701 Insert into a Binary Search Tree

```go
func insertIntoBST(root *TreeNode, val int) *TreeNode {
  if root == nil{
    root = new(TreeNode)
    root.Val = val
    return root
  }

  if root.Val < val{
    root.Right = insertIntoBST(root.Right, val)
    return root
  }else{
    root.Left = insertIntoBST(root.Left, val)
    return root
  }

}
```

703 Kth largest Element in a Stream

```java
//pq 的解法
PriorityQueue<Integer> queue;
    int k;
    public KthLargest(int k, int[] nums) {
        this.k = k;
        queue = new PriorityQueue<>((o1, o2) -> (o1 - o2));
        for(int num : nums) {
            if(queue.size() == k){
                if(queue.peek() > num)
                    continue;
                else {
                    queue.poll();
                    queue.add(num);
                }
```

```
            }else{
                queue.add(num);
            }
        }
    }

    public int add(int val) {
        if(queue.size() == k){
            if(queue.peek() <= val){
                queue.poll();
                queue.add(val);
            }
        }else{
            queue.add(val);
        }

        return queue.peek();
    }
```

执行结果: 通过 显示详情 >

执行用时: 851 ms , 在所有 Java 提交中击败了 5.04% 的用户

内存消耗: 44 MB , 在所有 Java 提交中击败了 14.46% 的用户

炫耀一下:

```
class KthLargest {
   List<Integer> res;
    int k;
    public KthLargest(int k, int[] nums) {
        res = new ArrayList<>();
        for(int num : nums)
            res.add(num);

        Collections.sort(res, (o1, o2) -> o2 - o1);
        this.k = k;
    }

    public int add(int val) {
        for(int i = 0; i <= res.size(); i++){
            if(i == res.size()){
              res.add(val);
              break;
            } else if(val >= res.get(i)) {
                res.add(i, val);
                break;
```

```
            }
        }

        return res.get(k - 1);
    }
}
```

704 Binary Search

执行用时： 32 ms ，在所有 C++ 提交中击败了 96.22% 的用户

内存消耗： 26.9 MB ，在所有 C++ 提交中击败了 59.21% 的用户

```
4
5
6
7
```

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left = 0, right = nums.size() - 1;
        while(left <= right){
            int mid = (left + right) / 2;

            if(nums[mid] == target){
                return mid;
            }else if(nums[mid] > target){
                right = mid - 1;
            }else{
                left = mid + 1;
            }
        }

        return -1;
    }
};
```

705 Design HashSet

```
class MyHashSet {
public:
    vector<list<int>> data;
```

```
    static const int base = 1007;
    static int hash(int key){
        return key % base;
    }

    /** Initialize your data structure here. */
    MyHashSet() :data(base){

    }

    void add(int key) {
        int pos = hash(key);
        for(auto it = data[pos].begin(); it != data[pos].end(); it++){
            if((*it) == key)
                return;
        }

        data[pos].push_back(key);
    }

    void remove(int key) {
        int pos = hash(key);
        for(auto it = data[pos].begin(); it != data[pos].end(); it++){
            if((*it) == key){
                data[pos].erase(it);
                return;
            }

        }
    }

    /** Returns true if this set contains the specified element */
    bool contains(int key) {
        int pos = hash(key);
        for(auto it = data[pos].begin(); it != data[pos].end(); it++){
            if((*it) == key)
                return true;
        }

        return false;
    }
};

/**
 * Your MyHashSet object will be instantiated and called as such:
 * MyHashSet* obj = new MyHashSet();
 * obj->add(key);
 * obj->remove(key);
 * bool param_3 = obj->contains(key);
```

```
 */
```

706 Design HashMap

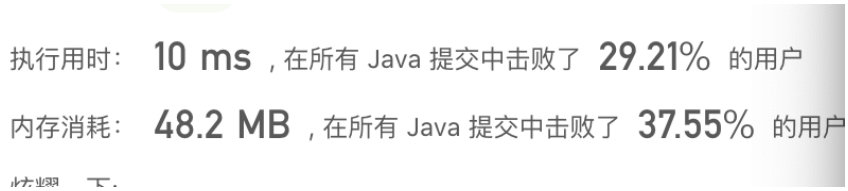
执行用时： 100 ms ，在所有 C++ 提交中击败了 95.92% 的用户

内存消耗： 52.1 MB ，在所有 C++ 提交中击败了 32.10% 的用户

```
class MyHashMap {
public:
    vector<list<pair<int, int>>> nodes;
    static const int BUCKETNUM = 1007;
    static int hash(int key){
        return key % BUCKETNUM;
    }
    /** Initialize your data structure here. */
    MyHashMap() : nodes(BUCKETNUM){ }

    /** value will always be non-negative. */
    void put(int key, int value) {
        int pos = hash(key);
        for(auto it = nodes[pos].begin(); it != nodes[pos].end(); it++){
            if((*it).first == key){
                (*it).second = value;
                return;
            }
        }

        nodes[pos].push_back({key, value});
    }

    /** Returns the value to which the specified key is mapped, or -1 if this map contains no mapping for the key */
    int get(int key) {
```

```
        int pos = hash(key);
         for(auto it = nodes[pos].begin(); it != nodes[pos].end(); it++){
            if((*it).first == key){
                return (*it).second;
            }
        }

        return -1;
    }

    /** Removes the mapping of the specified value key if this map contains a mapping for the key */
    void remove(int key) {
         int pos = hash(key);
         for(auto it = nodes[pos].begin(); it != nodes[pos].end(); it++){
            if((*it).first == key){
                nodes[pos].erase(it);
                return;
            }
        }

