《时代的眼泪》命题报告

——李欣隆 nzhtl1477 高中就读于成都七中

总结

- 2位选手获得100分的好成绩
  1位选手获得84分
  2位选手获得76分
  5位选手获得72分
  31位选手获得大于等于64分
  116位选手获得大于等于48分
  332位选手获得大于等于24分
  21位选手爆零

命题人

- 李欣隆 nzhtl1477
- 蔡承泽 ccz181078

题意

- 给定一个二维平面，初始给定一个大小为n的点集(x[i],y[i])
- 有m次查询，每次查询有多少点对(i,j)，其中l1<=x[i]<x[j]<=r1且l2<=y[i]<y[j]<=r2

语 录

- 为保护当事人隐私，以上图片均已打码
- 因为OI已经不太考这种东西了吧，所以这道题叫做《时代的眼泪》

12分部分分

- 满足n,m<=100
- 我们可以每次暴力枚举出所有(i,j)的二元组，然后检查一下是否满足题意即可
- O(n^2m)

24分部分分

- 满足n,m<=5000
- 我们可以每次找出在矩形中的每个点，发现问题就是求这些点的顺序对，顺序对和逆序对类似，使用归并排序或树状数组即可
- O(nmlogn)

16分部分分

- 对所有询问，满足l2[i]=1,r2[i]=n
- 可以发现这个问题实际上就是查询区间的顺序对
- 区间顺序对和区间逆序对一样都是不弱于整数矩阵乘法的，考虑根号算法

16分部分分

- 莫队算法：
- 给定m个询问，有一种特殊的方法将其排序，使得总转移量为O(nsqrtm)，这里不详细讲解其细节

16分部分分

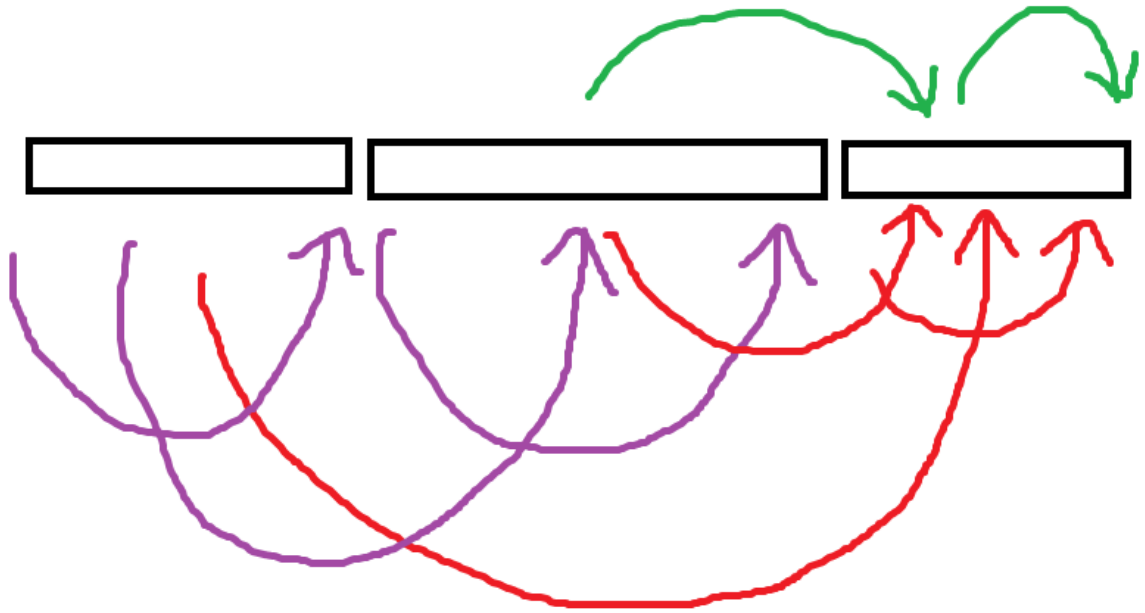
- 莫队算法：
- 考虑维护区间的答案时同时维护一个数据结构，支持插入，删除，和查询小于x的元素个数
- 用树状数组来维护比较高效，时间复杂度O(nsqrtmlogn)
- 可能拿不到满分

