```
**********21. Merge Two Sorted Lists**********

class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        if(!l1) return l2;
        if(!l2) return l1;
        if(l1->val < l2->val){
            l1->next = mergeTwoLists(l1->next, l2);
            return l1;
        }
        else{
            l2->next = mergeTwoLists(l1, l2->next);
            return l2;
        }
    }
}
***************53. Maximum Subarray***************************
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int res = INT_MIN, cur_sum = 0;
        for(auto i : nums){
            cur_sum += i;
            res = max(res, cur_sum);
            if(cur_sum < 0){
                cur_sum = 0;
            }
        }
        return res;
    }
};
******************Merge k Sorted Lists**********************
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        multimap<int, ListNode*> mp;
        for(auto i:lists){
            if(i)
                mp.insert(make_pair(i->val, i));
        }
        ListNode* dummy = new ListNode(0);
        ListNode *tmp = dummy;
        while(!mp.empty()){
            ListNode *cur_min = mp.begin()->second;
            tmp->next = new ListNode(cur_min->val);
            tmp = tmp->next;
            mp.erase(mp.begin());
            if(cur_min->next){
                cur_min = cur_min->next;
                mp.insert(make_pair(cur_min->val, cur_min));
            }
        }
        return dummy->next;
    }
```

```
};

******************198. House Robber************************
class Solution {
public:
    int rob(vector<int>& nums) {
    // x    x    x    x    x
    // |    |    |    |
    //bk2  bk1  cur  num
    //     |    |
    //    bk2  bk1  cur

        int back2 = 0, back1 = 0, cur = 0;
        for( int num :nums ){
            cur = max(num + back2, back1);
            back2 = back1;
            back1 = cur;
        }
        return cur;
    }
};
******************238. Product of Array Except Self************************
class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {

        vector<int> res(nums.size(), 1);

        //[2,3,4,5]
        //[1,1,1,1]
        //[1,       2,  2*3,  2*3*4]
        //[3*4*5, 4*5,  5,    1]
        for(int i = 1; i < nums.size(); i++){
            res[i] = res[i-1] * nums[i-1];
        }
        int tmp = 1;
        for(int i = nums.size()-1; i >= 0; i--){
            res[i] *= tmp;
            tmp *= nums[i];
        }
        return res;
    }
};

********************149. Max Points on a Line*********************
class Solution {
public:
    //建立map<double(斜率), int（计数）> mp的问题是：需要double很高的精度
    //比如94911151.0/94911150.0 == 94911152.0/94911151.0?
    //duplicate points 也算在同一直线上
    //vertical的points计算slope = infinity,所以单独计数
    //如果有如下的情况[ [1,1], [1,1], [1,0], [1,-1], [1,2] ]
    //有两个点重合，
    实际上四个点都是在vertical线上，但是我们需要分开完全重合和vertical的情况
    //因为重合的点实际上可以加入任意的斜率
```

```
    int maxPoints(vector<Point>& points) {
        if(points.size() <= 2) return points.size();
        int res = 0;
        for(int i = 0; i < points.size(); i++){
            map<pair<int,int>, int> mp;
            int dup = 1, vertical = 0, non_vertical = 0;
            for(int j = i+1; j < points.size(); j++){
                if(points[i].x == points[j].x){
                    if(points[i].y == points[j].y)
                        dup++;
                    else
                        vertical++;
                    continue;
                }
                int delta_x = points[i].x - points[j].x;
                int delta_y = points[i].y - points[j].y;
                int gcd = getGCD(delta_x, delta_y);
                mp[make_pair(delta_x/gcd, delta_y/gcd)]++;
                if (non_vertical < mp[make_pair(delta_x/gcd, delta_y/gcd)])
                    non_vertical = mp[make_pair(delta_x/gcd, delta_y/gcd)];
            }
            res = max(res, max(non_vertical, vertical)+dup);
        }
        return res;
    }
private:
    //getGCD(8,12) => getGCD(12,8) => getGCD(8, 4) =>getGCD(4, 0) => return 4;
    int getGCD(int a, int b){
        if (b == 0) return a;
        return getGCD(b, a%b);
    }
};
*****************50. Pow(x, n)************************
class Solution {
public:
    double myPow(double x, int n) {
        if(n == 0) return 1;
