

t078

[华为OD机试 2025 B卷 - 德州扑克 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD机试 2025 B卷 - 模拟数据序列化传输 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD 机试 2025 B卷 - 相同数字组成图形的周长 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD 机试 2025 B卷 - 跳格子2 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD 机试 2025 B卷 - 周末爬山 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD机试 2025 B卷 - 手机App防沉迷系统 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD机试 2025 B卷 - 字符串摘要 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD机试 2025 B卷 - 出错的或电路 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD 机试 2025 B卷 - 快递运输 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD 机试 2025 B卷 - 求解连续序列 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

[华为OD 机试 2025 B卷 - 投篮大赛 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

华为OD机试 2025 B卷 - 德州扑克 (C++ & Python & JAVA & JS & GO)-CSDN博客

德州扑克

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

五张牌，每张牌由牌大小和花色组成，牌大小2~10、J、Q、K、A，牌花色为红桃、黑桃、梅花、方块四种花色之一。

判断牌型：

牌型1，同花顺：同一花色的顺子，如红桃2红桃3红桃4红桃5红桃6。

牌型2，四条：四张相同数字 + 单张，如红桃A黑桃A梅花A方块A + 黑桃K。

牌型3，葫芦：三张相同数字 + 一对，如红桃5黑桃5梅花5 + 方块9梅花9。

牌型4，同花：同一花色，如方块3方块7方块10方块J方块Q。

牌型5，顺子：花色不一样的顺子，如红桃2黑桃3红桃4红桃5方块6。

牌型6，三条：三张相同+两张单。

说明：

1. 五张牌里不会出现牌大小和花色完全相同的牌。
2. 编号小的牌型较大，如同花顺比四条大，依次类推。
3. 包含A的合法的顺子只有10 J Q K A和A 2 3 4 5;类似K A 2 3 4的序列不认为是顺子。

输入描述

输入由5行组成，每行为一张牌大小和花色，牌大小为2~10、J、Q、K、A，花色分别用字符H、S、C、D表示红桃、黑桃、梅花、方块。

输出描述

输出牌型序号，5张牌符合多种牌型时，取最大的牌型序号输出。

用例1

输入

▼

Plain Text |

1 4 H
2 5 S
3 6 C
4 7 D
5 8 D

输出

1 5

说明

| 4 5 6 7 8构成顺子，输出5

用例2

输入

1 9 S
2 5 S
3 6 S
4 7 S
5 8 S

输出

1 1

说明

| 既是顺子又是同花，输出1，同花顺

题解

思路： 模拟

1. 接收输入，将牌面值转换为可比较的int类型，便于后面顺子判断。
2. 使用 哈希表 统计各个花色出现的次数，使用 哈希表 统计各牌面值出现的次数。判断输入牌是否为顺子(此时需要特殊看待 A2345 组成的顺子情况)。
3. 经过2处理得到的数据之后，接下来按照牌型序号顺序进行判断手中牌属于那种情况即可。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<map>
8 #include<set>
9 using namespace std;
10 // 牌面值映射
11 map<string, int> mp = {
12     {"2", 2},
13     {"3", 3},
14     {"4", 4},
15     {"5", 5},
16     {"6", 6},
17     {"7", 7},
18     {"8", 8},
19     {"9", 9},
20     {"10", 10},
21     {"J", 11},
22     {"Q", 12},
23     {"K", 13},
24     {"A", 14},
25 };
26 // 花色映射
27 map<string, int> colorMap = {
28     {"H", 0},
29     {"S", 1},
30     {"C", 2},
31     {"D", 3},
32 };
33
34 // 判断是否为顺子
35
36 bool isShunzi(vector<int> nums) {
37     string tmp = "";
38     for (int i = 0; i < nums.size(); i++) {
39         tmp += nums[i];
40     }
41     // 特殊处理 A2345
42     if (tmp == "142345") {
43         return true;
44     }
45     for (int i = 1; i < nums.size(); i++) {
```

```

46         if (nums[i] != nums[i-1] + 1) {
47             return false;;
48         }
49     }
50     return true;
51 }
52 // 统计出现字符个数及对应个数
53 map<int, int> countNum(vector<int> nums) {
54     map<int, int> numMapping;
55     for (int i = 0; i < nums.size(); i++) {
56         numMapping[nums[i]]++;
57     }
58     return numMapping;
59 }
60
61 int main() {
62     vector<int> nums(5), colors(5);
63     for (int i = 0; i < 5; i++) {
64         string value, color;
65         cin >> value >> color;
66         nums[i] = mp[value];
67         colors[i] = colorMap[color];
68     }
69     // 排序便于判断顺子
70     sort(nums.begin(), nums.end());
71     bool isShun = isShunzi(nums);
72     map<int, int> colorsCount = countNum(colors);
73     map<int, int> numsCount = countNum(nums);
74
75     // 分类判断
76     if (isShun && colorsCount.size() == 1) {
77         cout << 1;
78     } else if (numsCount.size() == 2 && ((*numsCount.begin()).second == 4 ||
79 | (*numsCount.begin()).second == 1)) {
80         cout << 2;
81     } else if (numsCount.size() == 2 && ((*numsCount.begin()).second == 3 ||
82 | (*numsCount.begin()).second == 2)) {
83         cout << 3;
84     } else if (colorsCount.size() == 1) {
85         cout << 4;
86     } else if (isShun) {
87         cout << 5;
88     } else if (numsCount.size() == 3){
89         bool flag = true;
90         for (auto item : numsCount) {
91             if (item.second != 1 && item.second != 3) {
92                 flag = false;
93                 break;

```

```
92         }
93     }
94     if (flag) {
95         cout << 6;
96     }
97 }
98 return 0;
99 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         // 牌面值映射
6         Map<String, Integer> mp = new HashMap<>();
7         String[] values = {"2", "3", "4", "5", "6", "7", "8", "9", "10",
8         "J", "Q", "K", "A"};
9         for (int i = 0; i < values.length; i++) {
10            mp.put(values[i], i + 2);
11        }
12
13         // 花色映射
14         Map<String, Integer> colorMap = new HashMap<>();
15         colorMap.put("H", 0);
16         colorMap.put("S", 1);
17         colorMap.put("C", 2);
18         colorMap.put("D", 3);
19
20         Scanner scanner = new Scanner(System.in);
21         List<Integer> nums = new ArrayList<>();
22         List<Integer> colors = new ArrayList<>();
23
24         for (int i = 0; i < 5; i++) {
25             String value = scanner.next();
26             String color = scanner.next();
27             nums.add(mp.get(value));
28             colors.add(colorMap.get(color));
29         }
30         scanner.close();
31
32         // 排序便于判断顺子
33         Collections.sort(nums);
34         boolean isShun = isShunzi(nums);
35
36         // 统计出现次数
37         Map<Integer, Integer> colorsCount = countFrequency(colors);
38         Map<Integer, Integer> numsCount = countFrequency(nums);
39
40         // 分类判断
41         if (isShun && colorsCount.size() == 1) {
42             System.out.println(1);
43         } else if (numsCount.size() == 2 && (numsCount.containsValue(4) |
| numsCount.containsValue(1))) {
44             System.out.println(2);
45 }
```

```

44         } else if (numsCount.size() == 2 && (numsCount.containsValue(3) ||
45 | numsCount.containsValue(2))) {
46             System.out.println(3);
47         } else if (colorsCount.size() == 1) {
48             System.out.println(4);
49         } else if (isShun) {
50             System.out.println(5);
51         } else if (numsCount.size() == 3) {
52             boolean flag = true;
53             for (int value : numsCount.values()) {
54                 if (value != 1 && value != 3) {
55                     flag = false;
56                     break;
57                 }
58             }
59             if (flag) {
60                 System.out.println(6);
61             }
62         }
63     }
64
65     // 判断是否为顺子
66     private static boolean isShunzi(List<Integer> nums) {
67         if (nums.equals(Arrays.asList(14, 2, 3, 4, 5))) return true;
68         for (int i = 1; i < nums.size(); i++) {
69             if (nums.get(i) != nums.get(i - 1) + 1) return false;
70         }
71         return true;
72     }
73
74     // 统计元素出现的次数
75     private static Map<Integer, Integer> countFrequency(List<Integer> lis-
76 t) {
77         Map<Integer, Integer> countMap = new HashMap<>();
78         for (int num : list) {
79             countMap.put(num, countMap.getOrDefault(num, 0) + 1);
80         }
81         return countMap;
82     }
83 }
```

Python

```
1 import sys
2 from collections import Counter
3
4 # 牌面值映射
5 mp = {
6     "2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8": 8, "9": 9, "10": 10,
7     "J": 11, "Q": 12, "K": 13, "A": 14
8 }
9
10 # 花色映射
11 color_map = {"H": 0, "S": 1, "C": 2, "D": 3}
12
13
14 def is_shunzi(nums):
15     """判断是否为顺子"""
16     if nums == [2, 3, 4, 5, 14]: # A,2,3,4,5 特殊顺子
17         return True
18     return all(nums[i] == nums[i - 1] + 1 for i in range(1, len(nums)))
19
20
21 def main():
22     # 读取输入
23     nums = []
24     colors = []
25
26     for _ in range(5):
27         value, color = sys.stdin.readline().strip().split()
28         nums.append(mp[value])
29         colors.append(color_map[color])
30
31     # 排序便于判断顺子
32     nums.sort()
33     is_shun = is_shunzi(nums)
34
35     # 统计出现次数
36     colors_count = Counter(colors)
37     nums_count = Counter(nums)
38
39     # 分类判断
40     if is_shun and len(colors_count) == 1:
41         print(1) # 同花顺
42     elif len(nums_count) == 2 and (4 in nums_count.values() or 1 in nums_count.values()):
43         print(2) # 四条
```

```
44     elif len(nums_count) == 2 and (3 in nums_count.values() or 2 in nums_c
45         ount.values()):
46         print(3) # 葫芦
47     elif len(colors_count) == 1:
48         print(4) # 同花
49     elif is_shun:
50         print(5) # 顺子
51     elif len(nums_count) == 3:
52         if sorted(nums_count.values()) == [1, 1, 3]: # 只有一种三条
53             print(6) # 三条
54
55 if __name__ == "__main__":
56     main()
```

JavaScript

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  // 牌面值映射
5  const mp = {
6      "2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8": 8, "9": 9, "10": 10,
7      "J": 11, "Q": 12, "K": 13, "A": 14
8  };
9
10 // 花色映射
11 const colorMap = { "H": 0, "S": 1, "C": 2, "D": 3 };
12
13 /**
14  * 判断是否为顺子
15  * @param {number[]} nums - 牌的数值数组 (已排序)
16  * @returns {boolean}
17 */
18 function isShunzi(nums) {
19     if (JSON.stringify(nums) === JSON.stringify([2, 3, 4, 5, 14])) {
20         return true; // A,2,3,4,5 特殊顺子
21     }
22     for (let i = 1; i < nums.length; i++) {
23         if (nums[i] !== nums[i - 1] + 1) {
24             return false;
25         }
26     }
27     return true;
28 }
29
30 /**
31  * 统计数组中每个元素的出现次数
32  * @param {number[]} arr
33  * @returns {Map<number, number>}
34 */
35 function countOccurrences(arr) {
36     let countMap = new Map();
37     for (const num of arr) {
38         countMap.set(num, (countMap.get(num) || 0) + 1);
39     }
40     return countMap;
41 }
42
43 // 读取输入
44 const rl = readline.createInterface({
```

```

45     input: process.stdin,
46     output: process.stdout
47   });
48
49   let lines = [];
50   rl.on("line", (line) => {
51     lines.push(line.trim());
52     if (lines.length === 5) {
53       rl.close();
54     }
55   });
56
57   rl.on("close", () => {
58     let nums = [], colors = [];
59
60     for (let i = 0; i < 5; i++) {
61       let [value, color] = lines[i].split(" ");
62       nums.push(mp[value]);
63       colors.push(colorMap[color]);
64     }
65
66     // 排序便于判断顺子
67     nums.sort((a, b) => a - b);
68     let isShun = isShunzi(nums);
69
70     // 统计出现次数
71     let colorsCount = countOccurrences(colors);
72     let numsCount = countOccurrences(nums);
73
74     // 分类判断
75     if (isShun && colorsCount.size === 1) {
76       console.log(1); // 同花顺
77     } else if (numsCount.size === 2 && [...numsCount.values()].includes
78 (4) || [...numsCount.values()].includes(1)) {
79       console.log(2); // 四条
80     } else if (numsCount.size === 2 && [...numsCount.values()].includes
81 (3) || [...numsCount.values()].includes(2)) {
82       console.log(3); // 葫芦
83     } else if (colorsCount.size === 1) {
84       console.log(4); // 同花
85     } else if (isShun) {
86       console.log(5); // 顺子
87     } else if (numsCount.size === 3) {
88       let counts = [...numsCount.values()].sort();
89       if (JSON.stringify(counts) === JSON.stringify([1, 1, 3])) {
90         console.log(6); // 三条
91       }
92     }
93   }

```

```
91 } );
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "sort"
8     "strings"
9 )
10 // 牌面值映射
11 var valueMap = map[string]int{
12     "2": 2,
13     "3": 3,
14     "4": 4,
15     "5": 5,
16     "6": 6,
17     "7": 7,
18     "8": 8,
19     "9": 9,
20     "10": 10,
21     "J": 11,
22     "Q": 12,
23     "K": 13,
24     "A": 14,
25 }
26 // 花色映射
27 var colorMap = map[string]int{
28     "H": 0,
29     "S": 1,
30     "C": 2,
31     "D": 3,
32 }
33
34 // 判断是否为顺子
35 func isShunzi(nums []int) bool {
36     tmp := ""
37     for _, num := range nums {
38         tmp += fmt.Sprintf("%d", num)
39     }
40     // 特殊处理
41     if tmp == "142345" {
42         return true
43     }
44     for i := 1; i < len(nums); i++ {
45         if nums[i] != nums[i-1]+1 {
```

```
46         return false
47     }
48 }
49     return true
50 }
51 // 统计出现字符个数及对应个数
52 func countNum(nums []int) map[int]int {
53     count := make(map[int]int)
54     for _, num := range nums {
55         count[num]++
56     }
57     return count
58 }
59
60 func main() {
61     reader := bufio.NewReader(os.Stdin)
62     nums := make([]int, 5)
63     colors := make([]int, 5)
64
65     for i := 0; i < 5; i++ {
66         line, _ := reader.ReadString('\n')
67         line = strings.TrimSpace(line)
68         parts := split(line, " ")
69         value, color := parts[0], parts[1]
70         nums[i] = valueMap[value]
71         colors[i] = colorMap[color]
72     }
73     // 排序便于判断顺子
74     sort.Ints(nums)
75     isShun := isShunzi(nums)
76     colorsCount := countNum(colors)
77     numsCount := countNum(nums)
78     // 分类判断
79     if isShun && len(colorsCount) == 1 {
80         fmt.Println(1)
81     } else if len(numsCount) == 2 {
82         first := -1
83         for _, v := range numsCount {
84             first = v
85             break
86         }
87         if first == 4 || first == 1 {
88             fmt.Println(2)
89         } else if first == 3 || first == 2 {
90             fmt.Println(3)
91         }
92     } else if len(colorsCount) == 1 {
93         fmt.Println(4)
```

```
94     } else if isShun {
95         fmt.Println(5)
96     } else if len(numsCount) == 3 {
97         flag := true
98         for _, count := range numsCount {
99             if count != 1 && count != 3 {
100                 flag = false
101                 break
102             }
103         }
104         if flag {
105             fmt.Println(6)
106         }
107     }
108 }
```

德州扑克

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

五张牌，每张牌由牌大小和花色组成，牌大小2~10、J、Q、K、A，牌花色为红桃、黑桃、梅花、方块四种花色之一。

判断牌型：

牌型1，同花顺：同一花色的顺子，如红桃2红桃3红桃4红桃5红桃6。

牌型2，四条：四张相同数字 + 单张，如红桃A黑桃A梅花A方块A + 黑桃K。

牌型3，葫芦：三张相同数字 + 一对，如红桃5黑桃5梅花5 + 方块9梅花9。

牌型4，同花：同一花色，如方块3方块7方块10方块J方块Q。

牌型5，顺子：花色不一样的顺子，如红桃2黑桃3红桃4红桃5方块6。

牌型6，三条：三张相同+两张单。

说明：

1. 五张牌里不会出现牌大小和花色完全相同的牌。
2. 编号小的牌型较大，如同花顺比四条大，依次类推。
3. 包含A的合法的顺子只有10 J Q K A和A 2 3 4 5;类似K A 2 3 4的序列不认为是顺子。

输入描述

输入由5行组成，每行为一张牌大小和花色，牌大小为2~10、J、Q、K、A，花色分别用字符H、S、C、D表示红桃、黑桃、梅花、方块。

输出描述

输出牌型序号，5张牌符合多种牌型时，取最大的牌型序号输出。

用例1

输入

```
Plain Text |  
1 4 H  
2 5 S  
3 6 C  
4 7 D  
5 8 D
```

输出

```
Plain Text |  
1 5
```

说明

| 4 5 6 7 8构成顺子，输出5

用例2

输入

```
Plain Text |  
1 9 S  
2 5 S  
3 6 S  
4 7 S  
5 8 S
```

输出

```
Plain Text |  
1 1
```

说明

| 既是顺子又是同花，输出1，同花顺

题解

思路： 模拟

1. 接收输入，将牌面值转换为可比较的int类型，便于后面顺子判断。
2. 使用 哈希表 统计各个花色出现的次数，使用 哈希表 统计各牌面值出现的次数。判断输入牌是否为顺子(此时需要特殊看待 A2345 组成的顺子情况)。
3. 经过2处理得到的数据之后，接下来按照牌型序号顺序进行判断手中牌属于那种情况即可。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<map>
8 #include<set>
9 using namespace std;
10 // 牌面值映射
11 map<string, int> mp = {
12     {"2", 2},
13     {"3", 3},
14     {"4", 4},
15     {"5", 5},
16     {"6", 6},
17     {"7", 7},
18     {"8", 8},
19     {"9", 9},
20     {"10", 10},
21     {"J", 11},
22     {"Q", 12},
23     {"K", 13},
24     {"A", 14},
25 };
26 // 花色映射
27 map<string, int> colorMap = {
28     {"H", 0},
29     {"S", 1},
30     {"C", 2},
31     {"D", 3},
32 };
33
34 // 判断是否为顺子
35
36 bool isShunzi(vector<int> nums) {
37     string tmp = "";
38     for (int i = 0; i < nums.size(); i++) {
39         tmp += nums[i];
40     }
41     // 特殊处理 A2345
42     if (tmp == "142345") {
43         return true;
44     }
45     for (int i = 1; i < nums.size(); i++) {
```

```

46         if (nums[i] != nums[i-1] + 1) {
47             return false;;
48         }
49     }
50     return true;
51 }
52 // 统计出现字符个数及对应个数
53 map<int, int> countNum(vector<int> nums) {
54     map<int, int> numMapping;
55     for (int i = 0; i < nums.size(); i++) {
56         numMapping[nums[i]]++;
57     }
58     return numMapping;
59 }
60
61 int main() {
62     vector<int> nums(5), colors(5);
63     for (int i = 0; i < 5; i++) {
64         string value, color;
65         cin >> value >> color;
66         nums[i] = mp[value];
67         colors[i] = colorMap[color];
68     }
69     // 排序便于判断顺子
70     sort(nums.begin(), nums.end());
71     bool isShun = isShunzi(nums);
72     map<int, int> colorsCount = countNum(colors);
73     map<int, int> numsCount = countNum(nums);
74
75     // 分类判断
76     if (isShun && colorsCount.size() == 1) {
77         cout << 1;
78     } else if (numsCount.size() == 2 && ((*numsCount.begin()).second == 4 ||
79 | (*numsCount.begin()).second == 1)) {
80         cout << 2;
81     } else if (numsCount.size() == 2 && ((*numsCount.begin()).second == 3 ||
82 | (*numsCount.begin()).second == 2)) {
83         cout << 3;
84     } else if (colorsCount.size() == 1) {
85         cout << 4;
86     } else if (isShun) {
87         cout << 5;
88     } else if (numsCount.size() == 3){
89         bool flag = true;
90         for (auto item : numsCount) {
91             if (item.second != 1 && item.second != 3) {
92                 flag = false;
93                 break;

```

```
92         }
93     }
94     if (flag) {
95         cout << 6;
96     }
97 }
98 return 0;
99 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         // 牌面值映射
6         Map<String, Integer> mp = new HashMap<>();
7         String[] values = {"2", "3", "4", "5", "6", "7", "8", "9", "10",
8         "J", "Q", "K", "A"};
9         for (int i = 0; i < values.length; i++) {
10            mp.put(values[i], i + 2);
11        }
12
13         // 花色映射
14         Map<String, Integer> colorMap = new HashMap<>();
15         colorMap.put("H", 0);
16         colorMap.put("S", 1);
17         colorMap.put("C", 2);
18         colorMap.put("D", 3);
19
20         Scanner scanner = new Scanner(System.in);
21         List<Integer> nums = new ArrayList<>();
22         List<Integer> colors = new ArrayList<>();
23
24         for (int i = 0; i < 5; i++) {
25             String value = scanner.next();
26             String color = scanner.next();
27             nums.add(mp.get(value));
28             colors.add(colorMap.get(color));
29         }
30         scanner.close();
31
32         // 排序便于判断顺子
33         Collections.sort(nums);
34         boolean isShun = isShunzi(nums);
35
36         // 统计出现次数
37         Map<Integer, Integer> colorsCount = countFrequency(colors);
38         Map<Integer, Integer> numsCount = countFrequency(nums);
39
40         // 分类判断
41         if (isShun && colorsCount.size() == 1) {
42             System.out.println(1);
43         } else if (numsCount.size() == 2 && (numsCount.containsValue(4) |
| numsCount.containsValue(1))) {
44             System.out.println(2);
45 }
```

```

44         } else if (numsCount.size() == 2 && (numsCount.containsValue(3) ||
45 | numsCount.containsValue(2))) {
46             System.out.println(3);
47         } else if (colorsCount.size() == 1) {
48             System.out.println(4);
49         } else if (isShun) {
50             System.out.println(5);
51         } else if (numsCount.size() == 3) {
52             boolean flag = true;
53             for (int value : numsCount.values()) {
54                 if (value != 1 && value != 3) {
55                     flag = false;
56                     break;
57                 }
58             }
59             if (flag) {
60                 System.out.println(6);
61             }
62         }
63     }
64
65     // 判断是否为顺子
66     private static boolean isShunzi(List<Integer> nums) {
67         if (nums.equals(Arrays.asList(14, 2, 3, 4, 5))) return true;
68         for (int i = 1; i < nums.size(); i++) {
69             if (nums.get(i) != nums.get(i - 1) + 1) return false;
70         }
71         return true;
72     }
73
74     // 统计元素出现的次数
75     private static Map<Integer, Integer> countFrequency(List<Integer> lis-
76 t) {
77         Map<Integer, Integer> countMap = new HashMap<>();
78         for (int num : list) {
79             countMap.put(num, countMap.getOrDefault(num, 0) + 1);
80         }
81         return countMap;
82     }
83 }
```

Python

```
1 import sys
2 from collections import Counter
3
4 # 牌面值映射
5 mp = {
6     "2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8": 8, "9": 9, "10": 10,
7     "J": 11, "Q": 12, "K": 13, "A": 14
8 }
9
10 # 花色映射
11 color_map = {"H": 0, "S": 1, "C": 2, "D": 3}
12
13
14 def is_shunzi(nums):
15     """判断是否为顺子"""
16     if nums == [2, 3, 4, 5, 14]: # A,2,3,4,5 特殊顺子
17         return True
18     return all(nums[i] == nums[i - 1] + 1 for i in range(1, len(nums)))
19
20
21 def main():
22     # 读取输入
23     nums = []
24     colors = []
25
26     for _ in range(5):
27         value, color = sys.stdin.readline().strip().split()
28         nums.append(mp[value])
29         colors.append(color_map[color])
30
31     # 排序便于判断顺子
32     nums.sort()
33     is_shun = is_shunzi(nums)
34
35     # 统计出现次数
36     colors_count = Counter(colors)
37     nums_count = Counter(nums)
38
39     # 分类判断
40     if is_shun and len(colors_count) == 1:
41         print(1) # 同花顺
42     elif len(nums_count) == 2 and (4 in nums_count.values() or 1 in nums_count.values()):
43         print(2) # 四条
```

```
44     elif len(nums_count) == 2 and (3 in nums_count.values() or 2 in nums_c
45         ount.values()):
46         print(3) # 葫芦
47     elif len(colors_count) == 1:
48         print(4) # 同花
49     elif is_shun:
50         print(5) # 顺子
51     elif len(nums_count) == 3:
52         if sorted(nums_count.values()) == [1, 1, 3]: # 只有一种三条
53             print(6) # 三条
54
55 if __name__ == "__main__":
56     main()
```

JavaScript

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  // 牌面值映射
5  const mp = {
6      "2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8": 8, "9": 9, "10": 10,
7      "J": 11, "Q": 12, "K": 13, "A": 14
8  };
9
10 // 花色映射
11 const colorMap = { "H": 0, "S": 1, "C": 2, "D": 3 };
12
13 /**
14  * 判断是否为顺子
15  * @param {number[]} nums - 牌的数值数组 (已排序)
16  * @returns {boolean}
17 */
18 function isShunzi(nums) {
19     if (JSON.stringify(nums) === JSON.stringify([2, 3, 4, 5, 14])) {
20         return true; // A,2,3,4,5 特殊顺子
21     }
22     for (let i = 1; i < nums.length; i++) {
23         if (nums[i] !== nums[i - 1] + 1) {
24             return false;
25         }
26     }
27     return true;
28 }
29
30 /**
31  * 统计数组中每个元素的出现次数
32  * @param {number[]} arr
33  * @returns {Map<number, number>}
34 */
35 function countOccurrences(arr) {
36     let countMap = new Map();
37     for (const num of arr) {
38         countMap.set(num, (countMap.get(num) || 0) + 1);
39     }
40     return countMap;
41 }
42
43 // 读取输入
44 const rl = readline.createInterface({
```

```

45     input: process.stdin,
46     output: process.stdout
47   });
48
49   let lines = [];
50   rl.on("line", (line) => {
51     lines.push(line.trim());
52     if (lines.length === 5) {
53       rl.close();
54     }
55   });
56
57   rl.on("close", () => {
58     let nums = [], colors = [];
59
60     for (let i = 0; i < 5; i++) {
61       let [value, color] = lines[i].split(" ");
62       nums.push(mp[value]);
63       colors.push(colorMap[color]);
64     }
65
66     // 排序便于判断顺子
67     nums.sort((a, b) => a - b);
68     let isShun = isShunzi(nums);
69
70     // 统计出现次数
71     let colorsCount = countOccurrences(colors);
72     let numsCount = countOccurrences(nums);
73
74     // 分类判断
75     if (isShun && colorsCount.size === 1) {
76       console.log(1); // 同花顺
77     } else if (numsCount.size === 2 && [...numsCount.values()].includes
78 (4) || [...numsCount.values()].includes(1)) {
79       console.log(2); // 四条
80     } else if (numsCount.size === 2 && [...numsCount.values()].includes
81 (3) || [...numsCount.values()].includes(2)) {
82       console.log(3); // 葫芦
83     } else if (colorsCount.size === 1) {
84       console.log(4); // 同花
85     } else if (isShun) {
86       console.log(5); // 顺子
87     } else if (numsCount.size === 3) {
88       let counts = [...numsCount.values()].sort();
89       if (JSON.stringify(counts) === JSON.stringify([1, 1, 3])) {
90         console.log(6); // 三条
91       }
92     }
93   }

