

## 701 Insert into a Binary Search Tree

```
1 func insertIntoBST(root *TreeNode, val int) *TreeNode {
2     if root == nil{
3         root = new(TreeNode)
4         root.Val = val
5         return root
6     }
7
8     if root.Val < val{
9         root.Right = insertIntoBST(root.Right, val)
10        return root
11    }else{
12        root.Left = insertIntoBST(root.Left, val)
13        return root
14    }
15
16 }
```

## 703 Kth largest Element in a Stream

```
1 //pq 的解法
2 PriorityQueue<Integer> queue;
3 int k;
4 public KthLargest(int k, int[] nums) {
5     this.k = k;
6     queue = new PriorityQueue<>((o1, o2) -> (o1 - o2));
7     for(int num : nums) {
8         if(queue.size() == k){
9             if(queue.peek() > num)
10                 continue;
11             else {
12                 queue.poll();
13                 queue.add(num);
14             }
15         }
16     }
17 }
```

```

15         }else{
16             queue.add(num);
17         }
18     }
19 }
20
21 public int add(int val) {
22     if(queue.size() == k){
23         if(queue.peek() <= val){
24             queue.poll();
25             queue.add(val);
26         }
27     }else{
28         queue.add(val);
29     }
30
31     return queue.peek();
32 }

```

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:

```

1  class KthLargest {
2      List<Integer> res;
3      int k;
4      public KthLargest(int k, int[] nums) {
5          res = new ArrayList<>();
6          for(int num : nums)
7              res.add(num);
8
9          Collections.sort(res, (o1, o2) -> o2 - o1);
10         this.k = k;
11     }
12
13     public int add(int val) {
14         for(int i = 0; i <= res.size(); i++){
15             if(i == res.size()){
16                 res.add(val);
17                 break;
18             } else if(val >= res.get(i)) {
19                 res.add(i, val);
20                 break;

```

```

21         }
22     }
23
24     return res.get(k - 1);
25 }
26 }

```

## 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

```

1  class Solution {
2  public:
3      int search(vector<int>& nums, int target) {
4          int left = 0, right = nums.size() - 1;
5          while(left <= right){
6              int mid = (left + right) / 2;
7
8              if(nums[mid] == target){
9                  return mid;
10             }else if(nums[mid] > target){
11                 right = mid - 1;
12             }else{
13                 left = mid + 1;
14             }
15         }
16
17         return -1;
18     }
19 };

```

## 705 Design HashSet

```

1  class MyHashSet {
2  public:
3      vector<list<int>>> data;

```

```

4      static const int base = 1007;
5      static int hash(int key){
6          return key % base;
7      }
8
9      /** Initialize your data structure here. */
10     MyHashSet() :data(base){
11
12     }
13
14     void add(int key) {
15         int pos = hash(key);
16         for(auto it = data[pos].begin(); it != data[pos].end(); it++){
17             if((*it) == key)
18                 return;
19         }
20
21         data[pos].push_back(key);
22     }
23
24     void remove(int key) {
25         int pos = hash(key);
26         for(auto it = data[pos].begin(); it != data[pos].end(); it++){
27             if((*it) == key){
28                 data[pos].erase(it);
29                 return;
30             }
31         }
32     }
33 }
34
35 /** Returns true if this set contains the specified element */
36 bool contains(int key) {
37     int pos = hash(key);
38     for(auto it = data[pos].begin(); it != data[pos].end(); it++){
39         if((*it) == key)
40             return true;
41     }
42
43     return false;
44 }
45 };
46
47 /**
48  * Your MyHashSet object will be instantiated and called as such:
49  * MyHashSet* obj = new MyHashSet();
50  * obj->add(key);
51  * obj->remove(key);
52  * 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

... —

```
1  class MyHashMap {
2  public:
3      vector<list<pair<int, int>>> nodes;
4      static const int BUCKETNUM = 1007;
5      static int hash(int key){
6          return key % BUCKETNUM;
7      }
8      /** Initialize your data structure here. */
9      MyHashMap() : nodes(BUCKETNUM){ }
10
11     /** value will always be non-negative. */
12     void put(int key, int value) {
13         int pos = hash(key);
14         for(auto it = nodes[pos].begin(); it != nodes[pos].end(); it++){
15             if((*it).first == key){
16                 (*it).second = value;
17                 return;
18             }
19         }
20
21         nodes[pos].push_back({key, value});
22     }
23
24     /** Returns the value to which the specified key is mapped, or -1 if this map
25     contains no mapping for the key */
26     int get(int key) {
```

```

26         int pos = hash(key);
27         for(auto it = nodes[pos].begin(); it != nodes[pos].end(); it++){
28             if((*it).first == key){
29                 return (*it).second;
30             }
31         }
32
33         return -1;
34     }
35
36     /** Removes the mapping of the specified value key if this map contains a
37     mapping for the key */
38     void remove(int key) {
39         int pos = hash(key);
40         for(auto it = nodes[pos].begin(); it != nodes[pos].end(); it++){
41             if((*it).first == key){
42                 nodes[pos].erase(it);
43                 return;
44             }
45         }
46     }
47 };
48

```

# 713 Subarray Product Less Than K

执行用时：10 ms，在所有 Java 提交中击败了 29.21% 的用户

内存消耗：48.2 MB，在所有 Java 提交中击败了 37.55% 的用户

非常差劲 下

