

# 分块思想与数据结构

浙江省余姚中学 李昊

# 分块思想有何应用？

- 一. 直接分块
- 二. 平衡两种针对性不同的算法
- 三. 定期重构
- 四. 莫队算法

# 直接分块

- 信息学竞赛中经常会出现类似于：
- 操作  $L\ R$ : 对第 $L$ 至第 $R$ 个数进行...
- 询问  $L\ R$ : 询问第 $L$ 至第 $R$ 个数...
- 往往它是需要复杂数据结构来维护，或者难以实现两个区间信息的合并，以致数据结构做不了的

# 直接分块

- 让我们来看看分块是怎么搞的
- 将连续 $x$ 个元素分为一个块
- 对于每一个操作 $[L, R]$ ，一定是中间包含了若干完整块，两边剩余不超过 $2x$ 个位置，即“ $[L, K \cdot X], [K \cdot X + 1, (K + 1) \cdot X] \cdots [K_2 \cdot Z + 1, R]$ ”这种形式
- 对于完整块可以统一处理，如果可以做到 $O(1)$ ，时间复杂度 $O(n/x)$
- 对于剩余的 $2x$ 个点，可以暴力处理 $O(x)$
- 当 $x = \sqrt{n}$ 时，每次操作的时间复杂度降到最低 $O(\sqrt{n})$
- 若上面两种情况的任意一种需要多一个 $\log$ 的复杂度，那么可以通过调整块的大小来使得每次操作的复杂度变为 $O(\sqrt{n \log n})$

# 看一道简单的例题

- $n$ 个数,  $q$ 个操作
- 操作1: 将一段区间内每个数增加 $x$
- 操作2: 询问一段区间内数的总和
- 数据范围:  $n, q \leq 100000$

# 直接分块的做法

- 按照前面所说的，将序列每 $x$ 个分一块
- 修改操作：
  - 对于完整的块，每个块有一个标记 $t$ ，表示该块内每个元素的值都需要加上 $t$ ，相当于线段树上的`lazy`标记
  - 对于多余的 $2x$ 个位置，直接暴力修改值
- 询问操作：
  - 对于完整的块，返回块内元素总和 $+t \times \text{块大小}$
  - 对于多余的 $2x$ 个位置，直接暴力累加
- 由于两种情况都可以 $\sqrt{n}$ 完成，所以 $x$ 设为 $\sqrt{n}$ ，总时间复杂度 $O(q\sqrt{n})$

# 平衡两种针对性不同的算法

- 信息学竞赛有这样一类题目，当题目中的参数比较小时，可以通过某种姿势暴力，当参数比较大时，又有另一种暴力，这时我们可以通过设置一个阈值，然后分两种情况做
- 这一类题目比较灵活多变，我们直接来看一道例题

# TC SRM589 Flipping Bits

- 给一个长度为 $n$ 的01字符串，再给一个正整数 $M$ 。
- 每次操作可以将一个位置取反，或者将一个长度为 $M$ 的倍数的前缀取反。问最少多少次操作，才能使字符串成为一个循环节为 $M$ 的循环串。
- 数据范围： $n \leq 300$



# Solution

- 每 $m$ 个划分为一段
- 有两种操作，单点翻转和整段翻转，如何平衡这两种操作？

# Solution

- 当 $m$ 比较小时可以直接枚举出答案，那么当单点修改完后，每一段为最后答案或者每一位都与最后答案相反，剩下的只是整段翻转的事了
- 但并不知道每段变成哪种情况会使得答案最优
- 所以从后往前对每一段dp一下，记录目前翻转次数的奇偶性
- 那么对于当前段只需数出与答案不相同的个数即可
- 时间复杂度 $O(2^m * n / m)$

# Solution

- 当 $m$ 比较大的时候，发现段数并不怎么多，枚举每一段是否需要翻转
- 对于 $i, i+m, i+m*2, \dots$  这些位置，发现它们可以划为一类，因为他们对其他位置没有影响，其他位置对他们也没有影响，只需要保证每一类里最后都是一样的就行，所以对于每一类数一下0多还是1多，把少的修改掉即可
- 时间复杂度 $O(n * 2^{\sqrt{n/m}})$

