

## 162. 买卖股票的最佳时机 II

### 162. 寻找峰值

难度 中等  178  收藏  分享  切换为英文  关注  反馈

峰值元素是指其值大于左右相邻值的元素。

给定一个输入数组 `nums`，其中 `nums[i] ≠ nums[i+1]`，找到峰值元素并返回其索引。

数组可能包含多个峰值，在这种情况下，返回任何一个峰值所在位置即可。

你可以假设 `nums[-1] = nums[n] = -∞`。

示例 1:

输入: `nums = [1,2,3,1]`  
输出: 2  
解释: 3 是峰值元素，你的函数应该返回其索引 2。

示例 2:

输入: `nums = [1,2,1,3,5,6,4]`  
输出: 1 或 5  
解释: 你的函数可以返回索引 1，其峰值元素为 2；  
或者返回索引 5，其峰值元素为 6。




说明:

你的解法应该是  $O(\log N)$  时间复杂度的。

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int l = 0, r = nums.size() - 1;
        while(l < r)
        {
            int mid = l + r >> 1;
            if(nums[mid] > nums[mid + 1]) r = mid;
            else l = mid + 1;
        }
        return r;
    }
};
```

## 164. 最大间距

## 164. 最大间距

难度 困难  139  收藏  分享  切换为英文  关注  反馈

给定一个无序的数组，找出数组在排序之后，相邻元素之间最大的差值。

如果数组元素个数小于 2，则返回 0。

示例 1:

输入: [3,6,9,1]

输出: 3

解释: 排序后的数组是 [1,3,6,9]，其中相邻元素 (3,6) 和 (6,9) 之间都存在最大差值 3。

示例 2:

输入: [10]

输出: 0

解释: 数组元素个数小于 2，因此返回 0。

说明:

- 你可以假设数组中所有元素都是非负整数，且数值在 32 位有符号整数范围内。
- 请尝试在线性时间复杂度和空间复杂度的条件下解决此问题。

```
class Solution {
public:
    int maximumGap(vector<int>& nums) {
        if (nums.size() < 2) return 0;
        int N = nums.size();
        int MaxNum = INT_MIN;
        int MinNum = INT_MAX;
        for (int i = 0; i < nums.size(); ++i) {
            MaxNum = max(nums[i], MaxNum);
            MinNum = min(nums[i], MinNum);
        }
        if (MaxNum == MinNum) return 0;
        double len = (MaxNum - MinNum) * 1.0 / (N - 1);

        int BucketSize = floor((MaxNum - MinNum) / len) + 1;

        vector<int> minbucket(BucketSize, INT_MAX);
        vector<int> maxbucket(BucketSize, INT_MIN);

        for (int i = 0; i < nums.size(); ++i) {
            int index = floor((nums[i] - MinNum) / len);
            minbucket[index] = min(minbucket[index], nums[i]);
            maxbucket[index] = max(maxbucket[index], nums[i]);
        }

        int delta = 0;
        int premax = maxbucket[0];
        for (int i = 1; i < BucketSize; ++i) {
            int curdelta = 0;
```

```

        if (minbucket[i] != INT_MAX) {
            delta = max(minbucket[i] - premax, delta);
            premax = maxbucket[i];
        }

    }
    return delta;
}
};

```

## 165. 比较版本号

### 165. 比较版本号

难度 **中等**   76   收藏   分享   切换为英文   关注   反馈

比较两个版本号 *version1* 和 *version2*。

如果 *version1* > *version2* 返回 1，如果 *version1* < *version2* 返回 -1，除此之外返回 0。

你可以假设版本字符串非空，并且只包含数字和 `.` 字符。

`.` 字符不代表小数点，而是用于分隔数字序列。

例如，`2.5` 不是“两个半”，也不是“差一半到三”，而是第二版中的第五个小版本。

你可以假设版本号的每一级的默认修订版号为 0。例如，版本号 `3.4` 的第一级（大版本）和第二级（小版本）修订号分别为 3 和 4。其第三级和第四级修订号均为 0。

#### 示例 1:

输入: *version1* = "0.1", *version2* = "1.1"  
输出: -1

#### 示例 2:

输入: *version1* = "1.0.1", *version2* = "1"  
输出: 1

#### 示例 3:

输入: *version1* = "7.5.2.4", *version2* = "7.5.3"  
输出: -1

#### 示例 4:

输入: *version1* = "1.01", *version2* = "1.001"  
输出: 0  
解释: 忽略前导零，“01”和“001”表示相同的数字“1”。

