算法讨论班--------第十九期 2016/1/8 (89, 121, 123)

主讲人：韩冬奇

89. Gray Code

The gray code is a binary numeral system where two successive values differ in only one bit.

Given a non-negative integer *n* representing the total number of bits in the code, print the sequence of gray code. A gray code sequence must begin with 0.

For example, given *n* = 2, return `[0,1,3,2]`. Its gray code sequence is:

```
00 - 0
01 - 1
11 - 3
10 - 2
```

Note:

For a given *n*, a gray code sequence is not uniquely defined.

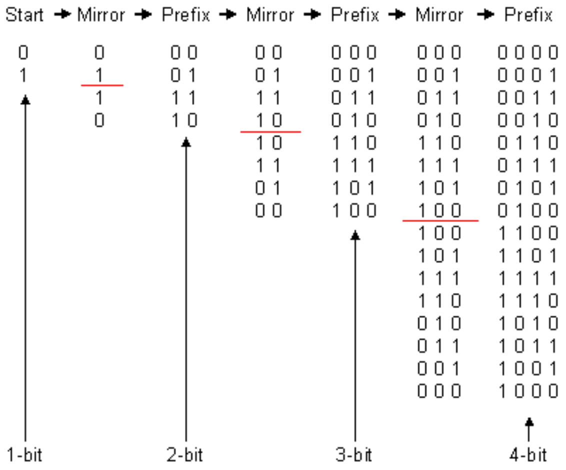
For example, `[0,2,3,1]` is also a valid gray code sequence according to the above definition.

For now, the judge is able to judge based on one instance of gray code sequence. Sorry about that.

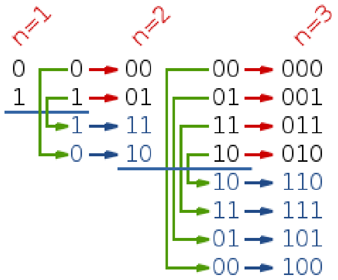
Tags: Backtracking

```
class Solution {
public:
    vector<int> grayCode(int n) {
        vector<int> re(1,0);
        int max_value = ((unsigned)0 - 1) >> (sizeof(int) * 8 - n);
        for (int i = 1;i <= max_value;i++) {
            re.push_back(i^(i >> 1));
        }
        return re;
    }
};
```

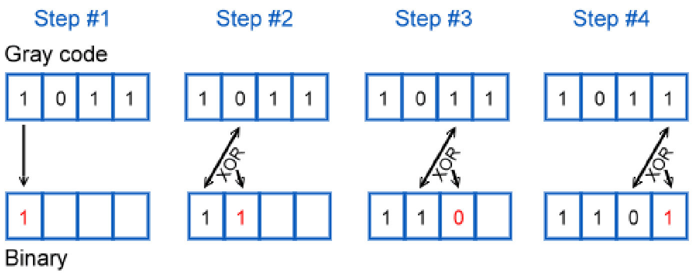
思路：

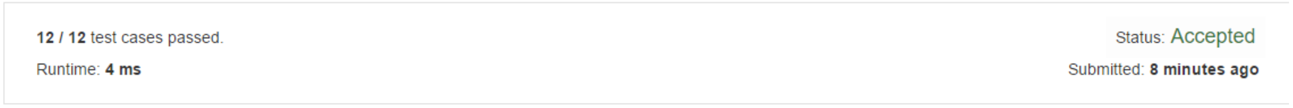


1. 除了最高位，格雷码的位元完全上下对称。如果我们生成了n位的格雷码，可将其顺序反转，然后在两个列表的前面分别补上0和1，就可生成n+1为的格雷码。(可用bitset实现)