# 定期重构

- 对于有修改有询问的题，有一种叫“定期重构”的做法
- 每修改 $\sqrt{m}$ 次，就把需要的信息统一处理一次，那么每次询问时，最多就 $\sqrt{m}$ 次修改对本次询问的影响还没处理，暴力处理即可
- 还是直接看题感受一下吧

# 定期重构

- $n$ 个数,  $q$ 个操作
- 操作1: 将一段区间内每个数增加 $x$
- 操作2: 询问一段区间内数的总和
- 数据范围:  $n, q \leq 100000$

# 定期重构

- 每 $\sqrt{q}$ 次修改操作，我们就把a数组重新算一遍
- 每次询问考虑剩余的 $\sqrt{q}$ 个修改操作对当前询问的影响
- 总时间复杂度 $O(n\sqrt{q})$

# 莫队算法

- 对于无修改，且如果知道区间  $[l, r]$  的答案就可以通过  $O(1)$  或者  $O(\log n)$  的时间算出  $[l, r+1]$ 、 $[l, r-1]$ 、 $[l+1, r]$ 、 $[l-1, r]$  的答案的题目，可以套用这一通用解法
- 先将序列分成  $\sqrt{n}$  分块，然后将所有询问做双关键字排序，第一关键字为询问的左端点所在的块，第二关键字为询问的右端点
- 那么两个询问  $[l_1, r_1]$ 、 $[l_2, r_2]$  之间转移的时间为  $(|l_1 - l_2| + |r_1 - r_2|) * O(\text{移动一步的复杂度})$ ，考虑总的端点挪动次数

# 莫队算法

- 先将所有询问按照第一关键字分成 $\sqrt{n}$ 类
- 对于左端点，在同一类内的转移，一次不超过 $\sqrt{n}$ ，在不同类之间转移不超过 $\sqrt{n}$ 次
- 对于右端点，在同一类内，右端点是单调不降的，所以同一类内最多转移 $n$ 次，在不同类之间转移不超过 $\sqrt{n}$ 次
- 那么总转移次数就是 $n\sqrt{n}$ 级别的了



# 小z的袜子

- 有 $n$ 个数， $m$ 个询问，每次询问从第 $l$ 个到第 $r$ 个数中拿出两个数，这两个数相等的概率
- 数据范围：  $n, m \leq 50000$

# 莫队算法

- 其实就是要统计区间内相等的对数
- 按照前面所说的将询问先排个序，考虑如何转移
- 开一个数组 $s$ ，表示目前区间内有 $s[i]$ 个 $i$
- 若新增一个数 $x$ 进区间，答案增加 $s[x]$ ， $s[x]+1$
- 删除一个数 $x$ ， $s[x]-1$ ，答案减少 $s[x]$
- 时间复杂度 $O(n\sqrt{t}(n))$

# CodeChef NOV 14 Chef and Churu

- 有一个长度为n的数组A，有n个函数，第i个函数的值为

$$\sum_{j=l[i]}^{r[i]} a[j]$$

- 有两种操作：
  - 修改A[i]
  - 询问第l~r个函数值的和。
- 
- 数据范围：n≤100000

# Solution

- 考虑询问操作的  $l=r$  该怎么做
- 直接对序列分块，维护块的总和即可，询问像最前面那道例题一样做即可， $O(\sqrt{n})$
- 询问其实还有更快的做法
- 修改操作，维护
  - 前  $i$  个块的总和
  - 每个数到它所在块左端点所有数的总和
- 询问操作
  - 可以  $O(1)$  查询前  $i$  个数的总和
  - 前  $r$  个数 - 前  $l-1$  个数就是答案了

# Solution

- 既然要查询的函数，那么我们把函数也分块
- 维护每个函数块的答案
- 每次询问时统计完整的被包含的块的和
- 统计剩余未被考虑的 $2 * \sqrt{n}$ 个函数的值
- 对于每个函数可以通过前 $i$ 块和、块内前缀和 $O(1)$ 查询其值。
- 对于每次修改，需要统计出对每个函数块的影响
- 这个可以通过预处理每个函数块中包含点 $i$ 个函数有几个函数轻松解决
- 时间复杂度 $O(n\sqrt{n})$

# 51nod 双马尾机器人

- 从 $1 \sim n$ 中选出尽可能少的数，使得剩下的每个数至少和一个被选择的数不互质， $k$ 不能被选择
- 数据范围：  $n, k \leq 1000$

