拉勾教育
—互联网人实战大学—

贪心算法：行相等的最少多米诺旋转

中等/贪心算法

学习目标

拉勾教育
—互联网人实战大学—

- 理解贪心算法的主要思想与应用场景
- 掌握贪心算法的流程
- 熟练应用贪心算法解决问题

题目描述

在一排多米诺骨牌中，A[i] 和 B[i] 分别代表第 i 个多米诺骨牌的上半部分和下半部分。（一个多米诺是两个从 1 到 6 的数字同列平铺形成的 —— 该平铺的每一半上都有一个数字。）

我们可以旋转第 i 张多米诺，使得 A[i] 和 B[i] 的值交换。

返回能使 A 中所有值或者 B 中所有值都相同的最小旋转次数。　如果无法做到，返回 -1.

输入：A = [2,1,2,4,2,2], B = [5,2,6,2,3,2]
输出：2

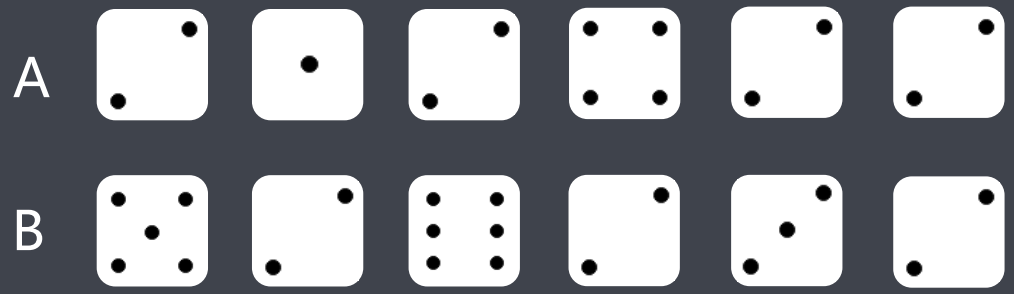
输入：A = [3,5,1,2,3], B = [3,6,3,3,4]
输出：-1

一. Comprehend 理解题意

拉勾教育
—互联网人实战大学—

行相等的最小多米诺旋转

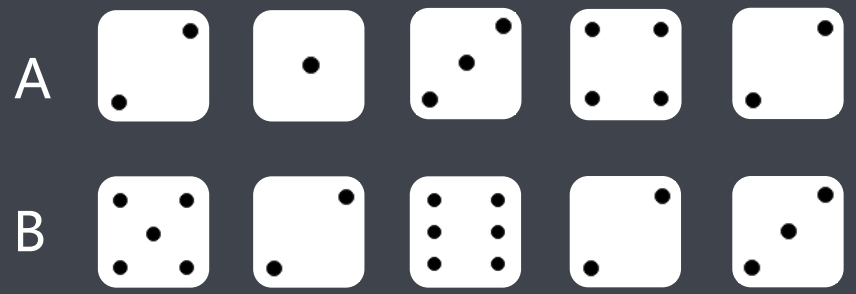
- 多米诺骨牌
  - 每张牌由2个1-6的数字A[i]， B[i]组成
- 多米诺旋转
  - 交换每张牌的A[i]与B[i]
- 行相等的最小多米诺旋转
  - 使得A或B中元素全部相同的最小旋转次数



一. Comprehend 理解题意

细节问题

- 可能不存在多米诺旋转使 A 中所有值或者 B 中所有值都相同



二. Choose 数据结构及算法思维选择

数据结构选择

- 输入的数据类型为两个整形数组表示每张多米诺骨牌的两个数字
- 输出的数据类型为一个整数
- 因为需要处理的是两个由1到6数字组成的整数集合，我们使用数组作为数据结构