1. 自然数*n*对应的二进制数为B,其格雷码为`G(i) = B(i+1) xor B(i)`。



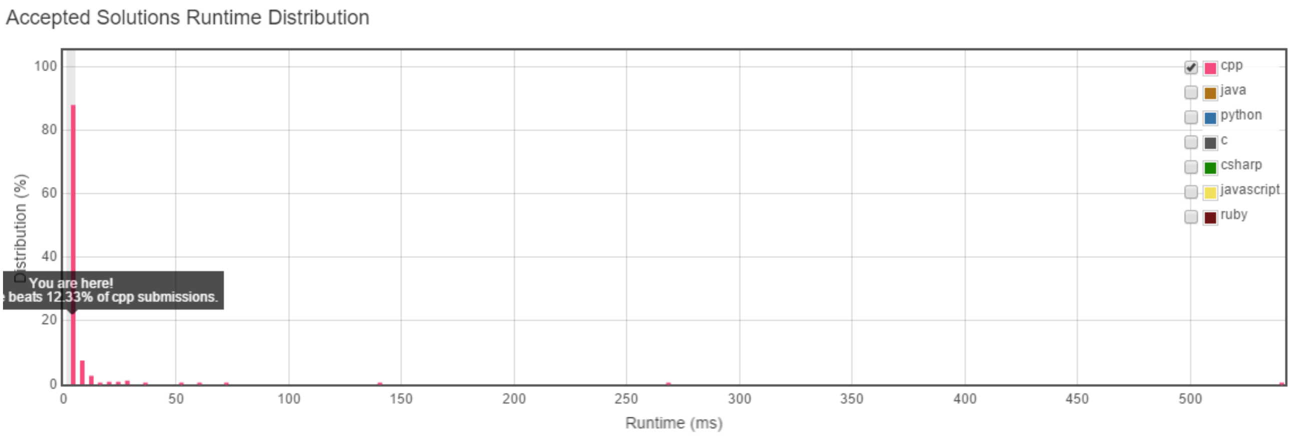


12 / 12 test cases passed.

Runtime: **4 ms**

Status: Accepted

Submitted: **8 minutes ago**



121. Best Time to Buy and Sell Stock

Say you have an array for which the ith element is the price of a given stock on day i.

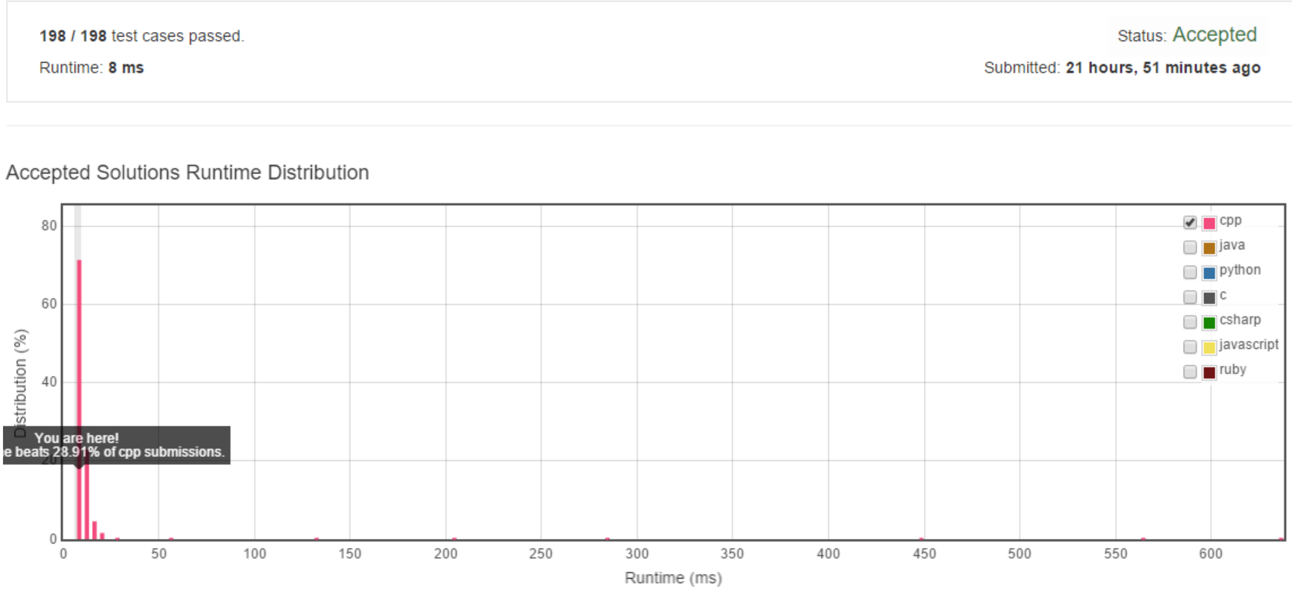
If you were only permitted to complete at most one transaction (ie, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Tags: Array, Dynamic Programming

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

思路：

设`dp[i]`为到第i天为止的最大收益，则显然有`dp[i+1] = max(dp[i], prices[i+1] - lowest)`。



198 / 198 test cases passed.

Runtime: 8 ms

Status: Accepted

Submitted: 21 hours, 51 minutes ago

Accepted Solutions Runtime Distribution

123. Best Time to Buy and Sell Stock III

Say you have an array for which the ith element is the price of a given stock on day i.