```
1 public int numSubarrayProductLessThanK(int[] nums, int k) {
2     int res = 0;
3
4     int left = 0, right = 0;
5     int product = 1;
6     while(right < nums.length){
7         while(right < nums.length){
8             product *= nums[right];
9             if(product >= k)
10                 break;
11
12             res += right - left + 1;
13             right++;
14         }
15
16         if(right == nums.length)
17             break;
18
19         while(left < right && product >= k){
20             product /= nums[left];
21             left++;
22         }
23
24         if(product < k)
25             res += right - left + 1;
26         right++;
27     }
28
29     return res;
30 }
```

# 716 Max Stack

执行结果： 通过 [显示详情 >](#)

[添加备注](#)

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：

```
1 public class MaxStack {
2     DoubleLinkedList dll;
3     TreeMap<Integer, List<Node>> map;
4     /** initialize your data structure here. */
5     public MaxStack() {
6         dll = new DoubleLinkedList();
7         map = new TreeMap<>();
8     }
9
10    public void push(int x) {
11        Node newNode = new Node(x);
12        dll.add(newNode);
13        map.putIfAbsent(x, new ArrayList<>());
14
15        map.get(x).add(newNode);
16    }
17
18    public int pop() {
19        int val = top();
20        List<Node> nodes = map.get(val);
21        nodes.remove(nodes.size() - 1);
22        dll.delete();
23
24        if(map.get(val).size() == 0)
25            map.remove(val);
26
27        return val;
28    }
29
30    public int top() {
31        return dll.tail.prev.val;
32    }
33
34    public int peekMax() {
35        return map.lastKey();
36    }
37 }
```



```

36     }
37
38     public int popMax() {
39         int maxVal = peekMax();
40         List<Node> nodes = map.get(maxVal);
41         Node node = nodes.get(nodes.size() - 1);
42         nodes.remove(nodes.size() - 1);
43         if(nodes.size() == 0)
44             map.remove(maxVal);
45
46         dll.delete(node);
47
48         return maxVal;
49     }
50 }
51
52 class DoubleLinkedList{
53     Node head;
54     Node tail;
55
56     public DoubleLinkedList(){
57         head = new Node(0);
58         tail = new Node(0);
59
60         head.next = tail;
61         tail.prev = head;
62     }
63
64     public void add(Node node){
65         node.next = tail;
66         node.prev = tail.prev;
67
68         tail.prev.next = node;
69         tail.prev = node;
70     }
71
72     public void delete(){
73         tail.prev.prev.next = tail;
74         tail.prev = tail.prev.prev;
75     }
76
77     public void delete(Node node){
78         node.prev.next = node.next;
79         node.next.prev = node.prev;
80     }
81 }
82
83 class Node{
84     public int val;

```

```

85     public Node next;
86     public Node prev;
87
88     public Node(int val){
89         this.val = val;
90     }
91 }

```

## 717 1-bit and 2-bit Characters

执行用时: **4 ms** , 在所有 Go 提交中击败了 **89.13%** 的用户

内存消耗: **2.7 MB** , 在所有 Go 提交中击败了 **47.83%** 的用户

炫耀一下:

```

1  func isOneBitCharacter(bits []int) bool {
2      index := 0
3
4      for ;index < len(bits);{
5          if bits[index] == 0{
6              index++
7          }else {
8              if index + 2 == len(bits){
9                  return false
10             }
11             index += 2
12         }
13     }
14
15     return true
16 }

```

## 718 Maximum Length of Repeated Subarray

```

1  //使用 DP 优化

```

```

2 //时间复 O(N*M)
3 func findLength(nums1 []int, nums2 []int) int {
4     res := 0
5     dp := make([][]int, len(nums1))
6     for i := len(nums1) - 1; i >= 0; i--{
7         dp[i] = make([]int, len(nums2))
8         for j := len(nums2) - 1; j >= 0; j--{
9             if nums1[i] == nums2[j]{
10                 if i == len(nums1) - 1 || j == len(nums2) - 1{
11                     dp[i][j] = 1
12                 }else{
13                     dp[i][j] = 1 + dp[i + 1][j + 1]
14                 }
15
16                 if dp[i][j] > res{
17                     res = dp[i][j]
18                 }
19             }
20         }
21     }
22
23     return res
24 }

```

执行结果： **超出时间限制** [显示详情 >](#)

[添加备注](#)

最后执行的输入：

[91,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,...] [查看全](#)

```

1 //超时
2 func findLength(nums1 []int, nums2 []int) int {
3     res := 0
4     for i := 0; i < len(nums1); i++){
5         for j := 0; j < len(nums2); j++){
6             for k := 0; i + k < len(nums1) && j + k < len(nums2); k++){
7                 if nums1[i + k] == nums2[j + k]{
8                     if res < k + 1{
9                         res = k + 1
10                    }
11                }else{
12                    break

```