```

```
91 } );
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "sort"
8     "strings"
9 )
10 // 牌面值映射
11 var valueMap = map[string]int{
12     "2": 2,
13     "3": 3,
14     "4": 4,
15     "5": 5,
16     "6": 6,
17     "7": 7,
18     "8": 8,
19     "9": 9,
20     "10": 10,
21     "J": 11,
22     "Q": 12,
23     "K": 13,
24     "A": 14,
25 }
26 // 花色映射
27 var colorMap = map[string]int{
28     "H": 0,
29     "S": 1,
30     "C": 2,
31     "D": 3,
32 }
33
34 // 判断是否为顺子
35 func isShunzi(nums []int) bool {
36     tmp := ""
37     for _, num := range nums {
38         tmp += fmt.Sprintf("%d", num)
39     }
40     // 特殊处理
41     if tmp == "142345" {
42         return true
43     }
44     for i := 1; i < len(nums); i++ {
45         if nums[i] != nums[i-1]+1 {
```

```
46         return false
47     }
48 }
49     return true
50 }
51 // 统计出现字符个数及对应个数
52 func countNum(nums []int) map[int]int {
53     count := make(map[int]int)
54     for _, num := range nums {
55         count[num]++
56     }
57     return count
58 }
59
60 func main() {
61     reader := bufio.NewReader(os.Stdin)
62     nums := make([]int, 5)
63     colors := make([]int, 5)
64
65     for i := 0; i < 5; i++ {
66         line, _ := reader.ReadString('\n')
67         line = strings.TrimSpace(line)
68         parts := split(line, " ")
69         value, color := parts[0], parts[1]
70         nums[i] = valueMap[value]
71         colors[i] = colorMap[color]
72     }
73     // 排序便于判断顺子
74     sort.Ints(nums)
75     isShun := isShunzi(nums)
76     colorsCount := countNum(colors)
77     numsCount := countNum(nums)
78     // 分类判断
79     if isShun && len(colorsCount) == 1 {
80         fmt.Println(1)
81     } else if len(numsCount) == 2 {
82         first := -1
83         for _, v := range numsCount {
84             first = v
85             break
86         }
87         if first == 4 || first == 1 {
88             fmt.Println(2)
89         } else if first == 3 || first == 2 {
90             fmt.Println(3)
91         }
92     } else if len(colorsCount) == 1 {
93         fmt.Println(4)
```

```
94     } else if isShun {
95         fmt.Println(5)
96     } else if len(numsCount) == 3 {
97         flag := true
98         for _, count := range numsCount {
99             if count != 1 && count != 3 {
100                 flag = false
101                 break
102             }
103         }
104         if flag {
105             fmt.Println(6)
106         }
107     }
108 }
```

| 来自: 华为OD机试 2025 B卷 – 德州扑克 (C++ & Python & JAVA & JS & GO)-CSDN博客

| 来自: 华为OD机试 2025 B卷 – 德州扑克 (C++ & Python & JAVA & JS & GO)-CSDN博客

华为OD机试 2025 B卷 - 模拟数据序列化传输

(C++ & Python & JAVA & JS & GO)-CSDN博客

模拟数据序列化传输

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

模拟一套简化的序列化传输方式，请实现下面的数据编码与解码过程

编码前数据格式为 [位置,类型,值]，多个数据的时候用逗号分隔，位置仅支持数字，不考虑重复等场景；类

型仅支持：Integer / String / Compose（Compose的数据类型表示该存储的数据也需要编码）。

编码后数据参考图示，数据区的格式是：位置#类型#长度#数据，类型存储需要编码，Integer->0；
String->1；Compose->2，长度是指数据的字符长度；数据仅允许数字、大小写字母、空格。



输入的编码字符长度不能超过1000，一个数据的格式错误，则解析剩下数据，其他错误输出
ENCODE_ERROR。

输入的解码字符不能超过1000，数据区异常则跳过继续解析剩余数据区，其他异常输出
DECODE_ERROR。

输入描述

输入有两行：

第一行是命令，1表示编码，2表示解码，第二行输入待编码、解码的字符串
数据最多嵌套10层，[1,Compose,[1,String,Second]] 为2层嵌套

输出描述

如果输入要求是编码，则输出编码结果；如果输入要求是解码，则输出解码结果；当异常时输出对应的错误字符

用例1

输入

```
Plain Text |  
1 1  
2 [1,String,I am Mary],[2, Integer,23],[3, Long,1000000], [4, Compose, [1, String,  
I am Kitty],[2, Integer,44]]
```

输出

```
Plain Text |  
1 1#1#9#I am Mary2#0#2#234#2#25#1#1#10#I am Kitty2#0#2#44
```

说明

由于Long型未不支持类型，所以数据[3,Long,1000000]自动被过滤

用例2

输入

```
Plain Text |  
1 2  
2 1#1#9#I am Mary2#0#2#233#0#3#8794#2#25#1#1#10#I am Kitty2#0#2#44
```

输出

```
Plain Text |  
1 [1,String,I am Mary],[2, Integer,23],[3, Integer,879],[4, Compose, [1, String,I  
am Kitty],[2, Integer,44]]
```

用例3

输入

Plain Text |

```
1 2  
2 xxx
```

输出

Plain Text |

```
1 DECODE_ERROR
```

说明

| 输入待解码数据不满足格式要求

用例4

输入

Plain Text |

```
1 1  
2 [1,String,I am Marry],[2,Integer,23]],
```

输出

Plain Text |

```
1 ENCODE_ERROR
```

说明

| 输入格式不满足输入格式要求

题解

思路： 字符串处理 + 递归

1. 此题考察字符串处理，以及考察处理嵌套类型时递归的应用。
2. 编码和解码的情况分别定义一个函数去进行处理。在编码和解码两种情况下对于嵌套类型的处理都可以使用递归去进行处理。
3. 编码和解码具体逻辑可以参照下面对应代码实现。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 using namespace std;
10
11 map<string,int> typeLength = {{"0", 7}, {"1", 6}, {"2", 7}};
12 map<string,string> typeToStringMp = {{"0", "Integer"}, {"1", "String"}, {"2", "Compose"}};
13 // 标志是否出现格式错误
14 bool flag = false;
15
16 struct Item {
17     // 位置
18     string pos;
19     // 长度
20     int count;
21     // 类型
22     string type;
23     // 值
24     string value;
25     vector<Item> children;
26     Item(string pos, string type, int count, string value) : pos(pos), count(count), value(value), type(type){}
27 };
28
29 // 最顶层按照括号进行切割
30 vector<string> splitTopLevelItem(string& input) {
31     vector<string> res;
32     int depth = 0;
33     int start = 0;
34     int n = input.size();
35     for (int i = 0; i < n; i++) {
36         char c = input[i];
37         if (c == '[') {
38             if (depth == 0) {
39                 start = i;
40             }
41             depth += 1;
42         } else if (c == ']') {
43             depth -= 1;
```

```

44         if (depth == 0) {
45             string tmp = input.substr(start, i - start + 1);
46             res.push_back(tmp);
47         }
48     }
49     // 出现括号乱序问题 右括号大于左括号数量
50     if (depth < 0) {
51         flag = true;
52         break;
53     }
54 }
55
56 // 括号左右对数不匹配
57 if (depth != 0) {
58     flag = true;
59 }
60 return res;
61 }
62
63 // 编码
64 vector<Item> encode(string& input) {
65     vector<Item> res;
66     if (input.empty()) {
67         return res;
68     }
69     // 解析输入是否已[ 开头 以 ] 结尾
70     if (input[0] != '[' || input.back() != ']') {
71         flag = true;
72         return res;
73     }
74
75     vector<string> ans = splitTopLevelItem(input);
76     // 处理字符串过程中括号对数不匹配, 异常情况
77     if (flag) {
78         return res;
79     }
80     int n = ans.size();
81
82     for (int i = 0; i < n; i++) {
83         string itemStr = ans[i];
84         // 取出左右边界括号
85         itemStr.pop_back();
86         itemStr = itemStr.substr(1, itemStr.size()-1);
87         vector<string> currentAns;
88         string tmp = "";
89         int m = itemStr.size();
90         int j = 0;
91

```

```

92     for ( ; j < m; j++) {
93         char c = itemStr[j];
94         if (c != ',') {
95             tmp += c;
96         } else {
97             currentAns.push_back(tmp);
98             tmp = "";
99             if (currentAns.size() == 2) {
100                 break;
101             }
102         }
103     }
104
105     string type = currentAns.back();
106     // 值类型
107     if (type == "Integer" || type == "String") {
108         string value = itemStr.substr(j+1);
109         int count = value.size();
110         string typeStr = currentAns[1] == "Integer" ? "0" : "1";
111         res.push_back({currentAns[0], typeStr, count, value});
112         // 复合类型
113     } else if (type == "Compose") {
114         // 下一层嵌套的内容
115         string childrenItemStr = itemStr.substr(j+1);
116         // 递归处理
117         vector<Item> children = encode(childrenItemStr);
118         // 计算长度
119         int m = children.size();
120         // 计算长度 嵌套类型的总长度 = 各子片段的长度 + 各子长度类型字符串的长
度
121         int count = 0;
122         string typeSrtr = "2";
123         for (int i = 0; i < m; i++) {
124             Item item = children[i];
125             count += item.count;
126             count += typeLength[item.type];
127         }
128         res.push_back({currentAns[0], typeSrtr, count, ""});
129         res.back().children = children;
130     }
131 }
132 return res;
133 }
134
135 // 递归格式化输出编码结果
136 void formatEncodePrint(vector<Item> items) {
137     int n = items.size();
138     for (int i = 0; i < n; i++) {

```

```

139         Item item = items[i];
140         if (item.type == "0" || item.type == "1") {
141             cout << item.pos << "#" << item.type << "#" << item.count <<
142             "#" << item.value;
143         } else {
144             cout << item.pos << "#" << item.type << "#" << item.count <<
145             "#";
146             formatEncodePrint(item.children);
147         }
148     }
149
150 // 递归格式化输出解码结果
151 void formatDecodePrint(vector<Item> items) {
152     int n = items.size();
153     for (int i = 0; i < n; i++) {
154         Item item = items[i];
155         if (item.type == "0" || item.type == "1") {
156             cout << "[" << item.pos << "," << typeToStringMp[item.type] <
157             < "," << item.value << "]";
158         } else {
159             cout << "[" << item.pos << "," << typeToStringMp[item.type] <
160             < ",";
161             formatDecodePrint(item.children);
162             cout << "]";
163         }
164         if (i != n-1) {
165             cout << ",";
166         }
167     }
168
169 // 通用 split 函数
170 vector<string> split(const string& str, const string& delimiter) {
171     vector<string> result;
172     size_t start = 0;
173     size_t end = str.find(delimiter);
174     while (end != string::npos) {
175         result.push_back(str.substr(start, end - start));
176         start = end + delimiter.length();
177         end = str.find(delimiter, start);
178     }
179     // 添加最后一个部分
180     result.push_back(str.substr(start));
181     return result;
182 }
```

```
183  
184 // 获取子复合类型的字符串内容  
185 string getChildrenInput(vector<string>& parts, int& index, int count) {  
186     // id  
187     string pos = "";  
188     // 类型  
189     string type = "";  
190     // 值  
191     string value = "";  
192     int subCount = -1;  
193  
194     // 结果  
195     string res = "";  
196  
197     int n = parts.size();  
198     int tmpCount = 0;  
199  
200     while (index < n && tmpCount < count) {  
201         if (pos == "") {  
202             pos = parts[index];  
203         } else if(type == "") {  
204             type = parts[index];  
205         } else if (subCount == -1){  
206             string subCountStr = parts[index];  
207             subCount = stoi(subCountStr);  
208         } else {  
209             tmpCount += typeLength[type] + subCount;  
210             if (type == "0" || type == "1") {  
211                 value = parts[index];  
212                 string remainngStr = value.substr(subCount);  
213                 value = value.substr(0, subCount);  
214                 parts[index] = remainngStr;  
215  
216                 res += pos + "#" + type + "#" + to_string(subCount) + "#"  
217                 + value;  
218                 index--;  
219             } else {  
220                 res += pos + "#" + type + "#" + to_string(subCount) + "#";  
221                 string childrenStr = getChildrenInput(parts, index, subCo  
unt);  
222                 res += childrenStr;  
223                 index--;  
224             }  
225             // 重置  
226             pos = "";  
227             type = "";  
228             value = "";  
229             subCount = -1;
```

```

229         }
230         index++;
231     }
232     // 无法找到指定长度的子片段
233     if (tmpCount != count) {
234         flag = true;
235     }
236     return res;
237 }
238

239 // 解码
240 vector<Item> decode(string& input) {
241     vector<Item> res;
242     vector<string> parts = split(input, "#");
243     int m = parts.size();
244     string pos = "";
245     string type = "";
246     string value = "";
247     int count = -1;
248     for (int i = 0; i < m; i++) {
249         // 按顺序赋值
250         if (pos == "") {
251             pos = parts[i];
252         } else if (type == "") {
253             type = parts[i];
254         } else if (count == -1) {
255             string countStr = parts[i];
256             count = stoi(countStr);
257         } else {
258             // 值类型
259             if (type == "1" || type == "0") {
260                 value = parts[i];
261                 // 两种情况:
262                 // 1. 最后一个数据, 长度应该等于count
263                 // 2. 不是最后一个数组, 长度应该大于count , 由 内容 + 下一个数据
264                 的位置编号
265                 if ((i != m - 1 && value.size() <= count) || (i == (m -
266                     1) && value.size() != count)) {
267                     flag = true;
268                     break;
269                 }
270                 string remainngStr = value.substr(count);
271                 value = value.substr(0, count);
272                 res.push_back({pos, type, count, value});
273                 parts[i] = remainngStr;
274                 // 因为最后一个元素的内容和下一个元素的id会在一起
275                 i--;
276             } else {

```

```

275         int tmpCount = 0;
276         int j = i;
277         string childrenInput = "";
278         // 通过长度找出复合类型的内容编码字符串
279         childrenInput = getChildrenInput(parts, j, count);
280         if (flag) {
281             break;
282         }
283         res.push_back({pos, type, count, ""});
284         vector<Item> childrenItem = decode(childrenInput);
285         res.back().children = childrenItem;
286         // 因为最后一个元素的内容和下一个元素的id会在一起
287         i = j-1;
288     }
289
290     // 重置
291     pos = "";
292     type = "";
293     count = -1;
294     value = "";
295 }
296 }
297
298 // 数据不完整
299 if (pos != "" || type != "" || count != -1 || value != "") {
300     flag = true;
301 }
302
303 return res;
304 }
305
306 int main() {
307     int command;
308     string input;
309     cin >> command;
310     cin.ignore();
311     getline(cin, input);
312     if (command == 1) {
313         vector<Item> items = encode(input);
314         if (flag) {
315             cout << "ENCODE_ERROR";
316         } else {
317             formatEncodePrint(items);
318         }
319
320     } else if (command == 2) {
321         vector<Item> items = decode(input);
322         if (flag) {

```

```
323         cout << "DECODE_ERROR";
324     } else {
325         formatDecodePrint(items);
326     }
327 }
328 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     // 类型长度映射
5     static Map<String, Integer> typeLength = new HashMap<>() {{
6         put("0", 7);
7         put("1", 6);
8         put("2", 7);
9     }};
10    // 类型字符串映射
11    static Map<String, String> typeToStringMp = new HashMap<>() {{
12        put("0", "Integer");
13        put("1", "String");
14        put("2", "Compose");
15    }};
16    // 标志是否出现格式错误
17    static boolean flag = false;
18
19    static class Item {
20        // 位置
21        String pos;
22        // 长度
23        int count;
24        // 类型 数字形式
25        String type;
26        // 值类型的值
27        String value;
28        // 复合类型的子数据
29        List<Item> children;
30
31        Item(String pos, String type, int count, String value) {
32            this.pos = pos;
33            this.type = type;
34            this.count = count;
35            this.value = value;
36            this.children = new ArrayList<>();
37        }
38    }
39
40    // 最顶层按照括号进行切割
41    static List<String> splitTopLevelItem(String input) {
42        List<String> res = new ArrayList<>();
43        int depth = 0;
44        int start = 0;
45        int n = input.length();
```

```

46         for (int i = 0; i < n; i++) {
47             char c = input.charAt(i);
48             if (c == '[') {
49                 if (depth == 0) {
50                     start = i;
51                 }
52                 depth++;
53             } else if (c == ']') {
54                 depth--;
55                 if (depth == 0) {
56                     String tmp = input.substring(start, i + 1);
57                     res.add(tmp);
58                 }
59             }
60             // 出现括号乱序问题 右括号大于左括号数量
61             if (depth < 0) {
62                 flag = true;
63                 break;
64             }
65         }
66     }
67
68     // 括号左右对数不匹配
69     if (depth != 0) {
70         flag = true;
71     }
72     return res;
73 }
74
75 // 编码
76 static List<Item> encode(String input) {
77     List<Item> res = new ArrayList<>();
78     if (input.isEmpty()) {
79         return res;
80     }
81
82     // 解析输入是否已[ 开头 以 ] 结尾
83     if (input.charAt(0) != '[' || input.charAt(input.length() - 1) !=
84         ']') {
85         flag = true;
86         return res;
87     }
88
89     List<String> ans = splitTopLevelItem(input);
90     // 处理字符串过程中括号对数不匹配,出现异常情况,结束
91     if (flag) {
92         return res;
93     }
94     int n = ans.size();

```

```

93
94     for (int i = 0; i < n; i++) {
95         String itemStr = ans.get(i);
96         // 取出左右边界括号
97         itemStr = itemStr.substring(1, itemStr.length() - 1);
98
99
100    List<String> currentAns = new ArrayList<>();
101    StringBuilder tmp = new StringBuilder();
102    int m = itemStr.length();
103    int j = 0;
104
105    for (; j < m; j++) {
106        char c = itemStr.charAt(j);
107        if (c != ',') {
108            tmp.append(c);
109        } else {
110            currentAns.add(tmp.toString());
111            tmp.setLength(0);
112            if (currentAns.size() == 2) {
113                break;
114            }
115        }
116    }
117    // 根据类型进行不同处理
118    String type = currentAns.get(currentAns.size() - 1);
119    // 值类型
120    if (type.equals("Integer") || type.equals("String")) {
121        String value = itemStr.substring(j + 1);
122        int count = value.length();
123        String typeStr = currentAns.get(1).equals("Integer") ?
124            "0" : "1";
125        res.add(new Item(currentAns.get(0), typeStr, count, value));
126    }
127    // 复合类型
128    } else if (type.equals("Compose")) {
129        // 下一层嵌套的内容
130        String childrenItemStr = itemStr.substring(j + 1);
131        // 递归处理 获得子元素
132        List<Item> children = encode(childrenItemStr);
133        // 计算长度
134        int m2 = children.size();
135        // 计算长度 嵌套类型的总长度 = 各子片段的长度 + 各子长度类型字符串
136        // 的长度
137        int count = 0;

```

的长度

```

138        String typeStr = "2";
139        for (int k = 0; k < m2; k++) {
140            Item item = children.get(k);
141            count += item.count;

```

```

138         count += typeLength.get(item.type);
139     }
140     // 添加一个复合类型
141     Item composeItem = new Item(currentAns.get(0), typeStr, c
142     ount, "");
143     composeItem.children = children;
144     res.add(composeItem);
145   }
146   return res;
147 }
148
149 // 递归格式化输出编码结果
150 static void formatEncodePrint(List<Item> items) {
151   for (Item item : items) {
152     if (item.type.equals("0") || item.type.equals("1")) {
153       System.out.print(item.pos + "#" + item.type + "#" + item.
154       count + "#" + item.value);
155     } else {
156       System.out.print(item.pos + "#" + item.type + "#" + item.
157       count + "#");
158       formatEncodePrint(item.children);
159     }
160   }
161
162 // 递归格式化输出解码结果
163 static void formatDecodePrint(List<Item> items) {
164   int n = items.size();
165   for (int i = 0; i < n; i++) {
166     Item item = items.get(i);
167     if (item.type.equals("0") || item.type.equals("1")) {
168       System.out.print("[ " + item.pos + "," + typeToStringMp.ge
169       t(item.type) + "," + item.value + " ]");
170     } else {
171       System.out.print("[ " + item.pos + "," + typeToStringMp.ge
172       t(item.type) + "," );
173       formatDecodePrint(item.children);
174       System.out.print(" ]");
175     }
176   }
177 }
178
179 // 通用 split 函数
180 static List<String> split(String str, String delimiter) {

```

```

181     List<String> result = new ArrayList<>();
182     int start = 0;
183     int end = str.indexOf(delimiter);
184     while (end != -1) {
185         result.add(str.substring(start, end));
186         start = end + delimiter.length();
187         end = str.indexOf(delimiter, start);
188     }
189     // 添加最后一个部分
190     result.add(str.substring(start));
191     return result;
192 }
193
194 // 关键: 获取子复合类型的字符串内容
195 static String getChildrenInput(List<String> parts, int[] index, int c
ount) {
196     String pos = "";
197     String type = "";
198     String value = "";
199     int subCount = -1;
200
201     StringBuilder res = new StringBuilder();
202
203     int n = parts.size();
204     int tmpCount = 0;
205
206     while (index[0] < n && tmpCount < count) {
207         if (pos.equals("")) {
208             pos = parts.get(index[0]);
209         } else if (type.equals("")) {
210             type = parts.get(index[0]);
211         } else if (subCount == -1) {
212             subCount = Integer.parseInt(parts.get(index[0]));
213         } else {
214             // 累计长度: 类型长度 + 数据长度
215             tmpCount += typeLength.get(type) + subCount;
216
217             if (type.equals("0") || type.equals("1")) {
218                 value = parts.get(index[0]);
219                 if (value.length() < subCount) {
220                     // 数据不足, 格式错误
221                     flag = true;
222                     break;
223                 }
224                 String valuePart = value.substring(0, subCount);
225                 String remainngStr = value.substring(subCount);
226                 parts.set(index[0], remainngStr);
227             }

```

```
228             res.append(pos).append("#").append(type).append("#").
229     append(subCount).append("#").append(valuePart);
230             // 因为下一条数据被拼接到当前字符串中, 所以退回一步, 让decode
231             // 处理
232             index[0]--;
233         } else {
234             // Compose 类型递归处理
235             res.append(pos).append("#").append(type).append("#").
236     append(subCount).append("#");
237
238             res.append(getChildrenInput(parts, index, subCount));
239             // 递归结束后回退一位,
240             index[0]--;
241         }
242         // 重置
243         pos = "";
244         type = "";
245         value = "";
246         subCount = -1;
247     }
248     index[0]++;
249
250     // 最后长度没对上则标记错误
251     if (tmpCount != count) {
252         flag = true;
253     }
254     return res.toString();
255 }
256
257 // 解码
258 static List<Item> decode(String input) {
259     List<Item> res = new ArrayList<>();
260     List<String> parts = split(input, "#");
261     int m = parts.size();
262     String pos = "";
263     String type = "";
264     String value = "";
265     int count = -1;
266     for (int i = 0; i < m; i++) {
267         if (pos.equals("")) {
268             pos = parts.get(i);
269         } else if (type.equals("")) {
270             type = parts.get(i);
271         } else if (count == -1) {
272             count = Integer.parseInt(parts.get(i));
273         } else {
274             if (type.equals("1") || type.equals("0")) {
```

```

273                     value = parts.get(i);
274                     // 两种情况:
275                     // 1. 最后一个数据, 长度应该等于count
276                     // 2. 不是最后一个数组, 长度应该大于count , 由 内容 + 下一个
277                     数据的位置编号
278                     if ((i != m - 1 && value.length() <= count) || (i ==
279                         m - 1 && value.length() != count)) {
280                         flag = true;
281                         break;
282                     }
283                     String remainngStr = value.substring(count);
284                     value = value.substring(0, count);
285                     res.add(new Item(pos, type, count, value));
286                     parts.set(i, remainngStr);
287                     i--;
288                 } else {
289                     int[] j = new int[]{i};
290                     String childrenInput = getChildrenInput(parts, j, cou
291                     nt);
292                     if (flag) {
293                         break;
294                     }
295                     res.add(new Item(pos, type, count, ""));
296                     List<Item> childrenItem = decode(childrenInput);
297                     res.get(res.size() - 1).children = childrenItem;
298                     i = j[0] - 1;
299                 }
300                 pos = "";
301                 type = "";
302                 count = -1;
303                 value = "";
304             }
305             if (!pos.equals("") || !type.equals("") || count != -1 || !value.
306             equals("")) {
307                 flag = true;
308             }
309         }
310         public static void main(String[] args) {
311             Scanner sc = new Scanner(System.in);
312             int command = sc.nextInt();
313             sc.nextLine();
314             String input = sc.nextLine();
315
316             if (command == 1) {
List<Item> items = encode(input);

```

```
317         if (flag) System.out.println("ENCODE_ERROR");
318         else formatEncodePrint(items);
319     } else if (command == 2) {
320         List<Item> items = decode(input);
321         if (flag) System.out.println("DECODE_ERROR");
322         else formatDecodePrint(items);
323     }
324 }
325 }
```

Python

```
1  from typing import List
2
3  # 类型长度映射
4  typeLength = {"0": 7, "1": 6, "2": 7}
5  # 类型字符串映射
6  typeToStringMp = {"0": "Integer", "1": "String", "2": "Compose"}
7  # 标志是否出现格式错误
8  flag = False
9
10
11 class Item:
12     def __init__(self, pos: str, type_: str, count: int, value: str):
13         self.pos = pos # 位置
14         self.type = type_ # 类型
15         self.count = count # 长度
16         self.value = value # 值
17         self.children: List[Item] = [] # 子项列表
18
19
20 def splitTopLevelItem(input_str: str) -> List[str]:
21     """最顶层按照括号进行切割"""
22     global flag
23     res = []
24     depth = 0
25     start = 0
26     n = len(input_str)
27     for i, c in enumerate(input_str):
28         if c == '[':
29             if depth == 0:
30                 start = i
31             depth += 1
32         elif c == ']':
33             depth -= 1
34             if depth == 0:
35                 res.append(input_str[start:i + 1])
36         if depth < 0:
37             flag = True # 括号乱序, 右括号过多
38             break
39         if depth != 0:
40             flag = True # 括号不匹配
41     return res
42
43
44 def encode(input_str: str) -> List[Item]:
45     """编码"""
```

```

46     global flag
47     res = []
48     if not input_str:
49         return res
50     # 判断格式
51     if input_str[0] != '[' or input_str[-1] != ']':
52         flag = True
53         return res
54
55     ans = splitTopLevelItem(input_str)
56     if flag:
57         return res
58     n = len(ans)
59
60     for i in range(n):
61         itemStr = ans[i]
62         # 去除左右括号
63         itemStr = itemStr[1:-1]
64         currentAns = []
65         tmp = ""
66         m = len(itemStr)
67         j = 0
68         while j < m:
69             c = itemStr[j]
70             if c != ',', ',':
71                 tmp += c
72             else:
73                 currentAns.append(tmp)
74                 tmp = ""
75                 if len(currentAns) == 2:
76                     break
77             j += 1
78         type_str = currentAns[-1]
79         # 根据类型进行不同处理
80         if type_str == "Integer" or type_str == "String":
81             value = itemStr[j + 1:]
82             count = len(value)
83             type_code = "0" if currentAns[1] == "Integer" else "1"
84             res.append(Item(currentAns[0], type_code, count, value))
85         elif type_str == "Compose":
86             childrenItemStr = itemStr[j + 1:]
87             children = encode(childrenItemStr)
88             count = 0
89             for item in children:
90                 count += item.count + typeLength[item.type]
91             composeItem = Item(currentAns[0], "2", count, "")
92             composeItem.children = children
93             res.append(composeItem)

```