16分部分分

- 莫队二次离线算法：
- 由于这里贡献可差分，所以有优化的余地
- 首先我们预处理出所有前缀[1,x]的答案，记做pre[x]，后缀[x,n]的答案，记做suf[x]

16分部分分

- 考虑从[l,x]转移到[l,y]时这个过程
- 如果加入pre[x+1,…y]这些元素，则加入了所有紫色与红色的贡献
- 我们想要的是绿色的贡献
- pre[x]是紫色的贡献



16分部分分

- 每次转移的时候加入pre[a]，x+1<=a<=y，然后减去pre[x]
- 这样得到的是红色的贡献
- 可以发现与我们想要的绿色的贡献的差距只有[x+1,y]对[1,l-1]这部分差距了
- 第一次离线：
- 于是这里开个vector，在l-1处push_back一个多元组表示这个差距的区间[x+1,y]以及这是第几次转移

16分部分分

- 这样问题变为了O(nsqrtm)次查询前缀中小于x的元素个数
- 注意到前缀只有[1,1],[1,2]…[1,n]这O(n)个
- 这里可以使用O(sqrtn)修改O(1)查询的根号平衡
- 第二次离线：
- 做一次扫描线，将第一次离线后的每个莫队转移时的询问处理出来
- 总时间复杂度O(nsqrtm)

16分部分分

- 复杂度是O(m+nsqrtm*k+n^(1+1/k))

- 当m= Ω(n^2)时应写为O(nsqrtm+m)
  当m=O(n^2)且m=poly(n)时为O(nsqrtm)，这道题实际上是这个case
  当m低于poly(n)时比较麻烦，不过一般不考虑这种情况

12分部分分

- 特殊性质：逆序对不超过c=50个
- 我们预处理出所有逆序对
- 每次查询看矩形中有几个逆序对，然后算出矩形中点数就知道顺序对个数了
- O(nlogn+m(c+logn))

12分部分分

- 特殊性质：所有查询的矩形包含或不相交
- 可以发现矩形之间构成一个树的关系
- 可以扫描线建树，将询问建成一个森林的形式，然后在这个询问的森林上启发式合并可以得到所有询问的答案。
- 如果启发式合并+树套树维护，会得到一个 O(mlog^3n) 的做法，这个复杂度可能只能通过部分数据。

12分部分分

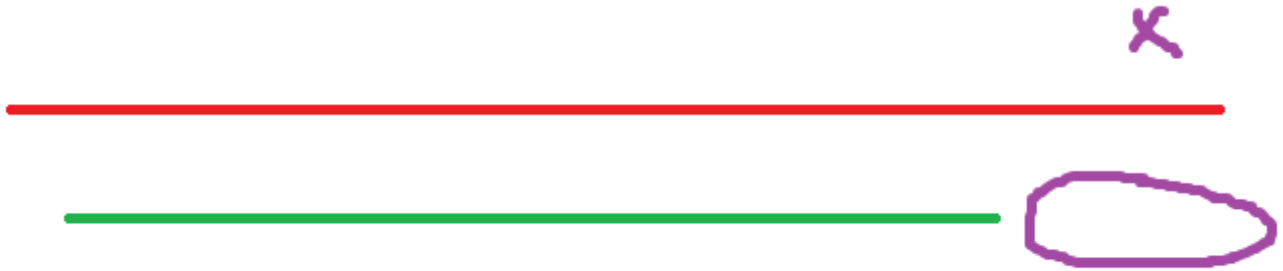
- 如果启发式合并，全局使用用可持久化值域线段树预处理，可以做到 O(mlog^2n)，这个复杂度可以通过所有部分分。
- 不过差不多出题完成之后我才意识到好像直接莫队跑这个点很高效，仔细研究了一下，最后还是给大家送温暖了
- 但实际上这种性质的询问莫队是可以被攻击的

