【华为OD机考 统一考试机试C卷】连续出牌数量（C++ Java JavaScript Python）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。根据考友反馈：目前抽到的试卷为B卷或C卷/D卷，其中C卷居多 ，按照之前的经验C卷D卷部分考题会复用A卷/B卷题，博主正积极从考过的同学收集C卷和D卷真题，可以查看下面的真题目录。

真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境 华为OD机考B卷C卷华为OD机考华为OD机考B卷华为OD机试B卷华为OD机试C卷华为OD机考C卷华为OD机考D卷题目华为OD机考C卷/D卷答案华为OD机考C卷/D卷解析华为OD机考C卷和D卷真题华为OD机考C卷和D卷题解

题目描述

有这么一款单人卡牌游戏，牌面由颜色和数字组成，颜色为红、黄、蓝、绿中的一种，数字为0-9中的一个。游戏开始时玩家从手牌中选取一张卡牌打出，接下来如果玩家手中有和他上一次打出的手牌颜色或者数字相同的手牌，他可以继续将该手牌打出，直至手牌打光或者没有符合条件可以继续打出的手牌。

现给定一副手牌，请找到最优的出牌策略，使打出的手牌最多。

输入描述

输入为两行，第一行是每张手牌的数字，数字由空格分隔，第二张为对应的每张手牌的颜色，用r y b g这4个字母分别代表4种颜色，字母也由空格分隔。手牌数量不超过10。

输出描述

输出一个数字，即最多能打出的手牌的数量。

用例

输入	1 4 3 4 5 r y b b r
输出	3

说明	如果打（1, r）-> (5, r)，那么能打两张。 如果打（4，y) -> (4, b) -> (3, b)，那么能打三张。

C++

```cpp
#include <iostream>
#include <vector>
#include <sstream>
#include <algorithm>
using namespace std;

// dfs函数：cards表示手牌是否可用，last_num表示上一张打出的牌的数字，last_color表示上一张打出的牌的颜色
int dfs(vector<string>& numbers, vector<string>& colors, string last_num, string last_color, vector<int>& cards) {
    int maxdepth = 0;
    for (int i = 0; i < cards.size(); i++) {
        if (cards[i] != 0) { // 如果这张牌还没被打出去
            if (numbers[i] == last_num || colors[i] == last_color) { // 如果这张牌的数字或颜色与上一张打出的牌相同
                cards[i] = 0; // 打出这张牌
                maxdepth = max(dfs(numbers, colors, numbers[i], colors[i], cards), maxdepth); // 继续搜索
                cards[i] = 1; // 恢复手牌
            }
        }
    }
    return maxdepth + 1; // 返回当前搜索深度+1
}

int main() {
    string input_str;
    getline(cin, input_str);
    vector<string> numbers;
    stringstream ss(input_str);
    string temp_str;
    while (ss >> temp_str) {
        numbers.push_back(temp_str);
    }
    getline(cin, input_str);
    vector<string> colors;
    ss.clear();
```

```
    ss.str(input_str);
    while (ss >> temp_str) {
        colors.push_back(temp_str);
    }
    vector<int> cards(numbers.size(), 1); // 初始化手牌
    int maxiter = 0;
    for (int i = 0; i < numbers.size(); i++) { // 枚举每一张牌
        cards[i] = 0; // 打出这张牌
        maxiter = max(dfs(numbers, colors, numbers[i], colors[i], cards), maxiter); // 进行搜索
        cards[i] = 1; // 恢复手牌
    }
    cout << maxiter << endl; // 输出最多能打出的牌数
    return 0;
}
```