```

13     }
14     }
15     }
16     }
17
18     return res
19 }
20

```

## 720 Longest Word In Dictionary

执行结果： 通过 [显示详情](#) >

执行用时： **48 ms** ，在所有 C++ 提交中击败了 **86.74%** 的用户

内存消耗： **19.4 MB** ，在所有 C++ 提交中击败了 **65.93%** 的用户

炫耀一下：



```

1  class Solution {
2  public:
3      string longestWord(vector<string>& words) {
4          if(words.size() == 1 && words[0].size() == 1)
5              return words[0];
6
7          sort(words.begin(), words.end());
8          unordered_map<string, int> map;
9          for(int i = 0; i < words.size(); i++){
10             map.insert({words[i], i});
11         }
12
13         string res = "";
14         vector<bool> dp(words.size(), false);
15         for(int i = 0; i < words.size(); i++){
16             if(words[i].size() == 1)
17                 dp[i] = true;
18             string frac = words[i].substr(0, words[i].size() - 1);
19             if(map.count(frac) != 0 && dp[map[frac]])
20                 dp[i] = true;
21
22             if(dp[i]){
23                 if(res.size() < words[i].size())
24                     res = words[i];
25                 else if(res.size() == words[i].size() && res > words[i])

```

```

26         res= words[i];
27
28     }
29 }
30
31
32
33     return res;
34 }
35 };

```

## 733 Flood Fill

执行结果： **通过** [显示详情 >](#)

[▶ 添加](#)

执行用时： **8 ms** ，在所有 C++ 提交中击败了 **85.42%** 的用户

内存消耗： **14 MB** ，在所有 C++ 提交中击败了 **9.47%** 的用户

... -- --

```

1  struct pair_hash
2  {
3      template <class T1, class T2>
4      size_t operator () (pair<T1, T2> const &pair) const
5      {
6          size_t h1 = hash<T1>()(pair.first); //用默认的 hash 处理 pair 中的第一个数据 x1
7          size_t h2 = hash<T2>()(pair.second); //用默认的 hash 处理 pair 中的第二个数据 x2
8          return h1 ^ h2;
9      }
10 };
11
12
13 class Solution {
14
15 public:
16     vector<vector<int>>> floodFill(vector<vector<int>>& image, int sr, int sc, int
newColor) {
17         queue<pair<int, int>> myQueue;
18         unordered_set<pair<int, int>, pair_hash> set;

```

```

19     int row = image.size();
20     int col = image[0].size();
21     auto isInRange = [&](int x, int y){return x >= 0 && y >= 0 && x < row && y
< col;};
22
23     vector<vector<int>> dir = {{1, 0}, {-1, 0}, {0, -1}, {0, 1}};
24     myQueue.push({sr, sc});
25     set.insert({sr, sc});
26     int standard = image[sr][sc];
27
28     while(!myQueue.empty()){
29         int size = myQueue.size();
30         for(int i = 0; i < size; i++){
31             auto coor = myQueue.front();
32             set.insert({coor.first, coor.second});
33             myQueue.pop();
34
35             image[coor.first][coor.second] = newColor;
36
37             for(int k = 0; k < 4; k++){
38                 int newX = coor.first + dir[k][0];
39                 int newY = coor.second + dir[k][1];
40
41                 if(set.count({newX, newY}) == 0){
42                     if(isInRange(newX, newY) && image[newX][newY] == standard){
43                         myQueue.push({newX, newY});
44                         set.insert({coor.first, coor.second});
45                     }
46                 }
47             }
48         }
49     }
50
51     return image;
52 }
53 };

```

## 735 Asteroid Collision

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:

```
1 public int[] asteroidCollision(int[] asteroids) {
2     List<Integer> ans = new ArrayList<>();
3
4     for(int i = 0; i < asteroids.length; i++){
5         if(ans.size() == 0 || ans.get(ans.size() - 1) < 0 || asteroids[i] > 0){
6             ans.add(asteroids[i]);
7         }else{
8             int flag = 0;
9             while(ans.size() > 0 && ans.get(ans.size() - 1) > 0){
10                 if(ans.get(ans.size() - 1) == -1 * asteroids[i]){
11                     ans.remove(ans.size() - 1);
12                     flag = 1;
13                     break;
14                 }else if(ans.get(ans.size() - 1) > -1 * asteroids[i]){
15                     flag = 1;
16                     break;
17                 }else{
18                     ans.remove(ans.size() - 1);
19                 }
20             }
21
22             if(flag == 0)
23                 ans.add(asteroids[i]);
24         }
25     }
26
27     int[] res = new int[ans.size()];
28     for(int i = 0; i < ans.size(); i++)
29         res[i] = ans.get(i);
30     return res;
31 }
```

执行结果: **通过** [显示详情](#)

[添加备注](#)

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:



```

1      public int[] asteroidCollision(int[] asteroids) {
2          List<Integer> res= new ArrayList<>();
3
4          int left = 0;
5          for(int i = 0; i < asteroids.length; i++){
6              if(res.size() == 0) {
7                  res.add(asteroids[i]);
8              }else if(res.get(res.size() - 1) * asteroids[i] > 0 ||
12 (res.get(res.size() - 1) < 0 && asteroids[i] > 0)){
13                 res.add(asteroids[i]);
14
15             }else{
16                 int attacker = asteroids[i];
17                 boolean settled = false;
18                 while(res.size() != 0 && res.get(res.size() - 1) > 0 && attacker <
19 0){
20                     int defend = res.get(res.size() - 1);
21                     if(Math.abs(defend) >= Math.abs(attacker)){
22                         if(Math.abs(defend) == Math.abs(attacker))
23                             res.remove(res.size() - 1);
24                         settled = true;
25                         break;
26                     }else{
27                         res.remove(res.size() - 1);
28                     }
29                 }
30                 if(!settled)
31                     res.add(attacker);
32             }
33
34             int[] ans = new int[res.size()];
35             for(int i = 0; i < res.size(); i++)
36                 ans[i] = res.get(i);
37             return ans;
38         }
39     }

```

## 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:

```
1  class Solution {
2  public:
3      vector<int> dailyTemperatures(vector<int>& temperatures) {
4          stack<int> monoStack;
5
6          int size = temperatures.size();
7          vector<int> ans(size, 0);
8          for(int i = size - 1; i >= 0; i--){
9              if(i == size - 1){
10                 ans[i] = 0;
11                 monoStack.push(i);
12             }
13             else{
14                 while(!monoStack.empty() && temperatures[i] >=
temperatures[monoStack.top()])
15                     monoStack.pop();
16
17                 if(monoStack.empty())
18                     ans[i] = 0;
19                 else
20                     ans[i] = monoStack.top() - i;
21
22                 monoStack.push(i);
23             }
24         }
25
26         return ans;
27     }
28 };
29 // 0 1 2 3 4 5 6 7
30 //[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

```

1  class Solution {
2      public int dominantIndex(int[] nums) {
3          int max = nums[0];
4          int index = 0;
5          for(int i = 0; i < nums.length; i++){
6              if(nums[i] > max){
7                  max = nums[i];
8                  index = i;
9              }
10         }
11
12         for(int i = 0; i < nums.length; i++){
13             if(nums[i] != max){
14                 if(nums[i] * 2 > max)
15                     return -1;
16             }
17         }
18
19         return index;
20     }
21 }

```

## 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：



```

1  class Solution {
2  public:
3      string shortestCompletingWord(string licensePlate, vector<string>& words) {
4          string res = "";
5          auto alpha = vector<int>(26, 0);
6          for(char ch : licensePlate){
7              if(ch >= 'a' && ch <= 'z')
8                  alpha[ch - 'a']++;
9              else if(ch >= 'A' && ch <= 'Z')
10                 alpha[ch - 'A']++;
11         }
12         //      cout << alpha['s' - 'a'] << endl;
13     }

```

```
14     for(string& str : words){
15         vector<int> strAlpha = vector<int>(26, 0);
16         for(char ch : str)
17             strAlpha[ch - 'a']++;
18
19         bool found = true;
20         for(int i = 0; i < 26; i++){
21             if(strAlpha[i] < alpha[i]){
22                 found = false;
23                 break;
24             }
25         }
26
27         if(found && (res == "" || res.size() > str.size()))
28             res = str;
29     }
30
31     return res;
32 }
33 };
```

## 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` which is a permutation of all the integers in the range `[0, n - 1]`.

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

- $0 \leq i < j < n$
- `nums[i] > nums[j]`

The number of **local inversions** is the number of indices `i` such that:

- $0 \leq i < n - 1$
- `nums[i] > nums[i + 1]`

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

### Example 1:

```
1 class Solution {
2 public:
3     bool isIdealPermutation(vector<int>& nums) {
4         int currentMax = nums[0];
5
6         for(int i = 0; i < nums.size(); i++){
7             currentMax = currentMax > nums[i] ? currentMax : nums[i];
8
9             if(i + 2 < nums.size()){
10                 if(nums[i + 2] < currentMax)
11                     return false;
12             }
13         }
14
15         return true;
16     }
17 };
```

```

1  class Solution {
2  public:
3      long globalInversion = 0;
4      vector<int> aux;
5      bool isIdealPermutation(vector<int>& nums) {
6          long localInversion = 0;
7          int n = nums.size();
8          aux = vector<int>(n);
9
10         for(int i = 0; i < n - 1; i++){
11             if(nums[i] > nums[i + 1])
12                 localInversion++;
13         }
14
15         mergeSort(nums, 0, n - 1);
16
17         return globalInversion == localInversion;
18     }
19
20     void mergeSort(vector<int>& nums, int left, int right){
21         if(left == right)
22             return;
23
24         int mid = (left + right) / 2;
25         mergeSort(nums, left, mid);
26         mergeSort(nums, mid + 1, right);
27
28         int index1 = left;
29         int index2 = mid + 1;
30
31         for(int i = left; i <= right; i++)
32             aux[i] = nums[i];
33
34         int index = left;
35         while(index1 <= mid || index2 <= right){
36             if(index1 > mid){
37                 nums[index++] = aux[index2++];
38             }else if(index2 > right){
39                 nums[index++] = aux[index1++];
40             }else if(aux[index1] <= aux[index2]){
41                 nums[index++] = aux[index1++];
42             }else{
43                 globalInversion += mid - index1 + 1;
44                 nums[index++] = aux[index2++];
45             }
46         }
47
48     }
49 };