        double t = myPow(x, n/2);
        if(n%2 == 0) return t*t; // x^4 = x^2 * x^2
        if(n > 0) return x*t*t; // x^5 = x^2 * x^2 *x
        else
            return t*t/x; // x^-5 = x^-2 * x^-2 /x;
    }
};

********************56. Merge Intervals**********************
struct Mycompr{
    bool operator()(const Interval &a, const Interval &b){
        return a.start < b.start;
    }
}mycompr;
class Solution {
public:
    vector<Interval> merge(vector<Interval>& intervals) {
        if(intervals.empty()) return {};
```

```
        sort(intervals.begin(), intervals.end(), mycompr);
        vector<Interval> res = {intervals[0]};
        for(int i = 1; i < intervals.size(); i++){
            if(intervals[i].start > res.back().end)
                res.push_back(intervals[i]);
            else{
                res.back().end = max(res.back().end, intervals[i].end);
            }
        }
        return res;
    }
};
******************152. Maximum Product Subarray************************
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int res = nums[0], mmin = nums[0], mmax = nums[0];
        for(int i = 1; i < nums.size(); i++){
            if(nums[i] < 0)
                swap(mmin, mmax);
            mmin = min(nums[i], nums[i]*mmin);
            mmax = max(nums[i], nums[i]*mmax);
            res = max(res, mmax);
        }
        return res;
    }
};
*********************150. Evaluate Reverse Polish Notation**********************
class Solution {
public:
    int evalRPN(vector<string>& tokens) {
        stack<int> sk;
        for(string &s : tokens){
            if(s != "+" && s != "-" && s != "*" && s != "/"){
                sk.push(stoi(s));
                continue;
            }
            int r = sk.top(); sk.pop();
            int l = sk.top(); sk.pop();
            if(s == "+")
                sk.push(l+r);

            else if(s == "-")
                sk.push(l-r);

            else if(s == "*")
                sk.push(l*r);

            else if(s == "/")
                sk.push(l/r);
        }
        return sk.top();
    }
};
******************173. Binary Search Tree Iterator************************
```

```
class BSTIterator {
public:
    stack<TreeNode*> sk;
    BSTIterator(TreeNode *root) {
        TreeNode *tmp = root;
        while(tmp){
            sk.push(tmp);
            tmp = tmp->left;
        }
    }

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

    /** @return the next smallest number */
    int next() {
        int res = sk.top()->val;
        TreeNode *tmp = sk.top()->right;
        sk.pop();
        while(tmp){
            sk.push(tmp);
            tmp = tmp->left;
        }
        return res;
    }
};
********************46. Permutations**********************
class Solution {
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> res;
        backtrack(res, nums, 0);
        return res;
    }
    void backtrack(vector<vector<int>> &res, vector<int> nums, int pos){
        if(pos == nums.size()-1)
            res.push_back(nums);
        for(int i = pos; i < nums.size(); i++){
            swap(nums[i], nums[pos]);
            backtrack(res, nums, pos+1);
            swap(nums[i], nums[pos]);
        }
    }
};
********************187. Repeated DNA Sequences**********************
class Solution {
public:
    vector<string> findRepeatedDnaSequences(string s) {
        if(s.size() <= 10)  return {};
        vector<int> charmap(26);
        charmap['A'-'A'] = 0;
        charmap['C'-'A'] = 1;
        charmap['G'-'A'] = 2;
```

```
        charmap['T'-'A'] = 3;
        //cout << hash("AACCG", charmap) << endl;
        map<int,int> mp;
        vector<string> res;
        for(int i = 0; i < 10; i++){
            int j = 0;
            while(i+j*10 <= s.size()-10){
                string tmp = s.substr(i+j*10, 10);
                //cout << tmp << endl;
                int hashint = hash(tmp, charmap);
                mp[hashint]++;
                if(mp[hashint] == 2)
                    res.push_back(tmp);
                j++;
            }
        }
        return res;

    }
    int hash(string s, vector<int> &charmap){
        int res = 0;
        for(char c: s){
            //cout << res << endl;
            res |= charmap[c-'A'];
            res <<= 2;