二. Choose 数据结构及算法思维选择

算法思维选择

- 题目要求寻找使得行相等的最小多米诺旋转
- 首先判断是否存在这样的旋转，再求解最小旋转次数
- 统计1——6这六个数字的出现次数

  在A中出现的次数

  在B中出现的次数

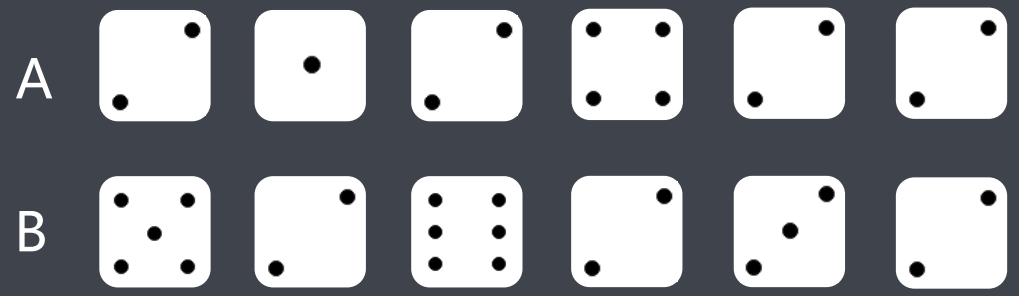
  在骨牌出现的次数（在多少个骨牌中出现过，骨牌上下都出现需记作一次）

三. Code 基本解法及编码实现

解题思路剖析

- 统计1到6这6个数字出现在多少张多米诺骨牌中以及分别在A、B出现的次数
- 只有数字2在所有牌中出现
- 且在A中出现的次数4 > 在B中出现的次数3

	1	2	3	4	5	6
A | B	1	6	1	1	1	1
A	1	4	0	1	0	0
B	0	3	1	0	1	1



三. Code 基本解法及编码实现

复杂度分析

- 时间复杂度
  - 统计1-6这6个数字出现的次数，只需要对多米诺骨牌浏览一遍，时间复杂度为O(n)
  - 寻找在所有多米诺骨牌出现的数字，以及最小的多米诺旋转，只需要对这个6×3的数组进行操作，时间复杂度为O(1)
  - 总时间复杂度为O(n)
- 空间复杂度
  - 空间消耗方面我们只需要使用常数级的变量空间，因此空间复杂度为O(1)

三. Code 基本解法及编码实现

Java编码实现

```java
public int minDominoRotations1(int[] A, int[] B) {
    int n = A.length;
    //记录1-6数字分别在A，B中出现的次数、在多米诺骨牌出现的次数
    int[] countA = new int[6];
    int[] countB = new int[6];
    int[] countAB = new int[6];
    //遍历骨牌，统计出现次数
    for (int i = 0; i < n; i++) {
        countA[A[i] - 1]++;
        countB[B[i] - 1]++;
        countAB[A[i] - 1]++;
        if (A[i] != B[i])
            countAB[B[i] - 1]++;
    }
    //在骨牌统计表找到出现次数等于骨牌总数的数字
    for (int i = 0; i < 6; i++) {
        if (countAB[i] == n) { //计算最小的旋转次数
            return (n - Math.max(countA[i], countB[i]));
        }
    }
    //找不到返回-1
    return -1;
}
```

执行结果： 通过 显示详情 >

执行用时： 7 ms , 在所有 Java 提交中击败了 33.33% 的用户

内存消耗： 46.1 MB , 在所有 Java 提交中击败了 85.16% 的用户

四. Consider 思考更优解

是否存在无效代码或者无效空间消耗

- 遍历所有骨牌是必须的吗？

是否有更好的算法思维

- 每张多米诺骨牌上只有两个数字，如果存在行相等则旋转的最终结果
  - A中的数字都是 A[0] 或者 A中的数字都是 B[0]
  - B中的数字都是 A[0] 或者 B中的数字都是 B[0]
- 故只需考察A[0]、B[0]在其他骨牌上的出现情况
- 若某骨牌中不存在A[0]、B[0]，则提前终止，无需浏览后面的骨牌

四. Consider 思考更优解

拉勾教育
—互联网人实战大学—

关键知识点：贪心算法（greedy algorithm）

- 求解最优子结构的问题，最优子结构就是局部最优解能决定全局最优解

[找零钱问题]

