JOIsc 好题选讲

宁波市镇海中学 梁晏成

Port Facility

- 有N件货物，第i件货物在A_i时刻进入港口，在B_i时刻运离港口。
- 有两艘不同的船可以用于暂存货物。当某一个货物进入港口时，你必须选择一艘船，将它放在这艘船内已经堆放的货物的最上方
- 当你将某个货物运离港口时，这个货物必须位于它所在船中所有货物的最上方（即每艘船内的货物形成一个栈的结构）
- 求有多少种方案能满足条件，对10^9+7取模
- 两个方案不同当且仅当存在一个货物，在两个方案中进入港口时该货物选择的船不同

- $N \leq 10^6$，$1 \leq A_i, B_i \leq 2N$，$A_i < B_i$，　A_i, B_i互不相同

Port Facility

- 考虑两个货物$i,j,(A_i < A_j)$，有以下三种情况：
- 1）$A_i < A_j < B_i < B_j$
- 那么如果这两个货物选择了同一艘船，在第i个货物离开港口的时候，第j个货物一定在第i个货物的上方，不满足条件
- 因此这两个货物必须放在不同的船上

- 2）$A_i < B_i < A_j < B_j$
- 3）$A_i < A_j < B_j < B_i$
- 不难发现上述两种情况下i,j两个货物可以看成独立的，不会互相影响

Port Facility

- 我们N^2枚举所有的i,j，如果二者不能位于同一艘船上，就在i,j之间连一条边。
- 这样就转化为判断是否是二分图，以及二分图染色方案数的经典问题了，求出连通块个数即可。

- 时间复杂度$O(N^2)$

Port Facility

- 方法一：
- 一种比较显然的想法是利用数据结构来优化连边。

- 可以发现连边和二维数点有一定的联系，我们使用复杂度为$O(N\log N)$主席树来做，具体地：
- 我们按照A_i从小到大来考虑，可以发现$[A_i,B_i]$会和所有$A_j<A_i<B_j<B_i$的所有货物$[A_j,B_j]$连边
- 因此我们将所有$A_j<A_i$的货物按照B_j插入主席树，然后考虑到i时，查询区间$[A_i,B_i]$并连边即可。
- $O(N\log N)$

Port Facility

- 方法二：
- 我们引入一种新的边：0边，表示这条边两侧的点颜色相同。
- 我们把原来的边称为1边，表示这条边两侧的点颜色不同

- 这样，如果我们要连边$i \to a, i \to b, \ldots, i \to z$，可以转化为：
- 1）建立1边：$i \to a$
- 2）建立0边：$a \to b, b \to c, \ldots, y \to z$
- 那么在之前主席树的过程中，我们不用主席树来维护，直接用set取所有出B_j位于$[A_i, B_i]$中的点j，将相邻的两个货物用0边连起来
- 如果一个点向左右都连了边，我们就可以将它删掉
- 简单分析一下可以发现时间复杂度为$O(N \log N)$

Port Facility

- 方法三：
- 分治，先考虑如何处理所有经过中点的区间：
- 将所有货物看成二维坐标上的一个点(A_i, B_i)
- 如果存在$(a,b),(c,d),(e,f)$满足$a < c < e \leq mid < b < d < f$，则不合法
- 否则将所有的区间分成两个队列进行连边

- 处理所有经过中点的区间和不经过中点的区间之间的连边，我们可以考虑把经过中点的区间按照左（右）端点排序，使用之前0边的技巧即可

- 时间复杂度$O(N \log N)$

Sparklers

- 有N个人站成一排，每个人手中有一束烟花，第i个人一开始站在X_i处。
- 一开始只有第K个人手中的烟花是在燃烧的。每一束烟花可以燃烧T秒，并且我们认为第T秒它并不会熄灭。
- 现在他们要互相合作使得每个人手中的烟花都燃烧过。一个人A手中的烟花能传递到另一个人B当且仅当：
- 1）两人位于同一位置
- 2）A手中的烟花已经燃烧了恰好T秒
- 3）B手中的烟花还没有燃烧过
- 每一秒钟一个人可以跑s的距离，询问能满足条件的s的最小值

- $N \leq 100000$，$T, X_i \leq 10^9$

Sparklers

- 首先注意到答案具有可二分性，我们不妨二分答案s然后进行检验
- 同时我们重新定义新的$s = T \times$ 原来的s，也就是一束烟花从点燃到熄灭一个人可以跑多远

- 假设当前有M个人的烟花已经燃烧过了，那么这M个人一定可以是连续的
- 用$L_{i,j}, R_{i,j}$表示$i \sim j$中所有人的烟花已经燃烧过了，现在手中烟花正在燃烧的人刚得到烟花时可以位于区间$[L_{i,j}, R_{i,j}]$中的任意一个点
- 转移时枚举传递给$i-1$或者$j+1$，假设给k，那么此时k的得到烟花时可以位于的区间为$[X_k - M \times s, X_k + M \times s] \cap [L_{i,j} - s, R_{i,j} + s]$
- 时间复杂度$O(N^2 \log ans)$

Sparklers

- 简单分析可以发现：
- 若$[L_{i,j}, R_{i,j}] \neq \emptyset$，那么
- $[L_{i,j}, R_{i,j}]$
- $= \cap_{k=i}^{j} [X_k - (j-i) \times s, X_k + (j-i) \times s]$
- $= [X_j - (j-i) \times s, X_i + (j-i) \times s]$，归纳即可证明
- 令$G_{i,j} = (X_i + (j-i) \times s) - (X_j - (j-i) \times s)$
- 那么$[L_{i,j}, R_{i,j}] \neq \emptyset$当且仅当存在一系列的区间$[\mathrm{K}, \mathrm{K}], [a, b], [c, d], \ldots [i, j]$，满足每个区间相较之前的区间仅在一个端点处向外拓展一位，并且任意区间$[x, y]$满足$G_{x,y} \geq 0$