```

---

## 763 Partition Labels

---

```
1      public List<Integer> partitionLabels(String S) {
2          List<Integer> res = new ArrayList<>();
3          int len = S.length();
4
5          int[] alpha = new int[26];
6          int[] record = new int[S.length()];
7          Arrays.fill(alpha, -1);
8
9          for(int i = len- 1; i >= 0; i--){
10             int pos = S.charAt(i) - 'a';
11             if(alpha[pos] != -1){
12                 record[i] = alpha[pos];
13             }else{
14                 record[i] = -1;
15             }
16
17             alpha[pos] = i;
18         }
19
20
21         int left = 0, right = 0;
22         while(right < len){
23             int index = left;
24             while(index < len){
25                 char ch = S.charAt(right);
26                 right = Math.max(record[index], right);
27
28                 if(index == right)
29                     break;
30
31                 index++;
32             }
33
34             res.add(right - left + 1);
35
36             right++;
37             left = right ;
38         }
39     }
```

```
40         return res;
41     }
```

## 767 Reorganize String

```
1  //358 变种
2  class Solution {
3  public:
4      string reorganizeString(string s) {
5          int size = s.size();
6          vector<int> alpha(26, 0);
7          for(char ch : s)
8              alpha[ch - 'a']++;
9
10         auto cmp = [&](pair<int, int>& p1, pair<int, int>& p2){return p1.second ==
p2.second ? p1.first > p2.first : p1.second < p2.second;};
11         priority_queue<pair<int, int>, vector<pair<int, int>>, decltype(cmp)>
pq(cmp);
12         string res = "";
13         for(int i = 0; i < 26; i++){
14             if(alpha[i] != 0)
15                 pq.push({i, alpha[i]});
16         }
17
18         deque<char> myQueue;
19         while(!pq.empty()){
20             pair<int, int> curPair = pq.top(); pq.pop();
21             char ch = (char)(curPair.first + 'a');
22             alpha[ch - 'a']--;
23             myQueue.push_back(ch);
24             res += string(1, ch);
25
26             if(myQueue.size() == 2){
27                 char curChar = myQueue.front(); myQueue.pop_front();
28                 if(alpha[curChar - 'a'] != 0){
29                     pq.push({curChar - 'a', alpha[curChar - 'a']});
30                 }
31             }
32         }
33
34         return res.size() == s.size() ? res : "";
35     }
36 };
```

## 771 Jewels and Stones

执行结果： **通过** [显示详情](#) >

[添加备注](#)

执行用时： **4 ms** ，在所有 C++ 提交中击败了 **45.86%** 的用户

内存消耗： **6.4 MB** ，在所有 C++ 提交中击败了 **5.34%** 的用户

炫耀一下：

```
1 class Solution {
2 public:
3     int numJewelsInStones(string jewels, string stones) {
4         vector<int> jew(256, 0);
5         for(char ch : jewels){
6             jew[ch]++;
7         }
8
9         int count = 0;
10        for(char ch : stones){
11            if(jew[ch] != 0)
12                count++;
13        }
14
15        return count;
16    }
17 };
```

## 772 Basic Calcutor

执行结果： **通过** [显示详情](#) >

[添加备注](#)

执行用时： **4 ms** ，在所有 C++ 提交中击败了 **85.96%** 的用户

内存消耗： **8.7 MB** ，在所有 C++ 提交中击败了 **45.26%** 的用户

评论区 下

```
1  /*
2     RPE规则:
3     1. 碰到 ( 直接入 stack
4     2 碰到 数字 直接进入 vector
5     3 碰到 ) 就一直pop 直到 (
```



```

6      4 碰到 ops 优先级小于自己的, 一直 push
7      碰到 大于等于的, 直接 pop给 vector
8  */
9  class Solution {
10 public:
11     int calculate(string s) {
12         vector<string> vec = getRPN(s);
13         int res = evl(vec);
14
15         return res;
16     }
17
18     vector<string> getRPN(string& s){
19         vector<string> res;
20         stack<string> myStack;
21         int num = -1;
22
23         for(int i = 0; i < s.size(); i++){
24             if(s[i] == ' ')
25                 continue;
26
27             if(s[i] >= '0' && s[i] <= '9'){
28                 if(num == -1)
29                     num = s[i] - '0';
30                 else{
31                     num *= 10;
32                     num += s[i] - '0';
33                 }
34             }else{
35                 if(num != -1){
36                     res.push_back(to_string(num));
37                     num = -1;
38                 }
39
40                 if(s[i] == '(')
41                     myStack.push("(");
42                 else if(s[i] == ')'){
43                     while(!myStack.empty()){
44                         if(myStack.top() == "("){
45                             myStack.pop();
46                             break;
47                         }
48
49                         res.push_back(myStack.top());      myStack.pop();
50                     }
51                 }else{
52                     while(!myStack.empty()){
53                         if(myStack.top() == "("){
54                             break;