对此算法的攻击

- 那么莫队算法在这样的数据下工作效率如何?
- 我们先关注一下区间上的情况

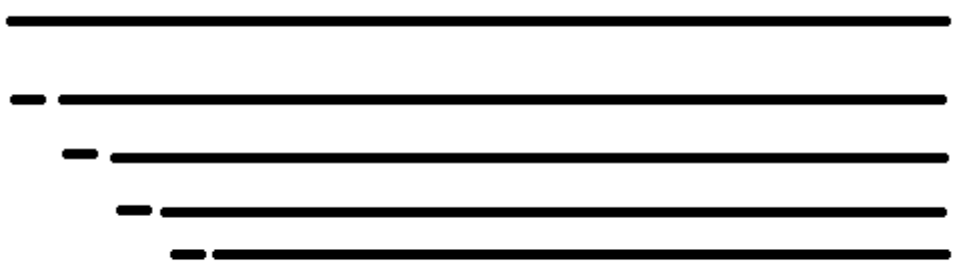
对此算法的攻击

- 可以发现如果我们构造了这样一个转移，会直接将原序列划分出一个长x的部分，这个部分之后只能在其中划分出子区间



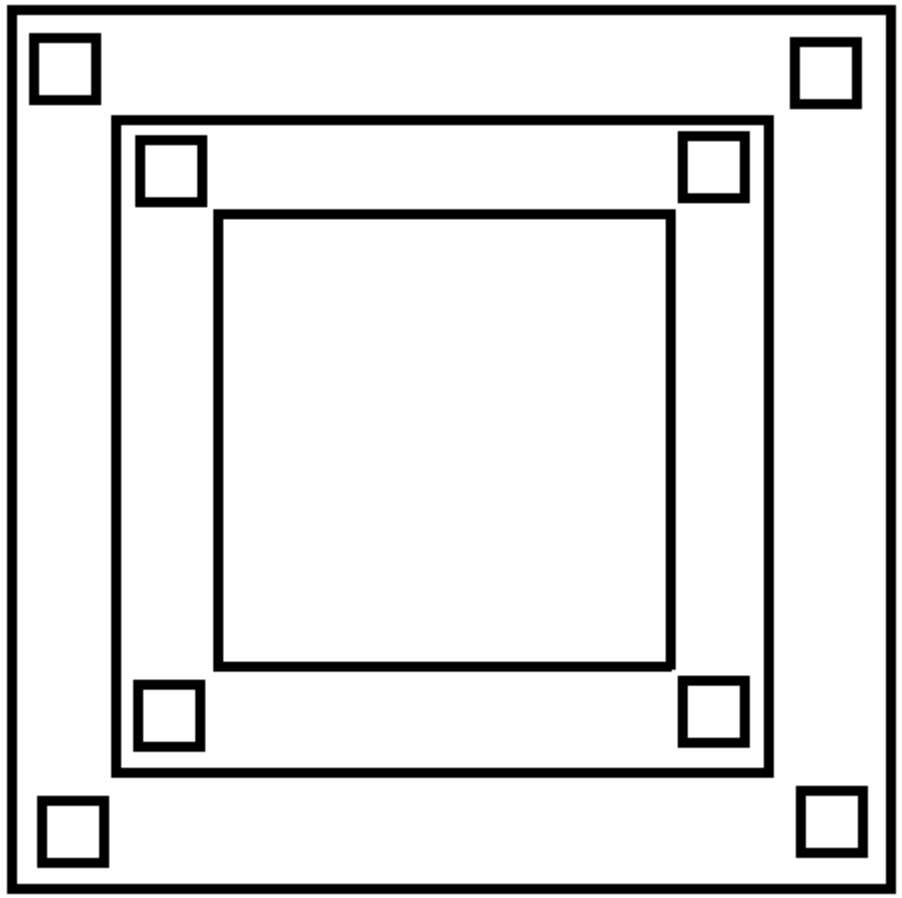
对此算法的攻击

- 我们分析后发现，块外的贡献大概没法卡（理论上感觉是对的，不过最近太忙没有进行严谨分析），块内还是有希望的：
- 如果这样一个结构在块内的话，莫队右端点会反复横跳，转移的复杂度会被卡满