    }
};

```

713 Subarry Product Less Than K




```java
public int numSubarrayProductLessThanK(int[] nums, int k) {
    int res = 0;

    int left = 0, right = 0;
    int product = 1;
    while(right < nums.length){
        while(right < nums.length){
            product *= nums[right];
            if(product >= k)
                break;

            res += right - left + 1;
            right++;
        }

        if(right == nums.length)
            break;

        while(left < right && product >= k){
            product /= nums[left];
            left++;
        }

        if(product < k)
            res += right - left + 1;
        right++;
    }

    return res;
}
```

716 Max Stack

执行结果: 通过 显示详情 > ▷ 添加备

执行用时: 22 ms , 在所有 Java 提交中击败了 64.91% 的用户

内存消耗: 40.3 MB , 在所有 Java 提交中击败了 76.61% 的用户

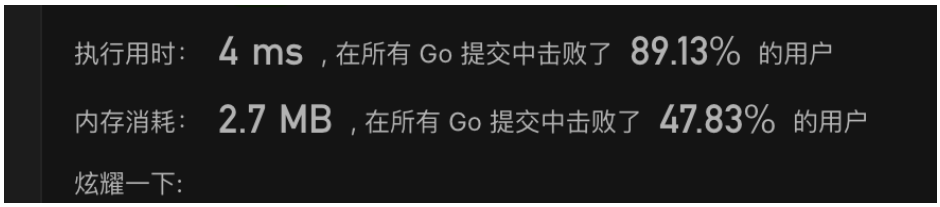
炫耀一下:

```java
public class MaxStack {
    DoubleLinkedList dll;
    TreeMap<Integer, List<Node>> map;
    /** initialize your data structure here. */
    public MaxStack() {
        dll = new DoubleLinkedList();
        map = new TreeMap<>();
    }

    public void push(int x) {
        Node newNode = new Node(x);
        dll.add(newNode);
        map.putIfAbsent(x, new ArrayList<>());

        map.get(x).add(newNode);
    }

    public int pop() {
        int val = top();
        List<Node> nodes = map.get(val);
        nodes.remove(nodes.size() - 1);
        dll.delete();

        if(map.get(val).size() == 0)
            map.remove(val);

        return val;
    }

    public int top() {
        return dll.tail.prev.val;
    }

    public int peekMax() {
        return map.lastKey();
```

```
    }

    public int popMax() {
        int maxVal = peekMax();
        List<Node> nodes = map.get(maxVal);
        Node node = nodes.get(nodes.size() - 1);
        nodes.remove(nodes.size() - 1);
        if(nodes.size() == 0)
            map.remove(maxVal);

        dll.delete(node);

        return maxVal;
    }
}

class DoubleLinkedList{
    Node head;
    Node tail;

    public DoubleLinkedList(){
        head = new Node(0);
        tail = new Node(0);

        head.next = tail;
        tail.prev = head;
    }

    public void add(Node node){
        node.next = tail;
        node.prev = tail.prev;

        tail.prev.next = node;
        tail.prev = node;
    }

    public void delete(){
        tail.prev.prev.next = tail;
        tail.prev = tail.prev.prev;
    }

    public void delete(Node node){
        node.prev.next = node.next;
        node.next.prev = node.prev;
    }
}

class Node{
    public int val;
```

```
    public Node next;
    public Node prev;

    public Node(int val){
        this.val = val;
    }
}
```

717 1-bit and 2-bit Characters




```go
func isOneBitCharacter(bits []int) bool {
  index := 0

  for ;index < len(bits);{
    if bits[index] == 0{
      index++
    }else {
      if index + 2 == len(bits){
        return false
      }
      index += 2
    }
  }

  return true
}
```

718 Maximum Length of Repeated Subarray

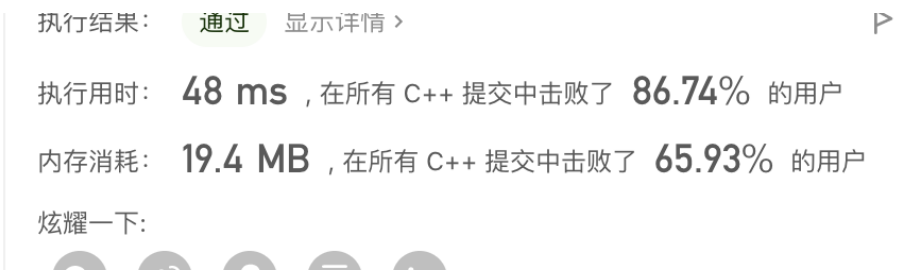
```
//使用 DP 优化
```

```go
//时间复 O(N*M)
func findLength(nums1 []int, nums2 []int) int {
    res := 0
    dp := make([][]int, len(nums1))
    for i := len(nums1) - 1; i >= 0; i--{
        dp[i] = make([]int, len(nums2))
        for j := len(nums2) - 1; j >= 0; j--{
            if nums1[i] == nums2[j]{
                if i == len(nums1) - 1 || j == len(nums2) - 1{
                    dp[i][j] = 1
                }else{
                    dp[i][j] = 1 + dp[i + 1][j + 1]
                }

                if dp[i][j] > res{
                    res = dp[i][j]
                }
            }
        }
    }

    return res
}
```


```go
//超时
func findLength(nums1 []int, nums2 []int) int {
  res := 0
  for i := 0; i < len(nums1); i++{
    for j := 0; j < len(nums2); j++{
      for k := 0; i + k < len(nums1) && j + k < len(nums2);k++{
        if nums1[i + k] == nums2[j + k]{
          if res < k + 1{
            res = k + 1
          }
        }else{
          break
```