        }
        return res;
    }
};
*****************236. Lowest Common Ancestor of a Binary Tree*****************
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if(!root || root == p || root == q) return root;
        TreeNode *left = lowestCommonAncestor(root->left, p, q);
        TreeNode *right = lowestCommonAncestor(root->right, p, q);
        if(!left)
            return right;
        if(!right)
            return left;
        return root;
    }
};
********************57. Insert Interval**********************
class Solution {
public:
    vector<Interval> insert(vector<Interval>& intervals, Interval newInterval) {
        vector<Interval> res;
        int i;
        for(i = 0; i < intervals.size(); i++){
            if(intervals[i].end < newInterval.start){
                res.push_back(intervals[i]);
            }
            else if(intervals[i].start > newInterval.end)
                break;
```

```
            else{
                newInterval.start = min(newInterval.start, intervals[i].start);
                newInterval.end = max(newInterval.end, intervals[i].end);
            }
        }
        res.push_back(newInterval);
        for(; i < intervals.size(); i++){
            res.push_back(intervals[i]);
        }
        return res;
    }
};
********************127. Word Ladder***********************
#     ##双向广搜 I
#     def ladderLength(self, beginWord, endWord, wordDict):
#         front, back, wordDict=set([beginWord]), set([endWord]), set(wordDict)
#         if endWord not in wordDict:
#             return 0
#         dist=2
#         length = len(beginWord)
#         wordDict.discard(beginWord) ##不会因为beginWord不在set中而引起KeyError
#         while front:
#             newFront=set()
#             for phrase in front:
#                 for i in range(length):
#                     for c in string.lowercase:
#                         neibor=phrase[:i]+c+phrase[i+1:]
#                         if neibor in back:
#                             return dist
#                         if neibor in wordDict:
#                             newFront.add(neibor)
#                             wordDict.remove(neibor)

#             if len(newFront)>len(back):
#                 front, back = back, newFront
#             else:
#                 front = newFront
#             dist+=1

#         return 0
**********************126. Word Ladder II************************
class Solution(object):
    def findLadders(self, beginWord, endWord, wordList):
        """
        :type beginWord: str
        :type endWord: str
        :type wordList: List[str]
        :rtype: List[List[str]]
        """
        front, back, wordSet = set([beginWord]), set([endWord]), set(wordList)
        if(endWord not in wordSet):
            return []
        wordSet.add(beginWord)
        nexts = {word:[] for word in wordSet}
        wordSet.discard(beginWord)
```

```
385         while front:
386             newfront = set()
387             for w in front:
388                 for i in range(len(w)):
389                     for c in string.ascii_lowercase:
390                         nei = w[:i] + c + w[i+1:]
391                         if nei in wordSet:
392                             nexts[w].append(nei)
393                             newfront.add(nei)
394             wordSet -= newfront
395             front = newfront
396 
397         print nexts
398         paths = []
399         path = [beginWord]
400         def foo(beginWord, endWord, nexts, path):
401             if beginWord == endWord:
402                 paths.append(path[:])
403                 return
404             for i in nexts[beginWord]:
405                 path.append(i)
406                 foo(i, endWord, nexts, path)
407                 path.pop()
408         foo(beginWord, endWord, nexts, path)
409         return paths
410 
411 ******************297. Serialize and Deserialize Binary Tree**************
412 class Codec {
413 public:
414     // Encodes a tree to a single string.
415     string serialize(TreeNode* root) {
416         if(!root) return "#";
417         string s = to_string(root->val) + "," + serialize(root->left) + "," +
        serialize(root->right);
418         return s;
419     }
420 
421     // Decodes your encoded data to tree.