对此算法的攻击

- 在二维平面上可以构造类似的结构，把这个放主对角线上
- 这样的结构能自适应所有块大小
- 但是如果有动态调整块大小的方
- 法，则这个方法会失效，在这个
- 情况下，该莫队算法的转移次数
- 应该不太能卡掉
- 同时这样的构造有小常数，所以
- 好的实现应该可以通过



对此算法的攻击

- 但是实际上好像选手都轻轻松松通过了这个部分分？
- 我不太明白为什么大部分人都没被卡掉，我自己写的O(n^7/4)是被卡到了2e8次转移的
- 这个部分分设计的没有达到预期，因为部分分分值过少而且比较难写，还能被通用莫队算法高效解决，本来想的是卡通用莫队4分，但实际上效果甚微。

24分部分分

- n在2e4~7e4之间，m在4e4~1.4e5之间
- 有一种被叫做“二维莫队”的方法，个人感觉还是叫做二维平面上的莫队比较好
- 这里相当于在做一个 四 维 空 间 上的莫队
- 时间复杂度为O(nm^0.75)，在随机数据下约4500是最优的块大小

24分部分分

- 经过测试，莫队总转移量在随机数据下为1.2e9
- 但是每次被转移的点实际上在这个转移量下只有1/3的概率在矩形中，所以贡献的位置只有4e8次

24分部分分

- 转移时可以像普通莫队那样维护一个树状数组，时间复杂度O(nm^0.75logn)，可能能通过2e4,3e4的部分分
- 也可以使用二次离线的方法
- 这里如果使用二次离线的方法，每次转移需要查询一个矩形内点的个数，会进行4次前缀小于x的元素个数的查询，可以优化到3次，我感觉这是极限了
- 所以总查询量实际上为1.2e9，难以通过

对此算法的攻击

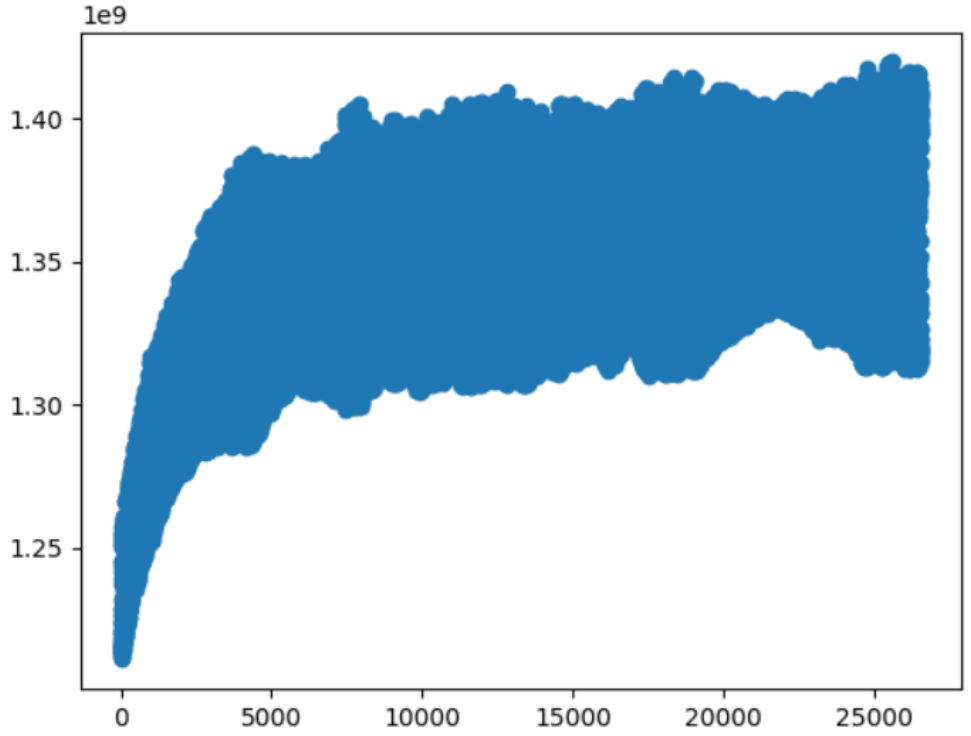
- 当莫队算法的块大小确定时很容易进行攻击，但如果随机块大小，则变成一个看上去很复杂的高维优化问题