# Solution

- 题目相当于被选择的数分解质因数后包含所有质因数
- 每个数分解质因数之后最多只含有一个大于 $\sqrt{n}$ 的质因子
- 以这个质因子为依据将所有数进行分类
- 同时记录质因子2、3、5、7、11、13、17、19、23、29、31有没有出现过
- $dp[i][j]$ 表示前*i*类数*j*中的质因子已经出现过最少需要选择几个数 (*j*为11个质因子有没有出现过的二进制状态)，那么只需保证每一类中至少选择了一个数方案就是合法的

# CodeChef OCT 14 Children Trips

- 有一棵边权为1或2的 $n$ 个点的树，每次询问从 $u$ 走到 $v$ 每天最多走 $k$ 步，且每天结束必须在节点上，最少要几天。
- 数据范围： $n$ 、询问次数 $\leq 100000$



# Solution

- 对 $k$ 分情况进行讨论
- $k \geq \sqrt{n}$
- 最多走 $\sqrt{n}$ 步，每一步二分能走到哪即可
- $k \leq \sqrt{n}$
- 对于每个 $k$ 一起处理，倍增，预处理 $2^i$ 步能走到哪里
- 时间复杂度 $O(n\sqrt{n} \log n)$

# Solution

- 怎么把 $\log$ 去掉?
- (1)  $k \leq \sqrt{n}$ 
  - 还是 $k$ 相同的询问一起处理
  - 找出每个点向上走长度 $k$ 走到哪
  - 根据这个新构造一棵树
  - dfs整棵新树, 记录根到目前节点的路径
  - 访问到某个点时处理这个点的所有询问
  - 直接在记录的路径上二分即可
  - 时间复杂度 $O(n\sqrt{n} + Q \log n)$

# Solution

- (2)  $k > \sqrt{n}$ 
  - 每个点  $u$  记录向上走长度  $i$  能走到哪 ( $i \leq \sqrt{n}$ ), 记为  $f[u][i]$
  - 每次走的时候如果剩余长度  $> \sqrt{n}$  就用  $f[u][\sqrt{n}]$  更新, 剩余长度减去实际走了多远
  - 否则走到  $f[u][\text{剩余长度}]$
  - 因为每步要么走  $\sqrt{n}$  远, 要么走完一次, 所以时间复杂度  $O(\sqrt{n})$

# CodeChef MAY 15 Chef and Balanced Strings

- 一个字符串如果他的字符都出现了偶数次则称为平衡串。询问一个字符串  $l \sim r$  内所有平衡子串长度的  $k$  次方和。
- 数据范围: 字符串长度  $n$ , 询问次数  $m \leq 100000$ ,  $1 \leq k \leq 2$

# Solution

- 如何快速判断一个子串是否是平衡串？
- 把前缀和根据每个字母出现次数的奇偶进行hash,  $s[r]=s[l-1]$ 即为平衡串
- 两个相邻的平衡串合并之后仍是平衡串。
- 对于每个 $i$ , 找到最小的 $a[i]$  ( $a[i]>i$ ), 使得 $[i, a[i]]$ 为平衡子串。  
则 $[i, a[i]]$ 、 $[i, a[a[i]+1]]$ 、 $[i, a[a[a[i]+1]+1]]\cdots$ 都是平衡串。
- 将这些位置划为一类

# Solution

- 分块预处理第*i*个块到第*j*个块的答案。每次询问的时候只需考虑添加 $2 * \sqrt{n}$ 个字符后答案的变化。
- 由于只有每一类内的两个位置配对才能组成平衡串，所以考虑每一类内如何维护
- $\sigma \{ (w_i - w_j)^2 \} = \sigma \{ w_i^2 \} * (\text{个数} - 1) - \sigma \{ w_i \}^2 + \sigma \{ w_i^2 \}$
- 具体的维护方法可以把式子展开，然后维护 $w_i$ 个数， $w_i$ 总和， $w_i$ 平方和
- $O(n\sqrt{n})$
- 或者套用莫队算法，维护方法类似

# Codeforces Round #221 (Div. 1) Tree and Queries

- 给出一个 $n$ 个点的树，每个点都有颜色， $Q$ 次询问，每次问 $v_i$ 的子树中至少出现 $k_i$ 次的颜色有几种
- 数据范围：  $n, Q \leq 10^5$