```

class Solution {
public:
    int compareVersion(string s1, string s2) {
        int i = 0, j = 0;
    }
};

```



```

while(i < s1.size() || j < s2.size())
{
    int x = i, y = j;
    while(x < s1.size() && s1[x] != '.') x ++;
    while(y < s2.size() && s2[y] != '.') y ++;
    int a = i == x ? 0 : atoi(s1.substr(i, x - i).c_str());
    int b = j == y ? 0 : atoi(s2.substr(j, y - j).c_str());
    if(a > b) return 1;
    if(a < b) return -1;
    i = x + 1;
    j = y + 1;
}
return 0;
};

```

## 166. 分数到小数

### 166. 分数到小数

难度 **中等**  126  收藏  分享  切换为英文  关注  反馈

给定两个整数，分别表示分数的分子 numerator 和分母 denominator，以字符串形式返回小数。

如果小数部分为循环小数，则将循环的部分括在括号内。

示例 1:

输入: numerator = 1, denominator = 2  
输出: "0.5"

示例 2:

输入: numerator = 2, denominator = 1  
输出: "2"

示例 3:

输入: numerator = 2, denominator = 3  
输出: "0.(6)"

```

class Solution {
public:
    string fractionToDecimal(int _n, int _d) {
        long long n = _n, d = _d;
        bool minus = false;
        if (n < 0) minus = !minus, n = -n;
        if (d < 0) minus = !minus, d = -d;
        string res = to_string(n / d);
        n %= d;
        if (!n)
        {
            if (minus && res != "0") return '-' + res;
            return res;
        }
    }
}

```

```

    res += '.';
    unordered_map<long long, int> hash;
    while (n)
    {
        if (hash[n])
        {
            res = res.substr(0, hash[n]) + '(' + res.substr(hash[n]) + ')';
            break;
        }
        else hash[n] = res.size();
        n *= 10;
        res += to_string(n / d);
        n %= d;
    }
    if (minus) res = '-' + res;
    return res;
}
};

```

## 167. 两数之和 II - 输入有序数组

### 167. 两数之和 II - 输入有序数组

难度 **简单**   282   收藏   分享   切换为英文   关注   反馈

给定一个已按照**升序排列**的有序数组，找到两个数使得它们相加之和等于目标数。

函数应该返回这两个下标值 index1 和 index2，其中 index1 必须小于 index2。

说明:

- 返回的下标值 (index1 和 index2) 不是从零开始的。
- 你可以假设每个输入只对应唯一的答案，而且你不可以重复使用相同的元素。

示例:

输入: numbers = [2, 7, 11, 15], target = 9

输出: [1,2]

解释: 2 与 7 之和等于目标数 9 。因此 index1 = 1, index2 = 2 。

```

class Solution {
public:
    vector<int> twoSum(vector<int>& numbers, int target) {
        for(int i = 0, j = numbers.size() - 1; i < numbers.size(); i++)
        {
            while(numbers[i] + numbers[j] > target) j--;
            if(numbers[i] + numbers[j] == target) return {i + 1, j + 1};
        }
        return {-1, -1};
    }
};

```

## 168. Excel表列名称

## 168. Excel表列名称

难度 简单

👍 210

❤ 收藏

🔗 分享

🌐 切换为英文

🔔 关注

📝 反馈

给定一个正整数，返回它在 Excel 表中相对应的列名称。

例如，

```
1 -> A
2 -> B
3 -> C
...
26 -> Z
27 -> AA
28 -> AB
...
```

示例 1:

```
输入: 1
输出: "A"
```

示例 2:

```
输入: 28
输出: "AB"
```






示例 3:

```
输入: 701
输出: "ZY"
```

```
class Solution {
public:
    string convertToTitle(int n) {
        string res;
        while (n) {
            res = char ((n - 1) % 26 + 'A') + res;
            n = (n - 1) / 26; //注意corner case: n=26
        }
        return res;
    }
};
```

## 169. 多数元素

## 169. 多数元素

难度 简单  566  收藏  分享  切换为英文  关注  反馈

给定一个大小为  $n$  的数组，找到其中的多数元素。多数元素是指在数组中出现次数大于  $\lfloor n/2 \rfloor$  的元素。

你可以假设数组是非空的，并且给定的数组总是存在多数元素。

示例 1:

输入: [3,2,3]  
输出: 3

示例 2:

输入: [2,2,1,1,1,2,2]  
输出: 2