```
        }
      }
    }
  }

  return res
}

```

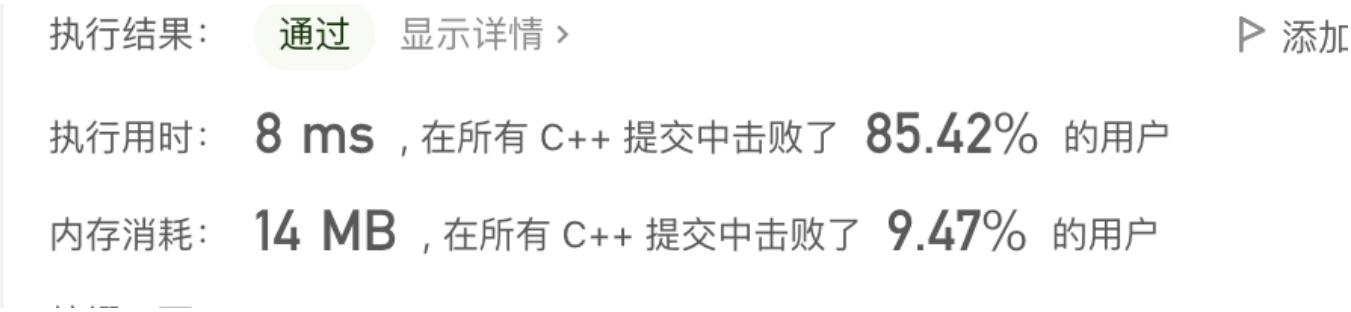
720 Longest Word In Dictionary




```cpp
class Solution {
public:
    string longestWord(vector<string>& words) {
        if(words.size() == 1 && words[0].size() == 1)
            return words[0];

        sort(words.begin(), words.end());
        unordered_map<string, int> map;
        for(int i = 0; i < words.size(); i++){
            map.insert({words[i], i});
        }

        string res = "";
        vector<bool> dp(words.size(), false);
        for(int i = 0; i < words.size(); i++){
            if(words[i].size() == 1)
                dp[i] = true;
            string frac = words[i].substr(0, words[i].size() - 1);
            if(map.count(frac) != 0 && dp[map[frac]])
                dp[i] = true;

            if(dp[i]){
                if(res.size() < words[i].size())
                    res = words[i];
                else if(res.size() == words[i].size() && res > words[i])
```

```
                    res=  words[i];

                }
            }



        return res;
    }
};
```

733 Flood Fill




```cpp
struct pair_hash
{
    template <class T1, class T2>
    size_t operator () (pair<T1, T2> const &pair) const
    {
        size_t h1 = hash<T1>()(pair.first); //用默认的 hash 处理 pair 中的第一个数据 X1
        size_t h2 = hash<T2>()(pair.second);//用默认的 hash 处理 pair 中的第二个数据 X2
        return h1 ^ h2;
    }
};


class Solution {

public:
    vector<vector<int>> floodFill(vector<vector<int>>& image, int sr, int sc, int newColor) {
        queue<pair<int, int>> myQueue;
        unordered_set<pair<int, int>, pair_hash> set;
```

```
        int row = image.size();
        int col = image[0].size();
        auto isInRange = [&](int x, int y){return x >= 0 && y >= 0 && x < row && y < col;};

        vector<vector<int>> dir = {{1, 0}, {-1, 0}, {0, -1}, {0, 1}};
        myQueue.push({sr, sc});
        set.insert({sr, sc});
        int standard = image[sr][sc];

        while(!myQueue.empty()){
            int size = myQueue.size();
            for(int i = 0; i < size; i++){
                auto coor = myQueue.front();
                set.insert({coor.first, coor.second});
                myQueue.pop();

                image[coor.first][coor.second] = newColor;

                for(int k = 0; k < 4; k++){
                    int newX = coor.first + dir[k][0];
                    int newY = coor.second + dir[k][1];

                    if(set.count({newX, newY}) == 0){
                        if(isInRange(newX, newY) && image[newX][newY] == standard){
                            myQueue.push({newX, newY});
                            set.insert({coor.first, coor.second});
                        }
                    }
                }
            }
        }

        return image;
    }
};
```

735 Asteroid Coollisioon

执行用时： 7 ms ，在所有 Java 提交中击败了 48.82% 的用户

内存消耗： 39.2 MB ，在所有 Java 提交中击败了 57.94% 的用户

炫耀一下：

```java
public int[] asteroidCollision(int[] asteroids) {
    List<Integer> ans = new ArrayList<>();

    for(int i = 0; i < asteroids.length; i++){
        if(ans.size() == 0 || ans.get(ans.size() - 1) < 0 || asteroids[i] > 0){
            ans.add(asteroids[i]);
        }else{
            int flag = 0;
            while(ans.size() > 0 && ans.get(ans.size() - 1) > 0){
                if(ans.get(ans.size() - 1) == -1 * asteroids[i]){
                    ans.remove(ans.size() - 1);
                    flag = 1;
                    break;
                }else if(ans.get(ans.size() - 1) > -1 * asteroids[i]){
                    flag = 1;
                    break;
                }else{
                    ans.remove(ans.size() - 1);
                }
            }

            if(flag == 0)
                ans.add(asteroids[i]);
        }
    }

    int[] res = new int[ans.size()];
    for(int i = 0; i < ans.size(); i++)
        res[i] = ans.get(i);
    return res;
}
```

执行结果： 通过 显示详情 > ▷ 添加备

执行用时： 8 ms ，在所有 Java 提交中击败了 26.84% 的用户

内存消耗： 39.2 MB ，在所有 Java 提交中击败了 60.03% 的用户

炫耀一下：

```java
    public int[] asteroidCollision(int[] asteroids) {
        List<Integer> res= new ArrayList<>();

        int left = 0;
        for(int i = 0; i < asteroids.length; i++){
            if(res.size() == 0) {
                res.add(asteroids[i]);
            }else if(res.get(res.size() - 1) * asteroids[i] > 0 ||
(res.get(res.size() - 1) < 0 && asteroids[i] > 0)){
                res.add(asteroids[i]);