422     TreeNode* deserialize(string data) {
423         //cout << data;
424         return helper(data);
425     }
426     TreeNode* helper(string &data){
427         if(data[0] == '#'){
428             if(data.size() > 1)
429                 data = data.substr(2);
430             return NULL;
431         }
432         int pos = data.find(',');
433         int num = stoi(data.substr(0, pos));
434         TreeNode *root = new TreeNode(num);
435         data = data.substr(pos+1);
436         root->left = helper(data);
437         root->right = helper(data);
438         return root;
```

```
    }

};
****************************************
****************************************
****************************************
*****************174. Dungeon Game************************
class Solution {
public:
    int calculateMinimumHP(vector<vector<int>>& dungeon) {
        if(dungeon.size()==0) return 0;
        int row=dungeon.size();
        int col=dungeon[0].size();
        for(int i=row-1; i>=0; i--) {

            for(int j=col-1; j>=0; j--) {
                //右下角元素，最小值为1，或者如果dungeon[i][j]=-10，则最小为11
                if(i==row-1 && j==col-1) dungeon[i][j]=max(1,
                1-dungeon[i][j]);
                else if(i==row-1) dungeon[i][j]=max(1,
                dungeon[i][j+1]-dungeon[i][j]);//最后一行，
                else if(j==col-1) dungeon[i][j]=max(1, dungeon[i+1][j]-dungeon[i][j]);
                else dungeon[i][j]=max(1, min(dungeon[i+1][j],
                dungeon[i][j+1])-dungeon[i][j]);
            }
        }
        return dungeon[0][0];
    }
};
******************339. Nested List Weight Sum************************
class Solution {
public:
    int depthSum(vector<NestedInteger>& nestedList) {
        int sum = 0;
        helper(nestedList, sum, 1);
        return sum;
    }
    void helper(vector<NestedInteger>& nestedList, int &sum, int level){
        for(auto i : nestedList){
            if(i.isInteger()){
                sum += i.getInteger() * level;
            }
            else{
                helper(i.getList(), sum, level+1);
            }
        }
    }
};
******************311. Sparse Matrix Multiplication**********************
class Solution {
public:
    vector<vector<int>> multiply(vector<vector<int>>& A, vector<vector<int>>& B) {
        vector<vector<int>> res(A.size(), vector<int>(B[0].size(), 0));
        for(int i = 0; i < A.size(); i++){
            for(int j = 0; j < A[0].size(); j++){
```

```
                if(A[i][j] != 0){
                    for(int k = 0; k < B[0].size(); k++){
                        if(B[j][k] != 0)
                            res[i][k] += A[i][j]*B[j][k];
                    }
                }
            }
        }
        return res;
    }
};
********************647. Palindromic Substrings**********************
class Solution {
public:
    int countSubstrings(string s) {
        int res = s.size();
        for(int i =0; i < s.size(); i++){
            int j = i-1, k = i+1;
            while(j >= 0 && k <= s.size() && s[j] == s[k])
                j--, k++, res++;
            j = i, k=i+1;
            while(j >= 0 && k <= s.size() && s[j] == s[k])
                j--, k++, res++;
        }
        return res;
    }
};
********************277. Find the Celebrity************************
// Forward declaration of the knows API.
bool knows(int a, int b);

class Solution {
public:
    int findCelebrity(int n) {
        int candi = 0;
        for(int i = 1; i < n; i++){
            if(knows(candi, i))
                candi = i;
        }

        for(int i = 0; i < n; i++){
            if(i != candi){
                if(knows(candi, i) || !knows(i, candi))
                    return -1;
            }
        }
        return candi;
    }
};
********************47. Permutations II************************
class Solution {
public:
    vector<vector<int>> permuteUnique(vector<int>& nums) {
        vector<vector<int>> res;
        sort(nums.begin(), nums.end());
```

```
        helper(res, nums, 0);
        return res;
    }
    void helper(vector<vector<int>> &res, vector<int> nums, int pos){
        if(pos == nums.size()-1)
            res.push_back(nums);
        for(int i = pos; i < nums.size(); i++){
            if(i != pos && nums[i] == nums[pos])
                continue;
            swap(nums[i], nums[pos]);
            helper(res, nums, pos+1);
        }
    }
};
******************364. Nested List Weight Sum II************************
class Solution {
public:
    int depthSumInverse(vector<NestedInteger>& nestedList) {
        int unweighted = 0, res = 0;
        while (!nestedList.empty()){
            vector<NestedInteger> tmp;
            for(auto i: nestedList){
                if(i.isInteger())
                    unweighted += i.getInteger();
                else
                    tmp.insert(tmp.begin(), i.getList().begin(), i.getList().end());
            }
            res += unweighted;
            nestedList = tmp;
        }
        return res;
    }
};
****************156. Binary Tree Upside Down*************************
class Solution {
public:
    TreeNode* upsideDownBinaryTree(TreeNode* root) {
        if(!root || !root->left) return root;
        TreeNode *newRoot = upsideDownBinaryTree(root->left);
        root->left->left = root->right;
        root->left->right = root;
        root->left = NULL;
        root->right = NULL;
        return newRoot;
    }
};
******************256. Paint House***********************
class Solution(object):
    def minCost(self, costs):
        """
              r g b      r g b
    house1    2 3 5  =>  2 3 5
    house2    1 1 9  =>  4 3 12
    house3    ...        ...