- 贪心法可以解决 图中的最小生成树、求哈夫曼编码 等问题

假如老板要找给我99元纸币，怎么找纸币数最少？

- 但有些问题贪心算法并不能获得最优解，如背包问题

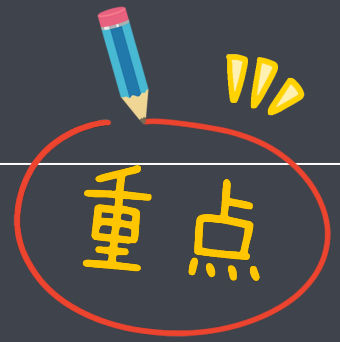
- 贪心法也可以用作辅

通过每一步的局部最优解来获得全局最优解 重点

四. Consider 思考更优解

关键知识点：贪心算法（greedy algorithm）

- 贪心算法的求解步骤

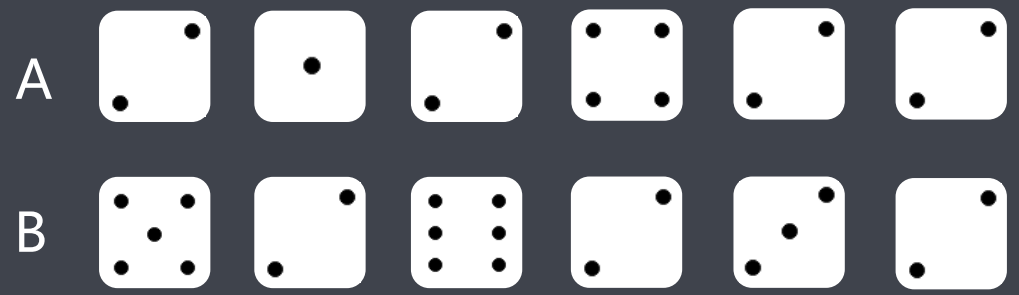


1. 创建数学模型来描述问题
2. 把求解的问题分成若干个子问题
3. 对每一子问题求解，得到子问题的局部最优解
4. 把子问题的解合并成原来问题的一个解

五. Code 最优解思路及编码实现

1.创建数学模型来描述问题

- 给定两个由1-6数字组成的数组A和B
- 通过多米诺旋转（交换A[i]和B[i] ），使得A或B中元素全部相同
- 要求进行尽可能少的交换来完成任务

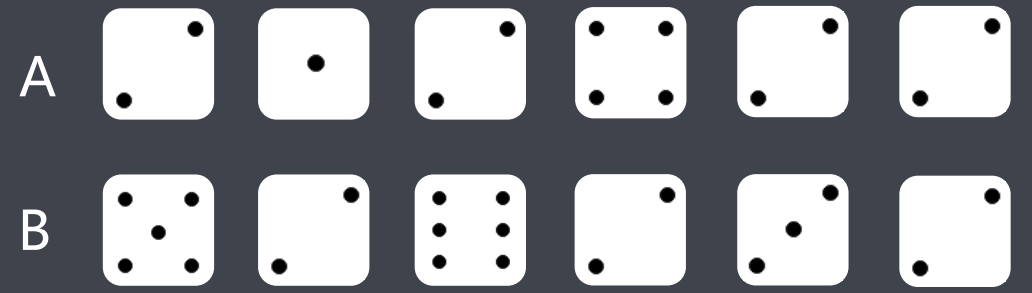


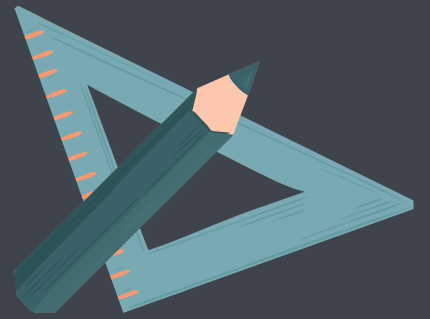
五. Code 最优解思路及编码实现

2.划分子问题

最小多米诺旋转如果存在一定是

- 将A或B中的元素全部变成A[0] 或
- 将A或B中的元素全部变成B[0]