```

94         return res
95
96
97     def formatEncodePrint(items: List[Item]):
98         """递归格式化输出编码结果"""
99         for item in items:
100             if item.type == "0" or item.type == "1":
101                 print(f"{item.pos}#{item.type}#{item.count}#{item.value}", end="")
102             else:
103                 print(f"{item.pos}#{item.type}#{item.count}#", end="")
104                 formatEncodePrint(item.children)
105
106
107     def formatDecodePrint(items: List[Item]):
108         """递归格式化输出解码结果"""
109         n = len(items)
110         for i, item in enumerate(items):
111             if item.type == "0" or item.type == "1":
112                 print(f"[{item.pos},{typeToStringMp[item.type]},{item.value}]", end="")
113             else:
114                 print(f"[{item.pos},{typeToStringMp[item.type]}," , end="")
115                 formatDecodePrint(item.children)
116                 print("]", end="")
117             if i != n - 1:
118                 print(",", end="")
119
120
121     def split(str_: str, delimiter: str) -> List[str]:
122
123         return str_.split(delimiter)
124
125
126     def getChildrenInput(parts: List[str], index: List[int], count: int) -> str:
127         """获取子复合类型的字符串内容"""
128         global flag
129         pos = ""
130         type_ = ""
131         value = ""
132         subCount = -1
133
134         res = ""
135         n = len(parts)
136         tmpCount = 0
137         # 根据长度来切割处理
138         while index[0] < n and tmpCount < count:

```

```

139         if pos == "":
140             pos = parts[index[0]]
141         elif type_ == "":
142             type_ = parts[index[0]]
143         elif subCount == -1:
144             subCount = int(parts[index[0]])
145         else:
146             tmpCount += typeLength[type_] + subCount
147             if type_ == "0" or type_ == "1":
148                 value = parts[index[0]]
149                 if len(value) < subCount:
150                     flag = True
151                     break
152                 valuePart = value[:subCount]
153                 remainngStr = value[subCount:]
154                 parts[index[0]] = remainngStr
155                 res += f"{pos}#{type_}#{subCount}#{valuePart}"
156                 index[0] -= 1 # 退回一步让decode处理
157             else:
158                 res += f"{pos}#{type_}#{subCount}#"
159
160             res += getChildrenInput(parts, index, subCount)
161             index[0] -= 1
162             pos = ""
163             type_ = ""
164             value = ""
165             subCount = -1
166             index[0] += 1
167
168         if tmpCount != count:
169             flag = True
170     return res
171
172
173 def decode(input_str: str) -> List[Item]:
174     """解码"""
175     global flag
176     res = []
177     parts = split(input_str, "#")
178     m = len(parts)
179     pos = ""
180     type_ = ""
181     value = ""
182     count = -1
183     i = 0
184     while i < m:
185         # 按顺序赋值
186         if pos == "":

```

```

187         pos = parts[i]
188     elif type_ == "":
189         type_ = parts[i]
190     elif count == -1:
191         count = int(parts[i])
192     else:
193         if type_ == "0" or type_ == "1":
194             value = parts[i]
195             if (i != m - 1 and len(value) <= count) or (i == m - 1 and len(value) != count):
196                 flag = True
197                 break
198             remainngStr = value[count:]
199             value = value[:count]
200             res.append(Item(pos, type_, count, value))
201             parts[i] = remainngStr
202             i -= 1
203         else:
204             j = [i]
205             childrenInput = getChildrenInput(parts, j, count)
206             if flag:
207                 break
208             res.append(Item(pos, type_, count, ""))
209             childrenItem = decode(childrenInput)
210             res[-1].children = childrenItem
211             i = j[0] - 1
212             pos = ""
213             type_ = ""
214             count = -1
215             value = ""
216             i += 1
217
218     if pos != "" or type_ != "" or count != -1 or value != "":
219         flag = True
220     return res
221
222
223 if __name__ == "__main__":
224     flag = False
225     command = int(input())
226     input_str = input()
227     if command == 1:
228         items = encode(input_str)
229         if flag:
230             print("ENCODE_ERROR")
231         else:
232             formatEncodePrint(items)
233 elif command == 2:

```

```
234     items = decode(input_str)
235     if flag:
236         print("DECODE_ERROR")
237     else:
238         formatDecodePrint(items)
```

JavaScript

```
1 const readline = require('readline');
2
3 // 类型长度映射
4 const typeLength = { "0": 7, "1": 6, "2": 7 };
5 // 类型字符串映射
6 const typeToStringMp = { "0": "Integer", "1": "String", "2": "Compose" };
7 // 标志是否出现格式错误
8 let flag = false;
9
10 // 定义数据结构Item, 保存位置pos, 类型type, 长度count, 值value及其子节点children
11 class Item {
12     constructor(pos, type, count, value) {
13         this.pos = pos;           // 位置标识
14         this.type = type;        // 类型 (0/1/2)
15         this.count = count;      // 长度
16         this.value = value;      // 存储值 (字符串)
17         this.children = [];       // 子项列表 (复合类型)
18     }
19 }
20
21 // 顶层根据中括号切割输入字符串, 提取一级子项
22 function splitTopLevelItem(input) {
23     const res = [];
24     let depth = 0;    // 括号深度
25     let start = 0;   // 当前子项起始索引
26     for (let i = 0; i < input.length; i++) {
27         const c = input[i];
28         if (c === '[') {
29             if (depth === 0) start = i; // 深度为0时, 记录开始位置
30             depth++;
31         } else if (c === ']') {
32             depth--;
33             if (depth === 0) {
34                 res.push(input.slice(start, i + 1)); // 取出一段完整子项
35             }
36         }
37         if (depth < 0) {
38             flag = true; // 括号不匹配, 右括号多了
39             break;
40         }
41     }
42     if (depth !== 0) flag = true; // 括号不匹配, 左括号多了
43     return res;
44 }
45
```

```

46 // 对输入字符串进行编码处理, 返回Item列表
47 function encode(input) {
48     const res = [];
49     if (!input) return res;
50     if (input[0] !== '[' || input[input.length - 1] !== ']') {
51         flag = true; // 非法格式, 必须以[]包围
52         return res;
53     }
54     const ans = splitTopLevelItem(input);
55     if (flag) return res;
56
57     // 遍历所有一级子项进行解析
58     for (let i = 0; i < ans.length; i++) {
59         let itemStr = ans[i];
60         // 去除两端括号
61         itemStr = itemStr.substring(1, itemStr.length - 1);
62
63         const currentAns = [];
64         let tmp = '';
65         let j = 0;
66         // 解析前两个逗号前的内容, 分别为位置和类型字符串
67         while (j < itemStr.length) {
68             const c = itemStr[j];
69             if (c !== ',') {
70                 tmp += c;
71             } else {
72                 currentAns.push(tmp);
73                 tmp = '';
74                 if (currentAns.length === 2) break; // 找到两个逗号后停止
75             }
76             j++;
77         }
78
79         const type = currentAns[currentAns.length - 1];
80         if (type === 'Integer' || type === 'String') {
81             // 普通类型, 后续全部为value
82             const value = itemStr.substring(j + 1);
83             const count = value.length;
84             const typeStr = currentAns[1] === 'Integer' ? '0' : '1';
85             res.push(new Item(currentAns[0], typeStr, count, value));
86         } else if (type === 'Compose') {
87             // 复合类型, 递归编码子内容
88             const childrenItemStr = itemStr.substring(j + 1);
89             const children = encode(childrenItemStr);
90             let count = 0;
91             // 计算长度 = 所有子项长度 + 各子项类型字符串长度
92             for (const item of children) {
93                 count += item.count + typeLength[item.type];

```

```

94         }
95         const composeItem = new Item(currentAns[0], '2', count, '');
96         composeItem.children = children;
97         res.push(composeItem);
98     }
99 }
100 return res;
101 }

103 // 递归格式化输出编码结果，连续打印
104 function formatEncodePrint(items) {
105     for (const item of items) {
106         if (item.type === '0' || item.type === '1') {
107             process.stdout.write(`#${item.pos}#${item.type}#${item.count}#${item.value}`);
108         } else {
109             process.stdout.write(`#${item.pos}#${item.type}#${item.count}#`);
110             formatEncodePrint(item.children);
111         }
112     }
113 }
114

115 // 递归格式化输出解码结果，打印为符合格式的字符串
116 function formatDecodePrint(items) {
117     for (let i = 0; i < items.length; i++) {
118         const item = items[i];
119         if (item.type === '0' || item.type === '1') {
120             process.stdout.write(`[${item.pos},${typeToStringMp[item.type]},${item.value}]`);
121         } else {
122             process.stdout.write(`[${item.pos},${typeToStringMp[item.type]},`);
123             formatDecodePrint(item.children);
124             process.stdout.write(']');
125         }
126         if (i !== items.length - 1) process.stdout.write(',');
127     }
128 }
129

130 // 简单分割字符串函数
131 function split(str, delimiter) {
132     return str.split(delimiter);
133 }
134

135 // 递归获取复合类型子内容字符串，供解码使用
136 function getChildrenInput(parts, index, count) {
137     let pos = '';
138     let type = '';
139     let value = '';

```

```

140     let subCount = -1;
141
142     let res = '';
143
144     let tmpCount = 0;
145     const n = parts.length;
146
147     while (index[0] < n && tmpCount < count) {
148         if (pos === '') {
149             pos = parts[index[0]];
150         } else if (type === '') {
151             type = parts[index[0]];
152         } else if (subCount === -1) {
153             subCount = parseInt(parts[index[0]], 10);
154         } else {
155             tmpCount += typeLength[type] + subCount;
156
157             if (type === '0' || type === '1') {
158                 value = parts[index[0]];
159                 if (value.length < subCount) {
160                     flag = true;
161                     break;
162                 }
163                 const valuePart = value.substring(0, subCount);
164                 const remainngStr = value.substring(subCount);
165                 parts[index[0]] = remainngStr;
166
167                 res += pos + '#' + type + '#' + subCount + '#' + valuePart;
168                 index[0]--;
169             } else {
170                 res += pos + '#' + type + '#' + subCount + '#';
171                 res += getChildrenInput(parts, index, subCount);
172                 index[0]--;
173             }
174             pos = '';
175             type = '';
176             value = '';
177             subCount = -1;
178         }
179         index[0]++;
180     }
181     if (tmpCount !== count) {
182         flag = true; // 长度不匹配, 解码异常
183     }
184     return res;
185 }
186
187 // 解码字符串为Item列表

```

```

188 function decode(input) {
189   const res = [];
190   const parts = split(input, '#');
191   const m = parts.length;
192   let pos = '';
193   let type = '';
194   let value = '';
195   let count = -1;
196
197   for (let i = 0; i < m; i++) {
198     // 按顺序赋值
199     if (pos === '') {
200       pos = parts[i];
201     } else if (type === '') {
202       type = parts[i];
203     } else if (count === -1) {
204       count = parseInt(parts[i], 10);
205     } else {
206       // 根据不同类型 不同处理方式
207       if (type === '1' || type === '0') {
208         value = parts[i];
209         // 两种情况:
210         // 1. 最后一个数据,长度应该等于count
211         // 2. 不是最后一个数组,长度应该大于count , 由 内容 + 下一个数据的位置编号
212         if ((i !== m - 1 && value.length <= count) || (i === m - 1 && value.length !== count)) {
213           flag = true;
214           break;
215         }
216         const remainngStr = value.substring(count);
217         value = value.substring(0, count);
218         res.push(new Item(pos, type, count, value));
219         parts[i] = remainngStr;
220         // 因为最后一个元素的内容和下一个元素的id会在一起
221         i--;
222       } else {
223         const j = [i];
224         const childrenInput = getChildrenInput(parts, j, count);
225         if (flag) break;
226         const newItem = new Item(pos, type, count, '');
227         newItem.children = decode(childrenInput);
228         res.push(newItem);
229         // 因为最后一个元素的内容和下一个元素的id会在一起
230         i = j[0] - 1;
231       }
232     // 重置
233     pos = '';
234     type = '';

```

```

235         count = -1;
236         value = '';
237     }
238 }
239 if (pos !== '' || type !== '' || count !== -1 || value !== '') {
240     flag = true; // 解码数据不完整
241 }
242 return res;
243 }
244

245 // ACM模式主程序，读取两行输入，第一行命令，第二行字符串
246 const rl = require('readline').createInterface({
247     input: process.stdin,
248     output: process.stdout,
249 });
250

251 const inputLines = [];
252 rl.on('line', line => {
253     inputLines.push(line.trim());
254     if (inputLines.length === 2) {
255         rl.close();
256     }
257 });
258

259 rl.on('close', () => {
260     flag = false;
261     const command = parseInt(inputLines[0]);
262     const inputStr = inputLines[1];
263

264     if (command === 1) {
265         const items = encode(inputStr);
266         if (flag) {
267             console.log("ENCODE_ERROR");
268         } else {
269             formatEncodePrint(items);
270             console.log();
271         }
272     } else if (command === 2) {
273         const items = decode(inputStr);
274         if (flag) {
275             console.log("DECODE_ERROR");
276         } else {
277             formatDecodePrint(items);
278             console.log();
279         }
280     }
281 });

```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 // 类型长度映射
12 var typeLength = map[string]int{"0": 7, "1": 6, "2": 7}
13
14 // 类型字符串映射
15 var typeToStringMp = map[string]string{"0": "Integer", "1": "String",
16 "2": "Compose"}
17
18 // 标志是否出现格式错误
19 var flag bool = false
20
21 // Item结构体定义
22 type Item struct {
23     pos      string // 位置
24     count    int    // 长度
25     typ      string // 类型 "0", "1", "2"
26     value    string // 值
27     children []Item // 子元素
28 }
29
30 // splitTopLevelItem 按中括号层级分割顶层子项
31 func splitTopLevelItem(input string) []string {
32     res := []string{}
33     depth := 0
34     start := 0
35     for i, c := range input {
36         if c == '[' {
37             if depth == 0 {
38                 start = i
39             }
40             depth++
41         } else if c == ']' {
42             depth--
43             if depth == 0 {
44                 res = append(res, input[start:i+1])
45             }
46         }
47     }
48 }
```

```

45     }
46     // 括号乱序
47     if depth < 0 {
48         flag = true
49         break
50     }
51 }
52 // 左右括号数量不一致
53 if depth != 0 {
54     flag = true
55 }
56 return res
57 }

59 // encode 编码函数，将字符串转为Item切片
60 func encode(input string) []Item {
61     res := []Item{}
62     if input == "" {
63         return res
64     }
65     // 判断格式
66     if input[0] != '[' || input[len(input)-1] != ']' {
67         flag = true
68         return res
69     }
70

71     ans := splitTopLevelItem(input)
72     if flag {
73         return res
74     }
75

76     for _, itemStr := range ans {
77         // 去除首尾中括号
78         itemStr = itemStr[1 : len(itemStr)-1]
79
80         currentAns := []string{}
81         tmp := ""
82         j := 0
83         for j < len(itemStr) {
84             c := itemStr[j]
85             if c != ',' {
86                 tmp += string(c)
87             } else {
88                 currentAns = append(currentAns, tmp)
89                 tmp = ""
90                 if len(currentAns) == 2 {
91                     break
92                 }

```

```

93         }
94         j++
95     }
96
97     typ := currentAns[len(currentAns)-1]
98     // 根据不同类型不同处理方式
99     if typ == "Integer" || typ == "String" {
100         value := itemStr[j+1:]
101         count := len(value)
102         typeStr := "0"
103         if currentAns[1] == "String" {
104             typeStr = "1"
105         }
106         res = append(res, Item{pos: currentAns[0], typ: typeStr, count: count, value: value})
107     } else if typ == "Compose" {
108         childrenItemStr := itemStr[j+1:]
109         // 递归处理复合类型
110         children := encode(childrenItemStr)
111         // 计算复合类型长度
112         count := 0
113         for _, item := range children {
114             count += item.count + typeLength[item.typ]
115         }
116         composeItem := Item{pos: currentAns[0], typ: "2", count: count, value: ""}
117         composeItem.children = children
118         res = append(res, composeItem)
119     }
120 }
121
122 return res
123
124 // formatEncodePrint 递归格式化打印编码结果
125 func formatEncodePrint(items []Item) {
126     for _, item := range items {
127         if item.typ == "0" || item.typ == "1" {
128             fmt.Printf("%s#%s#%d#%s", item.pos, item.typ, item.count, item.value)
129         } else {
130             fmt.Printf("%s#%s#%d#", item.pos, item.typ, item.count)
131             formatEncodePrint(item.children)
132         }
133     }
134 }
135
136 // formatDecodePrint 递归格式化打印解码结果
137 func formatDecodePrint(items []Item) {

```

```

138     for i, item := range items {
139         if item.typ == "0" || item.typ == "1" {
140             fmt.Printf("[%s,%s,%s]", item.pos, typeToStringMp[item.typ], item.v
141             alue)
142         } else {
143             fmt.Printf("[%s,%s,", item.pos, typeToStringMp[item.typ])
144             formatDecodePrint(item.children)
145             fmt.Printf("]")
146         }
147         if i != len(items)-1 {
148             fmt.Printf(",")
149         }
150     }
151
152 // split 按指定分隔符分割字符串
153 func split(str, delimiter string) []string {
154     return strings.Split(str, delimiter)
155 }
156
157 // getChildrenInput 递归获取复合类型子内容字符串, 供解码使用
158 func getChildrenInput(parts []string, index *int, count int) string {
159     pos := ""
160     typ := ""
161     value := ""
162     subCount := -1
163
164     res := ""
165
166     tmpCount := 0
167     n := len(parts)
168
169     for *index < n && tmpCount < count {
170         // 按顺序赋值
171         if pos == "" {
172             pos = parts[*index]
173         } else if typ == "" {
174             typ = parts[*index]
175         } else if subCount == -1 {
176             var err error
177             subCount, err = strconv.Atoi(parts[*index])
178             if err != nil {
179                 flag = true
180                 break
181             }
182         } else {
183             tmpCount += typeLength[typ] + subCount
184             // 不同类型不同处理方式

```

```

185         if typ == "0" || typ == "1" {
186             value = parts[*index]
187             if len(value) < subCount {
188                 flag = true
189                 break
190             }
191             valuePart := value[:subCount]
192             remainingStr := value[subCount:]
193             parts[*index] = remainingStr
194
195             res += pos + "#" + typ + "#" + strconv.Itoa(subCount) + "#" + val
196             uePart
197             *index--
198         } else {
199             res += pos + "#" + typ + "#" + strconv.Itoa(subCount) + "#"
200
201             res += getChildrenInput(parts, index, subCount)
202             *index--
203         }
204         // 重置
205         pos = ""
206         typ = ""
207         value = ""
208         subCount = -1
209     }
210     *index++
211 }
212 if tmpCount != count {
213     flag = true
214 }
215 return res
216 }

217 // decode 解码字符串为Item列表
218 func decode(input string) []Item {
219     res := []Item{}
220     parts := split(input, "#")
221     m := len(parts)
222     pos := ""
223     typ := ""
224     value := ""
225     count := -1
226     for i := 0; i < m; i++ {
227         if pos == "" {
228             pos = parts[i]
229         } else if typ == "" {
230             typ = parts[i]
231         } else if count == -1 {

```

```

232     var err error
233     count, err = strconv.Atoi(parts[i])
234     if err != nil {
235         flag = true
236         break
237     }
238 } else {
239     if typ == "1" || typ == "0" {
240         value = parts[i]
241         // 两种情况:
242         // 1. 最后一个数据, 长度应该等于count
243         // 2. 不是最后一个数组, 长度应该大于count , 由 内容 + 下一个数据的位置编号
244         if (i != m-1 && len(value) <= count) || (i == m-1 && len(value) != count) {
245             flag = true
246             break
247         }
248         remainingStr := ""
249         if len(value) > count {
250             remainingStr = value[count:]
251         }
252         value = value[:count]
253         res = append(res, Item{pos: pos, typ: typ, count: count, value: value})
254     }
255     parts[i] = remainingStr
256     // 因为一个元素的内容和下一个元素的id会在一起
257     i--
258 } else {
259     j := i
260     childrenInput := getChildrenInput(parts, &j, count)
261     if flag {
262         break
263     }
264     newItem := Item{pos: pos, typ: typ, count: count, value: ""}
265     newItem.children = decode(childrenInput)
266     res = append(res, newItem)
267     // 因为最后一个元素的内容和下一个元素的id会在一起
268     i = j - 1
269 }
270 pos = ""
271 typ = ""
272 count = -1
273 value = ""
274 }
275 // 不完整
276 if pos != "" || typ != "" || count != -1 || value != "" {
277     flag = true

```

```
278     }
279     return res
280 }
281
282 func main() {
283     reader := bufio.NewReader(os.Stdin)
284     flag = false
285
286     // 读取命令行参数 (1或2)
287     cmdLine, err := reader.ReadString('\n')
288     if err != nil {
289         fmt.Println("READ_ERROR")
290         return
291     }
292     cmdLine = strings.TrimSpace(cmdLine)
293     command, err := strconv.Atoi(cmdLine)
294     if err != nil {
295         fmt.Println("READ_ERROR")
296         return
297     }
298
299     // 读取字符串输入
300     inputLine, err := reader.ReadString('\n')
301     if err != nil {
302         fmt.Println("READ_ERROR")
303         return
304     }
305     inputLine = strings.TrimSpace(inputLine)
306
307     if command == 1 {
308         items := encode(inputLine)
309         if flag {
310             fmt.Println("ENCODE_ERROR")
311         } else {
312             formatEncodePrint(items)
313             fmt.Println()
314         }
315     } else if command == 2 {
316         items := decode(inputLine)
317         if flag {
318             fmt.Println("DECODE_ERROR")
319         } else {
320             formatDecodePrint(items)
321             fmt.Println()
322         }
323     }
324 }
```

来自: 华为OD机试 2025 B卷 – 模拟数据序列化传输 (C++ & Python & JAVA & JS & GO)-CSDN博客

华为OD机试2025B卷 - 相同数字组成图形的周长(C++ & Python & Java & JS & GO)-CSDN博客

相同数字组成图形的周长

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

有一个 64×64 的矩阵，每个元素的默认值为0，现在向里面填充数字，相同的数字组成一个实心图形，如下图所示是矩阵的局部（空白表示填充0）：

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----|---|---|---|---|---|---|---|---|---|---|----|
| 0 | | | | | | | | | | | |
| 1 | | | | 1 | | | | | | | |
| 2 | | | 1 | 1 | 1 | | | | | | |
| 3 | | | 1 | 1 | 1 | | | 2 | 2 | | |
| 4 | | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | | |
| 5 | | | 1 | 1 | 2 | 2 | 2 | 2 | 2 | | |
| 6 | | | | | 2 | 2 | 2 | 2 | 2 | | |
| 7 | | | | | 2 | 2 | 2 | 2 | 2 | | |
| 8 | | | | | | | | | | | |
| 9 | | | | | | | | | | | |
| 10 | | | | | | | | | | | |

数字1组成了蓝色边框的实心图形，数字2组成了红色边框的实心图形。单元格的边长规定为1个单位。

请根据输入，计算每个非0值填充出来的实心图形的周长。

输入描述

1. 第一行输入N，表示N个图形， $N > 0$ 且 $N < 64 \times 64$
2. 矩阵左上角单元格坐标标记作(0, 0)，第一个数字表示行号，第二个数字表示列号
3. 接下来是N行，每行第一个数是矩阵单元格填充的数字，后续每两个一组，表示填充该数字的单元格坐标
4. 答题者无需考虑数据格式非法的场景，题目用例不考察数据格式
5. 题目用例保证同一个填充值只会有一行输入数据

输出描述

- 一共输出N个数值，每个数值表示某一行输入表示图形的周长
- 输出顺序需和输入的隔行顺序保持一致，即第1个数是输入的第1个图形的周长，第2个数是输入的第2个图形的周长，以此类推。

用例1

输入

```
Plain Text |  
1 2  
2 1 1 3 2 2 2 3 2 4 3 2 3 3 3 4 4 1 4 2 4 3 4 4 5 2 5 3  
3 2 3 7 3 8 4 5 4 6 4 7 4 8 5 4 5 5 5 6 5 7 5 8 6 4 6 5 6 6 6 7 6 8 7 4 7 5  
7 6 7 7 7 8
```

输出

```
Plain Text |  
1 18 20
```

题解

思路： 逻辑分析

1. 主要要分析出这个规律：一个表格的周长为 4，代表四个方向每个方向一条1长度的边，当某个方向与另一个相同表格相邻时，暴露出去的边会少一个，贡献的周长-1。检测完四个边之后，就能知道这个方格能够贡献的周长。
2. 明白1的这个原理之后，我们初始构造一个 $64 * 64$ 地图数组，对于每次的输入，我们将对应坐标填充为指定值。然后遍历这些坐标，分别按照1的规律，处理每个坐标的方格能贡献的周长，累加就能知道这个图形的周长。
3. 按顺序输出每个图形的周长即可。

C++

```
1 #include <cstdint>
2 #include<iostream>
3 #include<vector>
4 #include<string>
5 #include <utility>
6 #include <sstream>
7 #include<algorithm>
8 #include<cmath>
9 #include<map>
10 using namespace std;
11
12
13 // 通用 split 函数
14 vector<int> split(const string& str, const string& delimiter) {
15     vector<int> result;
16     size_t start = 0;
17     size_t end = str.find(delimiter);
18     while (end != string::npos) {
19         result.push_back(stoi(str.substr(start, end - start)));
20         start = end + delimiter.length();
21         end = str.find(delimiter, start);
22     }
23     // 添加最后一个部分
24     result.push_back(stoi(str.substr(start)));
25     return result;
26 }
27
28 // 计算结果 一个方块的边长是4 四个面如果有一面与相同数字相邻则边长减一
29 int calislandPerimeter(vector<vector<int>>& grid, vector<int>& rowAndCol)
{
30     // 定义四个方向
31     int direct[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
32     int res = 0;
33
34     int n = rowAndCol.size();
35     for (int i = 1; i < n; i+=2) {
36         // 一个坐标代表一个方块 初始周长为4
37         int tmp = 4;
38         int x= rowAndCol[i];
39         int y = rowAndCol[i+1];
40         for (int j = 0; j < 4; j++) {
41             int nx = x + direct[j][0];
42             int ny = y + direct[j][1];
43             // 越界
44             if (nx < 0 || nx >= 64 || ny < 0 || ny >= 64) {
```

```

45         continue;
46     }
47     // 对应方向和相同数相邻 周长 -1
48     if (grid[x][y] == grid[nx][ny]) {
49         tmp--;
50     }
51 }
52
53     res += tmp;
54
55 }
56 return res;
57 }
58
59 int main() {
60     int n;
61     cin >> n;
62     // 存储结果
63     vector<int> res(n, 0);
64     vector<vector<int>> grid(64, vector<int>(64, 0));
65     cin.ignore();
66     for (int i = 0; i < n; i++) {
67         string input;
68         getline(cin, input);
69         vector<int> tmp = split(input, " ");
70         if (tmp.size() == 1) {
71             res[i] = 0;
72             continue;
73         }
74
75         int m = tmp.size();
76         for (int i = 1; i < m ; i+=2) {
77             grid[tmp[i]][tmp[i+1]] = tmp[0];
78         }
79         int calRes = calislandPerimeter(grid, tmp);
80         res[i] = calRes;
81     }
82     // 输出结果
83     for (int i = 0; i < n; i++) {
84         cout << res[i];
85         if (i != n-1) {
86             cout << " ";
87         }
88     }
89     return 0;
90 }

```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     static int[][] grid = new int[64][64];
5
6     // 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
7     public static int calIslandPerimeter(int[][] grid, List<Integer> rowAn
dCol) {
8         int[][] direct = {{-1,0}, {1,0}, {0,-1}, {0,1}};
9         int res = 0;
10        for (int i = 1; i < rowAndCol.size(); i += 2) {
11            int tmp = 4;
12            int x = rowAndCol.get(i);
13            int y = rowAndCol.get(i + 1);
14            for (int[] dir : direct) {
15                int nx = x + dir[0];
16                int ny = y + dir[1];
17                // 超过边界
18                if (nx < 0 || nx >= 64 || ny < 0 || ny >= 64) continue;
19                if (grid[x][y] == grid[nx][ny]) tmp--;
20            }
21            res += tmp;
22        }
23        return res;
24    }
25
26    public static void main(String[] args) {
27        Scanner sc = new Scanner(System.in);
28        int n = sc.nextInt();
29        sc.nextLine(); // 吸收换行符
30
31        int[] res = new int[n];
32        for (int i = 0; i < n; i++) {
33            String line = sc.nextLine();
34            String[] parts = line.split(" ");
35            if (parts.length == 1) {
36                res[i] = 0;
37                continue;
38            }
39            List<Integer> coords = new ArrayList<>();
40            for (String p : parts) {
41                coords.add(Integer.parseInt(p));
42            }
43            int id = coords.get(0);
```