Design an algorithm to find the maximum profit. You may complete at most two transactions.

Note: You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

Tags: Array, Dynamic Programming

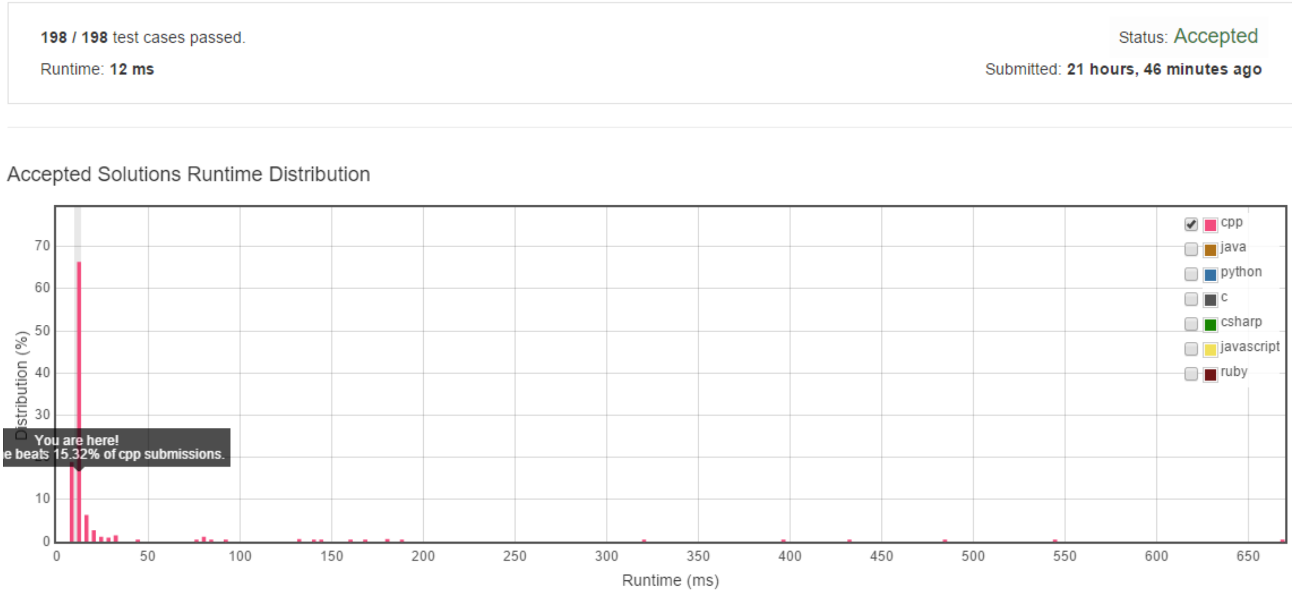
```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        if (prices.size() <= 1) return 0;
        int i;
        int re = 0;
        int *forward_dp = new int[prices.size()];
        int *back_dp = new int[prices.size()];
        forward_dp[0] = 0;
        back_dp[prices.size() - 1] = 0;

        int low = prices.front();
        for (i = 1; i < prices.size(); i++) {
            low = min(low, prices[i]);
            forward_dp[i] = max(forward_dp[i - 1], prices[i] - low);
        }

        int high = prices.back();
        for (i = prices.size() - 2; i >= 0; i--) {
            high = max(high, prices[i]);
            back_dp[i] = max(back_dp[i + 1], high - prices[i]);
        }

        for (i = 0;i < prices.size();i++) {
            re = max(re, forward_dp[i] + back_dp[i]);
        }
        return re;
    }
};
```

思路：

其最多允许进行两次交易，还包括进行一次交易或者不交易，后者也可以看作进行两次交易，如在当天进行一次或两次买入和卖出。

若发生了两次交易，则第二次买入需在第一次卖出以后进行，以`forward_dp[i]`表示第0天到第i天的最大收益，`back_dp[i]`表示第i天到最后一天的最大收益，显然有`dp[i] = forward_dp[i] + back_dp[i]`，其中`dp[i]`代表总收益。

由121题可知，`forward_dp[i+1] = max(forward_dp[i], prices[i+1] - lowest)`，则`back_dp[i] = max(back_dp[i+1], highest - prices[i]`。二者得到以后，即可从前向后扫描，计算`dp[i]`，找到最大值即可。



198 / 198 test cases passed.

Status: Accepted

Runtime: **12 ms**

Submitted: **21 hours, 46 minutes ago**

Accepted Solutions Runtime Distribution

You are here!
e beats 15.32% of cpp submissions.