    结果：house1涂r,house2涂g
```

```
601
602         """
603         r,g,b = 0,0,0
604         for i in range(len(costs)):
605             rr,gg,bb = r,g,b ##save the previous r,g,b values to rr,gg,bb
606             r = costs[i][0] + min(gg, bb)
607             g = costs[i][1] + min(rr, bb)
608             b = costs[i][2] + min(rr, gg)
609         return min([r,g,b])
610 ****************170. Two Sum III - Data structure design******************
611 class TwoSum {
612 public:
613
614     unordered_map<int,int> mp;
615     /** Initialize your data structure here. */
616     TwoSum() {
617
618     }
619
620     /** Add the number to an internal data structure.. */
621     void add(int number) {
622         // if (mp.find(number) == mp.end())
623         //     mp[number] = 1;
624         // else
625             mp[number]++;
626     }
627
628     /** Find if there exists any pair of numbers which sum is equal to the value. */
629     bool find(int value) {
630
631         for (auto it = mp.begin(); it != mp.end(); it++){
632             int target = value - it->first;
633             //如果要找的数字等于当前数字，则当前数字必须出现两次及以上
634             //如输入[2,2,3]，则保留{2:2，3:1},如果查找4，则返回true
635             if (target == (it->first)){
636                 if (it->second >=2)
637                 return true;
638             }
639             //如果不相等，则要求target出现过一次即可
640             else{
641                 if(mp.find(target) != mp.end())
642                 return true;
643             }
644         }
645         return false;
646     }
647 };
648
649 ****************367. Valid Perfect Square************************
650 class Solution {
651 public:
652     bool isPerfectSquare(int num) {
653
654         int low = 0, high = num;
655         while (low <= high) {
```

```
656             long mid = (low + high)/2;
657             if (mid * mid == num) {
658                 return true;
659             } else if (mid * mid < num) {
660                 low = (int) mid + 1;
661             } else {
662                 high = (int) mid - 1;
663             }
664         }
665         return false;
666     }
667 };
668 *****************698. Partition to K Equal Sum Subsets**********************
669 class Solution {
670 public:
671     bool canPartitionKSubsets(vector<int>& nums, int k) {
672         if(nums.empty()) return false;
673         int sum = accumulate(nums.begin(), nums.end(), 0);
674         if(sum % k != 0) return false;
675         sum = sum/k;
676         vector<bool> used(nums.size(), false);
677         return helper(nums, used, 0, k, sum, 0);
678     }
679     bool helper(vector<int> &nums, vector<bool> &used, int start, int k, int sum, int
    cur_sum){
680         if(k==1) return true;
681         if(cur_sum == sum){
682             return helper(nums, used, 0, k-1, sum, 0);
683         }
684         for(int i = start; i < used.size(); i++){
685             if(used[i] || cur_sum > sum) continue;
686             used[i] = true;
687             if( helper(nums, used, i+1, k, sum, cur_sum+nums[i]) ) return true;
688             used[i] = false;
689 
690         }
691         return false;
692     }
693 
694 };
695 ********************605. Can Place Flowers**********************
696 class Solution {
697 public:
698     bool canPlaceFlowers(vector<int>& flowerbed, int n) {
699         int cnt = 1, res = 0;
700         for(int i: flowerbed){
701             if(i == 0)
702                 cnt++;
703             else{
704                 res += (cnt-1)/2;
705                 cnt=0;
706             }
707         }
708         if(cnt) res+= cnt/2;
709         return res >= n;
```

```
710
711         }
712     };
713     ********************515. Find Largest Value in Each Tree Row*********************
714     class Solution {
715     public:
716         vector<int> largestValues(TreeNode* root) {
717             vector<int> res;
718             //if(!root) return res;
719             stack<pair<TreeNode*, int>>sk;
720             TreeNode* tmp = root;
721             int level = 0;