```
45     // 填充表格
46     for (int j = 1; j < coords.size(); j += 2) {
47         int x = coords.get(j);
48         int y = coords.get(j + 1);
49         grid[x][y] = id;
50     }
51
52     res[i] = calIslandPerimeter(grid, coords);
53 }
54
55 for (int i = 0; i < n; i++) {
56     System.out.print(res[i]);
57     if (i != n - 1) System.out.print(" ");
58 }
59 }
60 }
```

Python

```
1 def split(s):
2     return list(map(int, s.strip().split()))
3
4 # 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
5 def cal_island_perimeter(grid, row_and_col):
6     directions = [(-1,0), (1,0), (0,-1), (0,1)]
7     res = 0
8     for i in range(1, len(row_and_col), 2):
9         tmp = 4
10        x, y = row_and_col[i], row_and_col[i + 1]
11        for dx, dy in directions:
12            nx, ny = x + dx, y + dy
13
14            if 0 <= nx < 64 and 0 <= ny < 64 and grid[x][y] == grid[nx][n
15            y]:
16                tmp -= 1
17            res += tmp
18    return res
19
20 def main():
21     n = int(input())
22     grid = [[0] * 64 for _ in range(64)]
23     res = []
24
25     for _ in range(n):
26         line = input()
27         tmp = split(line)
28         if len(tmp) == 1:
29             res.append(0)
30             continue
31         # 填充
32         for i in range(1, len(tmp), 2):
33             x, y = tmp[i], tmp[i+1]
34             grid[x][y] = tmp[0]
35     perimeter = cal_island_perimeter(grid, tmp)
36     res.append(perimeter)
37
38     print(" ".join(map(str, res)))
39 main()
```

JavaScript

```

1  const readline = require("readline");
2
3  const rl = readline.createInterface({
4      input: process.stdin,
5      output: process.stdout
6  });
7
8  let inputLines = [];
9  rl.on("line", line => {
10     inputLines.push(line);
11 }).on("close", () => {
12     const n = parseInt(inputLines[0]);
13     const grid = Array.from({ length: 64 }, () => Array(64).fill(0));
14     const res = [];
15
16     const directions = [[-1,0],[1,0],[0,-1],[0,1]];
17     // 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
18     function calPerimeter(rowAndCol) {
19         let total = 0;
20         for (let i = 1; i < rowAndCol.length; i += 2) {
21             let x = rowAndCol[i], y = rowAndCol[i + 1];
22             let tmp = 4;
23             for (let [dx, dy] of directions) {
24                 let nx = x + dx, ny = y + dy;
25                 // 与相同方格相邻
26                 if (nx >= 0 && nx < 64 && ny >= 0 && ny < 64) {
27                     if (grid[x][y] === grid[nx][ny]) tmp--;
28                 }
29             }
30             total += tmp;
31         }
32         return total;
33     }
34
35     for (let i = 1; i <= n; i++) {
36         const parts = inputLines[i].trim().split(" ").map(Number);
37         if (parts.length === 1) {
38             res.push(0);
39             continue;
40         }
41         const id = parts[0];
42         // 填充图形
43         for (let j = 1; j < parts.length; j += 2) {
44             let x = parts[j], y = parts[j + 1];
45             grid[x][y] = id;
46         }
47     }
48
49     console.log(calPerimeter(res));
50 }
51
52 
```

```
46      }
47      res.push(calPerimeter(parts));
48  }
49
50  console.log(res.join(" "));
51});
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10 // 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
11 func calPerimeter(grid [][]int, coords []int) int {
12     dirs := []int{{-1, 0}, {1, 0}, {0, -1}, {0, 1}}
13     res := 0
14     for i := 1; i < len(coords); i += 2 {
15         tmp := 4
16         x := coords[i]
17         y := coords[i+1]
18         for _, d := range dirs {
19             nx := x + d[0]
20             ny := y + d[1]
21             // 与相同方格相邻
22             if nx >= 0 && nx < 64 && ny >= 0 && ny < 64 && grid[x][y] == grid[nx][ny] {
23                 tmp--
24             }
25         }
26         res += tmp
27     }
28     return res
29 }
30
31 func main() {
32     scanner := bufio.NewScanner(os.Stdin)
33     scanner.Scan()
34     n, _ := strconv.Atoi(scanner.Text())
35
36     grid := make([][]int, 64)
37     for i := 0; i < 64; i++ {
38         grid[i] = make([]int, 64)
39     }
40
41     res := make([]int, n)
42
43     for i := 0; i < n; i++ {
44         scanner.Scan()
```

```
45     line := scanner.Text()
46     parts := strings.Fields(line)
47     if len(parts) == 1 {
48         res[i] = 0
49         continue
50     }
51     coords := make([]int, len(parts))
52     for j, v := range parts {
53         coords[j], _ = strconv.Atoi(v)
54     }
55     id := coords[0]
56     // 填充表格
57     for j := 1; j < len(coords); j += 2 {
58         x, y := coords[j], coords[j+1]
59         grid[x][y] = id
60     }
61     res[i] = calPerimeter(grid, coords)
62 }
63
64 for i := 0; i < n; i++ {
65     fmt.Println(res[i])
66     if i != n-1 {
67         fmt.Println(" ")
68     }
69 }
70 }
```

相同数字组成图形的周长

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

有一个 64×64 的矩阵，每个元素的默认值为0，现在向里面填充数字，相同的数字组成一个实心图形，如下图所示是矩阵的局部（空白表示填充0）：

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----|---|---|---|---|---|---|---|---|---|---|----|
| 0 | | | | | | | | | | | |
| 1 | | | | 1 | | | | | | | |
| 2 | | | 1 | 1 | 1 | | | | | | |
| 3 | | | 1 | 1 | 1 | | | 2 | 2 | | |
| 4 | | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | | |
| 5 | | | 1 | 1 | 2 | 2 | 2 | 2 | 2 | | |
| 6 | | | | | 2 | 2 | 2 | 2 | 2 | | |
| 7 | | | | | 2 | 2 | 2 | 2 | 2 | | |
| 8 | | | | | | | | | | | |
| 9 | | | | | | | | | | | |
| 10 | | | | | | | | | | | |

数字1组成了蓝色边框的实心图形，数字2组成了红色边框的实心图形。单元格的边长规定为1个单位。
请根据输入，计算每个非0值填充出来的实心图形的周长。

输入描述

1. 第一行输入N，表示N个图形， $N > 0$ 且 $N < 64 \times 64$
2. 矩阵左上角单元格坐标标记作(0, 0)，第一个数字表示行号，第二个数字表示列号
3. 接下来是N行，每行第一个数是矩阵单元格填充的数字，后续每两个一组，表示填充该数字的单元格坐标
4. 答题者无需考虑数据格式非法的场景，题目用例不考察数据格式
5. 题目用例保证同一个填充值只会有一行输入数据

输出描述

- 一共输出N个数值，每个数值表示某一行输入表示图形的周长
- 输出顺序需和输入的隔行顺序保持一致，即第1个数是输入的第一个图形的周长，第2个数是输入的第二个图形的周长，以此类推。

用例1

输入

```
1 2  
2 1 1 3 2 2 2 3 2 4 3 2 3 3 3 4 4 1 4 2 4 3 4 4 5 2 5 3  
3 2 3 7 3 8 4 5 4 6 4 7 4 8 5 4 5 5 5 6 5 7 5 8 6 4 6 5 6 6 6 7 6 8 7 4 7 5  
7 6 7 7 7 8
```

输出

```
1 18 20
```

题解

思路： 逻辑分析

1. 主要要分析出这个规律：一个表格的周长为 **4**，代表四个方向每个方向一条1长度的边，当某个方向与另一个相同表格相邻时，暴露出去的边会少一个，贡献的周长-1。检测完四个边之后，就能知道这个方格能够贡献的周长。
2. 明白1的这个原理之后，我们初始构造一个 **64 * 64** 地图数组，对于每次的输入，我们将对应坐标填充为指定值。然后遍历这些坐标，分别按照1的规律，处理每个坐标的方格能贡献的周长，累加就能知道这个图形的周长。
3. 按顺序输出每个图形的周长即可。

C++

```
1 #include <cstdint>
2 #include<iostream>
3 #include<vector>
4 #include<string>
5 #include <utility>
6 #include <sstream>
7 #include<algorithm>
8 #include<cmath>
9 #include<map>
10 using namespace std;
11
12
13 // 通用 split 函数
14 vector<int> split(const string& str, const string& delimiter) {
15     vector<int> result;
16     size_t start = 0;
17     size_t end = str.find(delimiter);
18     while (end != string::npos) {
19         result.push_back(stoi(str.substr(start, end - start)));
20         start = end + delimiter.length();
21         end = str.find(delimiter, start);
22     }
23     // 添加最后一个部分
24     result.push_back(stoi(str.substr(start)));
25     return result;
26 }
27
28 // 计算结果 一个方块的边长是4 四个面如果有一面与相同数字相邻则边长减一
29 int calislandPerimeter(vector<vector<int>>& grid, vector<int>& rowAndCol)
{
30     // 定义四个方向
31     int direct[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
32     int res = 0;
33
34     int n = rowAndCol.size();
35     for (int i = 1; i < n; i+=2) {
36         // 一个坐标代表一个方块 初始周长为4
37         int tmp = 4;
38         int x= rowAndCol[i];
39         int y = rowAndCol[i+1];
40         for (int j = 0; j < 4; j++) {
41             int nx = x + direct[j][0];
42             int ny = y + direct[j][1];
43             // 越界
44             if (nx < 0 || nx >= 64 || ny < 0 || ny >= 64) {
```

```

45         continue;
46     }
47     // 对应方向和相同数相邻 周长 -1
48     if (grid[x][y] == grid[nx][ny]) {
49         tmp--;
50     }
51 }
52
53     res += tmp;
54
55 }
56
57 return res;
58 }

59 int main() {
60     int n;
61     cin >> n;
62     // 存储结果
63     vector<int> res(n, 0);
64     vector<vector<int>> grid(64, vector<int>(64, 0));
65     cin.ignore();
66     for (int i = 0; i < n; i++) {
67         string input;
68         getline(cin, input);
69         vector<int> tmp = split(input, " ");
70         if (tmp.size() == 1) {
71             res[i] = 0;
72             continue;
73         }
74
75         int m = tmp.size();
76         for (int i = 1; i < m ; i+=2) {
77             grid[tmp[i]][tmp[i+1]] = tmp[0];
78         }
79         int calRes = calislandPerimeter(grid, tmp);
80         res[i] = calRes;
81     }
82     // 输出结果
83     for (int i = 0; i < n; i++) {
84         cout << res[i];
85         if (i != n-1) {
86             cout << " ";
87         }
88     }
89     return 0;
90 }

```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     static int[][] grid = new int[64][64];
5
6     // 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
7     public static int calIslandPerimeter(int[][] grid, List<Integer> rowAn
dCol) {
8         int[][] direct = {{-1,0}, {1,0}, {0,-1}, {0,1}};
9         int res = 0;
10        for (int i = 1; i < rowAndCol.size(); i += 2) {
11            int tmp = 4;
12            int x = rowAndCol.get(i);
13            int y = rowAndCol.get(i + 1);
14            for (int[] dir : direct) {
15                int nx = x + dir[0];
16                int ny = y + dir[1];
17                // 超过边界
18                if (nx < 0 || nx >= 64 || ny < 0 || ny >= 64) continue;
19                if (grid[x][y] == grid[nx][ny]) tmp--;
20            }
21            res += tmp;
22        }
23        return res;
24    }
25
26    public static void main(String[] args) {
27        Scanner sc = new Scanner(System.in);
28        int n = sc.nextInt();
29        sc.nextLine(); // 吸收换行符
30
31        int[] res = new int[n];
32        for (int i = 0; i < n; i++) {
33            String line = sc.nextLine();
34            String[] parts = line.split(" ");
35            if (parts.length == 1) {
36                res[i] = 0;
37                continue;
38            }
39            List<Integer> coords = new ArrayList<>();
40            for (String p : parts) {
41                coords.add(Integer.parseInt(p));
42            }
43            int id = coords.get(0);
```

```
45     // 填充表格
46     for (int j = 1; j < coords.size(); j += 2) {
47         int x = coords.get(j);
48         int y = coords.get(j + 1);
49         grid[x][y] = id;
50     }
51
52     res[i] = calIslandPerimeter(grid, coords);
53 }
54
55 for (int i = 0; i < n; i++) {
56     System.out.print(res[i]);
57     if (i != n - 1) System.out.print(" ");
58 }
59 }
60 }
```

Python

```
1 def split(s):
2     return list(map(int, s.strip().split()))
3
4 # 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
5 def cal_island_perimeter(grid, row_and_col):
6     directions = [(-1,0), (1,0), (0,-1), (0,1)]
7     res = 0
8     for i in range(1, len(row_and_col), 2):
9         tmp = 4
10        x, y = row_and_col[i], row_and_col[i + 1]
11        for dx, dy in directions:
12            nx, ny = x + dx, y + dy
13
14            if 0 <= nx < 64 and 0 <= ny < 64 and grid[x][y] == grid[nx][n
15            y]:
16                tmp -= 1
17            res += tmp
18    return res
19
20 def main():
21     n = int(input())
22     grid = [[0] * 64 for _ in range(64)]
23     res = []
24
25     for _ in range(n):
26         line = input()
27         tmp = split(line)
28         if len(tmp) == 1:
29             res.append(0)
30             continue
31         # 填充
32         for i in range(1, len(tmp), 2):
33             x, y = tmp[i], tmp[i+1]
34             grid[x][y] = tmp[0]
35     perimeter = cal_island_perimeter(grid, tmp)
36     res.append(perimeter)
37
38     print(" ".join(map(str, res)))
39 main()
```

JavaScript

```

1  const readline = require("readline");
2
3  const rl = readline.createInterface({
4      input: process.stdin,
5      output: process.stdout
6  });
7
8  let inputLines = [];
9  rl.on("line", line => {
10     inputLines.push(line);
11 }).on("close", () => {
12     const n = parseInt(inputLines[0]);
13     const grid = Array.from({ length: 64 }, () => Array(64).fill(0));
14     const res = [];
15
16     const directions = [[-1,0],[1,0],[0,-1],[0,1]];
17     // 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
18     function calPerimeter(rowAndCol) {
19         let total = 0;
20         for (let i = 1; i < rowAndCol.length; i += 2) {
21             let x = rowAndCol[i], y = rowAndCol[i + 1];
22             let tmp = 4;
23             for (let [dx, dy] of directions) {
24                 let nx = x + dx, ny = y + dy;
25                 // 与相同方格相邻
26                 if (nx >= 0 && nx < 64 && ny >= 0 && ny < 64) {
27                     if (grid[x][y] === grid[nx][ny]) tmp--;
28                 }
29             }
30             total += tmp;
31         }
32         return total;
33     }
34
35     for (let i = 1; i <= n; i++) {
36         const parts = inputLines[i].trim().split(" ").map(Number);
37         if (parts.length === 1) {
38             res.push(0);
39             continue;
40         }
41         const id = parts[0];
42         // 填充图形
43         for (let j = 1; j < parts.length; j += 2) {
44             let x = parts[j], y = parts[j + 1];
45             grid[x][y] = id;
46         }
47     }
48
49     console.log(res);
50 }
51
52 
```

```
46      }
47      res.push(calPerimeter(parts));
48  }
49
50  console.log(res.join(" "));
51});
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10 // 计算结果 一个方块的边长是4, 如果有一面与相同数字相邻则边长减一
11 func calPerimeter(grid [][]int, coords []int) int {
12     dirs := []int{{-1, 0}, {1, 0}, {0, -1}, {0, 1}}
13     res := 0
14     for i := 1; i < len(coords); i += 2 {
15         tmp := 4
16         x := coords[i]
17         y := coords[i+1]
18         for _, d := range dirs {
19             nx := x + d[0]
20             ny := y + d[1]
21             // 与相同方格相邻
22             if nx >= 0 && nx < 64 && ny >= 0 && ny < 64 && grid[x][y] == grid[n
x][ny] {
23                 tmp--
24             }
25         }
26         res += tmp
27     }
28     return res
29 }
30
31 func main() {
32     scanner := bufio.NewScanner(os.Stdin)
33     scanner.Scan()
34     n, _ := strconv.Atoi(scanner.Text())
35
36     grid := make([][]int, 64)
37     for i := 0; i < 64; i++ {
38         grid[i] = make([]int, 64)
39     }
40
41     res := make([]int, n)
42
43     for i := 0; i < n; i++ {
44         scanner.Scan()
```

```
45     line := scanner.Text()
46     parts := strings.Fields(line)
47     if len(parts) == 1 {
48         res[i] = 0
49         continue
50     }
51     coords := make([]int, len(parts))
52     for j, v := range parts {
53         coords[j], _ = strconv.Atoi(v)
54     }
55     id := coords[0]
56     // 填充表格
57     for j := 1; j < len(coords); j += 2 {
58         x, y := coords[j], coords[j+1]
59         grid[x][y] = id
60     }
61     res[i] = calPerimeter(grid, coords)
62 }
63
64 for i := 0; i < n; i++ {
65     fmt.Println(res[i])
66     if i != n-1 {
67         fmt.Println(" ")
68     }
69 }
70 }
```

来自: 华为OD 机试 2025 B卷 – 相同数字组成图形的周长 (C++ & Python & JAVA & JS & GO)–
CSDN博客

来自: 华为OD 机试 2025 B卷 – 相同数字组成图形的周长 (C++ & Python & JAVA & JS & GO)–
CSDN博客

华为OD机试2025B卷 - 跳格子2(C++ & Python & Java & JS & Go)-CSDN博客

跳格子2

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

小明和朋友玩跳格子游戏，有 n 个连续格子组成的圆圈，每个格子有不同的分数，小朋友可以选择以任意格子起跳，但是不能跳连续的格子，不能回头跳，也不能超过一圈；给定一个代表每个格子得分的非负整数数组，计算能够得到的最高分数。

输入描述

给定一个数例，第一个格子和最后一个格子首尾相连，如：2 3 2

备注

- $1 \leq \text{nums.length} \leq 100$
- $1 \leq \text{nums}[i] \leq 1000$

输出描述

输出能够得到的最高分，如：3

用例1

输入

```
1 2 3 2
```

输出

```
1 3
```

说明

| 只能跳3这个格子，因为第一个格子和第三个格子首尾相连

用例2

输入

```
1 1 2 3 1
```

Plain Text |

输出

```
1 4
```

Plain Text |

说明

| $1 + 3 = 4$

题解

思路： 模拟 + 动态规划

1. 如果不考虑首尾相连情况下，这就是一道简单的 动态规划。但是我们可以通过一点小改动将这道题简化。将原输入数组通过切割得到 不包含第一个元素的数组， 不包含最后一个元素的数组。分别求其中可以得到的最大值。两者中的较大值就是结果。
2. 每个子数组求最大值的逻辑，定义 `intervalMaxValue = 0` 记录不包含前一个元素情况 前面可以得到的最大值。定义 `dp[]`，其中 `dp[i]` 到达i位置可以得到的最大值。从前往后进行遍历时进行以下操作：
 - `dp[i] = intervalMaxValue + nums[i-1];`
 - `intervalMaxValue = max(intervalMaxValue, dp[i-1])`
 - `res = max(res, dp[i])`
3. 根据1和2的逻辑，就能求出结果。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 using namespace std;
10
11 // 通用 split 函数
12 vector<int> split(const string& str, const string& delimiter) {
13     vector<int> result;
14     size_t start = 0;
15     size_t end = str.find(delimiter);
16     while (end != string::npos) {
17         result.push_back(stoi(str.substr(start, end - start)));
18         start = end + delimiter.length();
19         end = str.find(delimiter, start);
20     }
21     // 添加最后一个部分
22     result.push_back(stoi(str.substr(start)));
23     return result;
24 }
25
26 int calMaxValue(vector<int>& nums) {
27     int n = nums.size();
28     // 不包含前一个元素情况 前面可以得到的最大值
29     int intervalMaxValue = 0;
30     vector<int> dp(n+1, 0);
31     int res = 0;
32     for (int i = 1; i <= n; i++) {
33         dp[i] = intervalMaxValue + nums[i-1];
34         // 更新
35         intervalMaxValue = max(intervalMaxValue, dp[i-1]);
36         // 更新结果
37         res = max(dp[i], res);
38     }
39     return res;
40 }
41
42
43 int main() {
44     string input;
45     getline(cin, input);
```

```
46     vector<int> nums = split(input, " ");
47     int n = nums.size();
48
49     if (n == 1) {
50         cout << nums[0];
51         return 0;
52     }
53     // 不包含尾部元素
54     vector<int> sub1(nums.begin(), nums.begin() + n - 1);
55     // 不包含首元素
56     vector<int> sub2(nums.begin() + 1, nums.end());
57     int res1 = calMaxValue(sub1);
58     int res2 = calMaxValue(sub2);
59     int res = max(res1, res2);
60     cout << res;
61     return 0;
62 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     // 通用 split 函数, 空格切分
5     public static List<Integer> split(String input) {
6         List<Integer> res = new ArrayList<>();
7         for (String s : input.trim().split(" ")) {
8             res.add(Integer.parseInt(s));
9         }
10        return res;
11    }
12
13    // 计算不包含首/尾元素时的最大收益 (不能相邻)
14    public static int calMaxValue(List<Integer> nums) {
15        int n = nums.size();
16        // 不包含前一个元素情况 前面可以得到的最大值
17        int intervalMaxValue = 0;
18        int[] dp = new int[n + 1];
19        int res = 0;
20
21        for (int i = 1; i <= n; i++) {
22            dp[i] = intervalMaxValue + nums.get(i - 1);
23            intervalMaxValue = Math.max(intervalMaxValue, dp[i - 1]);
24            // 更新结果
25            res = Math.max(res, dp[i]);
26        }
27        return res;
28    }
29
30    public static void main(String[] args) {
31        Scanner sc = new Scanner(System.in);
32        String input = sc.nextLine();
33        List<Integer> nums = split(input);
34        int n = nums.size();
35
36        if (n == 1) {
37            System.out.println(nums.get(0));
38            return;
39        }
40
41        // 不包含尾部元素
42        List<Integer> sub1 = nums.subList(0, n - 1);
43        // 不包含首元素
44        List<Integer> sub2 = nums.subList(1, n);
45        // 取较大值
```

```
46         int res = Math.max(calMaxValue(sub1), calMaxValue(sub2));
47         System.out.println(res);
48     }
49 }
```

Python

```
Plain Text | ▾
```

```
1 def cal_max_value(nums):
2     n = len(nums)
3     dp = [0] * (n + 1)
4     # 不包含前一个元素情况 前面可以得到的最大值
5     interval_max = 0
6     res = 0
7     for i in range(1, n + 1):
8         dp[i] = interval_max + nums[i - 1]
9         interval_max = max(interval_max, dp[i - 1])
10        res = max(res, dp[i])
11    return res
12
13 # 输入处理
14 nums = list(map(int, input().strip().split()))
15 n = len(nums)
16
17 if n == 1:
18     print(nums[0])
19 else:
20     # 不包含最后一个
21     sub1 = nums[:-1]
22     # 不包含第一个
23     sub2 = nums[1:]
24     res = max(cal_max_value(sub1), cal_max_value(sub2))
25     print(res)
```

JavaScript

```
1 const readline = require('readline');
2
3 // 创建 readline 接口
4 const rl = readline.createInterface({
5     input: process.stdin,
6     output: process.stdout
7 });
8
9 let inputLines = [];
10
11 rl.on('line', function (line) {
12     inputLines.push(line.trim());
13 }).on('close', function () {
14     const input = inputLines[0];
15     const nums = input.split(' ').map(Number);
16
17     const n = nums.length;
18
19     if (n === 1) {
20         console.log(nums[0]);
21         return;
22     }
23     // 不包含最后一个
24     const sub1 = nums.slice(0, n - 1);
25     // 不包含第一个
26     const sub2 = nums.slice(1);
27
28     const res = Math.max(cal.MaxValue(sub1), cal.MaxValue(sub2));
29     console.log(res);
30 });
31
32 function cal.MaxValue(nums) {
33     const n = nums.length;
34     const dp = new Array(n + 1).fill(0);
35     //不包含前一个元素情况 前面可以得到的最大值
36     let intervalMax = 0;
37     let res = 0;
38
39     for (let i = 1; i <= n; i++) {
40         dp[i] = intervalMax + nums[i - 1];
41         intervalMax = Math.max(intervalMax, dp[i - 1]);
42         res = Math.max(res, dp[i]);
43     }
44     return res;
45 }
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 // 计算不包含相邻的最大和 (动态规划)
12 func calMaxValue(nums []int) int {
13     n := len(nums)
14     dp := make([]int, n+1)
15     // 不包含前一个元素情况 前面可以得到的最大值
16     intervalMax := 0
17     res := 0
18     for i := 1; i <= n; i++ {
19         dp[i] = intervalMax + nums[i-1]
20         if dp[i-1] > intervalMax {
21             intervalMax = dp[i-1]
22         }
23         if dp[i] > res {
24             res = dp[i]
25         }
26     }
27     return res
28 }
29
30 func main() {
31     scanner := bufio.NewScanner(os.Stdin)
32     scanner.Scan()
33     line := scanner.Text()
34     parts := strings.Fields(line)
35     nums := make([]int, len(parts))
36     for i, s := range parts {
37         nums[i], _ = strconv.Atoi(s)
38     }
39
40     n := len(nums)
41     if n == 1 {
42         fmt.Println(nums[0])
43         return
44     }
45     // 不包含尾部
```

```
46     sub1 := nums[:n-1]
47     // 不包含第一个
48     sub2 := nums[1:]
49
50     res := calMaxValue(sub1)
51     tmp := calMaxValue(sub2)
52     if tmp > res {
53         res = tmp
54     }
55     fmt.Println(res)
56 }
```

| 来自: 华为OD 机试 2025 B卷 – 跳格子2 (C++ & Python & JAVA & JS & GO)-CSDN博客

华为OD机试2025B卷 - 周末爬山(C++ & Python & Java & JS & Go)-CSDN博客

周末爬山

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

周末小明准备去爬山锻炼，0代表平地，山的高度使用1到9来表示，小明每次爬山或下山高度只能相差k及k以内，每次只能上下左右一个方向上移动一格，小明从左上角(0,0)位置出发

输入描述

第一行输入m n k(空格分隔)。代表m*n的二维山地图，k为小明每次爬山或下山高度差的最大值。

然后接下来输入山地图，一共m行n列，均以空格分隔。取值范围：

- $0 < m \leq 500$
- $0 < n \leq 500$
- $0 < k < 5$

输出描述

请问小明能爬到的最高峰多高，到该最高峰的最短步数，输出以空格分隔。同高度的山峰输出较短步数。
如果没有可以爬的山峰，则高度和步数都返回0。

备注

所有用例输入均为正确格式，且在取值范围内，考生不需要考虑不合法的输入格式。

用例1

输入

```
1 5 4 1
2 0 1 2 0
3 1 0 0 0
4 1 0 1 2
5 1 3 1 0
6 0 0 0 9
```

Plain Text

输出

```
Plain Text |  
1 2 2
```

说明

根据山地图可知，能爬到的最高峰在(0,2)位置，高度为2，[最短路径](#)为(0,0)–(0,1)–(0,2)，最短步数为2。

用例2

输入