            }else{
                int attacker = asteroids[i];
                boolean settled = false;
                while(res.size() != 0 && res.get(res.size() - 1) > 0 && attacker <
0){
                    int defend = res.get(res.size() - 1);
                    if(Math.abs(defend) >= Math.abs(attacker)){
                        if(Math.abs(defend) == Math.abs(attacker))
                            res.remove(res.size() - 1);
                        settled = true;
                        break;
                    }else{
                         res.remove(res.size() - 1);
                    }
                }

                if(!settled)
                    res.add(attacker);

            }
        }

        int[] ans = new int[res.size()];
        for(int i = 0; i < res.size(); i++)
            ans[i] = res.get(i);
        return ans;
    }

```

739 Daily Temperatures

执行用时： 124 ms ，在所有 C++ 提交中击败了 66.85% 的用户

内存消耗： 83.2 MB ，在所有 C++ 提交中击败了 56.87% 的用户

炫耀一下:

```cpp
class Solution {
public:
    vector<int> dailyTemperatures(vector<int>& temperatures) {
        stack<int> monoStack;

        int size = temperatures.size();
        vector<int> ans(size, 0);
        for(int i = size - 1; i >= 0; i--){
             if(i == size - 1){
                ans[i] = 0;
                monoStack.push(i);
             }
            else{
                while(!monoStack.empty() && temperatures[i] >= temperatures[monoStack.top()])
                    monoStack.pop();

                if(monoStack.empty())
                    ans[i] = 0;
                else
                    ans[i] = monoStack.top() - i;

                monoStack.push(i);
            }
        }

        return ans;
    }
};
// 0  1  2   3  4  5  6  7
//[73,74,75,71,69,72,76,73]
```

747 Largest Number At Least Twice of Others

执行用时： 0 ms ，在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 36.3 MB ，在所有 Java 提交中击败了 42.57% 的用户

```java
class Solution {
    public int dominantIndex(int[] nums) {
        int max = nums[0];
        int index = 0;
        for(int i = 0; i < nums.length; i++){
            if(nums[i] > max){
                max = nums[i];
                index = i;
            }
        }

        for(int i = 0; i < nums.length; i++){
            if(nums[i] != max){
                if(nums[i] * 2 > max)
                    return -1;
            }
        }

        return index;
    }
}
```

748 Shortest Completing Word

执行结果: 通过 显示详情 >

执行用时: 16 ms , 在所有 C++ 提交中击败了 65.69% 的用户

内存消耗: 15.5 MB , 在所有 C++ 提交中击败了 49.27% 的用户

炫耀一下:

```cpp
class Solution {
public:
    string shortestCompletingWord(string licensePlate, vector<string>& words) {
        string res = "";
        auto alpha = vector<int>(26, 0);
        for(char ch : licensePlate){
            if(ch >= 'a' && ch <= 'z')
                alpha[ch - 'a']++;
            else if(ch >= 'A' && ch <= 'Z')
                alpha[ch - 'A']++;
        }
//        cout << alpha['s' - 'a'] << endl;

```

```
        for(string& str : words){
            vector<int> strAlpha = vector<int>(26, 0);
            for(char ch : str)
                strAlpha[ch - 'a']++;

            bool found = true;
            for(int i = 0; i < 26; i++){
                if(strAlpha[i] < alpha[i]){
                    found = false;
                    break;
                }
            }

            if(found && (res == "" || res.size() > str.size()))
                res = str;
        }

        return res;
    }
};
```

775 Global and Local Inversion

775. Global and Local Inversions

难度 中等 69

You are given an integer array `nums` of length `n` wh
permutation of all the integers in the range `[0, n -`

The number of **global inversions** is the number of th
`(i, j)` where:

- `0 <= i < j < n`
- `nums[i] > nums[j]`

The number of **local inversions** is the number of indi

- `0 <= i < n - 1`
- `nums[i] > nums[i + 1]`

Return `true` *if the number of* ***global inversions*** *is e
number of* ***local inversions***.

Example 1:

```cpp
class Solution {
public:
    bool isIdealPermutation(vector<int>& nums) {
        int currentMax = nums[0];

        for(int i = 0; i < nums.size(); i++){
            currentMax = currentMax > nums[i] ? currentMax : nums[i];

            if(i + 2 < nums.size()){
                if(nums[i + 2] < currentMax)
                    return false;
            }
        }

        return true;
    }
};
```

```cpp
class Solution {
public:
    long globalInversion = 0;
    vector<int> aux;
    bool isIdealPermutation(vector<int>& nums) {
        long localInversion = 0;
        int n = nums.size();
        aux = vector<int>(n);

        for(int i = 0; i < n - 1; i++){
            if(nums[i] > nums[i + 1])
                localInversion++;
        }

        mergeSort(nums, 0, n - 1);

        return globalInversion == localInversion;
    }

    void mergeSort(vector<int>& nums, int left, int right){
        if(left == right)
            return;

        int mid = (left + right) / 2;
        mergeSort(nums, left, mid);
        mergeSort(nums, mid + 1, right);

        int index1 = left;
        int index2 = mid + 1;

        for(int i = left; i <= right; i++)
            aux[i] = nums[i];

        int index = left;
        while(index1 <= mid || index2 <= right){
            if(index1 > mid){
                nums[index++] = aux[index2++];
            }else if(index2 > right){
                nums[index++] = aux[index1++];
            }else if(aux[index1] <= aux[index2]){
                nums[index++] = aux[index1++];
            }else{
                globalInversion += mid - index1 + 1;
                nums[index++] = aux[index2++];
            }
        }