722             sk.push(make_pair(root, 0));
723             while(!sk.empty()){
724                 tmp = sk.top().first;
725                 level = sk.top().second;
726                 sk.pop();
727                 if(tmp){
728                     if(res.size() <= level){
729                         res.push_back(INT_MIN);
730                     }
731                     if(res[level]< tmp->val){
732                         res[level] = tmp->val;
733                     }
734                     sk.push(make_pair(tmp->left, level+1));
735                     sk.push(make_pair(tmp->right, level+1));
736                 }
737             }
738             return res;
739         }
740     };
741     ********************671. Second Minimum Node In a Binary Tree**********************
742     class Solution {
743     public:
744
745         int findSecondMinimumValue(TreeNode* root) {
746             if(!root || !root->left) return -1;
747             if(root->left->val == root->right->val) return -1;
748             int res = INT_MAX;
749             helper(root, res, root->val); //root->val is the the min val in the entire tree
750             return res;
751         }
752         void helper(TreeNode *root, int &res, int min){
753             if(!root) return;
754             if(root->val < res && root->val != min){
755                 res = root->val;
756             }
757             helper(root->left, res, min);
758             helper(root->right, res, min);
759         }
760
761     };
762     ******************254. Factor Combinations************************
763     class Solution {
764     public:
```

```
    vector<vector<int>> getFactors(int n) {
        vector<vector<int>> res;
        vector<int> tmp;
        backtrack(res, tmp, n, 2);
        return res;
    }
    void backtrack(vector<vector<int>> &res, vector<int> &tmp, int n, int start){
        if(n == 1){
            if(tmp.size() > 1)
                res.push_back(tmp);
            return;
        }
        for(int i = start; i <= n; i++){
            if(n%i == 0){
                tmp.push_back(i);
                backtrack(res, tmp, n/i, i);
                tmp.pop_back();
            }
        }
    }
};
*****************464. Can I Win************************
class Solution {
public:
    bool canIWin(int maxChoosableInteger, int desiredTotal) {
        if (maxChoosableInteger >= desiredTotal) return true;
        if((maxChoosableInteger - 1)*maxChoosableInteger/2 < desiredTotal)
            return false;
        map<int, bool> mp;
        return helper(maxChoosableInteger, desiredTotal, 0, mp);
    }
    bool helper(int maxnum, int total, int status, map<int,bool> &mp){
        if (mp.count(status)!=0) return mp[status];
        for(int i = 0; i < maxnum; i++){
            int cur = (1<<i);
            if((cur & status) == 0){
                if(i+1 >= total || !helper(maxnum, total-i-1, status|cur, mp)){
                    mp[status] = true;
                    return true;
                }
            }
        }
        mp[status] = false;
        return false;
    }
};
*******************366. Find Leaves of Binary Tree**********************
class Solution {
public:
    vector<vector<int>> findLeaves(TreeNode* root) {
        vector<vector<int>> res;
        helper(root, res);
        return res;
    }
    int helper(TreeNode *root, vector<vector<int>> &res){
```

```
820         if(!root) return 0;
821         int left = helper(root->left, res);
822         int right = helper(root->right, res);
823         int h = max(left+1, right+1);
824         if(res.size() < h){
825             res.push_back({});
826         }
827         res[h-1].push_back(root->val);
828         return h;
829     }
830 
831 };
832 ********************716. Max Stack**********************
833 class MaxStack {
834 public:
835     /** initialize your data structure here. */
836     MaxStack() { }
837     void push(int x) {
838         nums.push_back(x);
839         if(max_nums.empty() || x >= max_nums.back())
840             max_nums.push_back(x);