```

```

55         }
56
57         if(getPriority(myStack.top()[0]) <= getPriority(s[i])){
58             res.push_back(myStack.top());      myStack.pop();
59         }else{
60             break;
61         }
62     }
63
64     myStack.push(string(1, s[i]));
65 }
66
67 }
68 }
69
70 if(num != -1)
71     myStack.push(to_string(num));
72 while(!myStack.empty()){
73     res.push_back(myStack.top()); myStack.pop();
74 }
75
76 return res;
77 }
78
79
80
81 int getPriority(char ch){
82     if(ch == '+' || ch == '-')
83         return 1;
84     else
85         return 0;
86 }
87
88 int evl(vector<string>& vec) {
89     stack<int> myStack;
90     int res = 0;
91     for(string& str : vec){
92         if(str[0] >= '0' && str[0] <= '9'){ //number
93             myStack.push(parseInt(str));
94         }else{ //ops
95             int op1 = myStack.top();    myStack.pop();
96             int op2 = myStack.top();    myStack.pop();
97
98             int temp = 0;
99             if(str == "+")
100                 temp = op1 + op2;
101             else if(str == "-")
102                 temp = op2 - op1;
103             else if(str == "*")

```

```

104         temp = op1 * op2;
105     else
106         temp = op2 / op1;
107
108     myStack.push(temp);
109 }
110 }
111
112     return myStack.top();
113 }
114
115 int parseInt(string s){
116     int res = 0;
117     for(int i = 0; i < s.size(); i++){
118         if(res > INT_MAX / 10 || (res == INT_MAX / 10 && s[i] - '0' > INT_MAX
119 % 10))
120             return INT_MAX;
121
122         res *= 10;
123         res += s[i] - '0';
124     }
125     return res;
126 }
127 };

```

## 773 Sliding Puzzle

---

## 773. Sliding Puzzle

难度 困难

👍 226



On an  $2 \times 3$  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  $\begin{bmatrix} 1, 2, 3 \\ 4, 5, 0 \end{bmatrix}$ .

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:

1	2	3
4		5

**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

1.6k 0.000

```
1
2 class Solution {
3 public:
4     int row, col;
5     vector<vector<int>>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
6     const string FINAL = "123450";
7     int slidingPuzzle(vector<vector<int>>>& board) {
8         row = board.size();
9         col = board[0].size();
10    }
```

```

11     auto isInRange = [&](int i, int j){return i >= 0 && j >= 0 && i < row && j
    < col;};
12
13     unordered_set<string> set;
14     string start = getState(board);
15     if(start == FINAL)
16         return 0;
17
18     set.insert(start);
19
20     queue<string> myQueue;
21     myQueue.push(start);
22
23     int round = 0;
24     while(!myQueue.empty()){
25         int size = myQueue.size();
26         for(int i = 0; i < size; i++){
27             string curBoard = myQueue.front(); myQueue.pop();
28             //cout << " 当前" << curBoard << endl;
29             int pos = curBoard.find('0');
30             int posX = pos / 3;
31             int posY = pos % 3;
32
33             for(int k = 0; k < 4; k++){
34                 int newX = posX + dir[k][0];
35                 int newY = posY + dir[k][1];
36                 int newPos = newX * col + newY;
37
38                 string nextState = curBoard;
39
40                 if(isInRange(newX, newY)){
41                     swap(nextState[pos], nextState[newPos]);
42
43                     if(nextState == FINAL)
44                         return round + 1;
45                     if(set.count(nextState) == 0){
46                         myQueue.push(nextState);
47                         set.insert(nextState);
48                     }
49
50                 }
51             }
52
53         }
54
55         round++;
56         //     cout << "---" << endl;
57     }
58

```

```

59         return -1;
60     }
61
62     string getState(vector<vector<int>>& board){
63         string res;
64         for(int i = 0; i < row; i++){
65             for(int j = 0; j < col; j++){
66                 res += to_string(board[i][j]);
67             }
68         }
69
70         return res;
71     }
72 };

```

```

1  class Solution {
2  public:
3      int row, col;
4      vector<vector<int>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
5      const string FINAL = "123@450";
6      int slidingPuzzle(vector<vector<int>>& board) {
7          row = board.size();
8          col = board[0].size();
9
10         auto isInRange = [&](int i, int j){return i >= 0 && j >= 0 && i < row && j
< col;};
11
12         unordered_set<string> set;
13         string start = getState(board);
14         if(start == FINAL)
15             return 0;
16
17         set.insert(start);
18
19         queue<string> myQueue;
20         myQueue.push(start);
21
22         int round = 0;
23         while(!myQueue.empty()){
24             int size = myQueue.size();
25             for(int i = 0; i < size; i++){
26                 string curBoard = myQueue.front(); myQueue.pop();
27                 vector<vector<int>> newBord = getBoard(curBoard);
28
29                 pair<int, int> zeroPos = getZero(newBord);