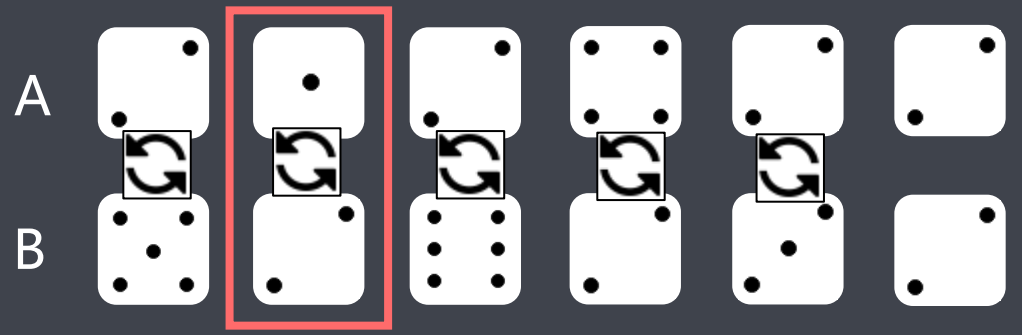




五. Code 最优解思路及编码实现

3.求解子问题

- 尝试将A中的元素全部变成A[0]
  - 需要交换(A[1], B[1])，(A[3], B[3])
  - 执行2次多米诺旋转操作



- 再尝试将B中的元素全部变成A[0]
  - 需要交换(A[0], B[0])，(A[2], B[2])，(A[4], B[4])
  - 执行3次多米诺旋转操作
- 尝试将A中的元素或B中的元素全部变成B[0]
  - 很快发现第二张牌的两个数字没有5，那么不可能将A中或B中元素全部变成B[0]

五. Code 最优解思路及编码实现

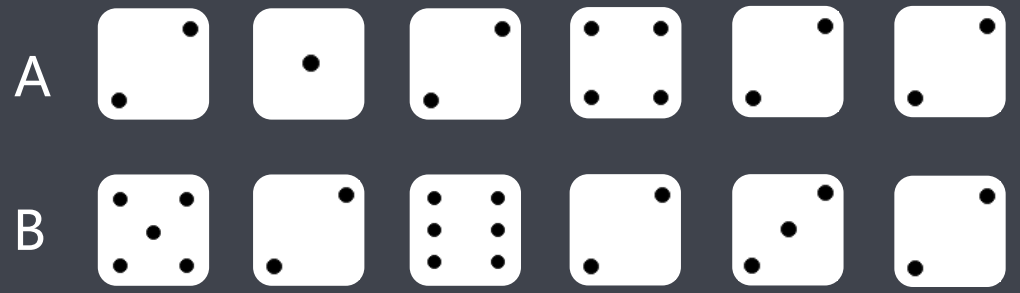
4.合并子问题的解

- 尝试将A中的元素全部变成A[0]

  执行2次多米诺旋转操作

- 再尝试将B中的元素全部变成A[0]

  执行3次多米诺旋转操作



- 尝试将A中的元素或B中的元素全部变成B[0]

  不可能将A中或B中元素全部变成B[0]

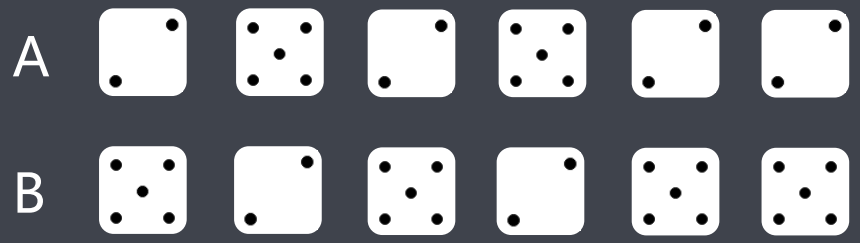
综上最小多米诺旋转次数为2

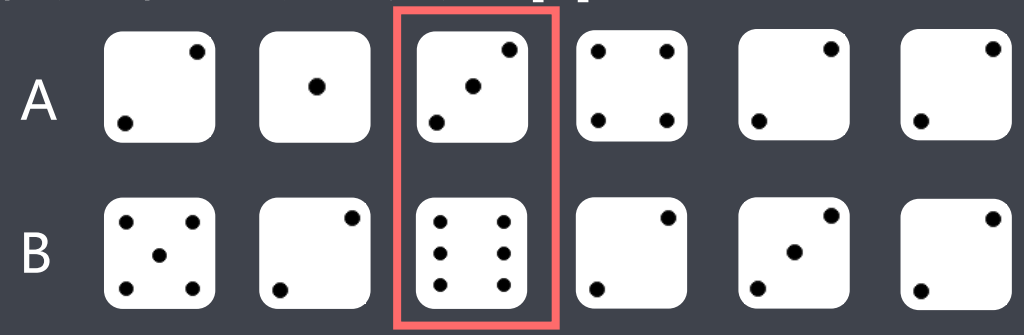
五. Code 最优解思路及编码实现

解题思路剖析

判断能否将A中或B中全部元素变成A[0]