    }
};
```

763 Partition Labels

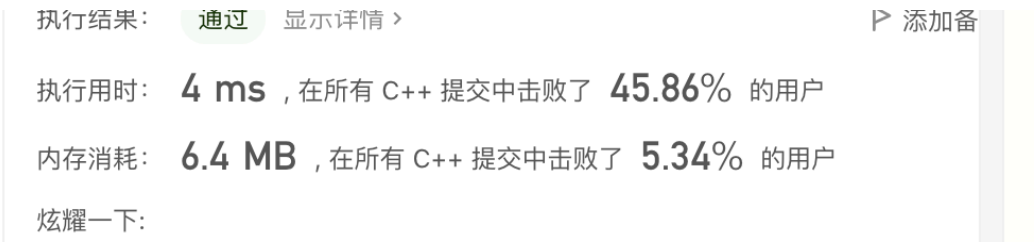
```java
    public List<Integer> partitionLabels(String S) {
        List<Integer> res = new ArrayList<>();
        int len = S.length();

        int[] alpha = new int[26];
        int[] record = new int[S.length()];
        Arrays.fill(alpha, -1);

        for(int i = len- 1; i >= 0; i--){
            int pos = S.charAt(i) - 'a';
            if(alpha[pos] != -1){
                record[i] = alpha[pos];
            }else{
                record[i] = -1;
            }

            alpha[pos] = i;
        }


        int left = 0, right = 0;
        while(right < len){
            int index = left;
            while(index < len){
                char ch = S.charAt(right);
                right = Math.max(record[index], right);

                if(index == right)
                    break;

                index++;
            }

            res.add(right - left + 1);

            right++;
            left = right ;
        }

```

```
        return res;
    }
```

767 Reorganize String

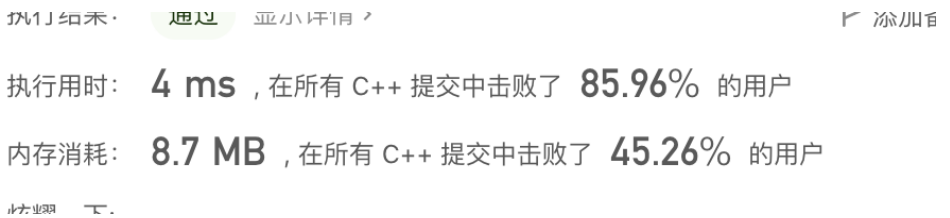
```
//358 变种
class Solution {
public:
    string reorganizeString(string s) {
        int size = s.size();
        vector<int> alpha(26, 0);
        for(char ch : s)
            alpha[ch - 'a']++;

        auto cmp = [&](pair<int, int>& p1, pair<int, int>& p2){return p1.second == p2.second ? p1.first > p2.first : p1.second < p2.second;};
        priority_queue<pair<int, int>, vector<pair<int, int>>, decltype(cmp)> pq(cmp);
        string res = "";
        for(int i = 0; i < 26; i++){
            if(alpha[i] != 0)
                pq.push({i, alpha[i]});
        }

        deque<char> myQueue;
        while(!pq.empty()){
            pair<int, int> curPair = pq.top(); pq.pop();
            char ch = (char)(curPair.first + 'a');
            alpha[ch - 'a']--;
            myQueue.push_back(ch);
            res += string(1, ch);

            if(myQueue.size() == 2){
                char curChar = myQueue.front(); myQueue.pop_front();
                if(alpha[curChar - 'a'] != 0){
                    pq.push({curChar - 'a', alpha[curChar - 'a']});
                }
            }
        }

        return res.size() == s.size() ? res : "";
    }
};
```

771 Jewels and Stones




```cpp
class Solution {
public:
    int numJewelsInStones(string jewels, string stones) {
        vector<int> jew(256, 0);
        for(char ch : jewels){
            jew[ch]++;
        }

        int count = 0;
        for(char ch : stones){
            if(jew[ch] != 0)
                count++;
        }

        return count;
    }
};
```

772 Basic Calcutor




```cpp
/*
  RPE规则:
    1. 碰到 ( 直接入 stack
    2 碰到 数字 直接进入 vector
    3 碰到 ) 就一直pop 直到 (
```

```
    4 碰到 ops 优先级小于自己的，一直 push
      碰到 大于等于的，直接 pop给 vector
*/
class Solution {
public:
    int calculate(string s) {
        vector<string> vec = getRPN(s);
        int res = evl(vec);

        return res;
    }

    vector<string> getRPN(string& s){
        vector<string> res;
        stack<string> myStack;
        int num = -1;

        for(int i = 0; i < s.size(); i++){
            if(s[i] == ' ')
                continue;

            if(s[i] >= '0' && s[i] <= '9'){
                if(num == -1)
                    num = s[i] - '0';
                else{
                    num *= 10;
                    num += s[i] - '0';
                }
            }else{
                if(num != -1){
                    res.push_back(to_string(num));
                    num = -1;
                }

                if(s[i] == '(')
                    myStack.push("(");
                else if(s[i] == ')'){
                    while(!myStack.empty()){
                        if(myStack.top() == "("){
                            myStack.pop();
                            break;
                        }

                        res.push_back(myStack.top());       myStack.pop();
                    }
                }else{
                    while(!myStack.empty()){
                        if(myStack.top() == "("){
                            break;
```