Sparklers

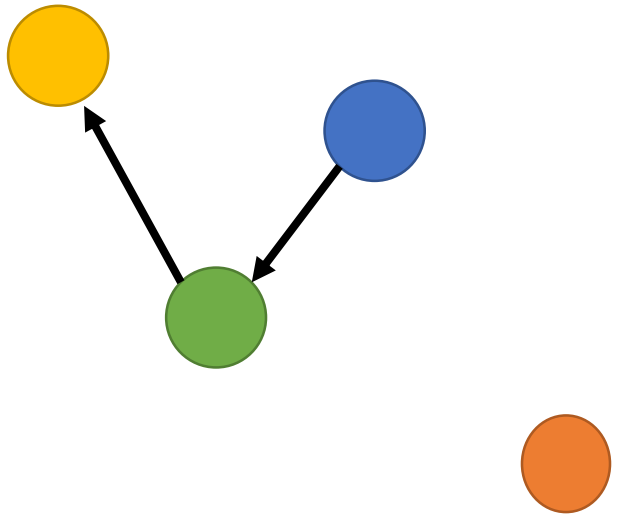
- 我们定义$A_i = X_i - 2 \times i \times s$
- 那么$G_{i,j} = (X_i + (j - i) \times s) - (X_j - (j - i) \times s)$
- $= X_i - 2 \times i \times s - \left(X_j - 2 \times j \times s\right) = A_i - A_j$
- 所以$G_{i,j} \geq 0 \Rightarrow A_i \geq A_j$

- 那么我们转化问题如下：
- 平面上有N个点(i, A_i)，开始时有两个点都位于(K, A_K)，每次可以选择一个点向远端移动一位，需要时刻保持左侧的点的高度≥右侧的点的高度

Sparklers

- 情况一：
- 如果存在操作能使左侧的点高度增大，那么我们贪心地走过去显然不劣；右侧同理

- 例如右图，如果左端点位于蓝点，右侧的点位于橙点
- 那么我们贪心让左端点从蓝点走到绿点再走到黄点，显然不劣



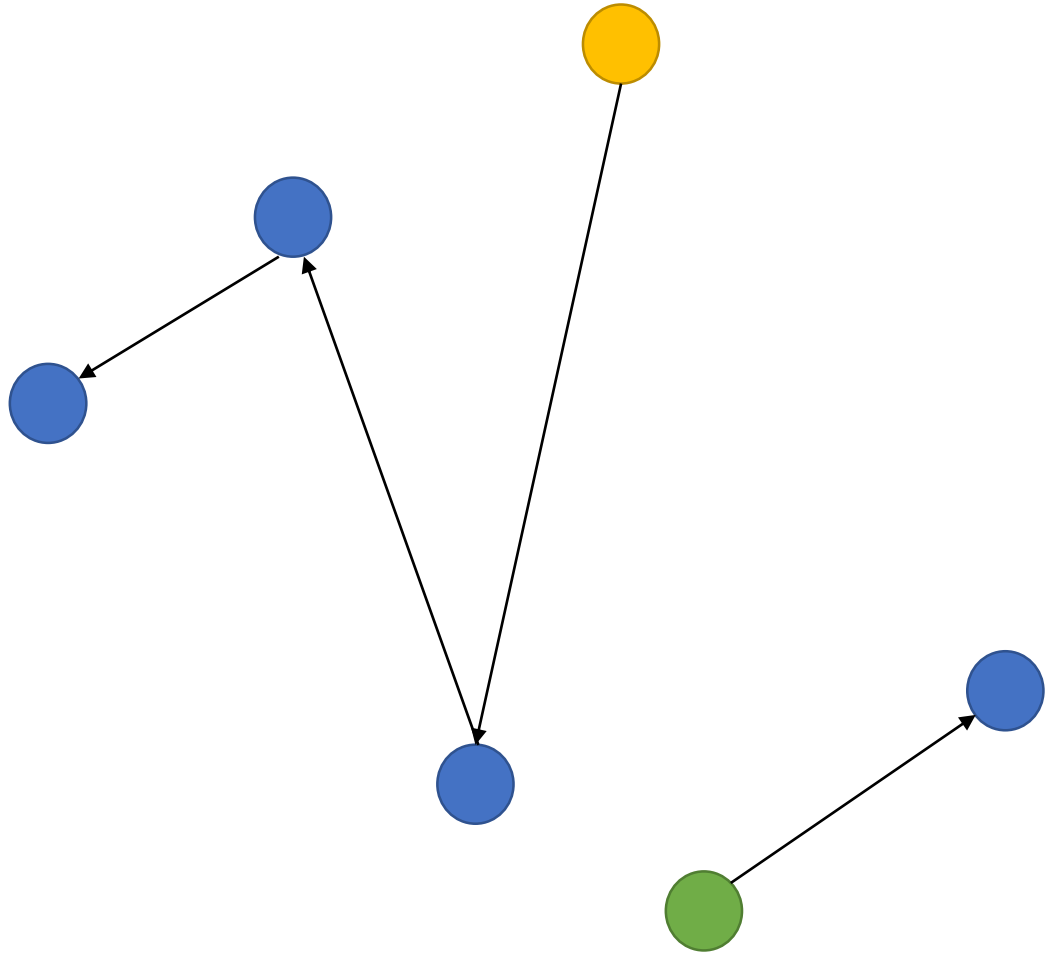
Sparklers

- 情况二：
- 如果不能进行之前的贪心，容易证明以后左侧的点不可能变高，且右侧的点不可能变低
- 考虑左侧的位于点X，点X左侧最低的点是Y，Y左侧（包括自己）最高的点是点Z
- 那么如果我们移动左侧的点，一定是连续操作使从$X \to Y \to Z$，否则没有意义；右侧同理
- 于是我们可以把没有用的点删去

Sparklers

- 删完无用点后可以得到类似这样一张图
- 可以发现在这张图中操作的顺序不会影响最后的结果
- 因此每次选择一个可行的操作进行即可

- 如果用ST表实现贪心的过程，总的时间复杂度为$O(N \log N \log ans)$



Cultivation

- 有一个$W \times H$的网格图，N个格子上长着草
- 接下来你可以进行一次操作（定义为“向上”，“向下”，“向左”，“向右”）
- 如果进行一次“向上”操作，那么当前所有长着草的格子的上方就会长出新的草（新的草不会再向上生长）。其余操作同理
- 问最少进行几次操作，能够使所有的格子都有草