```

```

30         for(int k = 0; k < 4; k++){
31             int newX = zeroPos.first + dir[k][0];
32             int newY = zeroPos.second + dir[k][1];
33
34             if(isInRange(newX, newY)){
35                 newBord[zeroPos.first][zeroPos.second] = newBord[newX]
[newY];
36                 newBord[newX][newY] = 0;
37
38                 string nextState = getState(newBord);
39                 if(nextState == FINAL)
40                     return round + 1;
41                 if(set.count(nextState) == 0){
42                     myQueue.push(nextState);
43                     set.insert(nextState);
44                 }
45
46                 newBord[newX][newY] = newBord[zeroPos.first]
[zeroPos.second];
47                 newBord[zeroPos.first][zeroPos.second] = 0;
48             }
49         }
50
51     }
52
53     round++;
54 }
55
56 return -1;
57 }
58
59 pair<int, int> getZero(vector<vector<int>>& board){
60     for(int i = 0; i < row; i++){
61         for(int j = 0; j < col; j++){
62             if(board[i][j] == 0)
63                 return {i, j};
64         }
65     }
66
67     return {-1, -1};
68 }
69
70 vector<vector<int>> getBoard(string& curBoard){
71     vector<vector<int>> res(row, vector<int>(col, 0));
72     for(int i = 0; i < 3; i++){
73         res[0][i] = curBoard[i] - '0';
74     }
75
76

```

```

77         for(int i = 4; i < 7; i++){
78             res[1][i - 4] = curBoard[i] - '0';
79         }
80
81         return res;
82     }
83
84     string getState(vector<vector<int>>& board){
85         string res = "";
86         for(int i = 0; i < row; i++){
87             for(int j = 0; j < col; j++){
88                 res += to_string(board[i][j]);
89             }
90
91             if(i != row - 1)
92                 res += "@";
93         }
94
95         return res;
96     }
97 };
98

```

## 777 Swap Adjacent in LR Strign

```

1  //39 / 94 超时
2  public class Solution {
3      /**
4       * The Algorithm is kind of like backtrack
5       * where I try for every possible way
6       * then to add into our result set
7       * @param start
8       * @param end
9       * @return
10      */
11
12      int counter = 0;
13      public boolean canTransform(String start, String end) {
14          backtrack(start.toCharArray(), end.toCharArray());
15          return counter != 0;
16      }

```



```

17
18 private void backtrack(char[] start, char[] end) {
19     if(counter >= 2){
20         return;
21     }
22
23     if(Arrays.equals(start, end)){
24         counter++;
25         return;
26     }
27
28     if(noExchange(start)){
29         return;
30     }
31
32     for(int i = 0; i < start.length - 1; i++){
33         if(start[i] == 'X' && start[i + 1] == 'L'){
34             start[i] = 'L';
35             start[i + 1] = 'X';
36
37             backtrack(start, end);
38
39             start[i] = 'X';
40             start[i + 1] = 'L';
41         }
42
43         if(counter >= 2)
44             return;
45
46         if(start[i] == 'R' && start[i + 1] == 'X'){
47             start[i] = 'X';
48             start[i + 1] = 'R';
49
50             backtrack(start, end);
51
52             start[i] = 'R';
53             start[i + 1] = 'X';
54         }
55     }
56 }
57
58 private boolean noExchange(char[] starts) {
59     for(int i = 0; i < starts.length - 1; i++){
60         if(starts[i] == 'X' && starts[i + 1] == 'L')
61             return false;
62         if(starts[i] == 'R' && starts[i + 1] == 'X')
63             return false;
64     }
65

```

```
66         return true;
67     }
68 }
```

```
1  //BFS 超时 56 / 94
2  public boolean canTransform(String start, String end) {
3      if(start.equals(end))
4          return true;
5
6      Set<String> visited = new HashSet<>();
7      Deque<String> queue = new ArrayDeque<>();
8      queue.add(start);
9      visited.add(start);
10
11     while(!queue.isEmpty()){
12         int size = queue.size();
13         for(int i =0; i < size; i++){
14             String cur = queue.removeFirst();
15
16             if(cur.equals(end))
17                 return true;
18
19             char[] curs = cur.toCharArray();
20             for(int k = 0; k < curs.length - 1; k++){
21                 if(curs[k] == 'X' && curs[k + 1] == 'L'){
22                     curs[k] = 'L';
23                     curs[k + 1] = 'X';
24
25                     String str = toStr(curs);
26                     if(!visited.contains(str)){
27                         queue.addLast(str);
28                         visited.add(str);
29                     }
30
31                     curs[k] = 'X';
32                     curs[k + 1] = 'L';
33                 }
34
35                 if(curs[k] == 'R' && curs[k + 1] == 'X'){
36                     curs[k] = 'X';
37                     curs[k + 1] = 'R';
38
39                     String str = toStr(curs);
40                     if(!visited.contains(str)){
41                         queue.addLast(str);
42                         visited.add(str);
43                     }
44                 }
45             }
46         }
47     }
48 }
```

```

44
45         curs[k] = 'R';
46         curs[k + 1] = 'X';
47     }
48 }
49 }
50 }
51
52     return false;
53 }
54
55 private String toStr(char[] curs) {
56     StringBuilder sb = new StringBuilder();
57     for(char ch : curs)
58         sb.append(ch);
59
60     return sb.toString();
61 }

```

```

1     public boolean canTransform(String start, String end) {
2         StringBuilder sb1 = new StringBuilder();
3         StringBuilder sb2 = new StringBuilder();
4         int len1 = start.length();
5         int len2 = end.length();
6
7         for(int i = 0; i < start.length(); i++){
8             char ch = start.charAt(i);
9             if(ch != 'X')
10                 sb1.append(ch);
11         }
12
13         for(int i = 0; i < end.length(); i++){
14             char ch = end.charAt(i);
15             if(ch != 'X')
16                 sb2.append(ch);
17         }
18
19         if(!sb1.toString().equals(sb2.toString()))
20             return false;
21
22         int up = 0, down = 0;
23
24
25         while(up < len1 && down < len2){
26             while(up < len1 && start.charAt(up) != 'L')