```
                        }

                        if(getPriority(myStack.top()[0]) <= getPriority(s[i])){
                            res.push_back(myStack.top());       myStack.pop();
                        }else{
                            break;
                        }
                    }

                    myStack.push(string(1, s[i]));
                }

            }
        }

        if(num != -1)
            myStack.push(to_string(num));
        while(!myStack.empty()){
            res.push_back(myStack.top()); myStack.pop();
        }

        return res;
    }



    int getPriority(char ch){
        if(ch == '+' || ch == '-')
            return 1;
        else
            return 0;
    }

    int evl(vector<string>& vec) {
        stack<int> myStack;
        int res = 0;
        for(string& str : vec){
            if(str[0] >= '0' && str[0] <= '9'){ //number
                myStack.push(parseInt(str));
            }else{  //ops
                int op1 = myStack.top();    myStack.pop();
                int op2 = myStack.top();    myStack.pop();

                int temp = 0;
                if(str == "+")
                    temp =  op1 + op2;
                else if(str == "-")
                    temp =  op2 - op1;
                else if(str == "*")
```

```
                    temp =  op1 * op2;
                else
                    temp =  op2 / op1;

                myStack.push(temp);
            }
        }

        return myStack.top();
    }

    int parseInt(string s){
        int res = 0;
        for(int i = 0; i < s.size(); i++){
            if(res > INT_MAX / 10 || (res == INT_MAX / 10 && s[i] - '0' > INT_MAX % 10))
                return INT_MAX;

            res *= 10;
            res += s[i] - '0';
        }

        return res;
    }
};
```

773 Sliding Puzzle

773. Sliding Puzzle

难度 困难 226

On an `2 x 3` board, there are five tiles labeled from `1` to `5`, and an empty square represented by `0`. A **move** consists of choosing `0` and a 4-directionally adjacent number and swapping it.

The state of the board is solved if and only if the board is `[[1,2,3],[4,5,0]]`.

Given the puzzle board `board`, return *the least number of moves required so that the state of the board is solved*. If it is impossible for the state of the board to be solved, return `-1`.

Example 1:


```
Input: board = [[1,2,3],[4,0,5]]
Output: 1
Explanation: Swap the 0 and the 5 in one move.
```

Example 2:

执行用时： 8 ms ，在所有 C++ 提交中击败了 89.91% 的用户

内存消耗： 8 MB ，在所有 C++ 提交中击败了 89.61% 的用户

```cpp

class Solution {
public:
    int row, col;
    vector<vector<int>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
    const string FINAL = "123450";
    int slidingPuzzle(vector<vector<int>>& board) {
        row = board.size();
        col = board[0].size();

```

```
        auto isInRange = [&](int i, int j){return i >= 0 && j >= 0 && i < row && j < col;};

        unordered_set<string> set;
        string start = getState(board);
        if(start == FINAL)
            return 0;

        set.insert(start);

        queue<string> myQueue;
        myQueue.push(start);

        int round = 0;
        while(!myQueue.empty()){
            int size = myQueue.size();
            for(int i = 0; i < size; i++){
                string curBoard = myQueue.front(); myQueue.pop();
                //cout <<" 当前" << curBoard << endl;
                int pos = curBoard.find('0');
                int posX = pos / 3;
                int posY = pos % 3;

                for(int k = 0; k < 4; k++){
                    int newX = posX + dir[k][0];
                    int newY = posY + dir[k][1];
                    int newPos = newX * col + newY;

                    string nextState = curBoard;

                    if(isInRange(newX, newY)){
                        swap(nextState[pos], nextState[newPos]);

                        if(nextState == FINAL)
                            return round + 1;
                        if(set.count(nextState) == 0){
                            myQueue.push(nextState);
                            set.insert(nextState);
                        }

                    }
                }

            }

            round++;
//            cout << "---" << endl;
        }

```

```
        return -1;
    }

    string getState(vector<vector<int>>& board){
        string res;
        for(int i = 0; i < row; i++){
            for(int j = 0; j < col; j++){
                res += to_string(board[i][j]);
            }
        }

        return res;
    }
};
```

```
class Solution {
public:
    int row, col;
    vector<vector<int>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
    const string FINAL = "123@450";
    int slidingPuzzle(vector<vector<int>>& board) {
        row = board.size();
        col = board[0].size();

        auto isInRange = [&](int i, int j){return i >= 0 && j >= 0 && i < row && j < col;};

        unordered_set<string> set;
        string start = getState(board);
        if(start == FINAL)
            return 0;

        set.insert(start);

        queue<string> myQueue;
        myQueue.push(start);

        int round = 0;
        while(!myQueue.empty()){
            int size = myQueue.size();
            for(int i = 0; i < size; i++){
                string curBoard = myQueue.front(); myQueue.pop();
                vector<vector<int>> newBord = getBoard(curBoard);

                pair<int, int> zeroPos = getZero(newBord);
```

```
                for(int k = 0; k < 4; k++){
                    int newX = zeroPos.first + dir[k][0];
                    int newY = zeroPos.second + dir[k][1];

                    if(isInRange(newX, newY)){
                        newBord[zeroPos.first][zeroPos.second] = newBord[newX]
[newY];
                        newBord[newX][newY] = 0;

                        string nextState = getState(newBord);
                        if(nextState == FINAL)
                            return round + 1;
                        if(set.count(nextState) == 0){
                            myQueue.push(nextState);
                            set.insert(nextState);
                        }

                        newBord[newX][newY] =  newBord[zeroPos.first]
[zeroPos.second];
                        newBord[zeroPos.first][zeroPos.second] = 0;
                    }
                }

            }

            round++;
        }

        return -1;
    }

    pair<int, int> getZero(vector<vector<int>>& board){
        for(int i = 0; i < row; i++){
            for(int j = 0; j < col; j++){
                if(board[i][j] == 0)
                    return {i, j};
            }
        }

        return {-1, -1};
    }

    vector<vector<int>> getBoard(string& curBoard){
        vector<vector<int>> res(row, vector<int>(col, 0));
        for(int i = 0; i < 3; i++){
            res[0][i] = curBoard[i] - '0';
        }


```

