# Stock Buy And Sell – Max One Transaction Allowed

| | |
|---|---|
| solved by | Senan |
| Platform | GeeksForGeeks |
| difficulty | Easy |
| tags | Logic |
| language | C++ |
| solved on | @22/11/2024 |
| link | https://www.geeksforgeeks.org/problems/buy-stock-2/1 |
| Completion | ☑ |

## Intuition

The problem is about maximizing profit from a stock's price data where you are allowed to buy and sell only once. The key observation is that for every price point, you need to know the minimum price encountered so far to calculate the profit from selling on that day.

## Approach

1. Initialize `mini` to the first element of the price array, representing the minimum price seen so far.
2. Initialize `profit` to 0, which will store the maximum profit observed.
3. Iterate through the price array starting from the second element:
    - If the current price is less than `mini`, update `mini`.
    - Otherwise, calculate the profit by subtracting `mini` from the current price and update `profit` if the new profit is larger.
4. Return the maximum profit observed.

## Complexity

### Time Complexity

- **O(n):** We traverse the `prices` array once, where `n` is the number of prices.

### Space Complexity

- **O(1):** Only a constant amount of extra space is used for variables like `mini` and `profit`.

## Code

```
class Solution {
  public:
    int maximumProfit(vector<int> &prices) {
        int mini = prices[0];
        int profit = 0;
        for(int i = 1; i < prices.size(); i++) {
```

```
            if(prices[i] < mini)
                mini = prices[i];
            else
                profit = max(profit, prices[i] - mini);
        }
        return profit;
    }
};
```