```

```

27         up++;
28         while(down < len2 && end.charAt(down) != 'L')
29             down++;
30
31         if(up == len1 || down == len2){
32             if(noSymbol(start, up + 1, 'L', true) && noSymbol(end, down + 1,
'L', true))
33                 break;
34             else
35                 return false;
36         }
37
38         if(up < down)
39             return false;
40         up++;
41         down++;
42     }
43
44     up = len1 - 1;
45     down = len2 - 1;
46
47     while(up >= 0 && down >= 0){
48         while(up >= 0 && start.charAt(up) != 'R')
49             up--;
50         while(down >= 0 && end.charAt(down) != 'R')
51             down--;
52
53         if(up == 0 || down == 0){
54             if(noSymbol(start, up - 1, 'R', false) && noSymbol(end, down - 1,
'R', false))
55                 break;
56             else
57                 return false;
58         }
59
60         if(up > down)
61             return false;
62         up--;
63         down--;
64     }
65
66     return true;
67 }
68
69 private boolean noSymbol(String str, int index, char ch, boolean l2r) {
70     if(l2r){
71         for(int i = index; i < str.length(); i++){
72             if(str.charAt(i) == ch)
73                 return false;

```

```

74         }
75
76         return true;
77     }else{
78         for(int i = index; i >= 0; i--){
79             if(str.charAt(i) == ch)
80                 return false;
81         }
82
83         return true;
84     }
85
86 }
87

```

## 780 Reaching Points

```

1  class Solution {
2      public boolean reachingPoints(int sx, int sy, int tx, int ty) {
3          while(sx < tx && sy < ty){
4              if(tx < ty)
5                  ty -= tx;
6              else
7                  tx -= ty;
8          }
9
10         if(sx == tx && sy <= ty)
11             return (ty - sy) % sx == 0;
12         else
13             return sy == ty && sx <= tx && (tx - sx) % sy == 0;
14     }
15 }

```

```

1      public boolean reachingPoints(int sx, int sy, int tx, int ty) {
2          Deque<Point> queue = new ArrayDeque<>();
3          Point root = new Point(sx, sy);
4          Point end = new Point(tx, ty);
5          Set<Point> visited = new HashSet<>();
6          queue.addLast(root);
7

```

```

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;

```

```

7      }
8
9      public void inorder(TreeNode root){
10         if(root == null)
11             return;
12
13         inorder(root.left);
14
15         if(pre != null)
16             res = Math.min(res, Math.abs((long)(root.val) - (long)(pre.val)));
17
18         pre = root;
19
20         inorder(root.right);
21     }

```

## 787 Cheapest Flights Within K Stops

```

1  //超时， 通过 27 / 49 个案例
2  class Solution {
3      Map<Integer, List<String>> map = new HashMap<>();
4      int res = Integer.MAX_VALUE;
5      public int findCheapestPrice(int n, int[][] flights, int src, int dst, int k) {
6          for(int[] flight : flights){
7              map.putIfAbsent(flight[0], new ArrayList<>());
8              map.get(flight[0]).add(flight[1] + "@" + flight[2]);
9          }
10
11          dfs(dst, k, src, 0, 0, new boolean[n]);
12
13          return res == Integer.MAX_VALUE ? -1 : res;
14      }
15
16      private void dfs(int dst, int k, int cur, int step, int price, boolean[]
visited) {
17          if(cur == dst && k + 1 >= step){
18              res = Math.min(res, price);
19              return;
20          }
21
22          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()){

```



```

20         for(int i = 0; i < map.get(ch); i++)
21             res.append(ch);
22     }
23
24     return res.toString();
25 }

```

## 797 All Paths From Source to Target

执行用时: **7 ms** , 在所有 Java 提交中击败了 **10.26%** 的用户

内存消耗: **39.9 MB** , 在所有 Java 提交中击败了 **92.86%** 的用户

此题思路如下:

```

1  //典型 DFS
2  List<List<Integer>> res = new ArrayList<>();
3  int n;
4  public List<List<Integer>> allPathsSourceTarget(int[][] graph) {
5      n = graph.length;
6      List<Set<Integer>> bags = new ArrayList<>();
7      for(int i = 0; i <= n - 1; i++)
8          bags.add(new HashSet<>());
9
10     for(int i = 0; i < graph.length; i++){
11         int[] edges = graph[i];
12         for(int num : edges)
13             bags.get(i).add(num);
14     }
15     List<Integer> path = new ArrayList<>();
16     path.add(0);
17     dfs(bags, new HashSet<>(), path, 0);
18     return res;
19 }
20
21 private void dfs(List<Set<Integer>> bags, HashSet<Integer> visited,
22 List<Integer> path, int bagNum) {
23     if(bagNum == n - 1){
24         res.add(new ArrayList<>(path));
25     }
26 }

```

```
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 }
```