```
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        unordered_map<int, int> hash;
        for (int i = 0; i < nums.size(); i++) {
            hash[nums[i]] += 1;
            if (hash[nums[i]] > nums.size() / 2)
                return nums[i];
        }
    }
};
/*
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int n = nums.size(), t;
        while (n > 1) {
            t = 0;
            for (int i = 0; i < n; i += 2) {
                if (i == n - 1 || nums[i] == nums[i + 1])
                    nums[t++] = nums[i];
            }
            n = t;
        }
        return nums[0];
    }
};

class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int cnt = 0, candidate;
        for (int i = 0; i < nums.size(); i++) {
            if (cnt == 0)
                candidate = nums[i];
            if (candidate == nums[i])
                cnt++;
            else
```

```
        cnt--;  
    }  
    return candidate;  
}  
};  
*/
```

## 171. Excel表列序号

### 171. Excel表列序号

难度 简单  133  收藏  分享  切换为英文  关注  反馈

给定一个Excel表格中的列名称，返回其相应的列序号。

例如，

```
A -> 1  
B -> 2  
C -> 3  
...  
Z -> 26  
AA -> 27  
AB -> 28  
...
```

示例 1:

```
输入: "A"  
输出: 1
```

示例 2:

```
输入: "AB"  
输出: 28
```

示例 3:

```
输入: "ZY"  
输出: 701
```

致谢:

特别感谢 @ts 添加此问题并创建所有测试用例。

```
class Solution {  
public:  
    int titleToNumber(string s) {  
        int num = 0;  
        for (int i = 0; i < s.length(); ++i) {  
            num = 26 * num + (s[i] - 'A' + 1); //注意'A'不是0而是1  
        }  
        return num;  
    }  
};
```



## 172. 阶乘后的零

### 172. 阶乘后的零

难度 简单

👍 263

❤ 收藏

🔗 分享

🌐 切换为英文

🔔 关注

📝 反馈

给定一个整数  $n$ ，返回  $n!$  结果尾数中零的数量。

示例 1:

输入: 3

输出: 0

解释:  $3! = 6$ ，尾数中没有零。

示例 2:

输入: 5

输出: 1

解释:  $5! = 120$ ，尾数中有 1 个零。

说明: 你算法的时间复杂度应为  $O(\log n)$ 。

```
class Solution {
public:
    int trailingZeroes(int n) {
        return n < 5 ? 0 : n/5 + trailingZeroes(n/5); //递归
    }
};
```

## 173. 二叉搜索树迭代器

### 173. 二叉搜索树迭代器

难度 中等

159

收藏

分享

切换为英文

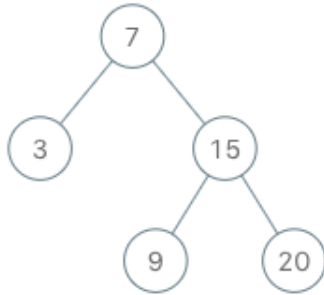
关注

反馈

实现一个二叉搜索树迭代器。你将使用二叉搜索树的根节点初始化迭代器。

调用 `next()` 将返回二叉搜索树中的下一个最小的数。

示例:



```
BSTIterator iterator = new BSTIterator(root);
iterator.next();    // 返回 3
iterator.next();    // 返回 7
iterator.hasNext(); // 返回 true
iterator.next();    // 返回 9
iterator.hasNext(); // 返回 true
iterator.next();    // 返回 15
iterator.hasNext(); // 返回 true
iterator.next();    // 返回 20
iterator.hasNext(); // 返回 false
```

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class BSTIterator {
public:
    stack<TreeNode*> stk;
    BSTIterator(TreeNode* root) {
        while(root)
        {
            stk.push(root);
            root = root->left;
        }
    }

    /** @return the next smallest number */
    int next() {
        auto p = stk.top();
        stk.pop();
```

```

    int res = p->val;
    p = p->right;
    while(p)
    {
        stk.push(p);
        p = p->left;
    }

    return res;
}

/** @return whether we have a next smallest number */
bool hasNext() {
    return !stk.empty();
}

};




/**
 * Your BSTIterator object will be instantiated and called as such:
 * BSTIterator* obj = new BSTIterator(root);
 * int param_1 = obj->next();
 * bool param_2 = obj->hasNext();
 */