java

```
import java.util.*;
import java.io.*;

public class Main {
    // dfs函数：cards表示手牌是否可用，last_num表示上一张打出的牌的数字，last_color表示上一张打出的牌的颜色
    public static int dfs(List<String> numbers, List<String> colors, String last_num, String last_color, List<Integer> cards) {
        int maxdepth = 0;
        for (int i = 0; i < cards.size(); i++) {
            if (cards.get(i) != 0) { // 如果这张牌还没被打出去
                if (numbers.get(i).equals(last_num) || colors.get(i).equals(last_color)) { // 如果这张牌的数字或颜色与上一张打出的牌相同
                    cards.set(i, 0); // 打出这张牌
                    maxdepth = Math.max(dfs(numbers, colors, numbers.get(i), colors.get(i), cards), maxdepth); // 继续搜索
                    cards.set(i, 1); // 恢复手牌
                }
            }
        }
        return maxdepth + 1; // 返回当前搜索深度+1
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input_str = sc.nextLine();
        List<String> numbers = new ArrayList<>();
```

```
        Scanner ss = new Scanner(input_str);
        while (ss.hasNext()) {
            String temp_str = ss.next();
            numbers.add(temp_str);
        }
        input_str = sc.nextLine();
        List<String> colors = new ArrayList<>();
        ss = new Scanner(input_str);
        while (ss.hasNext()) {
            String temp_str = ss.next();
            colors.add(temp_str);
        }
        List<Integer> cards = new ArrayList<>(Collections.nCopies(numbers.size(), 1)); // 初始化手牌
        int maxiter = 0;
        for (int i = 0; i < numbers.size(); i++) { // 枚举每一张牌
            cards.set(i, 0); // 打出这张牌
            maxiter = Math.max(dfs(numbers, colors, numbers.get(i), colors.get(i), cards), maxiter); // 进行搜索
            cards.set(i, 1); // 恢复手牌
        }
        System.out.println(maxiter); // 输出最多能打出的牌数
    }
}
```

javaScript

```
const readline = require('readline');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

let numbers = [];
let colors = [];

function dfs(numbers, colors, last_num, last_color, cards) {
  let maxdepth = 0;
  for (let i = 0; i < cards.length; i++) {
    if (cards[i] !== 0) {
      if (numbers[i] === last_num || colors[i] === last_color) {
        cards[i] = 0;
```

```
        maxdepth = Math.max(dfs(numbers, colors, numbers[i], colors[i], cards), maxdepth);
        cards[i] = 1;
      }
    }
  }
  return maxdepth + 1;
}

rl.on('line', (input) => {
  if (!numbers.length) {
    numbers = input.split(' ');
  } else if (!colors.length) {
    colors = input.split(' ');
    const cards = new Array(numbers.length).fill(1);
    let maxiter = 0;
    for (let i = 0; i < numbers.length; i++) {
      cards[i] = 0;
      maxiter = Math.max(dfs(numbers, colors, numbers[i], colors[i], cards), maxiter);
      cards[i] = 1;
    }
    console.log(maxiter);
    rl.close();
  }
});
```

python

```
import sys

# dfs函数：cards表示手牌是否可用，last_num表示上一张打出的牌的数字，last_color表示上一张打出的牌的颜色
def dfs(numbers, colors, last_num, last_color, cards):
    maxdepth = 0
    for i in range(len(cards)):
        if cards[i] != 0: # 如果这张牌还没被打出去
            if numbers[i] == last_num or colors[i] == last_color: # 如果这张牌的数字或颜色与上一张打出的牌相同
                cards[i] = 0 # 打出这张牌
                maxdepth = max(dfs(numbers, colors, numbers[i], colors[i], cards), maxdepth) # 继续搜索
                cards[i] = 1 # 恢复手牌
    return maxdepth + 1 # 返回当前搜索深度+1

```

```python
if __name__ == '__main__':
    input_str = sys.stdin.readline().strip()
    numbers = input_str.split()
    input_str = sys.stdin.readline().strip()
    colors = input_str.split()
    cards = [1] * len(numbers) # 初始化手牌
    maxiter = 0
    for i in range(len(numbers)): # 枚举每一张牌
        cards[i] = 0 # 打出这张牌
        maxiter = max(dfs(numbers, colors, numbers[i], colors[i], cards), maxiter) # 进行搜索
        cards[i] = 1 # 恢复手牌
    print(maxiter) # 输出最多能打出的牌数
```

文章目录

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷
题目描述
输入描述
输出描述
用例
C++
java
javaScript
python


机考真题
华为OD
No No No
CSDN @算法大师