- $W, H \leq 10^9$，$N \leq 300$

Cultivation

- 我们先来考虑一维的情况，例如 _ _ X X _ _ _ X _ X _ _
- 可以发现我们至少要进行4次操作，先“向左”2次，再“向右”2次。
- 而_ X _ _ _ X _需要进行3次操作

- 假设最左边的X到左侧的距离为A，相邻两个X直接的距离的最大值为B，最右边的X到右侧的距离为C。如果进行L次“向左”操作和R次“向右”操作，那么
- $L \geq A, R \geq C, L + R \geq B$
- $\Rightarrow \min L + R = \max(A + C, B)$
- 如果同时有多行，那么$\min L + R = \max(\max A + \max C, \max B)$

Cultivation

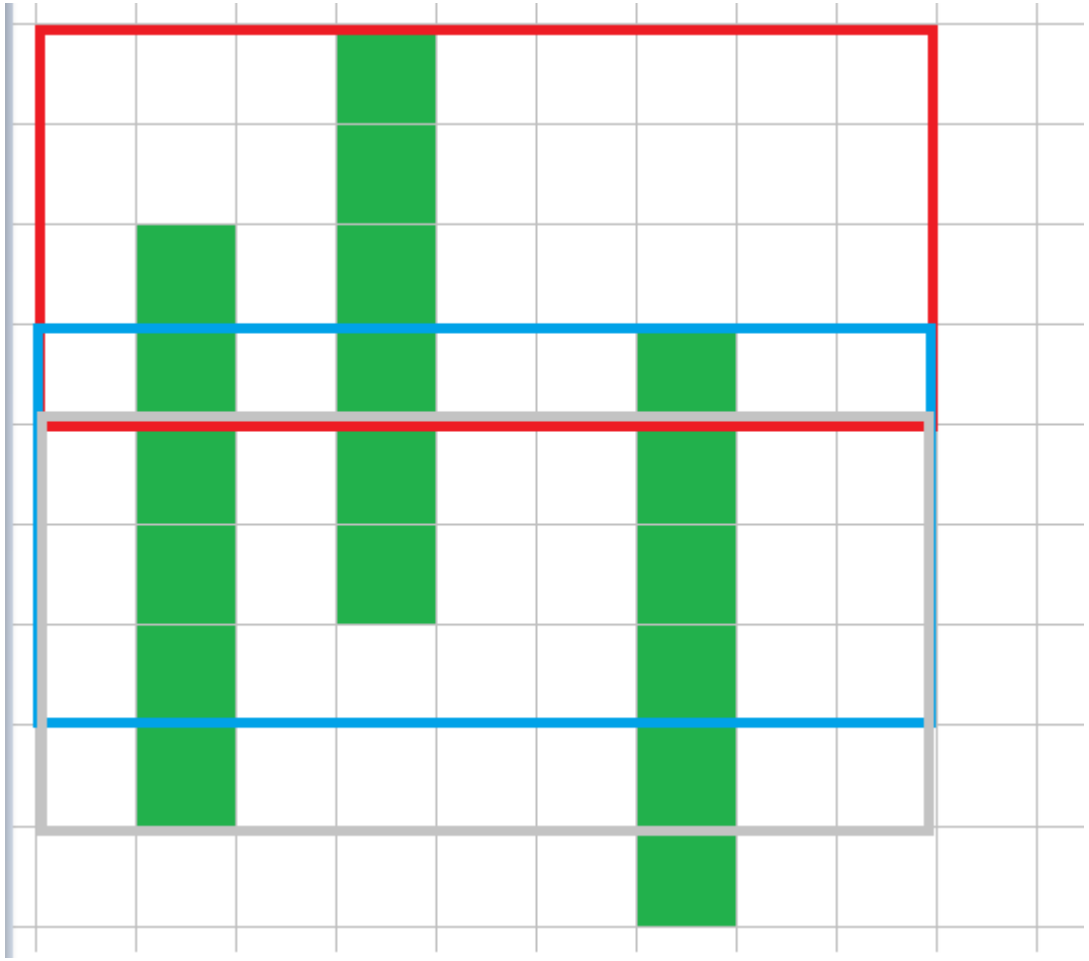
- 显然操作的顺序无关，接下来我们先考虑“向上”和“向下”，再考虑“向左”和“向右”
- 我们将“向上”X次，“向下”Y次的操作，转化为“向下”$X+Y$次后，再将整个网格向下移动X格

- 我们枚举$X+Y$，然后用单调队列在向下移动整个网格的同时维护$\max A, \max B, \max C$
- 时间复杂度$O(H*N)$

Cultivation

- 考虑只有哪些$X+Y$是有用的。

- 对于右图所示的$X+Y=5$的情况，只有三种有用的平移情况，即每一种平移方案一定是以某一块草的开头作为整个网格图的上边界
- 那么对于以某一块草(x_i, y_i)作为上边界的网格图，它的$\max A, \max B, \max C$一定在某一段草结束时或最下处取到



Cultivation

- 考虑(x_i, y_i) 和(x_j, y_j)之间的影响：
- 1）$x_j < x_i, X + Y \geq x_i - x_j - 1$ （影响第j段草结束时A, B, C的取值）
- 2）$X + Y \geq x_i - x_j + H - 1$（影响以第$i$段草的开头为上边界时，网格图最下处$A, B, C$的取值）

- 因此只需要考虑$O(N^2)$个不同的$X + Y$即可。总时间复杂度$O(N^3)$

Broken Device

- 你需要传递一个数X
- 你可以发送一份长度为150的01串。由于发送装置是损坏的，在这150位中，有指定的K位是损坏的，这些位置上只能发送0。你加密时知道这些信息。
- 你需要根据发送的01串，还原出X。注意，你解码时只知道发送装置是损坏的，而不知道K的值，也不知道哪些位置只能发送0

- $X \leq 10^{18}$，$K \leq 40$

Broken Device

- 显然X可以被写成一个60位01串。
- 我们计算得到最多可以浪费$150-60-40=50$位
- 那么我们考虑每3位为一段，每一段最多浪费1位，这样可以满足条件。
- 即如果这3位只有1位是损坏的，我们让它至少传递1位的有效信息；
- 如果没有任何一位损坏，我们让它至少传递2位的有效信息