```
        for(int i = 4; i < 7; i++){
            res[1][i - 4] = curBoard[i] - '0';
        }

        return res;
    }

    string getState(vector<vector<int>>& board){
        string res = "";
        for(int i = 0; i < row; i++){
            for(int j = 0; j < col; j++){
                res += to_string(board[i][j]);
            }

            if(i != row - 1)
                res += "@";
        }

        return res;
    }
};

```

777 Swap Adjacent in LR Strign

```
//39 / 94 超时
public class Solution {
    /**
     * The Algorithm is kind of like backtrack
     * where I try for every possible way
     * then to add into our result set
     * @param start
     * @param end
     * @return
     */

    int counter = 0;
    public boolean canTransform(String start, String end) {
        backtrack(start.toCharArray(), end.toCharArray());
        return counter != 0;
    }

```

```java

    private void backtrack(char[] start, char[] end) {
        if(counter >= 2){
            return;
        }

        if(Arrays.equals(start, end)){
            counter++;
            return;
        }

        if(noExchange(start)){
            return;
        }

        for(int i = 0; i < start.length - 1; i++){
            if(start[i] == 'X' && start[i + 1] == 'L'){
                start[i] = 'L';
                start[i + 1] = 'X';

                backtrack(start, end);

                start[i] = 'X';
                start[i + 1] = 'L';
            }

            if(counter >= 2)
                return;

            if(start[i] == 'R' && start[i + 1] == 'X'){
                start[i] = 'X';
                start[i + 1] = 'R';

                backtrack(start, end);;

                start[i] = 'R';
                start[i + 1] = 'X';
            }
        }
    }

    private boolean noExchange(char[] starts) {
        for(int i = 0; i < starts.length - 1; i++){
            if(starts[i] == 'X' && starts[i + 1] == 'L')
                return false;
            if(starts[i] == 'R' && starts[i + 1] == 'X')
                return false;
        }

```

```
        return true;
    }
}
```

```
//BFS 超时 56 / 94
public boolean canTransform(String start, String end) {
        if(start.equals(end))
            return true;

        Set<String> visited = new HashSet<>();
        Deque<String> queue = new ArrayDeque<>();
        queue.add(start);
        visited.add(start);

        while(!queue.isEmpty()){
            int size = queue.size();
            for(int i =0; i < size; i++){
                String cur = queue.removeFirst();

                if(cur.equals(end))
                    return true;

                char[] curs = cur.toCharArray();
                for(int k = 0; k < curs.length - 1; k++){
                    if(curs[k] == 'X' && curs[k + 1] == 'L'){
                        curs[k] = 'L';
                        curs[k + 1] = 'X';

                        String str = toStr(curs);
                        if(!visited.contains(str)){
                            queue.addLast(str);
                            visited.add(str);
                        }

                        curs[k] = 'X';
                        curs[k + 1] = 'L';
                    }

                    if(curs[k] == 'R' && curs[k + 1] == 'X'){
                        curs[k] = 'X';
                        curs[k + 1] = 'R';

                        String str = toStr(curs);
                        if(!visited.contains(str)){
                            queue.addLast(str);
                            visited.add(str);
                        }
```

```
                    curs[k] = 'R';
                    curs[k + 1] = 'X';
                }
            }
        }
    }

    return false;
}

private String toStr(char[] curs) {
    StringBuilder sb = new StringBuilder();
    for(char ch : curs)
        sb.append(ch);

    return sb.toString();
}
```

```
public boolean canTransform(String start, String end) {
    StringBuilder sb1 = new StringBuilder();
    StringBuilder sb2 = new StringBuilder();
    int len1 = start.length();
    int len2 = end.length();

    for(int i = 0; i < start.length(); i++){
        char ch = start.charAt(i);
        if(ch != 'X')
            sb1.append(ch);
    }

    for(int i = 0; i < end.length(); i++){
        char ch = end.charAt(i);
        if(ch != 'X')
            sb2.append(ch);
    }

    if(!sb1.toString().equals(sb2.toString()))
        return false;

    int up = 0, down = 0;


    while(up < len1 && down < len2){
        while(up < len1 && start.charAt(up) != 'L')
```

```
                up++;
            while(down < len2 && end.charAt(down) != 'L')
                down++;

            if(up == len1 || down == len2){
                if(noSymbol(start, up + 1, 'L', true) && noSymbol(end, down + 1, 'L', true))
                    break;
                else
                    return false;
            }

            if(up < down)
                return false;
            up++;
            down++;
        }

        up = len1 - 1;
        down = len2 -1;

        while(up >= 0 && down >= 0){
            while(up >= 0 && start.charAt(up) != 'R')
                up--;
            while(down >= 0 && end.charAt(down) != 'R')
                down--;

            if(up == 0 || down == 0){
                if(noSymbol(start, up - 1, 'R', false) && noSymbol(end, down - 1, 'R', false))
                    break;
                else
                    return false;
            }

            if(up > down)
                return false;
            up--;
            down--;
        }

        return true;
    }

    private boolean noSymbol(String str, int index, char ch, boolean l2r) {
        if(l2r){
            for(int i = index; i < str.length(); i++){
                if(str.charAt(i) == ch)
                    return false;
```

```
            }

            return true;
        }else{
            for(int i = index; i >= 0; i--){
                if(str.charAt(i) == ch)
                    return false;
            }

            return true;
        }

    }

```

780 Reaching Points

```
class Solution {
    public boolean reachingPoints(int sx, int sy, int tx, int ty) {
        while(sx < tx && sy < ty){
            if(tx < ty)
                ty %= tx;
            else
                tx %= ty;
        }

        if(sx == tx && sy <= ty)
            return (ty - sy) % sx == 0;
        else
            return sy == ty && sx <= tx && (tx - sx) % sy == 0;
    }
}
```