对此算法的攻击

- 实际上最终的数据使用了一种类似于梯度下降的方法对此算法进行了攻击，每次随机修改一个点，或者计算每个询问端点移动时的一个伪梯度，沿梯度方向移动端点
- 经测试，不基于梯度的优化算法，如遗传算法，爬山算法，模拟退火算法等比较低效
- 迭代一天后，总转移次数卡到了1.32e9次，大约多了10%

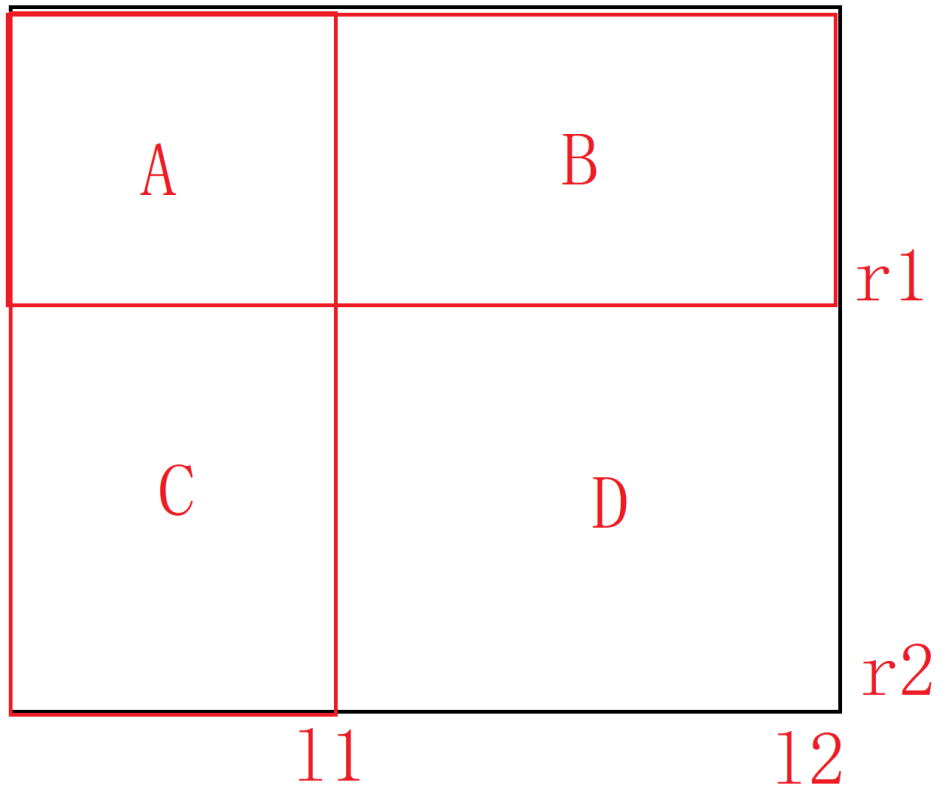
对此算法的攻击

- 有一种高维的构造方法使得相近点对之间距离比较短
- 比较复杂我没有仔细研究，使用网上的一个sobol序列进行了测试，约为1.26e9次转移次数，以其为梯度下降的初始解可以让收敛速度显著变快
- 如图，这里最优化的是多个块大小
- 以及随机块初始位置的均值