Broken Device

- 一个可行的构造方案如下：

- $001 \to 1$
- $010 \to 00$
- $011 \to 01$
- $100 \to 0$
- $101 \to 10$
- $110 \to 1$
- $111 \to 11$

Railway Trip

- 有N个火车站和K种火车，每个火车站都有一个等级$L_i(L_i \leq K)$。第j种火车只会在$L_i \geq j$的车站停车。每一种火车都会在铁道上不断地来回行驶。火车行驶的时间忽略不计。满足 $L_1 = L_N = K$
- 有Q次询问，每次给出起点和终点，问从起点前往终点，火车最少停多少次（中途经停也算火车停下）
- 当你位于车站时，你可以选择任意一个方向，并选择任意会在这个车站停车的火车。在某个车站下车并换乘另一趟火车只记作一次停下

- $N, Q \leq 10^5$

Railway Trip

- 问题可以转化为：
- 若点对x, y满足 $L_x \leq L_y$且x和y之间不存在k使得$L_k \geq L_x$，那么这两个点之间有一条无向边
- 多次询问两点间最短路

- 首先我们可以要求只能从点权较小的走向点权较大（可以相同）的，然后我们同时从S, T出发，问走到同一个点至少需要几步

Railway Trip

- 定义$L_{i,j}, R_{i,j}$表示从i出发走j步，最左和最右分别能走到哪里。
- 容易发现$L_{i,j+1}, R_{i,j+1}$只和$L_{i,j}, R_{i,j}$有关

- 不妨假设$S < T$，那么我们先从S出发，走k步使得$R_{i,k} < S \leq R_{i,k+1}$，然后再从$T$走最少的步数到达$R_{i,k}$
- 用倍增加速这一过程
- 时间复杂度$O((N+Q)\log N)$

Arranging Tickets

- 有N个车站按照顺时针围成一圈，编号为$1,2,\ldots,N$
- 有N种车票，其中第i种车票可以从第i个车站前往第$i+1$个车站；也可以从第$i+1$个车站前往第i种车站
- 只能以套票的形式购买车票，一组套票包含每一种车票恰好一张
- 现在有M类人，第i类人一共有C_i个，他们都要从第A_i个车站前往第B_i个车站，问要满足所有人的需求至少需要多少组套票

- $N \leq 200{,}000, M \leq 100{,}000, C_i \leq 10^9$

Arranging Tickets

- 一些约定：
- 为了表述方便，并且在所有部分分中N, M都可以看成同阶的。接下来时间复杂度中的M都写作N
- 从A_i前往B_i和从B_i前往A_i是等价的，因此我们假设$A_i < B_i$

Arranging Tickets

- 原题是在环上进行，这是很不方便的，我们考虑修改一下问题：
- 有M类区间A_i, B_i, C_i，起初所有的区间都覆盖在$[A_i, B_i)$上
- 你可以选择一些区间对$[1, N]$取补（定义这种操作为“翻转”），变为覆盖在$[1, A_i) \cup [B_i, N]$上
- 求一种方案，使得被覆盖次数最多的点被覆盖的次数最少。

- 我们考虑研究所有被翻转的区间，可以发现在最优的方案中一定存在一种，满足条件：
- 对于任意两个被翻转区间$[a, b), [c, d)$，必然有$[a, b) \cap [c, d) \neq \emptyset$
- 证明：采用反证法，若有两个区间不满足，那么这两个区间都不选择翻转，答案不劣

Arranging Tickets

- 有了上述结论，我们可以得到一个多项式的做法，具体地：
- 枚举一个点x，表示所有被翻转的区间原来都覆盖了点x。
- 二分最终的答案ans，枚举需要被翻转的区间个数t

- 我们将跨过点x的区间按照左端点排序。假设现在从左到右考虑到i，之前一共翻转了y个，点i现在被覆盖了z次
- 设现在要翻转now个区间，那么就有：
- $z - now + (t - now - y) \leq ans \Rightarrow now \geq \left\lceil \frac{z+t-y-ans}{2} \right\rceil$
- 那么我们选择所有左端点$\leq i$且没有被翻转的区间中，右端点最大的now个进行翻转，这一部分可以用堆维护。最后进行检验
- 时间复杂度$O(N^3 \log^2 N)$

Arranging Tickets

- 在之前的算法中，我们枚举了两个值x,t，我们考虑减少枚举的量。
- 令a_i,b_i表示初始状态和最终状态下第i个点被覆盖的次数。
- 假设所有的翻转的区间的交集为$[l,r)$，令b_k表示$b_l,\dots,b_{r-1}$中的最大值
- 那么$b_k=\max b_i$或$b_k=\max b_i-1$

- 证明如下：
- 若不满足，选择所有被翻转的区间中左端点最右的和右端点最左的，取消对它们的翻转
- 那么$\max b_i$不会变大且b_k至少$+1$，重复进行上述操作直到满足条件
- 这个结论在之后会被反复用到

Arranging Tickets

- 有了上述结论，我们可以发现如果答案为ans，并且我们枚举k，那么就有$b_k = ans$或$ans - 1$
- 因此可以得到$t = a_k - ans$或$a_k - (ans - 1)$

- 其余部分和之前相同
- 时间复杂度$O(N^2 \log^2 N)$

Arranging Tickets

- 接下来考虑减小k的枚举范围
- 结论：最优解中，存在一个k满足$a_k = \max a_i$

- 依然采用反证法证明：
- 假设所有被翻转的区间的交集为$[l, r)$，那么存在一个j满足$a_j \geq a_k + 1, j \notin [l, r)$
- 因为$j \notin [l, r)$，所以至少存在一个被翻转的区间翻转后覆盖了j而没有覆盖k，那么$b_j - a_j \geq b_k - a_k + 1$
- 那么我们得到$b_j \geq b_k + 2$，和之前的结论矛盾

Arranging Tickets

- 结论：假设a_l, a_r满足$a_l = a_r = \max a_i$，且l, r分别是最左端和最右端的两个

- 同样采用反证法，假设最优方案在$\mathrm{k} = t$处取到，
- 显然$a_t = \max a_i, l < t < r$，现在对最优方案进行考虑
- 一个显然的结论是不会存在被翻转的区间$[x, y)$满足$x > r$或$y \leq l$