# Solution

- 相当于在序列上询问区间中出现次数超过 $k_i$ 的数有几种
- 莫队算法模板题
- 强制在线？



# Solution

- 若  $k_i > \sqrt{n}$
- 可能的颜色最多  $\sqrt{n}$  种，对每一个颜色判断一下即可
- 如何做到  $O(1)$  回答某种颜色在某一区间内的出现次数？
- $sum1[i][j]$  表示在前  $i$  个块  $j$  出现了几次
- $sum2[i]$  表示在第  $i$  个元素所属的块中在第  $i$  个元素之前且与第  $i$  个元素颜色相同的位置有多少个
- 若  $k_i \leq \sqrt{n}$
- 对序列进行分块，记录从第  $i$  个块到第  $j$  个块出现超过  $k$  ( $k \leq \sqrt{n}$ ) 次的颜色有几种，对于多余的  $2 * \sqrt{n}$  个位置的颜色判断一下即可

## Codeforces Round #270 Design Tutorial: Increase the Constraints

- 给出两个01序列A和B，Q次询问，每次询问A的一个子串和B的一个子串异或之后的结果包含多少个1（两个子串长度相同）
- 数据范围：  $|A|, |B| \leq 2 \cdot 10^5$ ,  $Q \leq 4 \cdot 10^5$

# Solution

- 先考虑简化版本：如果询问的是A和B的子串（保证长度相同）？
- FFT
- 那么原题再把A分块就行了
- 当块的大小为 $\sqrt{n \log n}$ 时时间复杂度最优
- 时间复杂度 $O(n\sqrt{n \log n})$

# CodeChef JUNE 14 Sereja and Arcs

- 给定一个长度为 $n$ 的序列 $A$ ，对于每一对 $(i, j)$  ( $i < j$ )，如果满足 $A_i = A_j$ ，那么就在坐标系中画上一个颜色为 $A_i$ 的以 $(i, 0)$   $(j, 0)$  为直径的圆。问有多少对颜色不同的圆存在交点。
- 数据范围：  $1 \leq N, A_i \leq 100000$

# Solution

- 相交对数=总数-不相交对数。
- 不相交对数有两种类型:aabb和baab。
- 第一种比较好求，扫一遍就可以求出来。
- 现在来考虑第二种情况。

# Solution

- 设一阈值 $X$ , 然后分类讨论:
- (1) 如果相交的两种颜色中没有出现次数超过 $X$ 的颜色:
- 枚举这种颜色所有点对, 统计每一点对包含多少点对
- 树状数组维护, 此类点对最多 $nX$ 对
- 时间复杂度 $O(nX \log n)$

# Solution

- 如果相交的两种颜色中有出现次数超过 $X$ 的颜色：(假设这种颜色为 $a$ ，出现 $z$ 次)
- 对任意同一颜色的两个位置，设前者之前有 $x$ 次 $a$ 出现，后者之前有 $y$ 次 $a$ 出现，那么这个圆的贡献为  $(y-x) * (y-x-1) / 2 + x * (z-y)$
- 将此式分解然后线性扫一遍求和即可。
- 时间复杂度  $O(n^2/X)$
- 结合上一种情况，当 $X = \sqrt{n \log n}$ 时，总时间复杂度取到最优值  $O(n * \sqrt{n \log n})$

# XJOI 神奇的矩阵

- 有一个神奇的矩形。它的第一行每一个元素 $a(1, i)$ 都是给定的。对于每一个元素 $a(x, y)$  ( $x > 1$ )，它的值都是 $a(x-1, y)$ 在 $a(x-1, 1) \cdots a(x-1, y)$ 中出现过的次数。但由于这个矩阵很大，人们并不开心这么慢吞吞地计算整个矩阵的值，因此他们找到了你，并要求你快速知道某一个位置的值。有时这个矩阵的第一行还会被修改，你当然也要考虑这些修改的因素。
- 数据范围： $n, m, k \leq 10^5$



# 提示:

- 除了第一行，每两行为一循环周期

# Solution

- $x=2$ ?

- $x=3$ ?