841     }
842     int pop() {
843         int x = nums.back();
844         if(x == max_nums.back())
845             max_nums.pop_back();
846         nums.pop_back();
847         return x;
848     }
849     int top() {
850         return nums.back();
851     }
852     int peekMax() {
853         return max_nums.back();
854     }
855     int popMax() {
856         vector<int> tmp;
857         int x = max_nums.back();
858         while(x != nums.back()){
859             int val = nums.back();
860             nums.pop_back();
861             tmp.push_back(val);
862         }
863         nums.pop_back();
864         max_nums.pop_back();
865         while(!tmp.empty()){
866             int val = tmp.back();
867             tmp.pop_back();
868             nums.push_back(val);
869             if(max_nums.empty() || val >= max_nums.back())
                //3,1,2为例子，如果不压入2,1到max_nums,第二次popMax的时候会pop空栈
870                 max_nums.push_back(val);
871         }
872         return x;
873     }
```

```
private:
    vector<int> nums;
    vector<int> max_nums;
};
******************13. Roman to Integer**********************
class Solution {
public:
    /*
    罗马数字共有7个，即I（1）、V（5）、X（10）、L（50）、C（100）、D（500）和M（1000）。
    按照下述的规则可以表示任意正整数。需要注意的是罗马数字中没有“0”，与进位制无关。
    重复数次：一个罗马数字重复几次，就表示这个数的几倍。
    右加左减：
    在较大的罗马数字的右边记上较小的罗马数字，表示大数字加小数字。
    在较大的罗马数字的左边记上较小的罗马数字，表示大数字减小数字。
    */
    int romanToInt(string s) {
        unordered_map<char, int> T = { { 'I' , 1 },
                                       { 'V' , 5 },
                                       { 'X' , 10 },
                                       { 'L' , 50 },
                                       { 'C' , 100 },
                                       { 'D' , 500 },
                                       { 'M' , 1000 } };

        // IV: 4
        // V: 5
        // VI: 6
        //开始，sum = T['I'] = 1,
       int sum = T[s.back()];
       //从最后一位开始，比较倒数第二位与倒数第一位的大小，小于则减去，大于则加上
       //依次向前比较相邻位，直到第一位
        for(int i = s.size()-2; i >= 0; i--){
            if( T[s[i]] >= T[s[i+1]] ){
                sum += T[s[i]];
            }
            else{
                sum -= T[s[i]];
            }
        }
        return sum;
    }
};
******************12. Integer to Roman**********************
class Solution {
public:
    string intToRoman(int num) {
        vector<string> M = { "", "M", "MM", "MMM" };
        vector<string> C = { "", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"
        };
        vector<string> X = { "", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"
        };
        vector<string> I = { "", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"
        };
        return M[num / 1000] + C[(num % 1000) / 100] + X[(num % 100) / 10] + I[num % 10];
    }
```

```
};
******************76. Minimum Window Substring**********************
class Solution {
public:
/*
        0 1 2 3 4 5 6 7 8 9
        c c a c c a c b c c
        a b
        1.存储a,b的数量：m[a]=1, m[b]=1, count="ab".size()=2
        2.start=end=0，开始移动end，如果end的位置是a或者b，且m[s[end]] > 0,
        则count--；m[a or b]--
        这样当count==0时，找到了第一个窗口，此时start=0, end=8，m[a]=-1,m[b]=0,m[c]=-5
        此时因为有两个aa一个b，所以需要再尽可能向右移动start来减小窗口
        start移动时，m[s[start]]++, 且当m[s[start]]>0, count++,
        说明只有找到第二个a，count才能变为1

        初始化:m[a]=1,m[b]=1,count=2
        找到第一个窗口：m[a]=-1,m[b]=0,count=0
        (说明这窗口内有多余元素，但是不一定可以缩小，如oobaaacoo找abc)
        移动窗口左边，尝试缩小窗口
*/
    string minWindow(string s, string t) {
        if(s.size() < t.size()) return "";
        vector<int> charmap(128, 0);
        for(char c: t){
            charmap[c]++;
        }
        int i = 0, j = 0, start = 0, min_len = INT_MAX, cnt = t.size();
        while(j < s.size()){
            if(charmap[s[j]] > 0)
                cnt--;
            charmap[s[j]]--;
            j++;
            while(cnt == 0){
                if(min_len > j-i){
                    start = i;
                    min_len = j-i;
                }
                charmap[s[i]]++;
                if(charmap[s[i]] > 0)