Arranging Tickets

- 引理1：不存在一个被翻转的区间$[x,y)$满足$l < x < y \leq r$

- 反证，采用和之前类似的方法，假设不对这个区间翻转，对应的b数组为b'，那么：
- $b'_t = b_t + 1$
- $b'_l = b_l - 1$
- 考虑不对这个区间翻转的情况下，所有翻转的区间的交集为$[u,v)$，那么必然有$t \in [u,v)$，从而$b'_t - a_t \leq b'_l - a_l$，从而$b'_l \geq b'_t$
- 那么我们可以得到$b_l \geq b_t + 2$，矛盾

Arranging Tickets

- 我们要证明$k = l$和$k = r$都不能取到最优解的情况不存在，除了说明引理1中的情况不存在，还需要说明：
- 引理2：不存在两个同时被翻转的区间$[x1, y1), [x2, y2)$，满足$y2 \le r$且$x1 > l$。（等价于如果我们同时取消翻转这两个区间，答案不劣）

- 证明：
- 我们依然用b'表示同时取消翻转对应的b数组
- 还是应用之前多次运用的结论：$b_i - a_i$的值越接近t越小
- 那么就有：$\max(b_l, \dots, b_r) = \max(b_l, b_r)$，$b'$同理
- 那么我们就能得到$\max b_i = \max(j \le l, j \ge r)\, b_j$ ，b'同理
- 而这一部分b数组在同时取消翻转后显然不会增大，引理得证

Arranging Tickets

- 结论：假设a_l, a_r满足$a_l = a_r = \max a_i$，且l, r分别是最左端和最右端的两个
- 有了之前的两个引理，可以轻易得到这个结论

- 那么只需要考虑$k = l$或$k = r$的情况即可。
- 因此只需要二分答案ans，枚举$k = l$或$k = r$，然后枚举翻转的区间数量$a_k - ans$或$a_k - (ans - 1)$
- 由于上述结论对c_i是否为1并没有要求，因此可直接用之前的贪心进行检验，时间复杂度为$O(N \log^2 N)$


to be continued

Long Mansion

- 有N个房间，每个房间内有若干把钥匙，当你第一次进入某个房间时，可以获得这个房间内的所有钥匙。
- 第i个房间和第$i+1$个房间中有一扇门，门可以从两侧打开。每扇门都需要一把特定的钥匙。
- Q次询问如果一开始在房间S且没有任何钥匙，能否走到房间T

- $N, Q \leq 500000$

Long Mansion

- 我们用$[L_i, R_i]$表示从i出发能到达的最左和最右的房间
- 显然，若$j \in [L_i, R_i]$，那么必然有$[L_j, R_j] \subset [L_i, R_i]$

- 我们从左到右计算$[L_i, R_i]$，分情况考虑：
- 1）$R_i \geq i + 1$
- 2）$R_i = i$
- 同时维护一个深度dep，表示当前的区间外嵌套了多少层

Long Mansion

- $R_i \geq i+1$
- 显然$[L_{i+1}, R_{i+1}] \subset [L_i, R_i]$

- 我们先从$i+1$开始贪心向右走到最远的点x，此时如果可以走到i，那么$i+1$就可以走到$[L_i, R_i]$
- 否则只能走到$[i+1, x]$，令dep++

- 从某个点出发向某个方向走到最远的点可以通过预处理+二分做到每次$O(\log N)$

Long Mansion

- $R_i = i$
- 在这种情况下我们直接贪心地走，即不断重复下面的步骤：
- 1）向左走到最远
- 2）向右走到最远

- 从第二次开始，每来回走一次，至少会使dep减一
- 那么总的时间复杂度为$O(N \log N + Q)$

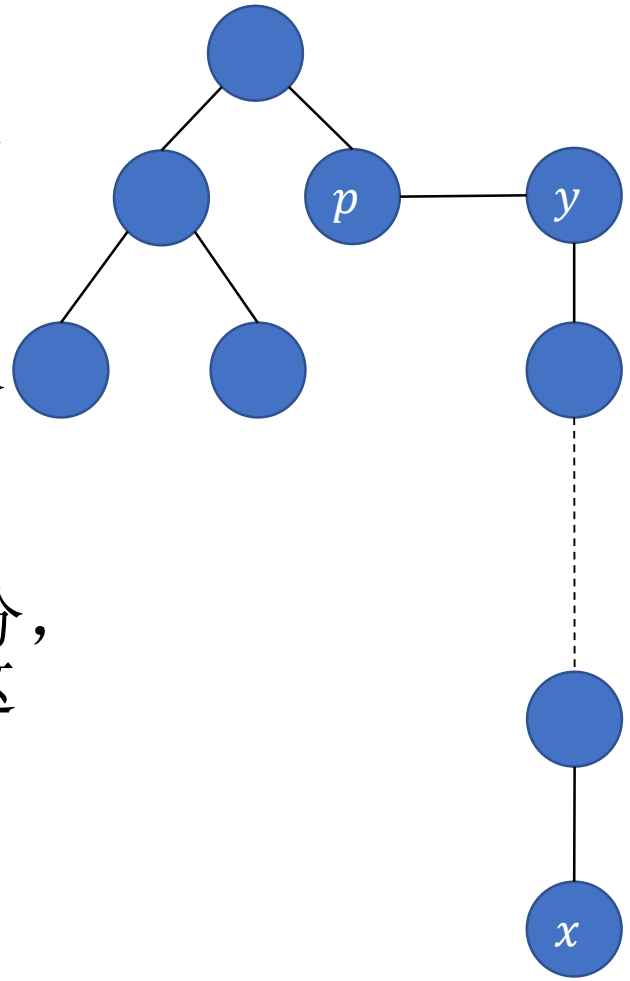
Natural Park

- 有一个N个点，M条边的无向图。每个点的度数≤ 7。每次你可以询问$(x, y, p[])$，交互库会返回只经过$p[]$中的点能否从x走到y。
- 最多询问45000次，求这张图。