# 解法一

- $a[2][i]$  表示  $a[1][i]$  在前  $i$  个数中出现了几次
- $a[3][i]$  表示有多少个数在前  $i$  个数中出现次数大于等于  $a[1][i]$
- $a[2]$  很好维护
- 考虑如何维护  $a[3]$

# 解法一

- $\text{sum}[i][j]$  表示第  $i$  块中  $a[2][x] = j$  的  $x$  有几个
- 只维护  $j \leq \sqrt{n}$  的
- 修改时，在每个块中其实最多只会影响2个  $\text{sum}$  值，所以直接维护即可
- 询问时，先求出  $a[2][i]$ 
  - 若  $a[2][i] > \sqrt{n}$ ，只考虑序列中出现超过  $\sqrt{n}$  次的数，每个数  $O(1)$  验证
  - 否则，先求出完整块中出现  $a[2][i]$  的出现次数，再验证序列中出现超过  $\sqrt{n}$  次的数，和  $i$  所在块中剩余的排在  $i$  前面的数
- 时间复杂度  $O(n\sqrt{n})$

## 解法二

- 对操作分块
- 每 $\sqrt{n}$ 操作就暴力重构一次，询问时只需考虑 $\sqrt{n}$ 次操作对当前询问影响即可

# Manthan, Codefest 16 Fibonacci-ish II

- 给你一个长度为 $n$ 的数组， $Q$ 次询问，每次将第 $l$ 个到第 $r$ 个数取出排序去重，求最小的数 $\times fib_1$  + 次小的数 $\times fib_2$  + 第三小的数 $\times fib_3$ .....的总和。
- 数据范围：  $n, Q \leq 30000$

# Solution

- 最外面是一个莫队算法
- 如何维护转移?
- 若加入或删除后数字种类不变则不需要管
- 否则, 用权值线段树维护
- 线段树的每个叶子上维护这个数 $\times \text{fib}[i]$ , 这个数 $\times \text{fib}[i+1]$
- 当插入一个数的时候, 给排在他后面的数统一乘上一个转移矩阵
- $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$
- 即可

## Educational Codeforces Round 6 Xors on Segments

- 给你一个长度为 $n$ 的数组 $A$ ,  $m$ 次询问, 每次询问给出 $(l, r)$
- 定义 $f(u, v) = u \text{ xor } (u+1) \text{ xor } \dots \text{ xor } v$
- 对于每一组询问输出 $f(A[x], A[y])$ 的最大值 ( $l \leq x, y \leq r, A[x] \leq A[y]$ )
- 数据范围  $1 \leq n \leq 5 \cdot 10^4, 1 \leq m \leq 5 \cdot 10^3, A[i] \leq 10^6$



# Solution

- $b[i] = 1 \text{ xor } 2 \text{ xor } 3 \cdots \text{ xor } i$ , 题目相当于求  $b[a[x]-1] \text{ xor } b[a[y]]$
- 对数组分块
- 预处理块到块的答案
- 询问时直接用完整块的答案, 多余的  $2\sqrt{n}$  个在可持久化 trie 上暴力找答案

# 2015北京网络赛 J Clarke and puzzle

- 给出 $n$ 位同学的五门成绩， $Q$ 次询问，每次给出一位同学的成绩，询问有多少位同学每一门成绩都比该同学差。
- 数据范围：  $n, Q \leq 50000$

# Solution

- 答案={第一门科目成绩比该同学差的人} and.....and {第五门科目成绩比该同学差的人}
- 怎么快速实现集合并?
- bitset
- 那么如何把 {第一门科目成绩比该同学差的人} 快速求出?
- 对成绩分块, 预处理
- $b[i][j]$  表示第  $i$  门科目成绩小于等于  $j * \sqrt{n}$  的人的集合
- 那么再加上剩余未被分进完整块的  $\sqrt{n}$  个人就是 {第一门科目成绩比该同学差的人} 了
- 所需时间复杂度  $O(\sqrt{n})$

# Solution

- 最后再把这五个得到的bitset并起来即可
- 时间复杂度  $O(\sqrt{n} * 5)$

# 集训队互测2015 Robot

- 初始有 $n$ 个机器人在一条数轴上，给定初始位置（所有机器人都做匀速运动）
- $m$ 次操作，每次操作给出当前时间（时间递增），有两种类型
- 改变一个机器人的运动速度
- 询问当前离原点最远的机器人
- 数据范围：  $n, m \leq 10^5$