```
Plain Text |  
1 5 4 3  
2 0 0 0 0  
3 0 0 0 0  
4 0 9 0 0  
5 0 0 0 0  
6 0 0 0 9
```

输出

```
Plain Text |  
1 0 0
```

说明

根据山地图可知，每次爬山距离3，无法爬到山峰上，步数为0。

题解

思路：[BFS](#) 求解

1. 定义 `visited[][]` 记录位置是否访问。定义 `res = grid[0][0]`, `stepCount = 0` ,表示能够到达的最高山峰高度和对应到达最高峰所用最小步数，初始设置高度设置为起点高度，步数设置为0.
2. 接下来使用队列模拟BFS算法从进行扩散，初始将 `(0, 0)` 加入队列，并标志 `visited[0][0] = true` . 使用 `tmpStepCount = 1` 表示当前扩散步数。接下来就是基础的BFS算法常规操作了，每一轮基本逻辑如下：

- a. 取出当前队列中的所有元素，每个坐标尝试向四周扩散，注意下面这些限制。不存在下面非法行为的将扩散得来的坐标加入下一轮要扩散的队列中。
 - 是否越界
 - 是否已经访问，不要重复访问
 - 是否满足高度差。
 - b. 在1的逻辑中，如果某个坐标能扩散到另一个 `(nx, ny)` 坐标高度能够大于 `res`，执行以下操作 `res = grid[nx][ny], stepCount = tmpStepCount`。BFS算法有的特点：第一次访问到某个高度的步数肯定是最小步数。
 - c. 每一轮模拟BFS扩散之后将 `tmpStepCount += 1`。
 - d. 重复1 2 3 逻辑直到队列为空。
3. 执行完BFS逻辑之后，此时输出 `res stepCount` 即可。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 #include<queue>
10 using namespace std;
11
12
13
14
15 int main() {
16     // 接收输入
17     int m ,n,k;
18     cin >> m >> n >> k;
19     vector<vector<int>> grid(m, vector<int>(n, 0));
20     for (int i = 0; i < m; i++) {
21         for (int j = 0; j < n; j++) {
22             cin >> grid[i][j];
23         }
24     }
25     // 记录访问装填
26     vector<vector<bool>> visited(m, vector<bool>(n, false));
27
28
29     // 模拟bfs扩散
30     queue<pair<int, int>> q;
31     q.push({0, 0});
32     // 当前所用步数
33     int tmpStepCount = 0;
34     visited[0][0] = true;
35
36     // 定义四个方向
37     int direct[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
38     // 能够到达的最高山峰高度(初始设为起点) 和所用步数
39     int res = grid[0][0] , stepCount = 0;
40
41     while (!q.empty()) {
42         tmpStepCount++;
43         queue<pair<int, int>> q_back;
44         while (!q.empty()) {
45             pair<int, int> tmp = q.front();
```

```

46     q.pop();
47     int x = tmp.first;
48     int y = tmp.second;
49     for (int i = 0; i < 4; i++) {
50         int nx = x + direct[i][0];
51         int ny = y + direct[i][1];
52         // 越过边界 或者已访问过
53         if (nx < 0 || nx >= m || ny < 0 || ny >= n || visited[nx]
54             [ny]) {
55             continue;
56         }
57         // 不能访问
58         if (abs(grid[x][y] - grid[nx][ny]) > k) {
59             continue;
60         }
61         // 更新能够到达的最高位置以及步数 bfs特性,最新到达指定高度山峰肯定是
62         // 最少步数,所以使用大于号
63         if (grid[nx][ny] > res) {
64             res = grid[nx][ny];
65             stepCount = tmpStepCount;
66         }
67         visited[nx][ny] = true;
68         q_back.push({nx, ny});
69     }
70     q = q_back;
71 }
72
73 cout << res << " " << stepCount;
74 return 0;
75 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     static int[][] direct = {
5         {-1, 0}, {1, 0}, {0, -1}, {0, 1} // 上 下 左 右 四个方向
6     };
7
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10        // 接收输入
11        int m = sc.nextInt();
12        int n = sc.nextInt();
13        int k = sc.nextInt();
14
15        // 地图高度
16        int[][] grid = new int[m][n];
17        for (int i = 0; i < m; i++)
18            for (int j = 0; j < n; j++)
19                grid[i][j] = sc.nextInt();
20
21        // 记录访问状态
22        boolean[][] visited = new boolean[m][n];
23
24        // 模拟 BFS 扩散
25        Queue<int[]> q = new LinkedList<>();
26        q.offer(new int[]{0, 0});
27        visited[0][0] = true;
28
29        int tmpStepCount = 0; // 当前扩散步数
30        int res = grid[0][0]; // 能够到达的最高山峰高度
31        int stepCount = 0; // 到达该山峰所用步数
32
33        while (!q.isEmpty()) {
34            tmpStepCount++;
35            Queue<int[]> q_back = new LinkedList<>();
36
37            while (!q.isEmpty()) {
38                int[] tmp = q.poll();
39                int x = tmp[0], y = tmp[1];
40
41                for (int[] d : direct) {
42                    int nx = x + d[0];
43                    int ny = y + d[1];
44
45                    // 越界或已访问
```

```
46             if (nx < 0 || nx >= m || ny < 0 || ny >= n || visited
47                 [nx] [ny])
48                 continue;
49
50             // 不满足高度差条件
51             if (Math.abs(grid[x] [y] - grid[nx] [ny]) > k)
52                 continue;
53
54             // 更新能够到达的最高位置以及最短步数
55             // 说明：只考虑高于起点的山峰，BFS保证第一次达到某个高度是最少步
56             // 数
57             if (grid[nx] [ny] > res) {
58                 res = grid[nx] [ny];
59                 stepCount = tmpStepCount;
60             }
61
62             visited[nx] [ny] = true;
63             q_back.offer(new int[] {nx, ny});
64         }
65         q = q_back;
66     }
67
68     System.out.println(res + " " + stepCount);
69 }
```

Python

```

1  from collections import deque
2
3  def main():
4      # 接收输入
5      m, n, k = map(int, input().split())
6      grid = [list(map(int, input().split())) for _ in range(m)]
7
8      # 记录访问状态
9      visited = [[False] * n for _ in range(m)]
10
11     # 初始化 BFS
12     q = deque()
13     q.append((0, 0))
14     visited[0][0] = True
15
16     tmpStepCount = 0
17     res = grid[0][0]           # 能够到达的最高山峰高度 初始设为起点
18     stepCount = 0             # 最短步数
19
20     # 四个方向: 上、下、左、右
21     directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
22
23     # 模拟 BFS 扩散
24     while q:
25         tmpStepCount += 1
26         q_back = deque()
27
28         while q:
29             x, y = q.popleft()
30
31             for dx, dy in directions:
32                 nx, ny = x + dx, y + dy
33
34                 # 越界或已访问
35                 if not (0 <= nx < m and 0 <= ny < n):
36                     continue
37                 if visited[nx][ny]:
38                     continue
39                 # 不满足高度差条件
40                 if abs(grid[x][y] - grid[nx][ny]) > k:
41                     continue
42
43                 # 更新能够到达的最高位置以及最短步数
44                 if grid[nx][ny] > res:
45                     res = grid[nx][ny]

```

```
46         stepCount = tmpStepCount
47
48         visited[nx][ny] = True
49         q_back.append((nx, ny))
50
51     q = q_back
52
53     print(res, stepCount)
54
55 if __name__ == "__main__":
56     main()
```

JavaScript

```

1  const readline = require('readline');
2  const rl = readline.createInterface({ input: process.stdin });
3
4  let lines = [];
5  rl.on('line', line => {
6      lines.push(line.trim());
7  });
8
9  rl.on('close', () => {
10     // 解析输入
11     const [m, n, k] = lines[0].split(' ').map(Number);
12     const grid = lines.slice(1, m + 1).map(row => row.split(' ').map(Number));
13
14     // 记录访问状态
15     const visited = Array.from({ length: m }, () => Array(n).fill(false));
16
17     // BFS 初始化
18     let q = [[0, 0]];
19     visited[0][0] = true;
20     let tmpStepCount = 0;
21
22     // 四个方向: 上下左右
23     const direct = [[-1, 0], [1, 0], [0, -1], [0, 1]];
24
25     let res = grid[0][0], stepCount = 0;
26
27     // 模拟 BFS 扩散
28     while (q.length > 0) {
29         tmpStepCount++;
30         let q_back = [];
31
32         while (q.length > 0) {
33             const [x, y] = q.shift();
34
35             for (let [dx, dy] of direct) {
36                 const nx = x + dx;
37                 const ny = y + dy;
38
39                 // 越界或已访问
40                 if (nx < 0 || nx >= m || ny < 0 || ny >= n || visited[nx]
41 [ny]) continue;
42
43                 // 高度差超过限制, 不能访问
44                 if (Math.abs(grid[x][y] - grid[nx][ny]) > k) continue;

```

```
44
45      // 更新最高峰和最短步数
46      if (grid[nx][ny] > res) {
47          res = grid[nx][ny];
48          stepCount = tmpStepCount;
49      }
50
51      visited[nx][ny] = true;
52      q_back.push([nx, ny]);
53  }
54
55  q = q_back;
56
57 }
58
59 // 输出结果
60 console.log(` ${res} ${stepCount}`);
61});
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 var directions = [4][2]int{{-1, 0}, {1, 0}, {0, -1}, {0, 1}} // 上下左右
12
13 func main() {
14     reader := bufio.NewReader(os.Stdin)
15
16     // 接收输入
17     line1, _ := reader.ReadString('\n')
18     parts := strings.Fields(line1)
19     m, _ := strconv.Atoi(parts[0])
20     n, _ := strconv.Atoi(parts[1])
21     k, _ := strconv.Atoi(parts[2])
22
23     grid := make([][]int, m)
24     for i := 0; i < m; i++ {
25         rowStr, _ := reader.ReadString('\n')
26         rowParts := strings.Fields(rowStr)
27         grid[i] = make([]int, n)
28         for j := 0; j < n; j++ {
29             grid[i][j], _ = strconv.Atoi(rowParts[j])
30         }
31     }
32
33     // 记录访问状态
34     visited := make([][]bool, m)
35     for i := range visited {
36         visited[i] = make([]bool, n)
37     }
38
39     // 初始化 BFS
40     type pair struct{ x, y int }
41     q := []pair{{0, 0}}
42     visited[0][0] = true
43     tmpStepCount := 0
44     res := grid[0][0]           // 能到达的最高山峰
45     stepCount := 0 // 最少步数
```

```

46
47     // 模拟 BFS 扩散
48     for len(q) > 0 {
49         tmpStepCount++
50         var q_back []pair
51
52         for _, cur := range q {
53             x, y := cur.x, cur.y
54
55             for _, d := range directions {
56                 nx, ny := x+d[0], y+d[1]
57
58                 // 越界或访问过
59                 if nx < 0 || nx >= m || ny < 0 || ny >= n || visited[nx][ny] {
60                     continue
61                 }
62                 // 不满足高度条件
63                 if abs(grid[x][y]-grid[nx][ny]) > k {
64                     continue
65                 }
66                 // 更新最高峰和最短步数
67                 if grid[nx][ny] > res {
68                     res = grid[nx][ny]
69                     stepCount = tmpStepCount
70                 }
71                 visited[nx][ny] = true
72                 q_back = append(q_back, pair{nx, ny})
73             }
74         }
75         q = q_back
76     }
77
78     fmt.Printf("%d %d\n", res, stepCount)
79 }
80
81 func abs(x int) int {
82     if x < 0 {
83         return -x
84     }
85     return x
86 }
```

周末爬山

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 200分题型

题目描述

周末小明准备去爬山锻炼，0代表平地，山的高度使用1到9来表示，小明每次爬山或下山高度只能相差k及k以内，每次只能上下左右一个方向上移动一格，小明从左上角(0,0)位置出发

输入描述

第一行输入 $m \ n \ k$ (空格分隔)。代表 $m \times n$ 的二维山地图， k 为小明每次爬山或下山高度差的**最大值**。

然后接下来输入山地图，一共m行n列，均以空格分隔。取值范围：

- $0 < m \leq 500$
- $0 < n \leq 500$
- $0 < k < 5$

输出描述

请问小明能爬到的最高峰多高，到该最高峰的最短步数，输出以空格分隔。同高度的山峰输出较短步数。

如果没有可以爬的山峰，则高度和步数都返回0。

备注

所有用例输入均为正确格式，且在取值范围内，考生不需要考虑不合法的输入格式。

用例1

输入

```
▼ Plain Text |  
1 5 4 1  
2 0 1 2 0  
3 1 0 0 0  
4 1 0 1 2  
5 1 3 1 0  
6 0 0 0 9
```

输出

```
▼ Plain Text |  
1 2 2
```

说明

根据山地图可知，能爬到的最高峰在(0,2)位置，高度为2，**最短路径**为(0,0)-(0,1)-(0,2)，最短步数为2。

用例2

输入

```
Plain Text |  
1 5 4 3  
2 0 0 0 0  
3 0 0 0 0  
4 0 9 0 0  
5 0 0 0 0  
6 0 0 0 9
```

输出

```
Plain Text |  
1 0 0
```

说明

根据山地图可知，每次爬山距离3，无法爬到山峰上，步数为0。

题解

思路： BFS 求解

1. 定义 `visited[][]` 记录位置是否访问。定义 `res = grid[0][0], stepCount = 0`，表示能够到达的最高山峰高度和对应到达最高峰所用最小步数，初始设置高度设置为起点高度，步数设置为0。
2. 接下来使用队列模拟BFS算法从进行扩散，初始将 `(0, 0)` 加入队列，并标志 `visited[0][0] = true`。使用 `tmpStepCount = 1` 表示当前扩散步数。接下来就是基础的BFS算法常规操作了，每一轮基本逻辑如下：
 - a. 取出当前队列中的所有元素，每个坐标尝试向四周扩散，注意下面这些限制。不存在下面非法行为的将扩散得来的坐标加入下一轮要扩散的队列中。
 - 是否越界
 - 是否已经访问，不要重复访问
 - 是否满足高度差。
 - b. 在a的逻辑中，如果某个坐标能扩散到另一个 `(nx, ny)` 坐标高度能够大于 `res`，执行以下操作 `res = grid[nx][ny], stepCount = tmpStepCount`。BFS算法有的特点：第一次访问到某个高度的步数肯定是最小步数。
 - c. 每一轮模拟BFS扩散之后将 `tmpStepCount += 1`。
 - d. 重复1 2 3 逻辑直到队列为空

3. 执行完BFS逻辑之后，此时输出 `res stepCount` 即可。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 #include<queue>
10 using namespace std;
11
12
13
14
15 int main() {
16     // 接收输入
17     int m ,n,k;
18     cin >> m >> n >> k;
19     vector<vector<int>> grid(m, vector<int>(n, 0));
20     for (int i = 0; i < m; i++) {
21         for (int j = 0; j < n; j++) {
22             cin >> grid[i][j];
23         }
24     }
25     // 记录访问装填
26     vector<vector<bool>> visited(m, vector<bool>(n, false));
27
28
29     // 模拟bfs扩散
30     queue<pair<int, int>> q;
31     q.push({0, 0});
32     // 当前所用步数
33     int tmpStepCount = 0;
34     visited[0][0] = true;
35
36     // 定义四个方向
37     int direct[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
38     // 能够到达的最高山峰高度(初始设为起点) 和所用步数
39     int res = grid[0][0] , stepCount = 0;
40
41     while (!q.empty()) {
42         tmpStepCount++;
43         queue<pair<int, int>> q_back;
44         while (!q.empty()) {
45             pair<int, int> tmp = q.front();
```

```

46     q.pop();
47     int x = tmp.first;
48     int y = tmp.second;
49     for (int i = 0; i < 4; i++) {
50         int nx = x + direct[i][0];
51         int ny = y + direct[i][1];
52         // 越过边界 或者已访问过
53         if (nx < 0 || nx >= m || ny < 0 || ny >= n || visited[nx]
54             [ny]) {
55             continue;
56         }
57         // 不能访问
58         if (abs(grid[x][y] - grid[nx][ny]) > k) {
59             continue;
60         }
61         // 更新能够到达的最高位置以及步数 bfs特性,最新到达指定高度山峰肯定是
62         // 最少步数,所以使用大于号
63         if (grid[nx][ny] > res) {
64             res = grid[nx][ny];
65             stepCount = tmpStepCount;
66         }
67         visited[nx][ny] = true;
68         q_back.push({nx, ny});
69     }
70     q = q_back;
71 }
72
73 cout << res << " " << stepCount;
74 return 0;
75 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     static int[][] direct = {
5         {-1, 0}, {1, 0}, {0, -1}, {0, 1} // 上 下 左 右 四个方向
6     };
7
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10        // 接收输入
11        int m = sc.nextInt();
12        int n = sc.nextInt();
13        int k = sc.nextInt();
14
15        // 地图高度
16        int[][] grid = new int[m][n];
17        for (int i = 0; i < m; i++)
18            for (int j = 0; j < n; j++)
19                grid[i][j] = sc.nextInt();
20
21        // 记录访问状态
22        boolean[][] visited = new boolean[m][n];
23
24        // 模拟 BFS 扩散
25        Queue<int[]> q = new LinkedList<>();
26        q.offer(new int[]{0, 0});
27        visited[0][0] = true;
28
29        int tmpStepCount = 0; // 当前扩散步数
30        int res = grid[0][0]; // 能够到达的最高山峰高度
31        int stepCount = 0; // 到达该山峰所用步数
32
33        while (!q.isEmpty()) {
34            tmpStepCount++;
35            Queue<int[]> q_back = new LinkedList<>();
36
37            while (!q.isEmpty()) {
38                int[] tmp = q.poll();
39                int x = tmp[0], y = tmp[1];
40
41                for (int[] d : direct) {
42                    int nx = x + d[0];
43                    int ny = y + d[1];
44
45                    // 越界或已访问
```

```

46                     if (nx < 0 || nx >= m || ny < 0 || ny >= n || visited
47             [nx] [ny])
48                 continue;
49
50             // 不满足高度差条件
51             if (Math.abs(grid[x] [y] - grid[nx] [ny]) > k)
52                 continue;
53
54             // 更新能够到达的最高位置以及最短步数
55             // 说明：只考虑高于起点的山峰，BFS保证第一次达到某个高度是最少步
56             // 数
57             if ( grid[nx] [ny] > res) {
58                 res = grid[nx] [ny];
59                 stepCount = tmpStepCount;
60             }
61
62             visited[nx] [ny] = true;
63             q_back.offer(new int[]{nx, ny});
64         }
65     }
66
67     System.out.println(res + " " + stepCount);
68 }
69 }
```

Python

```

1  from collections import deque
2
3  def main():
4      # 接收输入
5      m, n, k = map(int, input().split())
6      grid = [list(map(int, input().split())) for _ in range(m)]
7
8      # 记录访问状态
9      visited = [[False] * n for _ in range(m)]
10
11     # 初始化 BFS
12     q = deque()
13     q.append((0, 0))
14     visited[0][0] = True
15
16     tmpStepCount = 0
17     res = grid[0][0]           # 能够到达的最高山峰高度 初始设为起点
18     stepCount = 0             # 最短步数
19
20     # 四个方向: 上、下、左、右
21     directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
22
23     # 模拟 BFS 扩散
24     while q:
25         tmpStepCount += 1
26         q_back = deque()
27
28         while q:
29             x, y = q.popleft()
30
31             for dx, dy in directions:
32                 nx, ny = x + dx, y + dy
33
34                 # 越界或已访问
35                 if not (0 <= nx < m and 0 <= ny < n):
36                     continue
37                 if visited[nx][ny]:
38                     continue
39                 # 不满足高度差条件
40                 if abs(grid[x][y] - grid[nx][ny]) > k:
41                     continue
42
43                 # 更新能够到达的最高位置以及最短步数
44                 if grid[nx][ny] > res:
45                     res = grid[nx][ny]

```

```
46             stepCount = tmpStepCount
47
48             visited[nx][ny] = True
49             q_back.append((nx, ny))
50
51         q = q_back
52
53     print(res, stepCount)
54
55 if __name__ == "__main__":
56     main()
```

JavaScript

```

1  const readline = require('readline');
2  const rl = readline.createInterface({ input: process.stdin });
3
4  let lines = [];
5  rl.on('line', line => {
6      lines.push(line.trim());
7  });
8
9  rl.on('close', () => {
10     // 解析输入
11     const [m, n, k] = lines[0].split(' ').map(Number);
12     const grid = lines.slice(1, m + 1).map(row => row.split(' ').map(Number));
13
14     // 记录访问状态
15     const visited = Array.from({ length: m }, () => Array(n).fill(false));
16
17     // BFS 初始化
18     let q = [[0, 0]];
19     visited[0][0] = true;
20     let tmpStepCount = 0;
21
22     // 四个方向: 上下左右
23     const direct = [[-1, 0], [1, 0], [0, -1], [0, 1]];
24
25     let res = grid[0][0], stepCount = 0;
26
27     // 模拟 BFS 扩散
28     while (q.length > 0) {
29         tmpStepCount++;
30         let q_back = [];
31
32         while (q.length > 0) {
33             const [x, y] = q.shift();
34
35             for (let [dx, dy] of direct) {
36                 const nx = x + dx;
37                 const ny = y + dy;
38
39                 // 越界或已访问
40                 if (nx < 0 || nx >= m || ny < 0 || ny >= n || visited[nx]
41 [ny]) continue;
42
43                 // 高度差超过限制, 不能访问
44                 if (Math.abs(grid[x][y] - grid[nx][ny]) > k) continue;

```

```
44
45      // 更新最高峰和最短步数
46      if (grid[nx][ny] > res) {
47          res = grid[nx][ny];
48          stepCount = tmpStepCount;
49      }
50
51      visited[nx][ny] = true;
52      q_back.push([nx, ny]);
53  }
54
55  q = q_back;
56
57 }
58
59 // 输出结果
60 console.log(` ${res} ${stepCount}`);
61});
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 var directions = [4][2]int{{-1, 0}, {1, 0}, {0, -1}, {0, 1}} // 上下左右
12
13 func main() {
14     reader := bufio.NewReader(os.Stdin)
15
16     // 接收输入
17     line1, _ := reader.ReadString('\n')
18     parts := strings.Fields(line1)
19     m, _ := strconv.Atoi(parts[0])
20     n, _ := strconv.Atoi(parts[1])
21     k, _ := strconv.Atoi(parts[2])
22
23     grid := make([][]int, m)
24     for i := 0; i < m; i++ {
25         rowStr, _ := reader.ReadString('\n')
26         rowParts := strings.Fields(rowStr)
27         grid[i] = make([]int, n)
28         for j := 0; j < n; j++ {
29             grid[i][j], _ = strconv.Atoi(rowParts[j])
30         }
31     }
32
33     // 记录访问状态
34     visited := make([][]bool, m)
35     for i := range visited {
36         visited[i] = make([]bool, n)
37     }
38
39     // 初始化 BFS
40     type pair struct{ x, y int }
41     q := []pair{{0, 0}}
42     visited[0][0] = true
43     tmpStepCount := 0
44     res := grid[0][0]           // 能到达的最高山峰
45     stepCount := 0 // 最少步数
```

```

46
47     // 模拟 BFS 扩散
48     for len(q) > 0 {
49         tmpStepCount++
50         var q_back []pair
51
52         for _, cur := range q {
53             x, y := cur.x, cur.y
54
55             for _, d := range directions {
56                 nx, ny := x+d[0], y+d[1]
57
58                 // 越界或访问过
59                 if nx < 0 || nx >= m || ny < 0 || ny >= n || visited[nx][ny] {
60                     continue
61                 }
62                 // 不满足高度条件
63                 if abs(grid[x][y]-grid[nx][ny]) > k {
64                     continue
65                 }
66                 // 更新最高峰和最短步数
67                 if grid[nx][ny] > res {
68                     res = grid[nx][ny]
69                     stepCount = tmpStepCount
70                 }
71                 visited[nx][ny] = true
72                 q_back = append(q_back, pair{nx, ny})
73             }
74         }
75         q = q_back
76     }
77
78     fmt.Printf("%d %d\n", res, stepCount)
79 }
80
81 func abs(x int) int {
82     if x < 0 {
83         return -x
84     }
85     return x
86 }
```

来自: 华为OD 机试 2025 B卷 – 周末爬山 (C++ & Python & JAVA & JS & GO)-CSDN博客

| 来自: 华为OD 机试 2025 B卷 – 周末爬山 (C++ & Python & JAVA & JS & GO)–CSDN博客

华为OD机试 2025 B卷 - 手机App防沉迷系统

(C++ & Python & JAVA & JS & GO)-CSDN博客

手机App防沉迷系统

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

智能手机方便了我们生活的同时，也侵占了我们不少的时间。“手机App防沉迷系统”能够让我们每天合理地规划手机App使用时间，在正确的时间做正确的事。

它的大概原理是这样的：

1. 在一天24小时内，可以注册每个App的允许使用时段
2. 一个时间段只能使用一个App
3. App有优先级，数值越高，优先级越高。注册使用时段时，如果高优先级的App时间和低优先级的时段有冲突，则系统会自动注销低优先级的时段，如果App的优先级相同，则后添加的App不能注册。

请编程实现，根据输入数据注册App，并根据输入的时间点，返回时间点使用的App名称，如果该时间点没有注册任何App，请返回[字符串]“NA”。

输入描述

第一行表示注册的App数量 N (N ≤ 100)

第二部分包括 N 行，每行表示一条App注册数据

最后一行输入一个时间点，程序即返回该时间点使用的App

```
1 2
2 App1 1 09:00 10:00
3 App2 2 11:00 11:30
4 09:30
```

数据说明如下：

1. N行注册数据以空格分隔，四项数依次表示：App名称、优先级、起始时间、结束时间
2. 优先级1~5，数字越大，优先级越高
3. 时间格式 HH:MM，小时和分钟都是两位，不足两位前面补0
4. 起始时间需小于结束时间，否则注册不上
5. 注册信息中的时间段包含起始时间点，不包含结束时间点

输出描述

输出一个字符串，表示App名称，或NA表示空闲时间

示例1

输入

```
Plain Text |  
1 1  
2 App1 1 09:00 10:00  
3 09:30
```

输出

```
Plain Text |  
1 App1
```

说明

App1注册在9点到10点间，9点半可用的应用名是App1

示例2

输入

```
Plain Text |  
1 2  
2 App1 1 09:00 10:00  
3 App2 2 09:10 09:30  
4 09:20
```

输出

```
Plain Text |  
1 App2
```

说明

APP1和App2的时段有冲突，App2优先级高，注册App2之后，App1自动注销，因此输出App2。

示例3

输入

```
Plain Text |  
1 2  
2 App1 1 09:00 10:00  
3 App2 2 09:10 09:30  
4 09:50
```

输出

```
Plain Text |  
1 NA
```

题解

思路： 模拟

- 将所有app限制的起始时间和终止时间转换为 int类型，方便进行比较。
- 创建 `registeredApps` 数组用于存储当前情况下能够有效注册的app。
- 遍历每个app注册请求，查询已认定有效注册app数组中是否存在 与当前app注册请求时间范围有冲突并且优先级比自己低的，有则在有效注册数组移除那个app注册记录，将自己加入有效注册数组中。
- 根据输入的时间，在有效注册app记录数组中寻找对应的APP。

Java