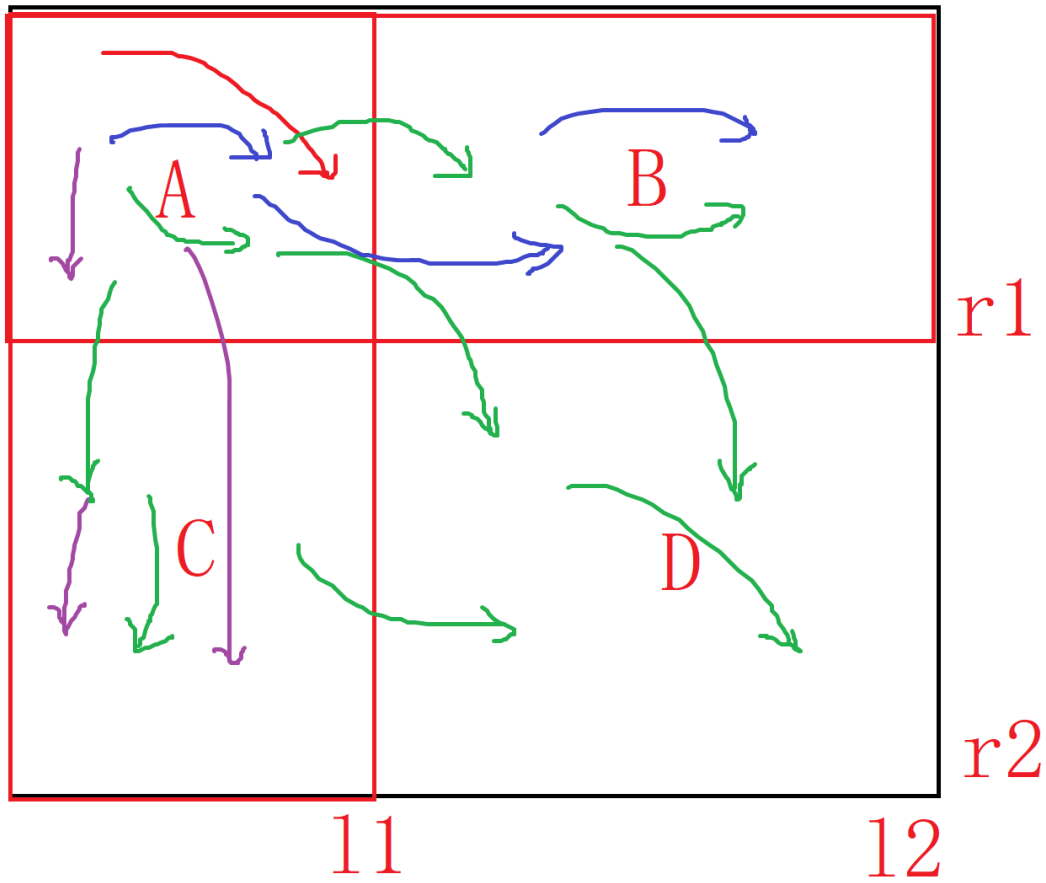
24分部分分

- 可以考虑将四维的莫队进行差分，变成双前缀的莫队形式
- A为[1,11-1,1,r1-1]的答案
- AB为[1,12,1,r1-1]的答案
- AC为[1,11-1,1,r2]的答案
- ABCD为[1,l2,1,r2]的答案
- D=ABCD-AB-AC+A