```
    public boolean reachingPoints(int sx, int sy, int tx, int ty) {
        Deque<Point> queue = new ArrayDeque<>();
        Point root = new Point(sx, sy);
        Point end  = new Point(tx, ty);
        Set<Point> visited = new HashSet<>();
        queue.addLast(root);

```

```
8
9           while(!queue.isEmpty()){
10              int size = queue.size();
11              for(int i = 0; i < size; i++){
12                  Point cur = queue.removeFirst();
13                  visited.add(cur);
14                  System.out.println(cur);
15                  if(cur.equals(end))
16                      return true;
17
18                  if(cur.x + cur.y > tx + ty || cur.x + cur.y < 0)
19                      continue;
20
21                  Point left = new Point(cur.x + cur.y, cur.y);
22                  Point right = new Point(cur.x, cur.x + cur.y);
23
24                  if(!visited.contains(left)){
25                      queue.addLast(left);
26                  }
27
28                  if(!visited.contains(right)){
29                      queue.addLast(right);
30                  }
31              }
32          }
33
34          return false;
35      }
36
```

783 Minimum Distance Between BST Nodes

执行用时： 0 ms ，在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 36 MB ，在所有 Java 提交中击败了 58.78% 的用户

```
1       TreeNode pre = null;
2       long res = Integer.MAX_VALUE;
3       public int minDiffInBST(TreeNode root) {
4           inorder(root);
5
6           return (int)res;
```

```
    }

    public void inorder(TreeNode root){
        if(root == null)
            return;

        inorder(root.left);

        if(pre != null)
            res = Math.min(res, Math.abs((long)(root.val) - (long)(pre.val)));

        pre = root;

        inorder(root.right);
    }
```

787 Cheapest Flights Within K Stops

```
//超时， 通过 27 / 49 个案例
class Solution {
   Map<Integer, List<String>> map = new HashMap<>();
    int res = Integer.MAX_VALUE;
    public int findCheapestPrice(int n, int[][] flights, int src, int dst, int k) {
        for(int[] flight : flights){
            map.putIfAbsent(flight[0], new ArrayList<>());
            map.get(flight[0]).add(flight[1] +  "@" +flight[2]);
        }

        dfs(dst, k, src, 0, 0, new boolean[n]);

        return res == Integer.MAX_VALUE ? -1 : res;
    }

    private void dfs(int dst, int k, int cur, int step, int price, boolean[]
visited) {
        if(cur == dst && k + 1 >= step){
            res = Math.min(res, price);
            return;
        }

        visited[cur] = true;
```

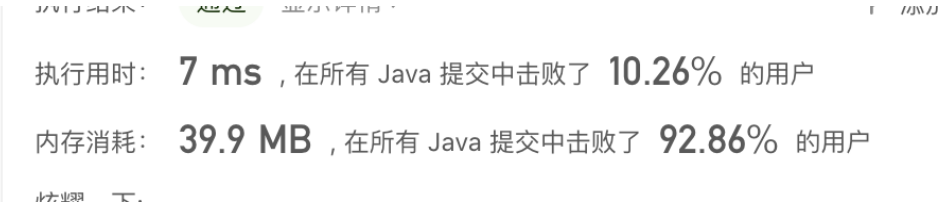
```
23
24          List<String> strings = map.get(cur);
25          if(strings == null || strings.size() == 0){
26              return;
27          }
28
29          for(String str : strings){
30              String[] split = str.split("@");
31              if(visited[Integer.parseInt(split[0])])
32                  continue;
33
34              visited[Integer.parseInt(split[0])] = true;
35              dfs(dst, k, Integer.parseInt(split[0]), step + 1, price +
Integer.parseInt(split[1]), visited);
36              visited[Integer.parseInt(split[0])] = false;
37          }
38
39      }
40  }
```

791 Custom Sort String

```
1       public String customSortString(String order, String str) {
2           Map<Character, Integer> map = new HashMap<>();
3
4           for(char ch : str.toCharArray())
5               map.put(ch, map.getOrDefault(ch, 0) + 1);
6
7           StringBuilder res = new StringBuilder();
8           for(int i =0; i < order.length(); i++){
9               char ch = order.charAt(i);
10              if(!map.containsKey(ch))
11                  continue;
12
13              for(int j = 0; j < map.get(ch); j++)
14                  res.append(ch);
15
16              map.remove(ch);
17          }
18
19          for(char ch : map.keySet()){
```

```
            for(int i = 0; i < map.get(ch); i++)
                res.append(ch);
        }

        return res.toString();
    }
```

797 All Paths From Source to Target




```
//典型 DFS
  List<List<Integer>> res = new ArrayList<>();
    int n;
    public List<List<Integer>> allPathsSourceTarget(int[][] graph) {
        n = graph.length;
        List<Set<Integer>> bags = new ArrayList<>();
        for(int i = 0; i <= n - 1; i++)
            bags.add(new HashSet<>());

        for(int i = 0; i < graph.length; i++){
            int[] edges = graph[i];
            for(int num : edges)
                bags.get(i).add(num);
        }
        List<Integer> path = new ArrayList<>();
        path.add(0);
        dfs(bags, new HashSet<>(), path, 0);
        return res;
    }

    private void dfs(List<Set<Integer>> bags, HashSet<Integer> visited,
List<Integer> path, int bagNum) {
        if(bagNum == n - 1){
            res.add(new ArrayList<>(path));
        }
```

```
25
26          Set<Integer> bag = bags.get(bagNum);
27
28          for(Integer nextBag : bag){
29              if(visited.contains(nextBag))
30                  continue;
31
32              visited.add(nextBag);
33              path.add(nextBag);
34              dfs(bags, visited, path, nextBag);
35              visited.remove(nextBag);
36              path.remove(path.size() - 1);
37          }
38
39      }
```