```
1 import java.util.*;
2
3 // 定义App类，用于存储App的相关信息
4 class App {
5     String name; // App名称
6     int priority; // App优先级
7     int startTime; // App允许使用的起始时间（以分钟为单位）
8     int endTime; // App允许使用的结束时间（以分钟为单位）
9
10    // App类的构造函数，用于创建App对象
11    public App(String name, int priority, int startTime, int endTime)
12    {
13        this.name = name;
14        this.priority = priority;
15        this.startTime = startTime;
16        this.endTime = endTime;
17    }
18
19    public class Main {
20        // 时间转换函数，将时间字符串转换为以分钟为单位的整数
21        public static int convertTime(String time) {
22            String[] parts = time.split(":");
23            int hours = Integer.parseInt(parts[0]);
24            int minutes = Integer.parseInt(parts[1]);
25            return hours * 60 + minutes;
26        }
27
28        public static void main(String[] args) {
29            Scanner scanner = new Scanner(System.in);
30            int n = scanner.nextInt(); // 读取App数量
31            List<App> apps = new ArrayList<>(); // 创建App列表，用于存储所有A
32            pp
33            for (int i = 0; i < n; i++) {
34                // 循环读取每个App的信息，并创建App对象添加到列表中
35                String appName = scanner.next();
36                int appPriority = scanner.nextInt();
37                String startTimeStr = scanner.next();
38                String endTimeStr = scanner.next();
39                int appStartTime = convertTime(startTimeStr);
40                int appEndTime = convertTime(endTimeStr);
41                apps.add(new App(appName, appPriority, appStartTime, appEn
42                dTime));
43            }
44        }
45    }
46}
```

```

43
44         String queryTimeStr = scanner.next();
45         int queryTime = convertTime(queryTimeStr); // 读取查询时间，并转
46         换为分钟
47         String appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
48
49         List<App> registeredApps = new ArrayList<>(); // 创建已注册App列
50         表
51         for (App app : apps) {
52             boolean canInsert = true;
53             List<Integer> needDeletePosition = new ArrayList<>();
54             if (app.startTime >= app.endTime) continue; // 如果起始时间
55             不小于结束时间，则跳过
56
57             // 遍历已注册的App列表，检查时间冲突
58             for (int i = registeredApps.size() - 1; i >= 0; i--) {
59                 App registered = registeredApps.get(i);
60                 if (Math.max(app.startTime, registered.startTime) < Ma
61                 th.min(app.endTime, registered.endTime)) {
62                     if (app.priority > registered.priority) {
63                         needDeletePosition.add(i); // 记录需要移除的位置
64                     } else {
65                         canInsert = false; // 当前App优先级不够高，无法插
66                         入
67                     }
68                 }
69             }
70             if (canInsert) {
71                 registeredApps.add(app); // 插入当前App
72                 for (int i = 0; i <= needDeletePosition.size() - 1; i+
73                 +) {
74                     registeredApps.remove((int) needDeletePosition.get
75                     (i)); // 移除冲突的App
76                 }
77             }
78         }
79
80         // 遍历已注册App列表，找到查询时间对应的App
81         for (App app : registeredApps) {
82             if (queryTime >= app.startTime && queryTime < app.endTim
83             e) {
84                 appAtTime = app.name; // 更新查询时间对应的App名称
85                 break; // 找到后退出循环
86             }
87         }
88
89         System.out.println(appAtTime); // 输出查询时间对应的App名称

```

```
83      }
84 }
```

Python

```
1 # 时间转换函数, 将时间字符串转换为以分钟为单位的整数
2     def convert_time(time_str):
3         hours, minutes = map(int, time_str.split(':'))
4         return hours * 60 + minutes
5
6     n = int(input()) # 读取App数量
7     apps = [] # 创建App列表, 用于存储所有App
8
9     for _ in range(n):
10        # 循环读取每个App的信息, 并创建App对象添加到列表中
11        app_name, app_priority, start_time_str, end_time_str = input().split()
12        app_priority = int(app_priority)
13        start_time = convert_time(start_time_str)
14        end_time = convert_time(end_time_str)
15        apps.append((app_name, app_priority, start_time, end_time))
16
17     query_time_str = input()
18     query_time = convert_time(query_time_str) # 读取查询时间, 并转换为分钟
19     app_at_time = "NA" # 初始化查询时间对应的App名称为"NA"
20
21     registered_apps = [] # 创建已注册App列表
22     for app in apps:
23         name, priority, start_time, end_time = app
24         if start_time >= end_time:
25             continue # 如果起始时间不小于结束时间, 则跳过
26         can_insert = True
27         need_delete_positions = []
28         # 遍历已注册的App列表, 检查时间冲突
29         for i in range(len(registered_apps) - 1, -1, -1):
30             registered = registered_apps[i]
31             if max(start_time, registered[2]) < min(end_time, registered[3]):
32                 if priority > registered[1]:
33                     need_delete_positions.append(i) # 记录需要移除的位置
34                 else:
35                     can_insert = False # 当前App优先级不够高, 无法插入
36         if can_insert:
37             registered_apps.append(app) # 插入当前App
38             for pos in need_delete_positions:
39                 registered_apps.pop(pos) # 移除冲突的App
40
41     # 遍历已注册App列表, 找到查询时间对应的App
42     for app in registered_apps:
43         if query_time >= app[2] and query_time < app[3]:
```

```
44         app_at_time = app[0]  # 更新查询时间对应的App名称
45         break  # 找到后退出循环
46
47     print(app_at_time)  # 输出查询时间对应的App名称
```

JavaScript

```

1 // 时间转换函数，将时间字符串转换为以分钟为单位的整数
2     function convertTime(timeStr) {
3         const [hours, minutes] = timeStr.split(':').map(Number);
4         return hours * 60 + minutes;
5     }
6
7     const readline = require('readline');
8     const rl = readline.createInterface({
9         input: process.stdin,
10        output: process.stdout
11    });
12
13    let n;
14    const apps = [];
15    let queryTimeStr;
16
17    rl.question('', (input) => {
18        n = parseInt(input); // 读取App数量
19        let count = 0;
20        rl.on('line', (line) => {
21            if (count < n) {
22                // 循环读取每个App的信息，并创建App对象添加到列表中
23                const [appName, appPriority, startTimeStr, endTimeStr] = line.split(' ');
24                const appStartTime = convertTime(startTimeStr);
25                const appEndTime = convertTime(endTimeStr);
26                apps.push({ name: appName, priority: parseInt(appPriority), startTime: appStartTime, endTime: appEndTime });
27                count++;
28            } else {
29                queryTimeStr = line; // 读取查询时间
30                rl.close();
31            }
32        });
33    });
34
35    rl.on('close', () => {
36        const queryTime = convertTime(queryTimeStr); // 将查询时间转换为分钟
37        let appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
38
39        const registeredApps = []; // 创建已注册App列表
40        for (const app of apps) {
41            if (app.startTime >= app.endTime) continue; // 如果起始时间不小于
结束时间，则跳过
42            let canInsert = true;

```

```
43     const needDeletePositions = [];
44     // 遍历已注册的App列表，检查时间冲突
45     for (let i = registeredApps.length - 1; i >= 0; i--) {
46         const registered = registeredApps[i];
47         if (Math.max(app.startTime, registered.startTime) < Math.m
48             in(app.endTime, registered.endTime)) {
49             if (app.priority > registered.priority) {
50                 needDeletePositions.push(i); // 记录需要移除的位置
51             } else {
52                 canInsert = false; // 当前App优先级不够高，无法插入
53             }
54         }
55     }
56     if (canInsert) {
57         registeredApps.push(app); // 插入当前App
58         for (const pos of needDeletePositions) {
59             registeredApps.splice(pos, 1); // 移除冲突的App
60         }
61     }
62 }

63 // 遍历已注册App列表，找到查询时间对应的App
64 for (const app of registeredApps) {
65     if (queryTime >= app.startTime && queryTime < app.endTime) {
66         appAtTime = app.name; // 更新查询时间对应的App名称
67         break; // 找到后退出循环
68     }
69 }
70
71 console.log(appAtTime); // 输出查询时间对应的App名称
72 }
```

C++

```
1 #include <iostream>
2     #include <vector>
3     #include <string>
4     #include <sstream>
5
6     // 定义App类, 用于存储App的相关信息
7     class App {
8     public:
9         std::string name; // App名称
10        int priority; // App优先级
11        int startTime; // App允许使用的起始时间 (以分钟为单位)
12        int endTime; // App允许使用的结束时间 (以分钟为单位)
13
14        // App类的构造函数, 用于创建App对象
15        App(std::string name, int priority, int startTime, int endTime)
16            : name(name), priority(priority), startTime(startTime), endTime(endTime) {}
17    };
18
19        // 时间转换函数, 将时间字符串转换为以分钟为单位的整数
20        int convertTime(const std::string& time) {
21            int hours, minutes;
22            char colon;
23            std::istringstream iss(time);
24            iss >> hours >> colon >> minutes; // 将时间字符串按照":"分割并转换为小时和分钟
25            return hours * 60 + minutes; // 将小时和分钟转换为分钟
26        }
27
28        int main() {
29            int n; // 读取App数量
30            std::cin >> n;
31
32            std::vector<App> apps; // 创建App列表, 用于存储所有App
33            for (int i = 0; i < n; i++) {
34                // 循环读取每个App的信息, 并创建App对象添加到列表中
35                std::string appName;
36                int appPriority, appStartTime, appEndTime;
37                std::string startTimeStr, endTimeStr;
38                std::cin >> appName >> appPriority >> startTimeStr >> endTimeStr;
39                appStartTime = convertTime(startTimeStr);
40                appEndTime = convertTime(endTimeStr);
41                apps.emplace_back(appName, appPriority, appStartTime, appEndTime);
42            }
43        }
44
```

```
42     }
43
44     std::string queryTimeStr;
45     std::cin >> queryTimeStr;
46     int queryTime = convertTime(queryTimeStr); // 读取查询时间，并转换为分
47     钟
48     std::string appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
49
50     std::vector<App> registeredApps; // 创建已注册App列表
51     for (const App& app : apps) {
52         bool canInsert = true;
53         std::vector<int> needDeletePosition;
54         if (app.startTime >= app.endTime) continue; // 如果起始时间不小于
55         结束时间，则跳过
56         // 遍历已注册的App列表，检查时间冲突
57         for (int i = registeredApps.size() - 1; i >= 0; --i) {
58             const App& registered = registeredApps[i];
59             // 如果存在时间冲突
60             if (std::max(app.startTime, registered.startTime) < std::min(app.endTime, registered.endTime)) {
61                 // 如果当前App的优先级高于已注册App的优先级
62                 if (app.priority > registered.priority) {
63                     // 记录起冲突需要移除的位置
64                     needDeletePosition.push_back(i);
65                 } else {
66                     // 之前注册app优先级比现在大或者相等，这个不能进行注册
67                     canInsert = false;
68                 }
69             }
70             // 未有冲突情况或者优先级是最高的那个情况下
71             if (canInsert) {
72                 registeredApps.push_back(app);
73                 // 移除已经起冲突的注册
74                 for (int i = 0; i < needDeletePosition.size(); i++) {
75                     registeredApps.erase(registeredApps.begin() + needDeletePosition[i]);
76                 }
77             }
78         }
79
80         // 遍历已注册App列表，找到查询时间对应的App
81         for (const App& app : registeredApps) {
82             if (queryTime >= app.startTime && queryTime < app.endTime) {
83                 appAtTime = app.name; // 更新查询时间对应的App名称
84                 break; // 找到后退出循环
85             }
86         }
87     }
88 }
```

```
86     }
87
88     std::cout << appAtTime << std::endl; // 输出查询时间对应的App名称
89
90     return 0;
91 }
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 // 定义App结构体，用于存储App的相关信息
12 type App struct {
13     name      string // App名称
14     priority   int    // App优先级
15     startTime int    // App允许使用的起始时间（以分钟为单位）
16     endTime    int    // App允许使用的结束时间（以分钟为单位）
17 }
18
19 // 时间转换函数，将时间字符串转换为以分钟为单位的整数
20 func convertTime(timeStr string) int {
21     parts := strings.Split(timeStr, ":")
22     hours, _ := strconv.Atoi(parts[0])
23     minutes, _ := strconv.Atoi(parts[1])
24     return hours*60 + minutes
25 }
26
27 func main() {
28     scanner := bufio.NewScanner(os.Stdin)
29     scanner.Scan()
30     n, _ := strconv.Atoi(scanner.Text()) // 读取App数量
31
32     apps := []App{} // 创建App列表，用于存储所有App
33     for i := 0; i < n; i++ {
34         scanner.Scan()
35         line := scanner.Text()
36         parts := strings.Split(line, " ")
37         appName := parts[0]
38         appPriority, _ := strconv.Atoi(parts[1])
39         startTimeStr := parts[2]
40         endTimeStr := parts[3]
41         appStartTime := convertTime(startTimeStr)
42         appEndTime := convertTime(endTimeStr)
43         apps = append(apps, App{name: appName, priority: appPriority, startTi
44         me: appStartTime, endTime: appEndTime})
45     }
46 }
```

```

45
46     scanner.Scan()
47     queryTimeStr := scanner.Text()
48     queryTime := convertTime(queryTimeStr) // 读取查询时间，并转换为分钟
49     appAtTime := "NA" // 初始化查询时间对应的App名称为"NA"
50
51     registeredApps := []App{} // 创建已注册App列表
52     for _, app := range apps {
53         if app.startTime >= app.endTime {
54             continue // 如果起始时间不小于结束时间，则跳过
55         }
56         canInsert := true
57         needDeletePositions := []int{}
58         // 遍历已注册的App列表，检查时间冲突
59         for i := len(registeredApps) - 1; i >= 0; i-- {
60             registered := registeredApps[i]
61             if max(app.startTime, registered.startTime) < min(app.endTime, registered.endTime) {
62                 if app.priority > registered.priority {
63                     needDeletePositions = append(needDeletePositions, i) // 记录需要
64                     移除的位置
65                 } else {
66                     canInsert = false // 当前App优先级不够高，无法插入
67                 }
68             }
69         }
70         if canInsert {
71             registeredApps = append(registeredApps, app) // 插入当前App
72             for _, pos := range needDeletePositions {
73                 registeredApps = append(registeredApps[:pos], registeredApps[pos+1:]...) // 移除冲突的App
74             }
75         }
76     }
77
78     // 遍历已注册App列表，找到查询时间对应的App
79     for _, app := range registeredApps {
80         if queryTime >= app.startTime && queryTime < app.endTime {
81             appAtTime = app.name // 更新查询时间对应的App名称
82             break // 找到后退出循环
83         }
84     }
85
86     fmt.Println(appAtTime) // 输出查询时间对应的App名称
87 }
88
89 // 辅助函数：返回两个整数中的最大值
func max(a, b int) int {

```

```
90     if a > b {
91         return a
92     }
93     return b
94 }
95
96 // 辅助函数: 返回两个整数中的最小值
97 func min(a, b int) int {
98     if a < b {
99         return a
100    }
101    return b
102 }
103
104 // 辅助函数: 反转整数切片
105 func reverse(slice []int) []int {
106     for i, j := 0, len(slice)-1; i < j; i, j = i+1, j-1 {
107         slice[i], slice[j] = slice[j], slice[i]
108     }
109     return slice
110 }
```

| 来自: 华为OD机试 2025 B卷 – 手机App防沉迷系统 (C++ & Python & JAVA & JS & GO)-CSDN博客

华为OD机试 2025 B卷 - 字符串摘要 (C++ & Python & JAVA & JS & GO)-CSDN博客

字符串摘要

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

给定一个字符串的摘要算法，请输出给定字符串的摘要值。

1. 去除字符串中非字母的符号。
2. 如果出现连续字符(不区分大小写)，则输出：该字符（小写）+ 连续出现的次数。
3. 如果是非连续的字符(不区分大小写)，则输出：该字符(小写) + 该字母之后字符串中出现的该字符的次数
4. 对按照以上方式表示后的字符串进行排序：字母和紧随的数字作为一组进行排序，数字大的在前，数字相同的，则按字母进行排序，字母小的在前。

输入描述

一行字符串，长度为[1,200]

输出描述

摘要字符串

用例1

输入

Plain Text

```
1 aabbcc
```

输出

Plain Text

```
1 a2b2c2
```

说明

无

用例2

输入

```
Plain Text |  
1 bAaAcBb
```

输出

```
Plain Text |  
1 a3b2b2c0
```

说明

bAaAcBb:第一个b非连续字母，该字母之后字符串中还出现了2次(最后的两个Bb)，所以输出b2,a连续出现3次，输出a3，c非连续，该字母之后字符串再没有出现过c，输出c0,Bb连续2次，输出b2对b2a3c0b2进行排序，最终输出a3b2b2c0。

题解

思路：模拟

1. 字符串中字符去除非字母字符，字母字符全部转换为小写。
2. 使用 哈希表 统计各个字符出现的数量，用于出现不连续字符的情况下使用。
3. 进行遍历，当 $s[i] \neq s[i-1]$ 作为条件，计算出 $s[i-1]$ 字符出现的连续个数 size，size = 1 为 $\text{count}[s[i-1]] - 1$ ，否则为 size。将 $\{s[i-1], \text{count}[s[i-1]] - 1\}$ 或 $\{s[i-1], \text{size}\}$ 插入结果数组中
4. 自定义排序，然后按照顺序输出结果。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<map>
8 using namespace std;
9
10 // 自定义排序
11 bool cmp(pair<char, int> p1, pair<char, int> p2) {
12     if (p1.second == p2.second) {
13         return p1.first < p2.first;
14     }
15     return p1.second > p2.second;
16 }
17
18
19 int main() {
20     string s;
21     cin >> s;
22     // 转换为小写并过滤非字母字符
23     string filtered = "";
24     for (char c : s) {
25         if (isalpha(c)) {
26             filtered += tolower(c);
27         }
28     }
29
30     // 统计每个字符的总频次
31     map<char, int> mp;
32     for (char c : filtered) {
33         mp[c]++;
34     }
35
36     s = filtered;
37     int n = s.size();
38     vector<pair<char, int>> res;
39     string tmp = "";
40     for (int i = 0; i < n; i++) {
41         if (i == 0) {
42             tmp += s[i];
43             continue;
44         }
45         // 连续终止
```

```

46         if (s[i] != s[i-1]) {
47             char c = s[i-1];
48             int length = tmp.size();
49             mp[c] -= length;
50             if (length == 1) {
51                 res.push_back({c, mp[c]});
52             } else {
53                 res.push_back({c, length});
54             }
55             tmp = string(1, s[i]);
56         } else {
57             tmp += s[i];
58         }
59     }
60     // 末尾边界处理
61     if (tmp != "") {
62         char c = tmp[0];
63         int length = tmp.size();
64         mp[c] -= length;
65         if (length == 1) {
66             res.push_back({c, mp[c]});
67         } else {
68             res.push_back({c, length});
69         }
70     }
71
72     sort(res.begin(), res.end(), cmp);
73     string resString = "";
74
75     for (int i = 0; i < res.size(); i++) {
76         pair<char,int> p = res[i];
77         resString += (string(1, p.first) + to_string(p.second));
78     }
79     cout << resString;
80     return 0;
81 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     // 自定义排序
5     static boolean cmp(Map.Entry<Character, Integer> p1, Map.Entry<Character, Integer> p2) {
6         if (p1.getValue().equals(p2.getValue())) {
7             return p1.getKey() < p2.getKey();
8         }
9         return p1.getValue() > p2.getValue();
10    }
11
12    public static void main(String[] args) {
13        Scanner sc = new Scanner(System.in);
14        String s = sc.nextLine();
15        StringBuilder filtered = new StringBuilder();
16
17        // 转换为小写并过滤非字母字符
18        for (char c : s.toCharArray()) {
19            if (Character.isAlphabetic(c)) {
20                filtered.append(Character.toLowerCase(c));
21            }
22        }
23
24        // 统计每个字符的总频次
25        Map<Character, Integer> mp = new HashMap<>();
26        for (char c : filtered.toString().toCharArray()) {
27            mp.put(c, mp.getOrDefault(c, 0) + 1);
28        }
29
30        StringBuilder tmp = new StringBuilder();
31        List<Map.Entry<Character, Integer>> res = new ArrayList<>();
32        for (int i = 0; i < filtered.length(); i++) {
33            if (i == 0 || filtered.charAt(i) != filtered.charAt(i - 1)) {
34                if (tmp.length() > 0) {
35                    char c = tmp.charAt(0);
36                    int length = tmp.length();
37                    mp.put(c, mp.get(c) - length);
38                    res.add(new AbstractMap.SimpleEntry<>(c, length == 1
? mp.get(c) : length));
39                }
40                tmp = new StringBuilder().append(filtered.charAt(i));
41            } else {
42                tmp.append(filtered.charAt(i));
43            }
44        }
45    }
46}
```

```
44 }
45
46     if (tmp.length() > 0) {
47         char c = tmp.charAt(0);
48         int length = tmp.length();
49         mp.put(c, mp.get(c) - length);
50         res.add(new AbstractMap.SimpleEntry<>(c, length == 1 ? mp.get
51             (c) : length));
52     }
53
54     // 排序
55     res.sort((p1, p2) -> cmp(p1, p2) ? -1 : 1);
56
57     StringBuilder result = new StringBuilder();
58     for (Map.Entry<Character, Integer> entry : res) {
59         result.append(entry.getKey()).append(entry.getValue());
60     }
61
62     System.out.println(result);
63     sc.close();
64 }
```

Python

```
1 def cmp(p1, p2):
2
3     if p1[1] == p2[1]:
4         return p1[0] < p2[0]
5     return p1[1] > p2[1]
6
7 s = input()
8
9 # 转换为小写并过滤非字母字符
10 filtered = ''.join([c.lower() for c in s if c.isalpha()])
11
12 # 统计每个字符的总频次
13 mp = {}
14 for c in filtered:
15     mp[c] = mp.get(c, 0) + 1
16
17 s = filtered
18 n = len(s)
19 res = []
20 tmp = ""
21
22 for i in range(n):
23     if i == 0:
24         tmp += s[i]
25         continue
26     # 连续终止
27     if s[i] != s[i - 1]:
28         c = s[i - 1]
29         length = len(tmp)
30         mp[c] -= length
31         if length == 1:
32             res.append((c, mp[c]))
33         else:
34             res.append((c, length))
35         tmp = s[i]
36     else:
37         tmp += s[i]
38
39 # 末尾边界处理
40 if tmp != "":
41     c = tmp[0]
42     length = len(tmp)
43     mp[c] -= length
44     if length == 1:
45         res.append((c, mp[c]))
```

```
46     else:
47         res.append((c, length))
48
49
50     res.sort(key=lambda x: (-x[1], x[0]))
51
52     res_string = ''.join(f'{x[0]}{x[1]}' for x in res)
53     print(res_string)
```

JavaScript

```
1 // 自定义排序
2 function cmp(p1, p2) {
3     if (p1[1] === p2[1]) {
4         return p1[0] < p2[0] ? -1 : 1;
5     }
6     return p1[1] > p2[1] ? -1 : 1;
7 }
8
9 const input = require('readline').createInterface({
10     input: process.stdin,
11     output: process.stdout
12 });
13
14 input.question('', (s) => {
15     let filtered = '';
16     // 转换为小写并过滤非字母字符
17     for (let c of s) {
18         if (/[^a-zA-Z]/.test(c)) {
19             filtered += c.toLowerCase();
20         }
21     }
22
23     // 统计每个字符的总频次
24     const mp = {};
25     for (let c of filtered) {
26         mp[c] = (mp[c] || 0) + 1;
27     }
28
29     const res = [];
30     let tmp = '';
31     for (let i = 0; i < filtered.length; i++) {
32         if (i === 0 || filtered[i] !== filtered[i - 1]) {
33             if (tmp.length > 0) {
34                 const c = tmp[0];
35                 const length = tmp.length;
36                 mp[c] -= length;
37                 res.push([c, length === 1 ? mp[c] : length]);
38             }
39             tmp = filtered[i];
40         } else {
41             tmp += filtered[i];
42         }
43     }
44
45     // 末尾边界处理
```

```
46     if (tmp.length > 0) {
47         const c = tmp[0];
48         const length = tmp.length;
49         mp[c] -= length;
50         res.push([c, length === 1 ? mp[c] : length]);
51     }
52
53     // 排序
54     res.sort(cmp);
55
56     // 输出
57     console.log(res.map(x => `${x[0]}${x[1]}`).join(''));
58
59     input.close();
60 });
});
```

Go

```
1 package main
2
3 import (
4     "fmt"
5     "sort"
6     "unicode"
7 )
8
9 // 定义一个字符-数字对
10 type CharCount struct {
11     char rune
12     count int
13 }
14
15 // 自定义排序规则
16 func customSort(p1, p2 CharCount) bool {
17     if p1.count == p2.count {
18         return p1.char < p2.char
19     }
20     return p1.count > p2.count
21 }
22
23 func main() {
24     // 输入字符串
25     var s string
26     fmt.Scanln(&s)
27
28     // 转换为小写并过滤非字母字符
29     filtered := ""
30     for _, c := range s {
31         if unicode.IsLetter(c) {
32             filtered += string(unicode.ToLower(c))
33         }
34     }
35
36     // 统计每个字符的总频次
37     frequency := make(map[rune]int)
38     for _, c := range filtered {
39         frequency[c]++
40     }
41
42     // 遍历并处理连续字符
43     var result []CharCount
44     n := len(filtered)
45     tmp := ""
```

```
46
47     for i := 0; i < n; i++ {
48         if i == 0 {
49             tmp += string(filtered[i])
50             continue
51         }
52         // 连续字符终止
53         if filtered[i] != filtered[i-1] {
54             c := rune(filtered[i-1])
55             length := len(tmp)
56             frequency[c] -= length
57             if length == 1 {
58                 result = append(result, CharCount{c, frequency[c]})  

59             } else {
60                 result = append(result, CharCount{c, length})
61             }
62             tmp = string(filtered[i])
63         } else {
64             tmp += string(filtered[i])
65         }
66     }
67
68     // 处理末尾剩余字符
69     if tmp != "" {
70         c := rune(tmp[0])
71         length := len(tmp)
72         frequency[c] -= length
73         if length == 1 {
74             result = append(result, CharCount{c, frequency[c]})  

75         } else {
76             result = append(result, CharCount{c, length})
77         }
78     }
79
80     // 按自定义规则排序
81     sort.Slice(result, func(i, j int) bool {
82         return customSort(result[i], result[j])
83     })
84
85     // 构建输出结果字符串
86     resString := ""
87     for _, item := range result {
88         resString += fmt.Sprintf("%c%d", item.char, item.count)
89     }
90
91     // 输出结果
92     fmt.Println(resString)
93 }
```

字符串摘要

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

给定一个字符串的摘要算法，请输出给定字符串的摘要值。

1. 去除字符串中非字母的符号。
2. 如果出现连续字符(不区分大小写)，则输出：该字符(小写) + 连续出现的次数。
3. 如果是非连续的字符(不区分大小写)，则输出：该字符(小写) + 该字母之后字符串中出现的该字符的次数
4. 对按照以上方式表示后的字符串进行排序：字母和紧随的数字作为一组进行排序，数字大的在前，数字相同的，则按字母进行排序，字母小的在前。

输入描述

一行字符串，长度为[1,200]

输出描述

摘要字符串

用例1

输入

| | |
|---|------------|
| ▼ | Plain Text |
| 1 | aabbcc |

输出

| | |
|---|------------|
| ▼ | Plain Text |
| 1 | a2b2c2 |

说明

无

用例2

输入

Plain Text |

1 bAaAcBb

输出

Plain Text |

1 a3b2b2c0

说明

bAaAcBb:第一个b非连续字母，该字母之后字符串中还出现了2次(最后的两个Bb)，所以输出b2,a连续出现3次，输出a3，c非连续，该字母之后字符串再没有出现过c，输出c0,Bb连续2次，输出b2对b2a3c0b2进行排序，最终输出a3b2b2c0。

