剑指 Offer II 035. 最小时间差

```cpp
class Solution {
public:
    int func(string &s) {
        return (s[0] - '0') * 10 * 60 + (s[1] - '0') * 60 + (s[3] - '0') * 10 + s[4] - '0';
    }
    int findMinDifference(vector<string>& timePoints) {
        sort(timePoints.begin(), timePoints.end());
        int ans = func(timePoints[0]) + 24 * 60 - func(timePoints[timePoints.size() - 1]);
        for (int i = 1; i < timePoints.size(); i++) {
            ans = min(ans, func(timePoints[i]) - func(timePoints[i - 1]));
        }
        return ans;
    }
};
```

剑指 Offer II 036. 后缀表达式

```cpp
class Solution {
public:
    int is_number(string &s) {
        return s.size() > 1 || (s[0] >= '0' && s[0] <= '9');
    }
    int evalRPN(vector<string>& tokens) {
        stack<int> sta;
        for (int i = 0; i < tokens.size(); i++) {
            if (is_number(tokens[i])) {
                sta.push(stoi(tokens[i]));
            } else {
                int b = sta.top();
                sta.pop();
                int a = sta.top();
                sta.pop();
                int c;
                if (tokens[i][0] == '+') c = a + b;
                else if (tokens[i][0] == '-') c = a - b;
                else if (tokens[i][0] == '*') c = a * b;
                else c = a / b;
                sta.push(c);
            }
        }
```

```
        return sta.top();
    }
};
```

剑指 Offer II 034. 外星语言是否排序

```
class Solution {
public:
    unordered_map<char, char> m;
    int func(string &a, string &b) {
        int n = min(a.size(), b.size());
        for (int i = 0; i < n; i++) {
            if (m[a[i]] < m[b[i]]) {
                return 0;
            }
            if (m[a[i]] > m[b[i]]) {
                return 1;
            }
        }
        if (a.size() > b.size()) {
            return 1;
        }
        return 0;
    }
    bool isAlienSorted(vector<string>& words, string order) {
        for (int i = 0; i < 26; i++) {
            m[order[i]] = i + 'a';
        }
        for (int i = 1; i < words.size(); i++) {
            if (func(words[i - 1], words[i])) {
                return false;
            }
        }
        return true;
    }
};
```

剑指 Offer II 033. 变位词组

```
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        vector<vector<string> > ans;
        unordered_map<string, int> m;
```

```
        for (auto &s : strs) {
            string temp = s;
            sort(temp.begin(), temp.end());
            if (m.count(temp) == 1) {
                ans[m[temp]].push_back(s);
            } else {
                m[temp] = ans.size();
                ans.push_back(vector<string>{s});
            }
        }
        return ans;
    }
};
```

剑指 Offer II 012. 左右两边子数组的和相等

```
class Solution {
public:
    int pivotIndex(vector<int>& nums) {
        int n = nums.size();
        vector<int> sum(n + 1);
        for (int i = 1; i <= n; i++) {
            sum[i] = sum[i - 1] + nums[i - 1];
        }
        for (int i = 0; i < n; i++) {
            if (sum[i] == sum[n] - sum[i + 1]) {
                return i;
            }
        }
        return -1;
    }
};
```

剑指 Offer II 010. 和为 k 的子数组

```
class Solution {
public:
    int subarraySum(vector<int>& nums, int k) {
        int n = nums.size();
        vector<int> sum(n + 1);
        for (int i = 1; i <= n; i++) {
            sum[i] = sum[i - 1] + nums[i - 1];
        }
        unordered_map<int, int> m;
```

```
        int ans= 0;
        for (int i = 0; i <= n; i++) {
            int temp = sum[i] - k;
            if (m.count(temp)) {
                ans += m[temp];
            }
            m[sum[i]]++;
        }
        return ans;
    }
};
```

剑指 Offer 56 - I. 数组中数字出现的次数

```
class Solution {
public:
    vector<int> singleNumbers(vector<int>& nums) {
        int t = 0;
        for (auto x : nums) {
            t ^= x;
        }
        int ind = 0;
        while ((t & (1 << ind)) == 0) {
            ind++;
        }
        int a = 0, b = 0;
        for (auto x : nums) {
            if ((x & (1 << ind)) == 0) {
                a ^= x;
            } else {
                b ^= x;
            }
        }
        return vector<int>{a, b};
    }
};
```

剑指 Offer 56 - II. 数组中数字出现的次数 II

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int bit2[32] = {0};
        for (auto x : nums) {
```

```
            for (int i = 0; x; i++) {
                if ((x & 1) != 0) {
                    bit2[i]++;
                }
                x >>= 1;
            }
        }
        int ans = 0;
        for (int i = 0; i <= 31; i++) {
            bit2[i] %= 3;
            if (bit2[i] != 0) {
                ans += 1 << i;
            }
        }
        return ans;
    }
};
```

剑指 Offer II 003. 前 n 个数字二进制中 1 的个数

```
class Solution {
public:
    vector<int> countBits(int n) {
        vector<int> ans(n + 1);
        for (int i = 1; i <= n; i++) {
            if (i & 1) {
                ans[i] = ans[i - 1] + 1;
            } else {
                ans[i] = ans[i / 2];
            }
        }
        return ans;
    }
};
```

剑指 Offer II 005. 单词长度的最大乘积

```
class Solution {
public:
    int maxProduct(vector<string>& words) {
        int n = words.size();
        vector<int> bit2(n);
        for (int i = 0; i < n; i++) {
            for (auto c : words[i]) {
```

```
                bit2[i] |= (1 << (c - 'a'));
            }
        }
        int ans = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if ((bit2[i] & bit2[j]) == 0) {
                    ans = max(ans, (int)(words[i].size() * words[j].size()));
                }
            }
        }
        return ans;
    }
};
```