Minimize the Heights II

Difficulty: **Medium** Accuracy: **15.06%** Submissions: **739K+** Points: **4**
Average Time: **25m**

Given an array **arr[]** denoting heights of **n** towers and a positive integer **k**.

For **each** tower, you must perform **exactly one** of the following operations **exactly once**.

- **Increase** the height of the tower by **k**
- **Decrease** the height of the tower by **k**

Find out the **minimum** possible difference between the height of the shortest and tallest towers after you have modified each tower.

You can find a slight modification of the problem here.
Note: It is **compulsory** to increase or decrease the height by k for each tower. After the operation, the resultant array should **not** contain any **negative integers**.

Examples :

C++ (12) | Start Timer

```cpp
class Solution {
  public:
    int getMinDiff(vector<int> &arr, int k) {
        // code here
        int n= arr.size();
        sort(arr.begin(),arr.end());

        int h= arr[n-1]-arr[0];
        for(int i=1;i<n;i++)
        {
            if(arr[i]-k<0){continue;}
           int small=min(arr[0]+k,arr[i]-k);
           int large=max(arr[n-1]-k,arr[i-1]+k);
           h=min(h,large-small);
        }

        return h;
    }
};
```

Stock Buy and Sell – Max one Transaction Allowed

Difficulty: **Easy** Accuracy: **49.33%** Submissions: **110K+** Points: **2**

Average Time: **10m**

Given an array **prices[]** of length n, representing the prices of the stocks on different days. The task is to find the maximum profit possible by buying and selling the stocks on different days when at most one transaction is allowed. Here one transaction means 1 buy + 1 Sell. If it is not possible to make a profit then **return 0**.

Note: Stock must be bought before being sold.

Examples:

Input: prices[] = [7, 10, 1, 3, 6, 9, 2]
Output: 8
Explanation: You can buy the stock on day 2 at price = 1 and sell it on day 5 at price = 9. Hence, the profit is 8.

Input: prices[] = [7, 6, 4, 3, 1]
Output: 0

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int maximumProfit(vector<int> &prices) {
        // code here
        int n=prices.size();
        int small=prices[0];
        int profit=0;

        for(int i=0;i<n;i++)
        {
            profit=max(profit,prices[i]-small);
            small=min(small,prices[i]);
        }

        return profit;
    }
};
```

Custom Input Compile & Run Subm

Stock Buy and Sell – Multiple Transaction Allowed

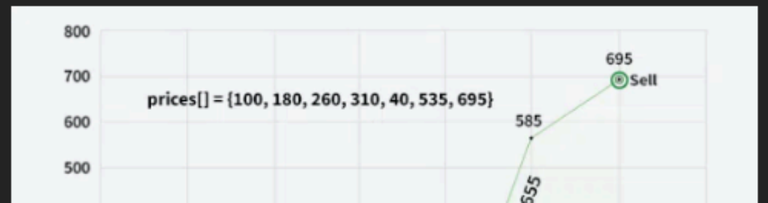
Difficulty: **Medium** Accuracy: **13.43%** Submissions: **185K+** Points: **4**

The cost of stock on each day is given in an array **price[]**. Each day you may decide to either buy or sell the stock i at **price[i]**, you can even buy and sell the stock on the same day. Find the **maximum profit** that you can get.

Note: A stock can only be sold if it has been bought previously and multiple stocks cannot be held on any given day.

Examples:

Input: prices[] = [100, 180, 260, 310, 40, 535, 695]
Output: 865
Explanation: Buy the stock on day 0 and sell it on day 3 => 310 – 100 = 210. Buy the stock on day 4 and sell it on day 6 => 695 – 40 = 655. Maximum Profit = 210 + 655 = 865.



C++ (12) | Start Timer

```cpp
class Solution {
  public:
    int maximumProfit(vector<int> &arr) {
        // code here
        int n =arr.size();
        int profit=0;
        for(int i=0;i<n-1;i++)
        {
            if(arr[i+1]>arr[i])
            {
                profit += arr[i+1]-arr[i];
            }
        }
        return profit;
    }
};
```

Custom Input | Compile & Run | Su

Smallest Positive Missing

Difficulty: **Medium** Accuracy: **25.13%** Submissions: **450K+** Points: **4**

You are given an integer array **arr[]**. Your task is to find the **smallest positive number** missing from the array.

Note: Positive number starts from 1. The array can have negative integers too.

Examples:

Input: arr[] = [2, -3, 4, 1, 1, 7]
Output: 3
Explanation: Smallest positive missing number is 3.

Input: arr[] = [5, 3, 2, 5, 1]
Output: 4
Explanation: Smallest positive missing number is 4.

Input: arr[] = [-8, 0, -1, -4, -3]
Output: 1

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int missingNumber(vector<int> &arr) {
        // code here
        int n=arr.size();
        sort(arr.begin(),arr.end());
        int s=1; //smallest positive

        for(int i=0;i<n;i++)
        {
            if(arr[i]==s)
            {
                s++;
            }
        }

        return s;

    }
};
```

Custom Input Compile & Run Sub