- $N, M \leq 1500$

Natural Park

- 我们首先考虑如何解决一个图是一棵树的问题。
- 假设我们现在已经得到了一棵以1为根的树，我们尝试加入点x
- 注意到在一棵树的bfs序中，前k个节点一定构成了一个连通块。那么我们在bfs序上可以通过二分查找得到一个点p，满足x可以通过一系列不在树上的点走到p。
- 若p和x不直接相连，那么我们在x到p的链上二分，直到找到一个和p直接相连的点y。我们把y加入这棵树即可。
- 询问数为$2N\log N$



Natural Park

- 注意到题目中有一个条件：每个点的度数≤ 7。
- 根据询问数45000的限制，我们可以大胆猜测最后的复杂度为$7N + 2N \log N$

- 考虑在树的基础上增加一个操作：
- 每当我们找到一条边(p, y)，我们不先把y加入到图中。我们将p点从图中去掉，对于剩下的每个连通块（至多7个），我们先判断是否存在和y直接相连的边，如果存在我们就递归处理这个连通块。

Long Distance Coach

- 有一辆客车从0前往X，车速为每秒走一个单位长度。途中有N个补给站位于$S_1, \ldots, S_N$。在起点或补给站可以给客车的饮水机加水，每一单位的水价格为W
- 起初车上有M个乘客，第i个乘客会在时刻$D_i + T \times k$喝一单位的水，如果此时饮水机没有水了，这位乘客就会下车并且索要C_i的赔偿费。不会有人在补给站的时候需要喝水。
- 司机会在所有$T \times k$的时刻喝一单位的水，此时饮水机必须要有水。
- 问如何分配在起点和补给站加水的量，使得总的花费最小

- $N, M \leq 200000$，$X \leq 10^{12}$，$0 < D_i < T$且D_i互不相同

Long Distance Coach

- 首先假设$S_1 < \cdots < S_N$，$D_1 < \cdots < D_M$
- 我们称一个人不会因没有水喝而下车为“选择”这个人

- 一个显然的贪心：
- 如果我们确定一个人不被选择，那么我们一定让他尽量早地下车

Long Distance Coach

- 考虑两个人$D_x < D_y$，如果我们不选择x而选择y，那么必然存在一个补给站S_i和k，满足$D_x + T \times k < S_i < D_y + T \times k$
- 而如果x在此时下车，所有满足$D_x + T \times k < D_z + T \times k < S_i$的人$z$都必须下车。即对于补给站$S_i$，如果不选择$x$，那么任意满足条件的$z$都不能选择。

- 因此假设我们$x, x+1, \ldots, y$都恰好在补给站S_i之前下车，那么需要满足：
- $D_y + T \times k < S_i$，$D_{y+1} + T \times k > S_i$
- 也就是$D_y < S_i \% T < D_{y+1}$

Long Distance Coach

- 令g_i, f_i表示考虑了人$i\sim n$，并且不选择/选择i的最小代价
- 定义函数$h(i,j,k)$为$i\sim j$在$(Tk, T(k+1))$下车的代价
- 我们考虑枚举一个补给点$S_k \in (Tx, T(x+1))$，我们假设i在S_k处下车，我们可以得到一个转移方程：
- $g_i = \min(g_{j+1}, f_{j+1}) + h(i,j,x)$
- 其中j满足$D_j + Tx < S_k < D_{j+1} + Tx$即$D_j < S_k\%T < D_{j+1}$

- 我们令$F_j = \min\{\left\lceil \frac{S_k}{T} \right\rceil$，$S_k$满足$D_j < S_k\%T < D_{j+1}\}$，现在改为枚举$j$，那么得到：
- $g_i = \min(g_{j+1}, f_{j+1}) + h(i,j,F_j)$

Long Distance Coach

- $g_i = \min(g_{j+1}, f_{j+1}) + h(i, j, F_j)$
- 如果F_j是有序的，这个问题很容易通过斜率优化+单调栈解决
- 考虑$i < x < y, F_x < F_y$，根据贪心的原则，y显然不能为最优决策点
- 形式化的证明：
- 由$g_{x+1} \leq \min(g_{y+1}, f_{y+1}) + h(x+1, y, F_y)$
- $g_{x+1} + h(i, x, F_x)$
- $\leq \min(g_{y+1}, f_{y+1}) + h(x+1, y, F_y) + h(i, x, F_x)$
- $< \min(g_{y+1}, f_{y+1}) + h(x+1, y, F_y) + h(i, x, F_y)$
- $< \min(g_{y+1}, f_{y+1}) + h(i, y, F_y)$

Long Distance Coach

- 那么我们在从后往前dp的时候，维护F_j使得它保持有序，剩下的用斜率优化+单调栈解决
- 时间复杂度为排序复杂度$O(N \log N)$
- 如果不维护F_j，也可以考虑cdq分治优化dp，复杂度不变

- 注意两个分子分母范围都为$long\ long$的分数大小比较

Abduction 2

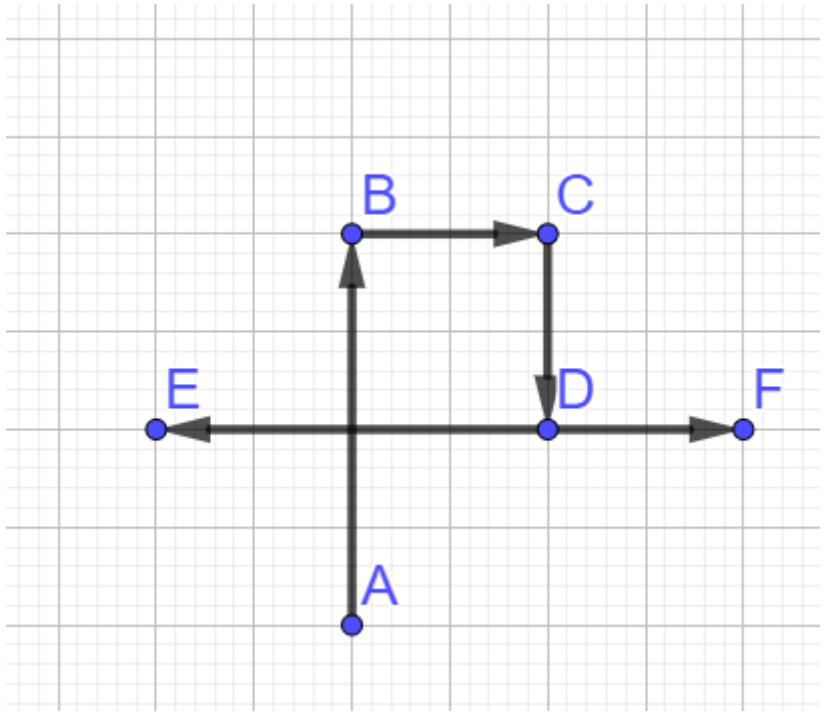
- 有$W+H$个街道，其中W条平行y轴，H条平行x轴，他们相交形成$W\times H$个十字路口。每条街道有一个重要度，重要度两两不相同。
- 给定一个起点，开始时你可以选择任意一个方向并顺着街道走下去，如果在十字路口拐弯可以走到重要度更大的街道就拐弯，方向随意。如果走到尽头那么就停止行走
- Q次询问一个起点，求走到尽头的最长路的长度