题解

思路：模拟

1. 字符串中字符去除非字母字符，字母字符全部转换为小写。
2. 使用 哈希表 统计各个字符出现的数量，用于出现不连续字符的情况下使用。
3. 进行遍历，当 $s[i] \neq s[i-1]$ 作为条件，计算出 $s[i-1]$ 字符出现的连续个数 size，size =1为 $count[s[i-1]] - 1$ ，否则为size。将{s[i-1], count[s[i-1]] - 1}或{s[i-1], size}插入结果数组中
4. 自定义排序，然后按照顺序输出结果。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<map>
8 using namespace std;
9
10 // 自定义排序
11 bool cmp(pair<char, int> p1, pair<char, int> p2) {
12     if (p1.second == p2.second) {
13         return p1.first < p2.first;
14     }
15     return p1.second > p2.second;
16 }
17
18
19 int main() {
20     string s;
21     cin >> s;
22     // 转换为小写并过滤非字母字符
23     string filtered = "";
24     for (char c : s) {
25         if (isalpha(c)) {
26             filtered += tolower(c);
27         }
28     }
29
30     // 统计每个字符的总频次
31     map<char, int> mp;
32     for (char c : filtered) {
33         mp[c]++;
34     }
35
36     s = filtered;
37     int n = s.size();
38     vector<pair<char, int>> res;
39     string tmp = "";
40     for (int i = 0; i < n; i++) {
41         if (i == 0) {
42             tmp += s[i];
43             continue;
44         }
45         // 连续终止
```

```

46         if (s[i] != s[i-1]) {
47             char c = s[i-1];
48             int length = tmp.size();
49             mp[c] -= length;
50             if (length == 1) {
51                 res.push_back({c, mp[c]});
52             } else {
53                 res.push_back({c, length});
54             }
55             tmp = string(1, s[i]);
56         } else {
57             tmp += s[i];
58         }
59     }
60     // 末尾边界处理
61     if (tmp != "") {
62         char c = tmp[0];
63         int length = tmp.size();
64         mp[c] -= length;
65         if (length == 1) {
66             res.push_back({c, mp[c]});
67         } else {
68             res.push_back({c, length});
69         }
70     }
71
72     sort(res.begin(), res.end(), cmp);
73     string resString = "";
74
75     for (int i = 0; i < res.size(); i++) {
76         pair<char,int> p = res[i];
77         resString += (string(1, p.first) + to_string(p.second));
78     }
79     cout << resString;
80     return 0;
81 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     // 自定义排序
5     static boolean cmp(Map.Entry<Character, Integer> p1, Map.Entry<Character, Integer> p2) {
6         if (p1.getValue().equals(p2.getValue())) {
7             return p1.getKey() < p2.getKey();
8         }
9         return p1.getValue() > p2.getValue();
10    }
11
12    public static void main(String[] args) {
13        Scanner sc = new Scanner(System.in);
14        String s = sc.nextLine();
15        StringBuilder filtered = new StringBuilder();
16
17        // 转换为小写并过滤非字母字符
18        for (char c : s.toCharArray()) {
19            if (Character.isAlphabetic(c)) {
20                filtered.append(Character.toLowerCase(c));
21            }
22        }
23
24        // 统计每个字符的总频次
25        Map<Character, Integer> mp = new HashMap<>();
26        for (char c : filtered.toString().toCharArray()) {
27            mp.put(c, mp.getOrDefault(c, 0) + 1);
28        }
29
30        StringBuilder tmp = new StringBuilder();
31        List<Map.Entry<Character, Integer>> res = new ArrayList<>();
32        for (int i = 0; i < filtered.length(); i++) {
33            if (i == 0 || filtered.charAt(i) != filtered.charAt(i - 1)) {
34                if (tmp.length() > 0) {
35                    char c = tmp.charAt(0);
36                    int length = tmp.length();
37                    mp.put(c, mp.get(c) - length);
38                    res.add(new AbstractMap.SimpleEntry<>(c, length == 1
? mp.get(c) : length));
39                }
40                tmp = new StringBuilder().append(filtered.charAt(i));
41            } else {
42                tmp.append(filtered.charAt(i));
43            }
44        }
45    }
46}
```

```
44 }
45
46     if (tmp.length() > 0) {
47         char c = tmp.charAt(0);
48         int length = tmp.length();
49         mp.put(c, mp.get(c) - length);
50         res.add(new AbstractMap.SimpleEntry<>(c, length == 1 ? mp.get
51             (c) : length));
52     }
53
54     // 排序
55     res.sort((p1, p2) -> cmp(p1, p2) ? -1 : 1);
56
57     StringBuilder result = new StringBuilder();
58     for (Map.Entry<Character, Integer> entry : res) {
59         result.append(entry.getKey()).append(entry.getValue());
60     }
61
62     System.out.println(result);
63     sc.close();
64 }
```

Python

```
1 def cmp(p1, p2):
2
3     if p1[1] == p2[1]:
4         return p1[0] < p2[0]
5     return p1[1] > p2[1]
6
7 s = input()
8
9 # 转换为小写并过滤非字母字符
10 filtered = ''.join([c.lower() for c in s if c.isalpha()])
11
12 # 统计每个字符的总频次
13 mp = {}
14 for c in filtered:
15     mp[c] = mp.get(c, 0) + 1
16
17 s = filtered
18 n = len(s)
19 res = []
20 tmp = ""
21
22 for i in range(n):
23     if i == 0:
24         tmp += s[i]
25         continue
26     # 连续终止
27     if s[i] != s[i - 1]:
28         c = s[i - 1]
29         length = len(tmp)
30         mp[c] -= length
31         if length == 1:
32             res.append((c, mp[c]))
33         else:
34             res.append((c, length))
35         tmp = s[i]
36     else:
37         tmp += s[i]
38
39 # 末尾边界处理
40 if tmp != "":
41     c = tmp[0]
42     length = len(tmp)
43     mp[c] -= length
44     if length == 1:
45         res.append((c, mp[c]))
```

```
46     else:
47         res.append((c, length))
48
49
50     res.sort(key=lambda x: (-x[1], x[0]))
51
52     res_string = ''.join(f"{x[0]}{x[1]}" for x in res)
53     print(res_string)
```

JavaScript

```
1 // 自定义排序
2 function cmp(p1, p2) {
3     if (p1[1] === p2[1]) {
4         return p1[0] < p2[0] ? -1 : 1;
5     }
6     return p1[1] > p2[1] ? -1 : 1;
7 }
8
9 const input = require('readline').createInterface({
10     input: process.stdin,
11     output: process.stdout
12 });
13
14 input.question('', (s) => {
15     let filtered = '';
16     // 转换为小写并过滤非字母字符
17     for (let c of s) {
18         if (/[^a-zA-Z]/.test(c)) {
19             filtered += c.toLowerCase();
20         }
21     }
22
23     // 统计每个字符的总频次
24     const mp = {};
25     for (let c of filtered) {
26         mp[c] = (mp[c] || 0) + 1;
27     }
28
29     const res = [];
30     let tmp = '';
31     for (let i = 0; i < filtered.length; i++) {
32         if (i === 0 || filtered[i] !== filtered[i - 1]) {
33             if (tmp.length > 0) {
34                 const c = tmp[0];
35                 const length = tmp.length;
36                 mp[c] -= length;
37                 res.push([c, length === 1 ? mp[c] : length]);
38             }
39             tmp = filtered[i];
40         } else {
41             tmp += filtered[i];
42         }
43     }
44
45     // 末尾边界处理
```

```
46     if (tmp.length > 0) {
47         const c = tmp[0];
48         const length = tmp.length;
49         mp[c] -= length;
50         res.push([c, length === 1 ? mp[c] : length]);
51     }
52
53     // 排序
54     res.sort(cmp);
55
56     // 输出
57     console.log(res.map(x => `${x[0]}${x[1]}`).join(''));
58
59     input.close();
60 });
});
```

Go

```
1 package main
2
3 import (
4     "fmt"
5     "sort"
6     "unicode"
7 )
8
9 // 定义一个字符-数字对
10 type CharCount struct {
11     char rune
12     count int
13 }
14
15 // 自定义排序规则
16 func customSort(p1, p2 CharCount) bool {
17     if p1.count == p2.count {
18         return p1.char < p2.char
19     }
20     return p1.count > p2.count
21 }
22
23 func main() {
24     // 输入字符串
25     var s string
26     fmt.Scanln(&s)
27
28     // 转换为小写并过滤非字母字符
29     filtered := ""
30     for _, c := range s {
31         if unicode.IsLetter(c) {
32             filtered += string(unicode.ToLower(c))
33         }
34     }
35
36     // 统计每个字符的总频次
37     frequency := make(map[rune]int)
38     for _, c := range filtered {
39         frequency[c]++
40     }
41
42     // 遍历并处理连续字符
43     var result []CharCount
44     n := len(filtered)
45     tmp := ""
```

```
46
47     for i := 0; i < n; i++ {
48         if i == 0 {
49             tmp += string(filtered[i])
50             continue
51         }
52         // 连续字符终止
53         if filtered[i] != filtered[i-1] {
54             c := rune(filtered[i-1])
55             length := len(tmp)
56             frequency[c] -= length
57             if length == 1 {
58                 result = append(result, CharCount{c, frequency[c]})  

59             } else {
60                 result = append(result, CharCount{c, length})
61             }
62             tmp = string(filtered[i])
63         } else {
64             tmp += string(filtered[i])
65         }
66     }
67
68     // 处理末尾剩余字符
69     if tmp != "" {
70         c := rune(tmp[0])
71         length := len(tmp)
72         frequency[c] -= length
73         if length == 1 {
74             result = append(result, CharCount{c, frequency[c]})  

75         } else {
76             result = append(result, CharCount{c, length})
77         }
78     }
79
80     // 按自定义规则排序
81     sort.Slice(result, func(i, j int) bool {
82         return customSort(result[i], result[j])
83     })
84
85     // 构建输出结果字符串
86     resString := ""
87     for _, item := range result {
88         resString += fmt.Sprintf("%c%d", item.char, item.count)
89     }
90
91     // 输出结果
92     fmt.Println(resString)
93 }
```

| 来自: 华为OD机试 2025 B卷 – 字符串摘要 (C++ & Python & JAVA & JS & GO)-CSDN博客

| 来自: 华为OD机试 2025 B卷 – 字符串摘要 (C++ & Python & JAVA & JS & GO)-CSDN博客

华为OD机试 2025 B卷 - 出错的或电路 (C++ & Python & JAVA & JS & GO)-CSDN博客

出错的或电路

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

某生产[门电路](#)的厂商发现某一批次的或门电路不稳定，具体表现为计算两个二进制数的或操作时，第一个二进制数中某两个比特位会出现交换，交换的比特位置是随机的，但只交换这两个位，其他位不变。很明显，这个交换可能会影响最终的或结果，也可能不会有影响。为了评估影响和定位出错的根因，工程师需要研究在各种交换的可能下，最终的或结果发生改变的情况有多少种。

输入描述

第一行有一个正整数N; 其中 $1 \leq N \leq 1000000$ 。

第二行有一个长为N的[二进制数](#)，表示与电路的第一个输入数，即会发生比特交换的输入数。

第三行有一个长为N的二进制数，表示与电路的第二个输入数。注意第二个输入数不会发生比特交换。

输出描述

输出只有一个整数，表示会影响或结果的交换方案个数。

示例1

输入

```
1 3
2 010
3 110
```

输出

```
1 1
```

说明

原本010和110的或结果是110，但第一个输入数可能会发生如下三种交换：

- 1、交换第1个比特和第2个比特，第一个输入数变为100，计算结果为110，计算结果不变
- 2、交换第1个比特和第3个比特，第一个输入数变为010，计算结果为110，计算结果不变
- 3、交换第2个比特和第3个比特，第一个输入数变为001，计算结果为111，计算结果改变
故只有一种交换会改变计算结果。

示例2

输入

```
Plain Text |  
1 6  
2 011011  
3 110110
```

输出

```
Plain Text |  
1 4
```

说明

原本011011和110110的或结果是111111，但是第一个输入数发生如下比特交换会影响最终计算结果：

- 1、交换第1个比特和第3个比特，第一个输入数变为110011，计算结果变为110111
- 2、交换第1个比特和第6个比特，第一个输入数变为111010，计算结果变为111110
- 3、交换第3个比特和第4个比特，第一个输入数变为010111，计算结果变为110111
- 4、交换第4个比特和第6个比特，第一个输入数变为011110，计算结果变为111100
其他交换都不会影响计算结果，故输出4.

题解

思路：逻辑

1. 根据与运算的特征，影响结果只会出现这两种情况
 - a. 第一个二进制对应位1，第二个二进制对应位为0。此时将第一个二进制换成0，结果就会发生改变。
 - b. 第一个二进制对应位0，第二个二进制对应为0的情况。此时将第一个二进制换成1，结果就会改变。
2. 根据1的推论，我们使用 `zeroCount, oneCount` 分别统计第一个二进制中0的数量，1的数量。使用 `diffCount0 diffCount1` 统计 `第一个二进制对应为0，第二个也为0` 的数量和 `第一个二进`

制位1，第二个二进制为0 的数量。

3. 如果影响结果一种情况的话：

a. 第一种情况的数量就为： `diffCount1 * zeroCount`

b. 第二种情况的数量为： `diffCount0 * oneCount`

4. 结果要求两种情况同时考虑，这时候需要注意，这两种情况实际上有重合片段，重合片段会发生在第一个二进制发生交换的两个比特一个属于 `diffCount1` 的情况，一个属于 `diffCount0` 的情况。这部分的数量就是 `diffCount0 * diffCount1`

5. 根据3 4 可以推断出最后的结果为 `diffCount1 * zeroCount + diffCount0 * oneCount - diffCount0 * diffCount1`

| 结果可能非常大，选择比int的数据类型存储结果。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 using namespace std;
10
11 int main() {
12     int n;
13     cin >> n;
14     string input1, input2;
15     cin >> input1;
16     cin >> input2;
17     // 统计第一个二进制数 1 的数量 0 的数量
18     long oneCount1, zeroCount;
19     oneCount1 = zeroCount = 0;
20     // 统计 第一个二进制 对应位为1 第二个二进制 为 0 的数量
21     long diffCount1;
22     // 统计 第一个二进制位0 第二个二进制也为0 的数量
23     long diffCount0;
24
25     diffCount1 = diffCount0 = 0;
26
27     for (int i = 0; i < n; i++) {
28         if (input1[i] == '1') {
29             oneCount1++;
30             if (input2[i] == '0') {
31                 diffCount1++;
32             }
33         } else {
34             zeroCount++;
35             if (input2[i] == '0') {
36                 diffCount0++;
37             }
38         }
39     }
40
41     long res = diffCount0 * oneCount1 + diffCount1 * zeroCount - diffCount0 * diffCount1;
42     cout << res;
43     return 0;
44 }
```

Java

```
Plain Text |
```

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         int n = Integer.parseInt(sc.nextLine());
8         String input1 = sc.nextLine();
9         String input2 = sc.nextLine();
10        // 分别统计第一个二进制 0比特的数量 1比特的数量
11        long oneCount1 = 0, zeroCount = 0;
12        // 第一个为0 第二个为0 的数量
13        long diffCount0 = 0;
14        // 第一个为1 第二个为0的数量
15        long diffCount1 = 0;
16
17        for (int i = 0; i < n; i++) {
18            if (input1.charAt(i) == '1') {
19                oneCount1++;
20                if (input2.charAt(i) == '0') {
21                    diffCount1++;
22                }
23            } else {
24                zeroCount++;
25                if (input2.charAt(i) == '0') {
26                    diffCount0++;
27                }
28            }
29        }
30
31        // 计算符合条件的组合数，减去重叠部分
32        long res = diffCount0 * oneCount1 + diffCount1 * zeroCount - diffC
33        ount0 * diffCount1;
34        System.out.println(res);
35    }
}
```

Python

```
1 def main():
2     n = int(input())
3     input1 = input().strip()
4     input2 = input().strip()
5     # 分别统计第一个二进制 0比特的数量 1比特的数量
6     one_count1 = zero_count = 0
7     # 第一个为0 第二个为0 的数量
8     diff_count0 = 0
9     # 第一个为1 第二个为0的数量
10    diff_count1 = 0
11
12    for i in range(n):
13        if input1[i] == '1':
14            one_count1 += 1
15            if input2[i] == '0':
16                diff_count1 += 1
17        else:
18            zero_count += 1
19            if input2[i] == '0':
20                diff_count0 += 1
21
22    # 计算符合条件的组合数，减去重叠部分
23    res = diff_count0 * one_count1 + diff_count1 * zero_count - diff_count
24    0 * diff_count1
25
26    print(res)
27
28    main()
```

JavaScript

```
1 const readline = require('readline');
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });
7
8 let lines = [];
9
10 rl.on('line', line => {
11     lines.push(line.trim());
12     if (lines.length === 3) rl.close();
13 });
14
15 rl.on('close', () => {
16     const n = parseInt(lines[0]);
17     const input1 = lines[1];
18     const input2 = lines[2];
19     // 分别统计第一个二进制 0比特的数量 1比特的数量
20     let oneCount1 = 0n, zeroCount = 0n;
21     // 第一个为0 第二个为0 的数量
22     let diffCount0 = 0n;
23     // 第一个为1 第二个为0的数量
24     let diffCount1 = 0n;
25
26     for (let i = 0; i < n; i++) {
27         const a = input1[i], b = input2[i];
28         if (a === '1') {
29             oneCount1++;
30             if (b === '0') diffCount1++;
31         } else {
32             zeroCount++;
33             if (b === '0') diffCount0++;
34         }
35     }
36
37     // 注意所有数都需要显式转换为 BigInt 类型
38     const res = diffCount0 * oneCount1 + diffCount1 * zeroCount - diffCount0 * diffCount1;
39     console.log(res.toString());
40 });
```

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strings"
8 )
9
10
11 func main() {
12     reader := bufio.NewReader(os.Stdin)
13
14     // 读取第一行: n
15     var n int
16     fmt.Fscanln(reader, &n)
17
18     // 读取第二行: input1
19     input1Line, _ := reader.ReadString('\n')
20     input1 := strings.TrimSpace(input1Line)
21
22     // 读取第三行: input2
23     input2Line, _ := reader.ReadString('\n')
24     input2 := strings.TrimSpace(input2Line)
25
26     // 分别统计第一个二进制 0比特的数量 1比特的数量
27     var oneCount1, zeroCount int64
28     // 第一个为0 第二个为0 的数量
29     var diffCount0 int64
30     // 第一个为1 第二个为0的数量
31     var diffCount1 int64
32
33     for i := 0; i < n; i++ {
34         a := input1[i]
35         b := input2[i]
36
37         if a == '1' {
38             oneCount1++
39             if b == '0' {
40                 diffCount1++
41             }
42         } else {
43             zeroCount++
44             if b == '0' {
45                 diffCount0++
```

```
46         }
47     }
48 }
49
50 // 按照公式计算: 去掉重复计算项
51 res := diffCount0*oneCount1 + diffCount1*zeroCount - diffCount0*diffCoun
52 t1
53 fmt.Println(res)
54 }
```

| 来自: 华为OD机试 2025 B卷 – 出错的或电路 (C++ & Python & JAVA & JS & GO)–CSDN博客

华为OD机试2025B卷 - 快递运输(C++ & Python & Java & JS & Go)-CSDN博客

快递运输

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

一辆运送快递的货车，运送的快递放在大小不等的长方体快递盒中，为了能够装载更多的快递，同时不能让货车超载，需要计算最多能装多少个快递。注：快递的体积不受限制，快递数最多1000个，货车载重最大50000

输入描述

第一行输入每个快递的重量，用英文逗号隔开，如 5,10,2,11

第二行输入货车的载重量，如 20

输出描述

输出最多能装多少个快递，如 3

示例1

输入

```
1 5,10,2,11  
2 20
```

输出

```
1 3
```

说明

货车的载重量为20，最多只能放三个快递5、10、2，因此输出3

题解一

思路: 贪心

1. 为了尽可能装更多数量快递，优先选择轻的快递。
2. 将输入重量进行排序，从前往后装快递，直到不能装为止。
3. 按照1 2 逻辑处理，输出最后计算而来的数量。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 using namespace std;
10
11
12 // 通用 split 函数
13 vector<int> split(const string& str, const string& delimiter) {
14     vector<int> result;
15     size_t start = 0;
16     size_t end = str.find(delimiter);
17     while (end != string::npos) {
18         result.push_back(stoi(str.substr(start, end - start)));
19         start = end + delimiter.length();
20         end = str.find(delimiter, start);
21     }
22     // 添加最后一个部分
23     result.push_back(stoi(str.substr(start)));
24     return result;
25 }
26
27
28 int main() {
29     string input;
30     getline(cin, input);
31     vector<int> weights = split(input, ",");
32     // 从小到大排序
33     sort(weights.begin(), weights.end());
34     int capacity;
35     cin >> capacity;
36     int n = weights.size();
37     int currentSum = 0;
38     int res = 0;
39     for (int i = 0; i < n; i++) {
40         if (currentSum + weights[i] > capacity) {
41             break;
42         }
43         currentSum += weights[i];
44         res++;
45     }
}
```

```
46     cout << res;
47     return 0;
48 }
```

Java

```
Plain Text |
```

```
1 import java.util.*;
2
3 public class Main {
4     // 通用 split 函数
5     public static List<Integer> split(String str, String delimiter) {
6         List<Integer> result = new ArrayList<>();
7         for (String s : str.split(delimiter)) {
8             result.add(Integer.parseInt(s));
9         }
10        return result;
11    }
12
13    public static void main(String[] args) {
14        Scanner sc = new Scanner(System.in);
15        String input = sc.nextLine();
16        List<Integer> weights = split(input, ",");
17
18        // 从小到大排序
19        Collections.sort(weights);
20
21        int capacity = sc.nextInt();
22        int currentSum = 0, res = 0;
23
24        for (int weight : weights) {
25            if (currentSum + weight > capacity) break;
26            currentSum += weight;
27            res++;
28        }
29
30        System.out.println(res);
31    }
32 }
```

Python

```
1 def split(s, delimiter):
2     return list(map(int, s.strip().split(delimiter)))
3
4 def main():
5     input_line = input()
6     weights = split(input_line, ",")
7
8     # 从小到大排序
9     weights.sort()
10
11    capacity = int(input())
12    current_sum = 0
13    res = 0
14
15    for w in weights:
16        if current_sum + w > capacity:
17            break
18        current_sum += w
19        res += 1
20
21    print(res)
22
23 main()
```

JavaScript

```
1 const readline = require('readline');
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });
7
8 let lines = [];
9
10 rl.on('line', line => {
11     lines.push(line);
12     if (lines.length === 2) rl.close();
13 }).on('close', () => {
14     let weights = lines[0].split(',').map(Number);
15     let capacity = parseInt(lines[1]);
16
17     // 从小到大排序
18     weights.sort((a, b) => a - b);
19
20     let currentSum = 0;
21     let res = 0;
22     for (let w of weights) {
23         if (currentSum + w > capacity) break;
24         currentSum += w;
25         res++;
26     }
27
28     console.log(res);
29 });
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9     "sort"
10    )
11
12 func split(s, delimiter string) []int {
13     parts := strings.Split(s, delimiter)
14     res := make([]int, len(parts))
15     for i, v := range parts {
16         res[i], _ = strconv.Atoi(v)
17     }
18     return res
19 }
20
21 func main() {
22     scanner := bufio.NewScanner(os.Stdin)
23     scanner.Scan()
24     input := scanner.Text()
25     weights := split(input, ",")
26
27     // 从小到大排序
28     sort.Ints(weights)
29
30     scanner.Scan()
31     capacity, _ := strconv.Atoi(scanner.Text())
32
33     currentSum := 0
34     res := 0
35     for _, w := range weights {
36         if currentSum+w > capacity {
37             break
38         }
39         currentSum += w
40         res++
41     }
42
43     fmt.Println(res)
44 }
```

题解二

思路：经典[01背包问题](#)。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 using namespace std;
10
11
12 // 通用 split 函数
13 vector<int> split(const string& str, const string& delimiter) {
14     vector<int> result;
15     size_t start = 0;
16     size_t end = str.find(delimiter);
17     while (end != string::npos) {
18         result.push_back(stoi(str.substr(start, end - start)));
19         start = end + delimiter.length();
20         end = str.find(delimiter, start);
21     }
22     // 添加最后一个部分
23     result.push_back(stoi(str.substr(start)));
24     return result;
25 }
26
27
28 int main() {
29     string input;
30     getline(cin, input);
31     vector<int> weights = split(input, ",");
32     int capacity;
33     cin >> capacity;
34     vector<int> dp(capacity + 1, 0);
35
36     int n = weights.size();
37     // 背包问题
38     for (int i = 0; i < n; i++) {
39         for (int j = capacity; j >= weights[i]; j--) {
40             dp[j] = max(dp[j], dp[j - weights[i]] + 1);
41         }
42     }
43     cout << dp[capacity];
44     return 0;
45 }
```

Java

```
1 import java.util.*;
2
3 public class Main {
4     // 通用 split 函数
5     public static List<Integer> split(String str, String delimiter) {
6         List<Integer> result = new ArrayList<>();
7         for (String s : str.split(delimiter)) {
8             result.add(Integer.parseInt(s));
9         }
10        return result;
11    }
12
13    public static void main(String[] args) {
14        Scanner sc = new Scanner(System.in);
15        String input = sc.nextLine();
16        List<Integer> weights = split(input, ",");
17        int capacity = sc.nextInt();
18        int[] dp = new int[capacity + 1];
19
20        // 背包问题
21        for (int weight : weights) {
22            for (int j = capacity; j >= weight; j--) {
23                dp[j] = Math.max(dp[j], dp[j - weight] + 1);
24            }
25        }
26
27        System.out.println(dp[capacity]);
28    }
29 }
```

Python

```
1 def split(s, delimiter):
2     return list(map(int, s.strip().split(delimiter)))
3
4 def main():
5     input_line = input()
6     weights = split(input_line, ',')
7     capacity = int(input())
8     dp = [0] * (capacity + 1)
9
10    # 背包问题
11    for weight in weights:
12        for j in range(capacity, weight - 1, -1):
13            dp[j] = max(dp[j], dp[j - weight] + 1)
14
15    print(dp[capacity])
16
17 main()
```

JavaScript

```
1 const readline = require('readline');
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });
7
8 let lines = [];
9
10 rl.on('line', line => {
11     lines.push(line);
12 }).on('close', () => {
13     let weights = lines[0].split(',').map(Number);
14     let capacity = parseInt(lines[1]);
15     let dp = Array(capacity + 1).fill(0);
16
17     // 背包问题
18     for (let weight of weights) {
19         for (let j = capacity; j >= weight; j--) {
20             dp[j] = Math.max(dp[j], dp[j - weight] + 1);
21         }
22     }
23
24     console.log(dp[capacity]);
25 });
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 func split(s, delimiter string) []int {
12     parts := strings.Split(s, delimiter)
13     result := make([]int, len(parts))
14     for i, p := range parts {
15         result[i], _ = strconv.Atoi(p)
16     }
17     return result
18 }
19
20 func main() {
21     scanner := bufio.NewScanner(os.Stdin)
22     scanner.Scan()
23     input := scanner.Text()
24     weights := split(input, ",")
25
26     scanner.Scan()
27     capacity, _ := strconv.Atoi(scanner.Text())
28     dp := make([]int, capacity+1)
29
30     // 背包问题
31     for _, weight := range weights {
32         for j := capacity; j >= weight; j-- {
33             if dp[j-weight]+1 > dp[j] {
34                 dp[j] = dp[j-weight] + 1
35             }
36         }
37     }
38     fmt.Println(dp[capacity])
39 }
```

华为OD机试2025B卷 - 求解连续序列(C++ & Python & JAVA & JS & GO)-CSDN博客

求解连续序列

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

已知连续正整数数列{K}=K₁, K₂, K₃...K_i的各个数相加之和为S, i=N (0<S<100000, 0<N<100000), 求此数列K。

输入描述

输入包含两个参数,

1. 连续正整数数列和S
2. 数列里数的个数N。

输出描述

如果有解输出数列K, 如果无解输出-1。

示例1

输入

```
1 525 6
```

输出

```
1 85 86 87 88 89 90
```

示例2

输入

▼

Plain Text |

1 3 5

输出

▼

Plain Text |

1 -1

题解

思路： 数学原理

1. 假设连续整数序列的第一个数为 x ，将 x 带入数列中表达形式可以变为 $x \ x+1 \ \dots \ x + n-1$ ，根据特征其实可以将这个序列转换为 $nx + \text{所有数于第一个数差值和}$ 。差值满足等差数列。可以利用 $S = n \cdot (2a_1 + (n-1)d) / 2$ 求和。
2. 直到1的原理之后，我们计算出等差数列和 sum ，然后判断 $(s - sum) \% n$ 是否等于0。
 - 不等于0，直接输出-1。不存在该序列。
 - 等于0，说明存在合法序列。知道首个值及长度接下来就比较简单，可以构造出序列即可。