```

## 174. 地下城游戏

---

## 174. 地下城游戏

难度 困难  192  收藏  分享  切换为英文  关注  反馈

一些恶魔抓住了公主 (P) 并将她关在了地下城的右下角。地下城是由  $M \times N$  个房间组成的二维网格。我们英勇的骑士 (K) 最初被安置在左上角的房间里，他必须穿过地下城并通过对抗恶魔来拯救公主。

骑士的初始健康点数为一个正整数。如果他的健康点数在某一时刻降至 0 或以下，他会立即死亡。

有些房间由恶魔守卫，因此骑士在进入这些房间时会失去健康点数（若房间里的值为负整数，则表示骑士将损失健康点数）；其他房间要么是空的（房间里的值为 0），要么包含增加骑士健康点数的魔法球（若房间里的值为正整数，则表示骑士将增加健康点数）。

为了尽快到达公主，骑士决定每次只向右或向下移动一步。

编写一个函数来计算确保骑士能够拯救到公主所需的最低初始健康点数。

例如，考虑到如下布局的地下城，如果骑士遵循最佳路径 右 -> 右 -> 下 -> 下，则骑士的初始健康点数至少为 7。

-2 (K)	-3	3
-5	-10	1
10	30	-5 (P)

说明:

- 骑士的健康点数没有上限。
- 任何房间都可能对骑士的健康点数造成威胁，也可能增加骑士的健康点数，包括骑士进入的左上角房间以及公主被监禁的右下角房间。

```
class Solution {
public:
    bool check(long long initial, vector<vector<int>>& dungeon) {
        int n = dungeon.size(), m = dungeon[0].size();
        vector<vector<long long>> f(n, vector<long long>(m, 0));
        f[0][0] = initial + dungeon[0][0];
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++) {
                if (i > 0 && f[i - 1][j] > 0)
                    f[i][j] = max(f[i][j], f[i - 1][j] + dungeon[i][j]);
                if (j > 0 && f[i][j - 1] > 0)
                    f[i][j] = max(f[i][j], f[i][j - 1] + dungeon[i][j]);
            }
        return f[n - 1][m - 1] > 0;
    }

    int calculateMinimumHP(vector<vector<int>>& dungeon) {
        long long l = 1, r = 100000000000001;
        while (l < r) {
            int mid = (l + r) >> 1;
            if (check(mid, dungeon)) r = mid;
            else l = mid + 1;
        }
        return l;
    }
}
```

```
};
/*
class Solution {
public:
    int calculateMinimumHP(vector<vector<int>>& dungeon) {
        int n = dungeon.size(), m = dungeon[0].size();
        vector<vector<long long>> f(n, vector<long long>(m, 10000000000001));
        f[n - 1][m - 1] = max(-dungeon[n - 1][m - 1], 0) + 1;
        for (int i = n - 1; i >= 0; i--)
            for (int j = m - 1; j >= 0; j--) {
                if (i < n - 1)
                    f[i][j] = min(f[i][j], f[i + 1][j] - dungeon[i][j]);
                if (j < m - 1)
                    f[i][j] = min(f[i][j], f[i][j + 1] - dungeon[i][j]);
                f[i][j] = f[i][j] <= 0 ? 1 : f[i][j];
            }
        return f[0][0];
    }
};
*/
```

## 179. 最大数

### 179. 最大数

难度 **中等**   269   收藏   分享   切换为英文   关注   反馈

给定一组非负整数，重新排列它们的顺序使之组成一个最大的整数。

示例 1:

输入: [10,2]  
输出: 210

示例 2:

输入: [3,30,34,5,9]  
输出: 9534330

说明: 输出结果可能非常大，所以你需要返回一个字符串而不是整数。

```
class Solution {
public:
    static bool cmp(int x, int y) {
        string sx = to_string(x), sy = to_string(y);
        return sx + sy > sy + sx;
    }
    string largestNumber(vector<int>& nums) {
        sort(nums.begin(), nums.end(), cmp);
        string ans = "";
        for (int i = 0; i < nums.size(); i++)
            ans += to_string(nums[i]);

        for (int i = 0; i < ans.length(); i++)
            if (i == ans.length() - 1 || ans[i] != '0')
                break;
        return ans;
    }
};
```

```
        return ans.substr(i, ans.length() - i);  
    return ans;  
}  
};
```