
        int[] ans = new int[nums.length];
        int positiveIndex = 0, negativeIndex = 1;

        for (int i=0;i<nums.length;i++) {
            if (nums[i] > 0) {
                ans[positiveIndex] = nums[i];
                positiveIndex += 2;
            }
            if (nums[i] < 0) {
                ans[negativeIndex] = nums[i];
                negativeIndex += 2;
            }
        }
        return ans;
    }
}
```

3. **Next Permutation**

Approach 1 :- Find all the permutations then search the input and return the next one. TC:- O(N!*N) SC:- O(1)

Approach 2:- Find the break point(jidhar se ele ko swap karne se aur badi array mil jayegi). To find the break point we will the array backward and store the ele where the valu of arr[i]<arr[i+1]. Breakpoint nhi mila toh yahi max hai bhai reverse karke bhejdo)Agar break point mil gaya toh breaking pt se leke end tak min number doondho aur use swap kardo. Fir pure right half ko reverse kardo. TC:-



Dry run :-

2 1 5 4 3 0 0

Breakpt= 1

Next smallest no is 3 if we swap 1 and 3 we get 2 3 5 4 1 00

Reversing the right half we will get 2 3 0 0 1 4 5 which is indeed the answer

```
class Solution {
    public void nextPermutation(int[] nums) {
        int brkpt=-1;
        int nextbig=-1;
        for(int i=nums.length-2;i>=0;i--){
            if(nums[i]<nums[i+1]){
                brkpt=i;
                break;
            }
        }
        if(brkpt==-1){
            reverse(nums,0);
        }

        else{
            for(int i=nums.length-1;i>=0;i--){
                if(nums[i]>nums[brkpt]){
                    nextbig=i;
                    break;
                }
            }

            swap(nums,brkpt,nextbig);
            reverse(nums,brkpt+1);
        }
    }
    void swap(int[] nums,int i,int j){
        int temp=nums[i];
        nums[i]=nums[j];
        nums[j]=temp;
    }
    void reverse(int[] nums,int start){
        int i=start;
        int j=nums.length-1;
        while(i<j){
            swap(nums,i,j);
            i++;
            j--;
```