- $W,H\leq 50000$，$Q\leq 100$

Abduction 2

- 考虑单次询问，我们用$solve(x,y,k)$表示当前位于(x,y)并且(x,y)是一个拐点，方向为k的最长路
- 经过预处理我们可以很容易地用倍增在$O(\log N)$的时间复杂度内找到下一个拐点，然后暴力枚举方向。
- 这么做的时间复杂度看似是$O(N^2 \log N)$，但实际上状态数只有$O(N)$个。

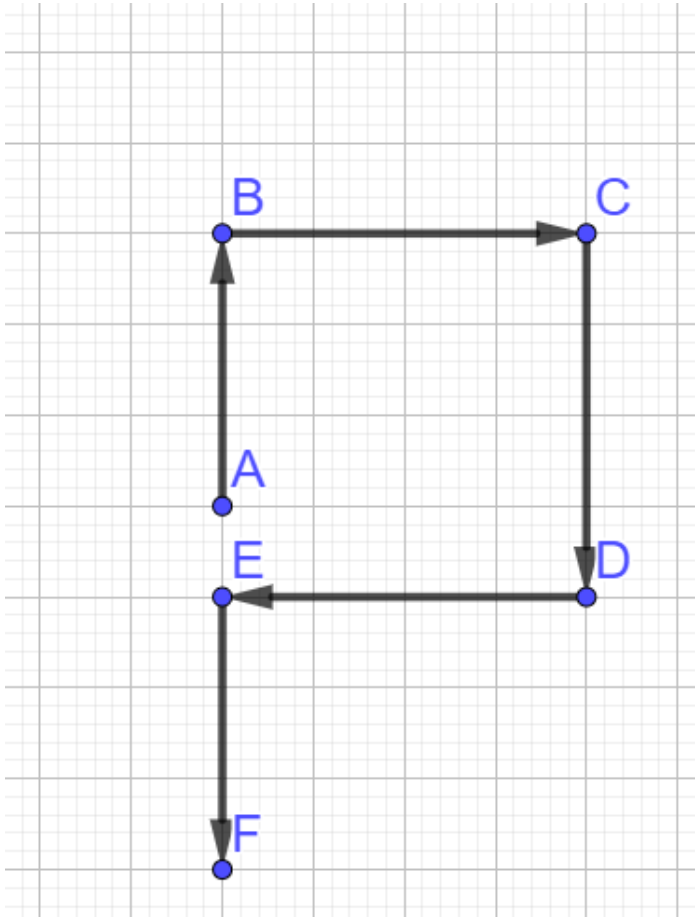
Abduction 2



- 如图所示的$AB \to BC \to CD \to DF$情况不会出现
- 对于重要度，我们有$DF > CD > BC > AB$
- 所以AB在经过和EF的交点时就会转弯
- 那么我们可以发现走过的街道只能不断远离原点

