Description | Editorial | Solutions | Submissions ...

268. Missing Number

Solved

Easy | Topics | Companies

Given an array `nums` containing `n` distinct numbers in the range `[0, n]`, return *the only number in the range that is missing from the array.*

Example 1:

> **Input:** nums = [3,0,1]
> **Output:** 2
> **Explanation:** n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

Example 2:

> **Input:** nums = [0,1]
> **Output:** 2
> **Explanation:** n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

Code

C++ | Auto

```cpp
class Solution
{
    public:
    int missingNumber(vector<int>& nums)
    {
        int res=0;
        sort(nums.begin(), nums.end());
        for (int i=0;i<nums.size();i++)
        {
            if (i!=nums[i])
            {
                res=i;
                break;
            }
            else
            {
                res=i+1;
            }
        }
        return res;
    }
};
```

Saved to local

Description | Editorial | Solutions | Submissions

121. Best Time to Buy and Sell Stock

Solved

Easy | Topics | Companies

You are given an array `prices` where `prices[i]` is the price of a given stock on the i^{th} day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

Example 1:

> **Input:** prices = [7,1,5,3,6,4]
> **Output:** 5
> **Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
> Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Example 2:

> **Input:** prices = [7,6,4,3,1]

Code | Last Executed Input

C++ | Auto

```cpp
class Solution
{
    public:
    int maxProfit(vector<int>& prices)
    {
        if (prices.size()==0)
        {
            return 0;
        }
        int fprice=prices[0];
        int profit=0;
        for (int i=1;i<prices.size();i++)
        {
            if (prices[i]<fprice)
            {
                fprice=prices[i];
            }
            else
            {
                profit=max(profit, prices[i]-fprice);
            }
        }
        return profit;
    }
```

Saved to local

Ln 6, Col 26