# Solution

- 超哥线段树维护线段

# CodeChef APRIL 15 Black-white Board Game

- 有一个 $n \times n$ 的矩形，每行的黑色格子都是连续的一段区间  $(l[i], r[i])$ 。两人玩游戏，第一个人选择一个逆序对数为偶数的排列 $a$ ，且 $(i, a[i])$ 这个格子为黑色，
- 第二个人类似，但排列的逆序对数需为奇数。出现过的排列不能再出现。给出排列多的人胜。
- 数据范围： $n \leq 100000$

# Solution

- 相当于求该矩形的行列式
- 先把所有行排序, 设  $l[i]$  为第  $i$  行黑色格子左端点,  $r[i]$  为第  $i$  行黑色格子右端点
- 以  $l[i]$  为第一关键字,  $r[i]$  为第二关键字将行排序
- 然后我们来模拟高斯消元的过程
- 将所有  $l[i]$  相同的行分为一类
- 我们从小到大枚举每一类, 对于某一类, 我们取出  $r[i]$  最小的一行, 然后用这一行去消同一类中其余所有的行
- 那么这一类中的其它行的黑色格子左端点都变成了  $r[i]+1$ , 将他们都划到第  $r[i]+1$  类即可
- 也就是说我们需要一个可以取最小值和可合并的数据结构
- 左偏树是一个不错的选择, 时间复杂度  $O(n \log n)$



# XJOI 不可视境界线

- 有 $n$ 个敌人，第 $i$ 个敌人战斗力为 $A_i$ ，打败该第 $i$ 个敌人至少需要 $B_i$ 战斗力，与第 $i$ 个敌人需要受到 $\sqrt{a^2 + (A_i - C)^2} - a$ 点伤害( $a$ 为之前所受伤害， $C$ 为当前战斗力)，战斗后战斗力会变成 $A_i$ ，初始 $C=0$
- 与第 $i$ 个敌人战斗后就不能再和 $1 \sim i$ 号敌人战斗了，问战胜 $n$ 号敌人后至少受多少伤害。
- 数据范围： $n \leq 10^5$

# Solution

- 第 $i$ 个点能从第 $j$ 个点转移 ( $j < i \ \&\& \ A[j] \geq B[i]$ )
- 战斗后受到的总伤害为  $\text{sqrt}(f[j]^2 + (A[j] - A[i])^2)$
- 欧几里得距离!
- 最近点对!
- KD tree!

# Codeforce k-Maximum Subsequence Sum

- 给一系列数，要求支持操作： 1. 修改某个数的值 2. 读入  $l, r, k$ ，询问在  $[l, r]$  内选不相交的不超过  $k$  个子段，最大的和是多少。
- 数据范围：  $n, m \leq 10^5$ ,  $k \leq 20$

# Solution

- $K=1$  ?
- 经典线段树
- $K=20$
- 维护20个东西  $K^2$ 合并
- 每次复杂度  $O(K^2 \log n)$

# Solution

- 先来看一下网络流该怎么做
- 源点向每个点连边，容量为1费用为0，每个点拆点， $i$ 向 $i'$ 连边容量为1，费用为 $a_i$ ，然后 $i'$ 向 $i+1$ 连边，再限制流量不超过 $k$
- 这个网络流显然是对的
- 在增广的过程中，要么找一个目前最大子段和，要么走反向边
- 用线段树来模拟这一过程：
  - 找出目前区间最大子段和，累加进答案，将这一子段 $\times -1$
  - 重复上一步骤 $k$ 次
- 时间复杂度 $O(n \log n k)$

# Codeforces Round #189 (Div. 1) Ping-Pong

- 有一些区间，能从  $(a, b)$  走到  $(c, d)$  的条件是  $c < a < d$  或者  $c < b < d$ ，有以下两种操作：
- 新加一个区间（后加入的区间保证比新加入的区间长）
- 询问能否从第  $a$  个加入的区间走到第  $b$  个加入的区间
- 初始区间个数为 0
- 数据范围：操作数  $\leq 10^5$

# Solution

- 如果后加入的区间能走到先加入的区间，则这两个区间能互相走到
- 若存在这样的区间就把他们合并成一个区间
- 为什么是对的？
- 若之后有一条线段能和合并过的线段合并，合并线段中必然包含一条能和该线段合并的线段。
- 这样就把互相能走到的线段都合并起来了，剩下的就是只能单向走的，判断下就可以。