https://leetcode.com/discuss/interview-question/344650/Amazon-Online-Assessment-Questions

目录

Find Pair With Given Sum

https://leetcode.com/problems/two-sum/ 注意题干细节，边界条件

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int>temp;
        int out_index = 0;
        std::vector<int> output (2,0);
        for(int i=0;i<nums.size();i++){
            if(temp.count(target-nums[i])){
                if(i > out_index){
                    out_index = i;
                    output[0] = temp[target-nums[i]];
                    output[1] = i;
                }
            }
            temp[nums[i]] = i;
        }
        if(out_index == 0){
            return {};
        }
        else{
            return output;
        }
    }
};
```

Movies on Flight

双指针大法好

```
vector<int> MoviesOnFlight(vector<int> movieDurations, int d){
    d = d-30;
    vector<int> newM = movieDurations;
    sort(begin(newM), end(newM));
    int maximum = 0;
    int sum = 0;
    vector<int> ans(2,0);
    for (int i=0;i<newM.size(); i++){
        for (int j=newM.size()-1;j>i; j--){
            sum = newM[i]+ newM[j];
            if (sum <= d){
                if (sum > maximum){
                    maximum = newM[i] + newM[j];
                    auto it1 = find(movieDurations.begin(), movieDurations.end(),
newM[i]);
                    ans[0] = distance(movieDurations.begin(), it1);
                    auto it2 = find(movieDurations.begin()+ans[0]+1, movieDurations.end(),
newM[j]);
                    ans[1] = distance(movieDurations.begin()+ans[0]+1, it2)+ans[0]+1;
                }
            }
            else
                continue;
        }
    }
    sort(begin(ans), end(ans));
    return ans;
}
```

Substrings with exactly K distinct chars

https://leetcode.com/problems/two-sum/ 滑动窗口做结

```
class Solution {
public:
    int subarraysWithKDistinct(vector<int>& A, int K) {
        int res = 0;
        vector<int> m(A.size() + 1);
        for(int i = 0, j = 0, prefix = 0, cnt = 0; i < A.size(); ++i)
        {
            if (m[A[i]]++ == 0) ++cnt;
            if (cnt > K) --m[A[j++]], --cnt, prefix = 0;
            while (m[A[j]] > 1) ++prefix, --m[A[j++]];
            if (cnt == K) res += prefix + 1;
        }
        return res;
    }
};
```

Path With Maximum Score/Max Of Min Altitudes

DP 注意细看 conor case

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int maxScore(vector<vector<int> >& grid){
    if (grid.empty() || (grid.size() == 1 && grid[0].size()==1)) return 0;
    if (grid.size()==1){
        for (unsigned int i = 2; i < grid[0].size()-1; i++){
            grid[0][i] = min(grid[0][i], grid[0][i-1]);
        }
        return grid[grid.size()-1][grid[0].size()-2];
    }
    if (grid[0].size()==1){
        for (unsigned int i = 2; i < grid.size()-1; i++){
            grid[i][0] = min(grid[i][0], grid[i-1][0]);
        }
        return grid[grid.size()-2][grid[0].size()-1];
    }

    for (unsigned int i = 2; i < grid[0].size(); i++){
        grid[0][i] = min(grid[0][i], grid[0][i-1]);
    }
    for (unsigned int i = 2; i < grid.size(); i++){
        grid[i][0] = min(grid[i][0], grid[i-1][0]);
    }

    for (unsigned int i = 1; i < grid.size(); i++){
        for (unsigned int j = 1; j < grid[0].size(); j++){
            if(i == grid.size()-1 && j == grid[0].size()-1) grid[i][j] = max(grid[i-1][j],
grid[i][j-1]);
            else grid[i][j] = max(min(grid[i-1][j], grid[i][j]),min(grid[i][j-1],
grid[i][j]));
        }
    }
    return grid[grid.size()-1][grid[0].size()-1];
}
```

Longest Palindromic Substring

https://leetcode.com/problems/longest-palindromic-substring/ 马拉车算法

```
class Solution {
public:
    string longestPalindrome(string s) {
         string t ="$#";
        for (int i = 0; i < s.size(); ++i) {
            t += s[i];
            t += '#';
        }
        int p[t.size()] = {0}, id = 0, mx = 0, resId = 0, resMx = 0;
        for (int i = 1; i < t.size(); ++i) {
            p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
            while (t[i + p[i]] == t[i - p[i]]) ++p[i];
            if (mx < i + p[i]) {
                mx = i + p[i];
                id = i;
            }
            if (resMx < p[i]) {
                resMx = p[i];
                resId = i;
            }
        }
        return s.substr((resId - resMx) / 2, resMx - 1);
    }
};
```

Substrings of size K with K distinct chars

重复就重头来

```
vector<string> kSubstring(string s, int k) {
    vector<string> result = {};
    if (s.empty() || k == 0) return result;
    unordered_map<char, int> letter;
    int start;
    for(int i=0;i<s.size();i++){
        if(letter.count(s[i]) && letter.at(s[i]) > start) start = letter[s[i]]+1;
        letter[s[i]] = i;
        if(i-start+1 == k && find(result.begin(),result.end(),s.substr(start, k)) == 
result.end()){
            result.push_back(s.substr(start, k));
            start += 1;
        }
    }
    return result;
}
```

Most Common Word

https://leetcode.com/problems/most-common-word/

```
class Solution {
public:
    string mostCommonWord(string paragraph, vector<string>& banned) {
        unordered_set<string> ban(banned.begin(), banned.end());
        unordered_map<string, int> strCnt;
        int mx = 0;
        for (auto &c : paragraph) c = isalpha(c) ? tolower(c) : ' ';
        istringstream iss(paragraph);
        string t = "", res = "";
        while (iss >> t) {
            if (!ban.count(t) && ++strCnt[t] > mx) {
                mx = strCnt[t];
                res = t;
            }
        }
        return res;
    }
};
```

K Closest Points to Origin

https://leetcode.com/problems/k-closest-points-to-origin/

```
class Solution {
public:
  vector<vector<int>> kClosest(vector<vector<int>>& points, int K) {
    partial_sort(points.begin(), points.begin() + K, points.end(), [](vector<int>&
p, vector<int>& q) {
        return p[0] * p[0] + p[1] * p[1] < q[0] * q[0] + q[1] * q[1];
    });
    return vector<vector<int>>(points.begin(), points.begin() + K);
  }
};
```

Merge Two Sorted Lists

https://leetcode.com/problems/merge-two-sorted-lists/

```
class Solution {
public:
  ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
    if (!l1) return l2;
    if (!l2) return l1;
    ListNode *head = l1->val < l2->val ? l1 : l2;
    ListNode *nonhead = l1->val < l2->val ? l2 : l1;
    head->next = mergeTwoLists(head->next, nonhead);
    return head;
  }
};
```

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        ListNode *dummy = new ListNode(), *cur = dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                cur->next = l1;
                l1 = l1->next;
            } else {
                cur->next = l2;
                l2 = l2->next;
            }
            cur = cur->next;
        }
        cur->next = l1 ? l1 : l2;
        return dummy->next;
    }
};
```