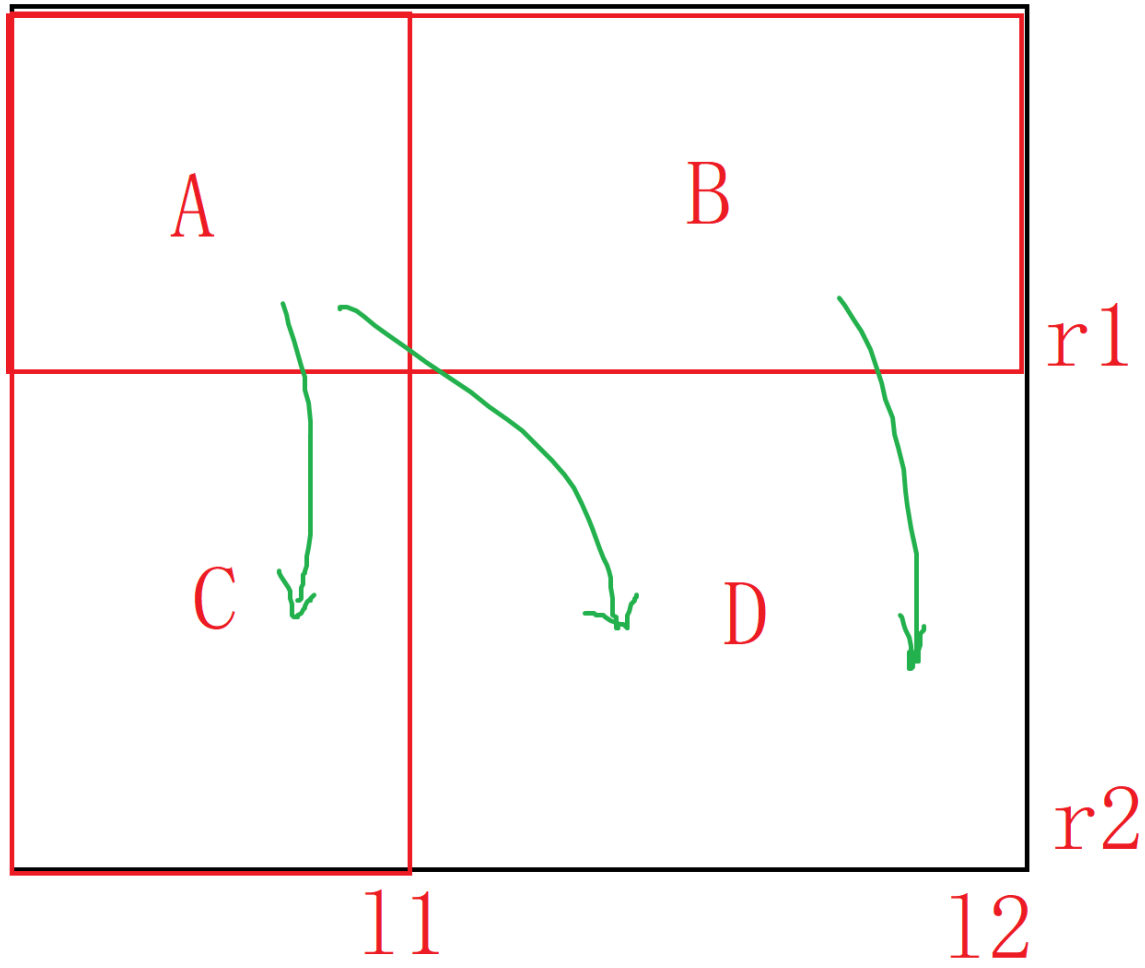
24分部分分

- D=ABCD-AB-AC+A
- -(B,D)-(C,D)-(A,D)
- (A,D)之间的贡献是平凡的，即为
- size(A)*size(D)



24分部分分

- 不失一般性，考虑(B,D)之间的贡献
- 可以差分为
- (B,D)=(AB,CD)-(A,C)-(A,D)
- 我们发现(AB,CD)与(A,C)均共用了
- 一端的限制，所以从四维问题变为
- 了一个三维问题
- 4-side -> 3-side



24分部分分

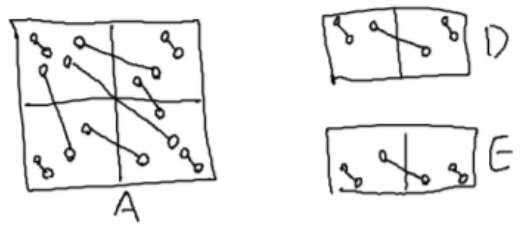
- 进行三维的莫队二次离线可以做到O(nm^2/3)的复杂度
- 转移次数因为有很多种写法，每维的转移还不对称，所以不太好计算，推测在9e8~1.6e9之间
- 对此算法没有进行攻击

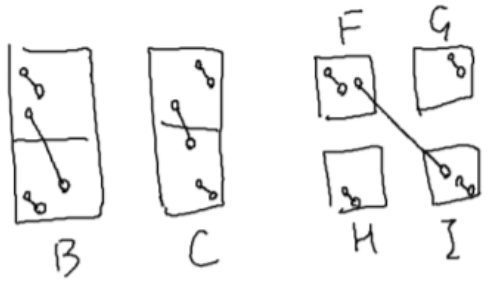
100分算法

- “第十三分块”，难度约7/10
- 约2019.10由蔡承泽同学提出，共同完成
- 考虑对平面进行分治，同时在这个分治结构上进行分块

100分算法

- 我们对原点集建立一棵树套树
- 树套树是一个DAG，共O(nlog^2n)个节点
- 先考虑如何维护树套树的
- 每个节点的答案
- 这里ans(B),ans(C),ans(D),ans(E)即为
- 树套树上节点的区间顺序对
- 本题可以分治的重点在于F与I的贡献
- 是平凡的