- 若能，返回旋转次数少的那个，不用再检查B[0]
  - 如果A[0] == B[0]，再检查B[0]没有意义
  - 如果A[0] != B[0]，仅当骨牌中只有A[0]、B[0]两个数字才能将A中或B中全部元素变成B[0]，此时最小旋转次数将和A[0]一样
- 若不能，并且A[0] != B[0]，再判断能否将A中或B中全部元素变成B[0]





五. Code 最优解思路及编码实现

复杂度分析

- 时间复杂度：

  检查是否可以将A或B中所有元素变成A[0]或B[0]，最坏情况可能需要浏览完所有多米诺骨牌，时间复杂度为O(n)

- 空间复杂度：

  空间消耗方面我们只需要使用常数级的变量空间，空间复杂度为O(1)

五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

Java编码实现

```java
public int minDominoRotations(int[] A, int[] B) {
    int n = A.length;
    int rotations = check(A[0], A, B, n);//元素全部变成A[0]，最少需要多少次旋转
    //如果A[0]==B[0]，那么不用继续检查B[0]
    //如果A[0]!=B[0] 且可以将A或B中的元素全部变成A[0]，那么也不用再检查B[0]
    if (rotations != -1 || A[0] == B[0])
        return rotations;
    else
        return check(B[0], A, B, n); //全部变成B[0]，最少需要多少次旋转
}

//检查将A或者B中元素全部变成x需要多少次旋转
public int check(int x, int[] A, int[] B, int n) {
    //rotationsA存将A中元素变成x需要多少次旋转，rotationsB存将B中元素变成x需要多少次旋转
    int rotationsA = 0, rotationsB = 0;
    //遍历骨牌判断是否能完成任务（在A中完成或者在B中完成）
    for (int i = 0; i < n; i++) {
        // 如果当前多米诺骨牌上没有数字x，那么不可能完成任务
        if (A[i] != x && B[i] != x) return -1;
            // 如果当前多米诺骨牌上A[i]不是x，那么rotationsA需要+1
        else if (A[i] != x) rotationsA++;
            // 如果当前多米诺骨牌上B[i]不是x，那么rotationsB需要+1
        else if (B[i] != x) rotationsB++;
    }
    // 返回最小旋转次数
    return Math.min(rotationsA, rotationsB);
}
```

执行结果： 通过 显示详情 >

执行用时： 4 ms ，在所有 Java 提交中击败了 99.06% 的用户

内存消耗： 46.2 MB ，在所有 Java 提交中击败了 77.42% 的用户

六. Change 变形延伸

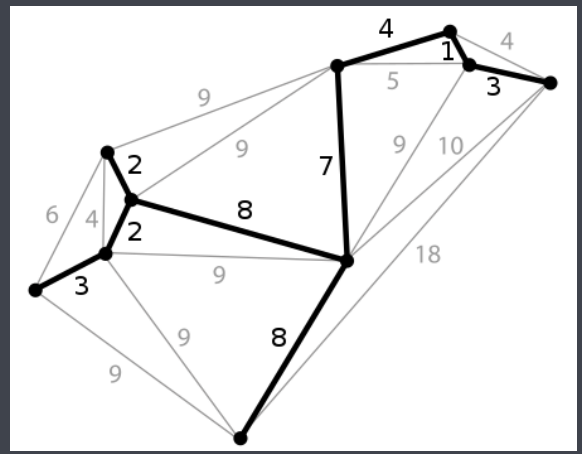
拉勾教育
—互联网人实战大学—

题目变形

每一张骨牌上有3个数字，每次旋转可互换3个数字中相邻的两个，求使得A、B、C中某一个数组数字全部一样的最小多米诺旋转次数？

延伸扩展

- 贪心算法是一种基础的算法思维
  - 适用条件：最优子结构可获得全局最优解
- 贪心算法在实际工作中的应用
  - 图中的最小生成树算法
  - 单源最短路径迪杰斯特拉算法



总结

- 理解贪心算法的主要思想与应用场景
- 掌握贪心算法的流程
- 熟练应用贪心算法解决问题

课后练习

拉勾教育
—互联网人实战大学—

1. 分发饼干（leetcode 455 /简单）
2. 买卖股票的最佳时机 II（leetcode 122 /简单）
3. 用最少数量的箭引爆气球（leetcode 452/中等）
4. 移掉K位数字 (leetcode 402 /中等)

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容