                    cnt++;
                i++;
            }
        }
        if(min_len == INT_MAX)
            return "";
        return s.substr(start, min_len);
    }
};

********************252. Meeting Rooms**********************
struct mystruct{
    bool operator() (Interval &a, Interval &b){
        return a.start < b.start;
    }
```

```
} mycompr;
class Solution {
public:
    bool canAttendMeetings(vector<Interval>& intervals) {
        sort(intervals.begin(), intervals.end(), mycompr);
        for(int i = 1; i < intervals.size(); i++){
            if(intervals[i].start < intervals[i-1].end)
                return false;
        }
        return true;
    }
};
********************253. Meeting Rooms II**********************
class Solution {
public:
    int minMeetingRooms(vector<Interval>& intervals) {
        vector<int> starts, ends;
        for(auto &i : intervals){
            starts.push_back(i.start);
            ends.push_back(i.end);
        }
        sort(starts.begin(), starts.end());
        sort(ends.begin(), ends.end());
        int res = 0, av = 0;
        for(int i = 0, j = 0; i < intervals.size(); ){
            if(starts[i] < ends[j]){
                if(av == 0)
                    res++;
                else
                    av--;
                i++;
            }
            else{
                av++;
                j++;
            }
        }
        return res;
    }
};
******************68. Text Justification************************
class Solution {
public:
    vector<string> fullJustify(vector<string>& words, int maxWidth) {

        vector<string> res;
        for(int i = 0; i < words.size(); ){
            int j = i;
            int cur_size = words[j].size();
            while(cur_size <= maxWidth){
                j++;
                cur_size += (words[j].size()+1);
            }

            cur_size -= words[j].size()+j-i;//total length of words, except the blanks
```

```
            int total_blanks = maxWidth - cur_size;
            int total_words = j-i-1;//how many words are there
            if(total_words == 0)
                total_words = 1;
            int blanks = total_blanks / total_words;//how many blanks in each space

            int k = i;
            string s = "";
            while(k -i < total_blanks%total_words){
                if(j < words.size())
                    s +=  words[k] + string(blanks + 1, ' ');
                else
                    s += words[k] + " ";
                k++;
            }
            while(k < j){
                if(j < words.size())
                    s +=  words[k] + string(blanks, ' ');
                else
                    s += words[k] + " ";
                k++;
            }
            while(s.size() > maxWidth)
                s.pop_back();

            res.push_back(s);
            i = j;
        }

        while(res.back().size() < maxWidth){
            res.back() += " ";
        }
        return res;
    }
};
*******3. Longest Substring Without Repeating Characters*******
class Solution {
public:
    //abcadcbb
    //start = -1; mp['a'] = 0; res = max(0,1) = 1;
    //start = -1; mp['b'] = 1; res= max(1,2) = 2;
    //start = -1; mp['c'] = 2; res = max(2,3) = 3
    //start = max(0, -1) = 0; mp['a'] = 3; res = max(3,3)= 3
    //start = max(-1, 0) = 0; mp['d'] = 4; res= max(3,4) = 4
    int lengthOfLongestSubstring(string s) {
        // vector<int> charmap(256, -1);
        // int res = 0, start = -1;
        // for(int i = 0; i < s.size(); i++){
        //     start = max(charmap[s[i]], start);
        //     charmap[s[i]] = i;
        //     res = max(res, i-start);
        // }
        // return res;
        vector<int> charmap(256, -1);
```

```
1088            int res = 0, start = -1;
1089            for(int i = 0; i < s.size(); i++){
1090                start = max(start, charmap[s[i]]);
1091                charmap[s[i]] = i;
1092                res = max(res, i-start);
1093            }
1094            return res;
1095        }
1096    };
```