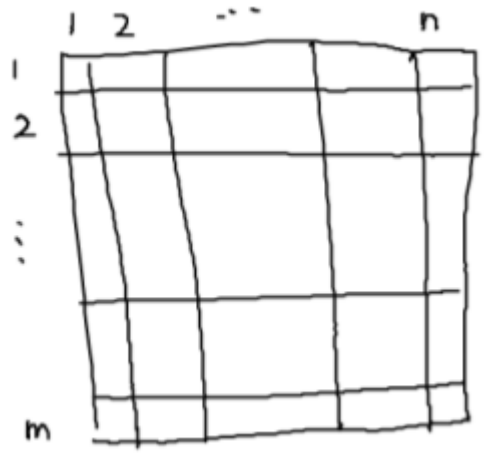


size (A) = size (B) + size (C)

ans (A) = size (F) · size (I)
+ ans (D) + ans (E)
+ ans (B) + ans (C)
- ans (F) - ans (G) - ans (H) - ans (I)

100分算法

- 每次查询先将其所对应的O(log^2n)个树套树节点提取出来
- 然后考虑在上面做类似的容斥



$$ans(1..m,1..n)=\sum_{i=1}^{m}\sum_{j=1}^{n} size(i,j)\sum_{k=1}^{i-1}\sum_{l=1}^{j-1} size(k,l)$$

$$+\sum_{i=1}^{m} ans(i,1..n)$$

$$+\sum_{j=1}^{n} ans(1..m,j)$$

} 区间逆序数

$$-\sum_{i=1}^{m}\sum_{j=1}^{n} ans(i,j)$$

$(1,1)\sim(m,n)$ 为树套树节点

100分算法

- 于是我们通过一个对平面的分治成功将问题转换为了很多次区间逆序对
- 可以发现询问的不均并不会导致总复杂度变差，时间复杂度:

$$T(n) = \sum_{i=0}^{k} T'(n, 2^k, 2m)$$

$$T'(n, d, m) = \max_{\sum m_i = m} \sum_{i=1}^{d} \frac{n}{d} \sqrt{m_i}$$

解得 $T(n) = O\left(m \log n + n\sqrt{m}\right)$。

总的时间复杂度为 $O\left(n\sqrt{m} + (n+m)\log^2 n\right)$，空间复杂度和具体实现有关，一般的实现需要$O(n+m)$或$O\left((n+m)\log n\right)$空间，这里开了 1GB 内存放过了 $O((n+m)\log^2 n)$ 空间的实现，std是 $O\left((n+m)\log n\right)$的，将树套树换成了在压缩trie上合并。

???分算法

- 区间逆序对可以做到O(n^(2ω/(ω+1)))！QωQ
- 带入目前最优的ω=2.373得n^1.41
- 如何o(n^1.5)实现区间逆序对？

???分算法

- 乘的两个矩阵分别是 序列块i权值块k的点数 和 权值块1..k-1序列块j的点数
- 乘积是序列块i,j间，权值块间贡献
- B为序列/权值块的个数
- 这样做1次n=m=k=B的矩阵乘法可以算出块间贡献
- 这里矩阵乘法复杂度为O(B^(2ω/(ω+1)))
- 块内贡献暴力维护，计算出时间复杂度O(n^1.41)

???分算法

- 这个算法是目前理论最优的
- 但是这样的算法应该not practical，只存在于理论

总结

- 有个别选手写了复杂度不是很高的分块算法，却没有获得理想的成绩，个人认为还是在这种高维情况下，分块算法的常数比莫队算法高了很多，建议能写莫队还是写莫队吧。

Q&A

- QAQ

Thanks for listening