C++

```
1 #include<iostream>
2 #include<vector>
3 #include<string>
4 #include <utility>
5 #include <sstream>
6 #include<algorithm>
7 #include<cmath>
8 #include<map>
9 using namespace std;
10
11 int main() {
12     long long s,n;
13     cin >> s >> n;
14
15     // 正整数性质决定的
16     if (n > s) {
17         cout << -1;
18         return 0;
19     }
20
21     // 等差数列公式 设第一个为x 差值是一个等差数列,
22     // 计算得来的是等差数列的和
23     long long sum = (1LL * n * (n-1)) / 2;
24     // 不能整除说明不存在
25     if ((s - sum) % n != 0) {
26         cout << -1;
27     } else{
28         long long start = (s - sum) / n;
29         for (int i = 0; i < n; i++) {
30             cout << start + i;
31             if (i != n - 1) {
32                 cout << " ";
33             }
34         }
35     }
36     return 0;
37 }
```

Java

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         long s = sc.nextLong();
7         long n = sc.nextLong();
8
9         // 正整数性质决定的
10        if (n > s) {
11            System.out.println(-1);
12            return;
13        }
14
15        // 等差数列求和, 首项为 x, 差值为 1, 求和公式为: x*n + n*(n-1)/2 = s
16        long sum = n * (n - 1) / 2;
17
18        if ((s - sum) % n != 0) {
19            System.out.println(-1);
20        } else {
21            long start = (s - sum) / n;
22            for (int i = 0; i < n; i++) {
23                System.out.print(start + i);
24                if (i != n - 1) {
25                    System.out.print(" ");
26                }
27            }
28        }
29    }
30}
```

Python

```
1 s, n = map(int, input().split())
2
3 # 正整数性质决定的
4 if n > s:
5     print(-1)
6 else:
7     # 等差数列求和: 首项为 x, 差值为 1
8     sum_val = n * (n - 1) // 2
9     if (s - sum_val) % n != 0:
10        print(-1)
11    else:
12        start = (s - sum_val) // n
13        res = [str(start + i) for i in range(n)]
14        print(" ".join(res))
```

JavaScript

```
1 const readline = require('readline');
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });
7
8 let inputLines = [];
9
10 rl.on('line', function(line) {
11     inputLines.push(line.trim());
12 }).on('close', function() {
13     let [s, n] = inputLines[0].split(' ').map(BigInt);
14
15     // 正整数性质决定的
16     if (n > s) {
17         console.log(-1);
18         return;
19     }
20
21     // 等差数列和公式 等差数列公式 设第一个为x 差值是一个等差数列,
22     let sum = n * (n - 1n) / 2n;
23
24     if ((s - sum) % n !== 0n) {
25         console.log(-1);
26     } else {
27         let start = (s - sum) / n;
28         let res = [];
29         for (let i = 0n; i < n; i++) {
30             res.push((start + i).toString());
31         }
32         console.log(res.join(' '));
33     }
34 });
35 
```

Go

```
1 package main
2
3 import (
4     "fmt"
5     "math/big"
6 )
7
8 func main() {
9     var sStr, nStr string
10    fmt.Scan(&sStr, &nStr)
11
12    s := new(big.Int)
13    n := new(big.Int)
14    s.SetString(sStr, 10)
15    n.SetString(nStr, 10)
16
17    // 如果 n > s 输出 -1
18    if n.Cmp(s) == 1 {
19        fmt.Println(-1)
20        return
21    }
22
23    one := big.NewInt(1)
24    two := big.NewInt(2)
25
26    // 等差数列和公式 等差数列公式 设第一个为x 差值是一个等差数列,
27    nMinus1 := new(big.Int).Sub(n, one)
28    sum := new(big.Int).Mul(n, nMinus1)
29    sum.Div(sum, two)
30
31    remain := new(big.Int).Sub(s, sum)
32    mod := new(big.Int)
33    mod.Mod(remain, n)
34    // 不能整除说明不存在
35    if mod.Cmp(big.NewInt(0)) != 0 {
36        fmt.Println(-1)
37        return
38    }
39
40    start := new(big.Int).Div(remain, n)
41    res := make([]string, 0, n.Int64())
42    for i := int64(0); i < n.Int64(); i++ {
43        val := new(big.Int).Add(start, big.NewInt(i))
44        res = append(res, val.String())
45    }
```

```
46     fmt.Println(fmt.Sprint(res)[1 : len(fmt.Sprint(res))-1]) // 去掉 []
47 }
```

| 来自: 华为OD 机试 2025 B卷 – 求解连续序列 (C++ & Python & JAVA & JS & GO)-CSDN博客

华为OD机试2025B卷 - 投篮大赛(C++ & Python & Java & JS & Go)-CSDN博客

投篮大赛

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

你现在是一场采用特殊赛制投篮大赛的记录员。这场比赛由若干回合组成，过去几回合的得分可能会影响以后几回合的得分。

比赛开始时，记录是空白的。

你会得到一个记录操作的字符串列表 ops，其中ops[i]是你需要记录的第i项操作，ops遵循下述规则：

- 整数x - 表示本回合新获得分数x
- “+” - 表示本回合新获得的得分是前两次得分的总和。
- “D” - 表示本回合新获得的得分是前一次得分的两倍。
- “C” - 表示本回合没有分数，并且前一次得分无效，将其从记录中移除。

请你返回记录中所有得分的总和。

输入描述

输入为一个字符串数组

输出描述

输出为一个整形数字

提示

1. $1 \leq \text{ops.length} \leq 1000$
2. $\text{ops}[i]$ 为 “C”、“D”、“+”，或者一个表示整数的字符串。整数范围是 $[-3 * 10^4, 3 * 10^4]$
3. 需要考虑异常的存在，如有异常情况，请返回-1
4. 对于“+”操作，题目数据不保证记录此操作时前面总是存在两个有效的分数
5. 对于“C”和“D”操作，题目数据不保证记录此操作时前面存在一个有效的分数
6. 题目输出范围不会超过整型的最大范围，不超过 $2^{63} - 1$

用例1

输入

Plain Text |

1 5 2 C D +

输出

Plain Text |

1 30

说明

Plain Text |

1 “5”–记录加5，记录现在是[5]
2
3 “2”–记录加2，记录现在是[5, 2]
4
5 “C”–使前一次得分的记录无效并将其移除，记录现在是[5].
6
7 “D”–记录加 $2 \times 5 = 10$ ，记录现在是[5, 10].
8
9 “+”–记录加 $5 + 10 = 15$ ，记录现在是[5, 10, 15].
10
11 所有得分的总和 $5 + 10 + 15 = 30$

用例2

输入

Plain Text |

1 5 -2 4 C D 9 + +

输出

Plain Text |

1 27

说明

“5”–记录加5，记录现在是[5]
“-2”–记录加-2，记录现在是[5,-2]
“4”–记录加4，记录现在是[5,-2,4]
“C”–使前一次得分的记录无效并将其移除，记录现在是[5,-2].
“D”–记录加 $2 \times -2 = 4$ ，记录现在是[5, -2, -4].
“9”–记录加9，记录现在是[5, -2, -4, 9].
“+”–记录加 $-4 + 9 = 5$ ，记录现在是[5, -2, -4, 9, 5].
“+”–记录加 $-9 + 5 = 14$ ，记录现在是[5, -2, -4, 9, 5, 14].
所以得分的总和 $5 - 2 - 4 + 9 + 5 + 14 = 27$

用例3

输入

Plain Text |

```
1 1
```

输出

Plain Text |

```
1 1
```

用例4

输入

Plain Text |

```
1 +
```

输出

Plain Text |

```
1 -1
```

题解

思路：模拟

1. 按照题目要求进行 **模拟** 计算即可。定义 **res[]** 用于存储每一轮的得分情况。对于不同指令的处理情况如下：

- 数字 **x**, 直接在**res**数组尾部添加**x**。
- **+** , 首先判断**res**的长度是否大于等于2, 不满足说明出现异常, 直接结束(异常)。满足的话, 计算**res**尾部两个元素和得到**y**, 将**y**添加至**res**的尾部。
- **D** ,首先判断**res**的长度是否大于等于1, 不满足说明出现异常, 直接结束(异常).满足的话, 将 **re**
s最后一个元素 * 2 得到 **y**,将**y**添加到**res**的尾部。
- **C** ,首先判断**res**的长度是否大于等于1, 不满足说明出现异常, 直接结束(异常).满足的话, 移除
res的尾部元素。
- 出现其它情况直接结束, 异常情况。

2. 按照1的思路, 如果出现异常的话直接输出 **-1** 。没有出现异常的话, 累计**res**数组的和就是结果。

C++

```
1 #include <ctime>
2 #include<iostream>
3 #include<vector>
4 #include<string>
5 #include <utility>
6 #include <sstream>
7 #include<algorithm>
8 #include<list>
9 #include<queue>
10 #include <regex>
11 using namespace std;
12
13 // 通用 split 函数
14 vector<string> split(const string& str, const string& delimiter) {
15     vector<string> result;
16     size_t start = 0;
17     size_t end = str.find(delimiter);
18     while (end != string::npos) {
19         result.push_back(str.substr(start, end - start));
20         start = end + delimiter.length();
21         end = str.find(delimiter, start);
22     }
23     // 添加最后一个部分
24     result.push_back(str.substr(start));
25     return result;
26 }
27
28 // 判断是否符合整数形式
29 bool isInteger(const std::string& str) {
30     std::regex pattern(R"(^-?\d+$)"); // 匹配可选的 "-" 号后跟 1 个或多个数字
31     return std::regex_match(str, pattern);
32 }
33
34
35 int main() {
36     string s;
37     getline(cin ,s);
38     vector<string> ans = split(s, " ");
39     // 记录是否出现异常
40     bool flag = false;
41     vector<int> res;
42     for (int i = 0; i < ans.size(); i++) {
43         string tmp = ans[i];
44         if (tmp == "C") {
45             if (res.size() == 0) {
```

```
46             flag = true;
47             break;
48         }
49         res.pop_back();
50     } else if (tmp == "D") {
51         if (res.size() == 0) {
52             flag = true;
53             break;
54         }
55         res.push_back(res[res.size() - 1] * 2);
56     } else if (tmp == "+") {
57         if (res.size() < 2) {
58             flag = true;
59             break;
60         }
61         res.push_back(res[res.size() - 1] + res[res.size() - 2]);
62     // 数字
63     } else {
64         if (!isInteger(tmp)){
65             flag = true;
66             break;
67         }
68         res.push_back(stoi(tmp));
69     }
70 }
71 if (flag) {
72     cout << -1;
73     return 0;
74 }
75 int sum = 0;
76 for (int i = 0; i < res.size(); i++) {
77     sum += res[i];
78 }
79 cout << sum;
80 return 0;
81 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     // 通用 split 方法
5     private static List<String> split(String str, String delimiter) {
6         return Arrays.asList(str.split(delimiter));
7     }
8
9     // 判断是否为整数
10    private static boolean isInteger(String str) {
11        return str.matches("-?\\d+"); // 正则匹配整数
12    }
13
14    public static void main(String[] args) {
15        Scanner scanner = new Scanner(System.in);
16        String s = scanner.nextLine();
17        scanner.close();
18
19        List<String> ops = split(s, " ");
20        List<Integer> res = new ArrayList<>();
21        boolean flag = false;
22
23        for (String op : ops) {
24            if (op.equals("C")) {
25                if (res.isEmpty()) {
26                    flag = true;
27                    break;
28                }
29                res.remove(res.size() - 1);
30            } else if (op.equals("D")) {
31                if (res.isEmpty()) {
32                    flag = true;
33                    break;
34                }
35                res.add(res.get(res.size() - 1) * 2);
36            } else if (op.equals("+")) {
37                if (res.size() < 2) {
38                    flag = true;
39                    break;
40                }
41                res.add(res.get(res.size() - 1) + res.get(res.size() -
42                2));
43            } else {
44                if (!isInteger(op)) {
45                    flag = true;
46                }
47            }
48        }
49
50        if (flag) {
51            System.out.println("非法操作");
52        } else {
53            System.out.println(res);
54        }
55    }
56}
```

```
45             break;
46         }
47         res.add(Integer.parseInt(op));
48     }
49 }
50
51 if (flag) {
52     System.out.println(-1);
53 } else {
54     System.out.println(res.stream().mapToInt(Integer::intValue).su
55 m());
56 }
57 }
```

Python

```
1 import sys
2 import re
3
4 def is_integer(s):
5     return bool(re.fullmatch(r"-?\d+", s)) # 正则匹配整数
6
7 def main():
8     s = sys.stdin.readline().strip()
9     ops = s.split(" ") # 直接用 split 分割字符串
10    res = []
11
12    for op in ops:
13        if op == "C":
14            if not res:
15                print(-1)
16                return
17            res.pop()
18        elif op == "D":
19            if not res:
20                print(-1)
21                return
22            res.append(res[-1] * 2)
23        elif op == "+":
24            if len(res) < 2:
25                print(-1)
26                return
27            res.append(res[-1] + res[-2])
28        else:
29            if not is_integer(op):
30                print(-1)
31                return
32            res.append(int(op))
33
34    print(sum(res))
35
36 if __name__ == "__main__":
37     main()
```

JavaScript

```
1 const readline = require("readline");
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });
7
8 // 判断是否为整数
9 function isInteger(str) {
10     return /^-?\d+$/ .test(str);
11 }
12
13 rl.on("line", function (line) {
14     let ops = line.trim().split(" ");
15     let res = [];
16
17     for (let op of ops) {
18         if (op === "C") {
19             if (res.length === 0) {
20                 console.log(-1);
21                 rl.close();
22                 return;
23             }
24             res.pop();
25         } else if (op === "D") {
26             if (res.length === 0) {
27                 console.log(-1);
28                 rl.close();
29                 return;
30             }
31             res.push(res[res.length - 1] * 2);
32         } else if (op === "+") {
33             if (res.length < 2) {
34                 console.log(-1);
35                 rl.close();
36                 return;
37             }
38             res.push(res[res.length - 1] + res[res.length - 2]);
39         } else {
40             if (!isInteger(op)) {
41                 console.log(-1);
42                 rl.close();
43                 return;
44             }
45             res.push(parseInt(op, 10));
46         }
47     }
48     console.log(res);
49     rl.close();
50 }
51
52 rl.on("SIGINT", () => {
53     process.exit(0);
54 })
```

```
46      }
47    }
48
49    console.log(res.reduce((sum, num) => sum + num, 0));
50    rl.close();
51  });

```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "regexp"
8     "strconv"
9     "strings"
10    )
11
12 // 判断是否为整数
13 func isInteger(s string) bool {
14     matched, _ := regexp.MatchString(`^-?\d+$`, s)
15     return matched
16 }
17
18 func main() {
19     scanner := bufio.NewScanner(os.Stdin)
20     scanner.Scan()
21     s := scanner.Text()
22
23     ops := strings.Fields(s) // 直接按空格分割
24     res := []int{}
25     flag := false
26
27     for _, op := range ops {
28         if op == "C" {
29             if len(res) == 0 {
30                 flag = true
31                 break
32             }
33             res = res[:len(res)-1] // 删除最后一个元素
34         } else if op == "D" {
35             if len(res) == 0 {
36                 flag = true
37                 break
38             }
39             res = append(res, res[len(res)-1]*2)
40         } else if op == "+" {
41             if len(res) < 2 {
42                 flag = true
43                 break
44             }
45             res = append(res, res[len(res)-1]+res[len(res)-2])
46         }
47     }
48 }
```

```
46     } else {
47         if !isInteger(op) {
48             flag = true
49             break
50         }
51         num, _ := strconv.Atoi(op)
52         res = append(res, num)
53     }
54 }
55
56 if flag {
57     fmt.Println(-1)
58 } else {
59     sum := 0
60     for _, v := range res {
61         sum += v
62     }
63     fmt.Println(sum)
64 }
65 }
```

投篮大赛

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

题目描述

你现在是一场采用特殊赛制投篮大赛的记录员。这场比赛由若干回合组成，过去几回合的得分可能会影响以后几回合的得分。

比赛开始时，记录是空白的。

你会得到一个记录操作的字符串列表 ops，其中ops[i]是你需要记录的第i项操作，ops遵循下述规则：

- 整数x – 表示本回合新获得分数x
- “+” – 表示本回合新获得的得分是前两次得分的总和。
- “D” – 表示本回合新获得的得分是前一次得分的两倍。
- “C” – 表示本回合没有分数，并且前一次得分无效，将其从记录中移除。

请你返回记录中所有得分的总和。

输入描述

输入为一个字符串数组

输出描述

输出为一个整形数字

提示

1. $1 \leq \text{ops.length} \leq 1000$
2. $\text{ops}[i]$ 为 “C”、“D”、“+”，或者一个表示整数的字符串。整数范围是 $[-3 * 10^4, 3 * 10^4]$
3. 需要考虑异常的存在，如有异常情况，请返回-1
4. 对于“+”操作，题目数据不保证记录此操作时前面总是存在两个有效的分数
5. 对于“C”和“D”操作，题目数据不保证记录此操作时前面存在一个有效的分数
6. 题目输出范围不会超过整型的最大范围，不超过 $2^{63} - 1$

用例1

输入

```
Plain Text |  
1 5 2 C D +
```

输出

```
Plain Text |  
1 30
```

说明

```
Plain Text |  
1 “5”–记录加5，记录现在是[5]  
2  
3 “2”–记录加2，记录现在是[5,2]  
4  
5 “C”–使前一次得分的记录无效并将其移除，记录现在是[5].  
6  
7 “D”–记录加 $2 \times 5 = 10$ ，记录现在是[5, 10].  
8  
9 “+”–记录加 $5 + 10 = 15$ ，记录现在是[5, 10, 15].  
10  
11 所有得分的总和 $5 + 10 + 15 = 30$ 
```

用例2

输入

Plain Text |

1 5 -2 4 C D 9 + +

输出

Plain Text |

1 27

说明

“5”–记录加5，记录现在是[5]

“-2”–记录加-2，记录现在是[5,-2]

“4”–记录加4，记录现在是[5,-2,4]

“C”–使前一次得分的记录无效并将其移除，记录现在是[5,-2].

“D”–记录加 $2 \times -2 = -4$ ，记录现在是[5, -2, -4].

“9”–记录加9，记录现在是[5, -2, -4, 9].

“+”–记录加 $-4 + 9 = 5$ ，记录现在是[5, -2, -4, 9, 5].

“+”–记录加 $-9 + 5 = -4$ ，记录现在是[5, -2, -4, 9, 5, 14].

所以得分的总和 $5 - 2 - 4 + 9 + 5 + 14 = 27$

用例3

输入

Plain Text |

1 1

输出

Plain Text |

1 1

用例4

输入

Plain Text |

1 +

输出

Plain Text |

1 -1

题解

思路：模拟

1. 按照题目要求进行 模拟 计算即可。定义 `res[]` 用于存储每一轮的得分情况。对于不同指令的处理情况如下：
 - 数字 `x`, 直接在`res`数组尾部添加`x`。
 - `+`, 首先判断`res`的长度是否大于等于2, 不满足说明出现异常, 直接结束(异常)。满足的话, 计算`res`尾部两个元素和得到`y`, 将`y`添加至`res`的尾部。
 - `D`, 首先判断`res`的长度是否大于等于1, 不满足说明出现异常, 直接结束(异常).满足的话, 将 `res`最后一个元素 `* 2` 得到 `y`, 将`y`添加到`res`的尾部。
 - `C`, 首先判断`res`的长度是否大于等于1, 不满足说明出现异常, 直接结束(异常).满足的话, 移除`res`的尾部元素。
 - 出现其它情况直接结束, 异常情况。
2. 按照1的思路, 如果出现异常的话直接输出 `-1` 。没有出现异常的话, 累计`res`数组的和就是结果。

C++

```
1 #include <ctime>
2 #include<iostream>
3 #include<vector>
4 #include<string>
5 #include <utility>
6 #include <sstream>
7 #include<algorithm>
8 #include<list>
9 #include<queue>
10 #include <regex>
11 using namespace std;
12
13 // 通用 split 函数
14 vector<string> split(const string& str, const string& delimiter) {
15     vector<string> result;
16     size_t start = 0;
17     size_t end = str.find(delimiter);
18     while (end != string::npos) {
19         result.push_back(str.substr(start, end - start));
20         start = end + delimiter.length();
21         end = str.find(delimiter, start);
22     }
23     // 添加最后一个部分
24     result.push_back(str.substr(start));
25     return result;
26 }
27
28 // 判断是否符合整数形式
29 bool isInteger(const std::string& str) {
30     std::regex pattern(R"(^-?\d+$)"); // 匹配可选的 "-" 号后跟 1 个或多个数字
31     return std::regex_match(str, pattern);
32 }
33
34
35 int main() {
36     string s;
37     getline(cin ,s);
38     vector<string> ans = split(s, " ");
39     // 记录是否出现异常
40     bool flag = false;
41     vector<int> res;
42     for (int i = 0; i < ans.size(); i++) {
43         string tmp = ans[i];
44         if (tmp == "C") {
45             if (res.size() == 0) {
```

```
46             flag = true;
47             break;
48         }
49         res.pop_back();
50     } else if (tmp == "D") {
51         if (res.size() == 0) {
52             flag = true;
53             break;
54         }
55         res.push_back(res[res.size() - 1] * 2);
56     } else if (tmp == "+") {
57         if (res.size() < 2) {
58             flag = true;
59             break;
60         }
61         res.push_back(res[res.size() - 1] + res[res.size() - 2]);
62 // 数字
63     } else {
64         if (!isInteger(tmp)){
65             flag = true;
66             break;
67         }
68         res.push_back(stoi(tmp));
69     }
70 }
71 if (flag) {
72     cout << -1;
73     return 0;
74 }
75 int sum = 0;
76 for (int i = 0; i < res.size(); i++) {
77     sum += res[i];
78 }
79 cout << sum;
80 return 0;
81 }
```

JAVA

```
1 import java.util.*;
2
3 public class Main {
4     // 通用 split 方法
5     private static List<String> split(String str, String delimiter) {
6         return Arrays.asList(str.split(delimiter));
7     }
8
9     // 判断是否为整数
10    private static boolean isInteger(String str) {
11        return str.matches("-?\\d+"); // 正则匹配整数
12    }
13
14    public static void main(String[] args) {
15        Scanner scanner = new Scanner(System.in);
16        String s = scanner.nextLine();
17        scanner.close();
18
19        List<String> ops = split(s, " ");
20        List<Integer> res = new ArrayList<>();
21        boolean flag = false;
22
23        for (String op : ops) {
24            if (op.equals("C")) {
25                if (res.isEmpty()) {
26                    flag = true;
27                    break;
28                }
29                res.remove(res.size() - 1);
30            } else if (op.equals("D")) {
31                if (res.isEmpty()) {
32                    flag = true;
33                    break;
34                }
35                res.add(res.get(res.size() - 1) * 2);
36            } else if (op.equals("+")) {
37                if (res.size() < 2) {
38                    flag = true;
39                    break;
40                }
41                res.add(res.get(res.size() - 1) + res.get(res.size() -
42                    2));
42            } else {
43                if (!isInteger(op)) {
44                    flag = true;
```

```
45         break;
46     }
47     res.add(Integer.parseInt(op));
48   }
49 }
50
51 if (flag) {
52   System.out.println(-1);
53 } else {
54   System.out.println(res.stream().mapToInt(Integer::intValue).su
55 m());
56 }
57 }
```

Python

```
1 import sys
2 import re
3
4 def is_integer(s):
5     return bool(re.fullmatch(r"-?\d+", s)) # 正则匹配整数
6
7 def main():
8     s = sys.stdin.readline().strip()
9     ops = s.split(" ") # 直接用 split 分割字符串
10    res = []
11
12    for op in ops:
13        if op == "C":
14            if not res:
15                print(-1)
16                return
17            res.pop()
18        elif op == "D":
19            if not res:
20                print(-1)
21                return
22            res.append(res[-1] * 2)
23        elif op == "+":
24            if len(res) < 2:
25                print(-1)
26                return
27            res.append(res[-1] + res[-2])
28        else:
29            if not is_integer(op):
30                print(-1)
31                return
32            res.append(int(op))
33
34    print(sum(res))
35
36 if __name__ == "__main__":
37     main()
```

JavaScript

```
1 const readline = require("readline");
2
3 const rl = readline.createInterface({
4     input: process.stdin,
5     output: process.stdout
6 });
7
8 // 判断是否为整数
9 function isInteger(str) {
10     return /^-?\d+$/ .test(str);
11 }
12
13 rl.on("line", function (line) {
14     let ops = line.trim().split(" ");
15     let res = [];
16
17     for (let op of ops) {
18         if (op === "C") {
19             if (res.length === 0) {
20                 console.log(-1);
21                 rl.close();
22                 return;
23             }
24             res.pop();
25         } else if (op === "D") {
26             if (res.length === 0) {
27                 console.log(-1);
28                 rl.close();
29                 return;
30             }
31             res.push(res[res.length - 1] * 2);
32         } else if (op === "+") {
33             if (res.length < 2) {
34                 console.log(-1);
35                 rl.close();
36                 return;
37             }
38             res.push(res[res.length - 1] + res[res.length - 2]);
39         } else {
40             if (!isInteger(op)) {
41                 console.log(-1);
42                 rl.close();
43                 return;
44             }
45             res.push(parseInt(op, 10));
46         }
47     }
48     console.log(res);
49     rl.close();
50 }
51
52 rl.on("close", () => process.exit(0));
```

```
46      }
47  }
48
49  console.log(res.reduce((sum, num) => sum + num, 0));
50  rl.close();
51});
```

Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "regexp"
8     "strconv"
9     "strings"
10    )
11
12 // 判断是否为整数
13 func isInteger(s string) bool {
14     matched, _ := regexp.MatchString(`^-?\d+$`, s)
15     return matched
16 }
17
18 func main() {
19     scanner := bufio.NewScanner(os.Stdin)
20     scanner.Scan()
21     s := scanner.Text()
22
23     ops := strings.Fields(s) // 直接按空格分割
24     res := []int{}
25     flag := false
26
27     for _, op := range ops {
28         if op == "C" {
29             if len(res) == 0 {
30                 flag = true
31                 break
32             }
33             res = res[:len(res)-1] // 删除最后一个元素
34         } else if op == "D" {
35             if len(res) == 0 {
36                 flag = true
37                 break
38             }
39             res = append(res, res[len(res)-1]*2)
40         } else if op == "+" {
41             if len(res) < 2 {
42                 flag = true
43                 break
44             }
45             res = append(res, res[len(res)-1]+res[len(res)-2])
46         }
47     }
48 }
```

```
46     } else {
47         if !isInteger(op) {
48             flag = true
49             break
50         }
51         num, _ := strconv.Atoi(op)
52         res = append(res, num)
53     }
54 }
55
56 if flag {
57     fmt.Println(-1)
58 } else {
59     sum := 0
60     for _, v := range res {
61         sum += v
62     }
63     fmt.Println(sum)
64 }
65 }
```

| 来自: 华为OD 机试 2025 B卷 – 投篮大赛 (C++ & Python & JAVA & JS & GO)-CSDN博客

| 来自: 华为OD 机试 2025 B卷 – 投篮大赛 (C++ & Python & JAVA & JS & GO)-CSDN博客