- 并且我们可以发现AB和DE也不能同时出现，即走过的街道只能在拐点处相交

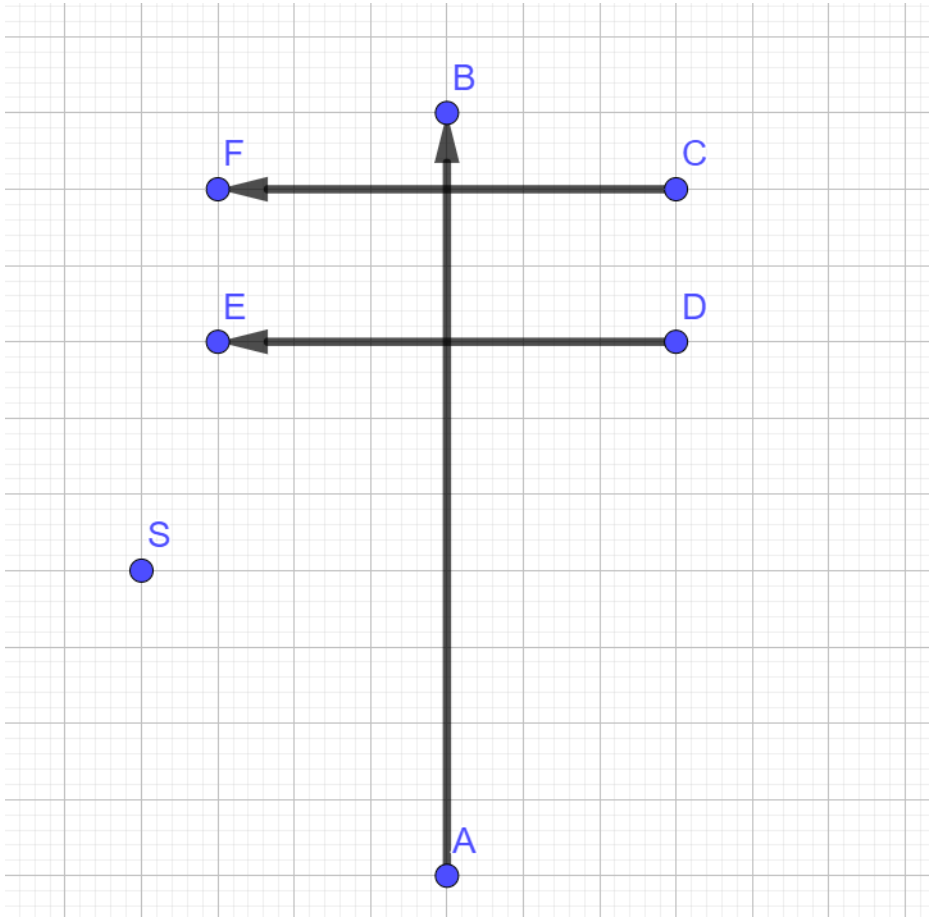
Abduction 2



- 考虑左图的情况，对于重要度我们有：
- $EF > DE > CD > BC > AB$
- 由$EF = AB$可以得到矛盾

- 因此同一条街道不会被经过两次
- 这种情况可以认为是之前情况的特殊形式

Abduction 2

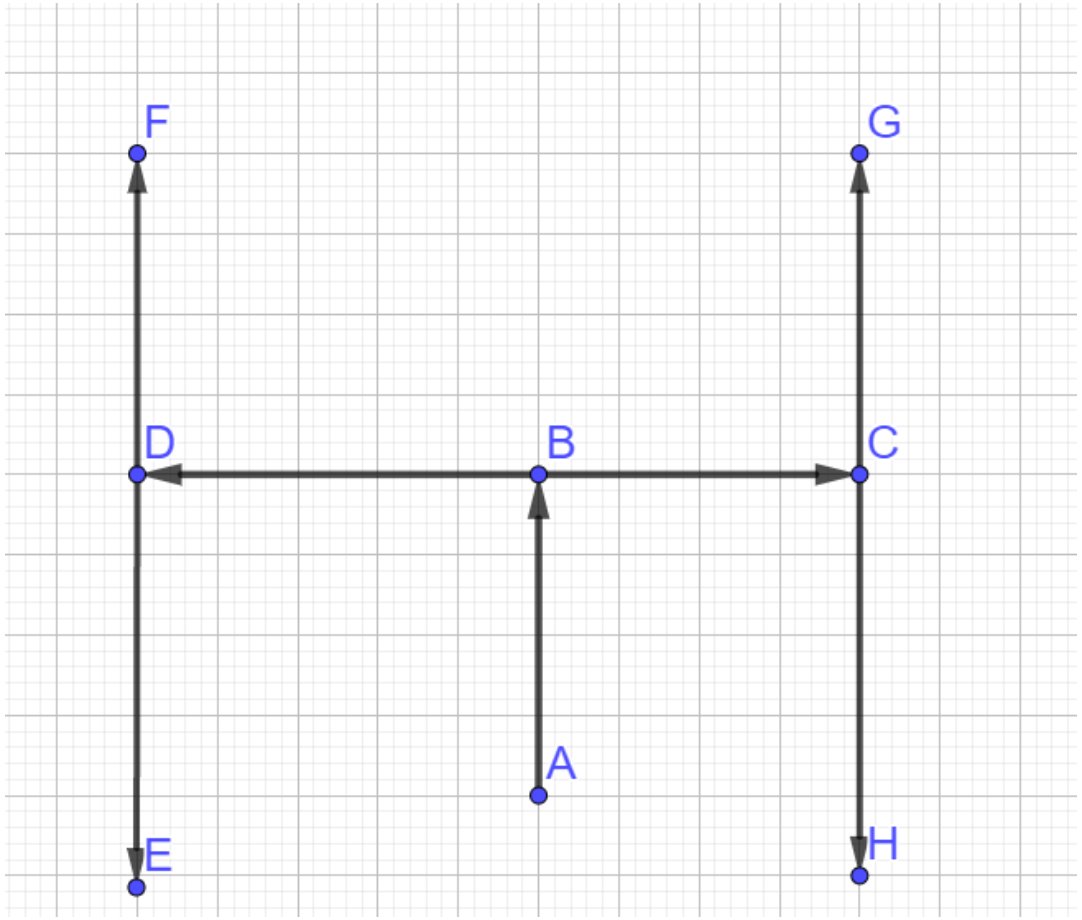


- 考虑如图的情况，ED为S上方第一条重要度$>AB$的街道
- 那么即使CF比DE的重要度高，$AB \cap CF$也不会成为拐点（因为不会出现重新回到AB的情况）
- 所以每个街道最多只会有两个拐出点（对于一条竖直街道，拐出点分别在S上方和S下方）
- 因此单次询问的状态数为$O(N)$

Abduction 2

- 考虑进行Q次询问，如果记忆化，复杂度能否低于$O(NQ)$
- 对某一个询问，考虑两条平行且位于起点的同一侧的街道X,Y，满足X比Y更靠近起点且X的重要程度比Y大，那么在这次询问中显然不会进入街道Y，我们可以忽略街道Y
- 对于这次询问，我们将所有不会被忽略的街道按照重要程度从小到大排序，那么如果排序后两个在同侧的平行街道之间没有一个垂直的街道，较大的那个平行街道也可以被删去
- 那么前$\frac{N}{\sqrt{Q}}$条街道中竖直和水平的街道数量一定是$O(\frac{N}{\sqrt{Q}})$的
- 加下来分两种情况讨论

Abduction 2



- 考虑$\frac{N}{\sqrt{Q}}$之后的某条街道CD
- 由于前$\frac{N}{\sqrt{Q}}$条街道中竖直的街道数量是$O(\frac{N}{\sqrt{Q}})$的，因此CD的长度至为$O(\frac{N}{\sqrt{Q}})$
- 而且CD之间的点不会再被水平方向上访问
- 由于总点数是N^2，因此对于水平街道，复杂度为$O(N\sqrt{Q})$
- 竖直街道同理可证

Abduction 2

- 考虑排名在$\frac{N}{\sqrt{Q}}$之前的街道
- 由于每条街道只有两个拐出点，因此每次询问状态数为$O(\frac{N}{\sqrt{Q}})$
- 总共Q次询问，因此这一部分的总状态数也是$O(N\sqrt{Q})$

Dragon 2

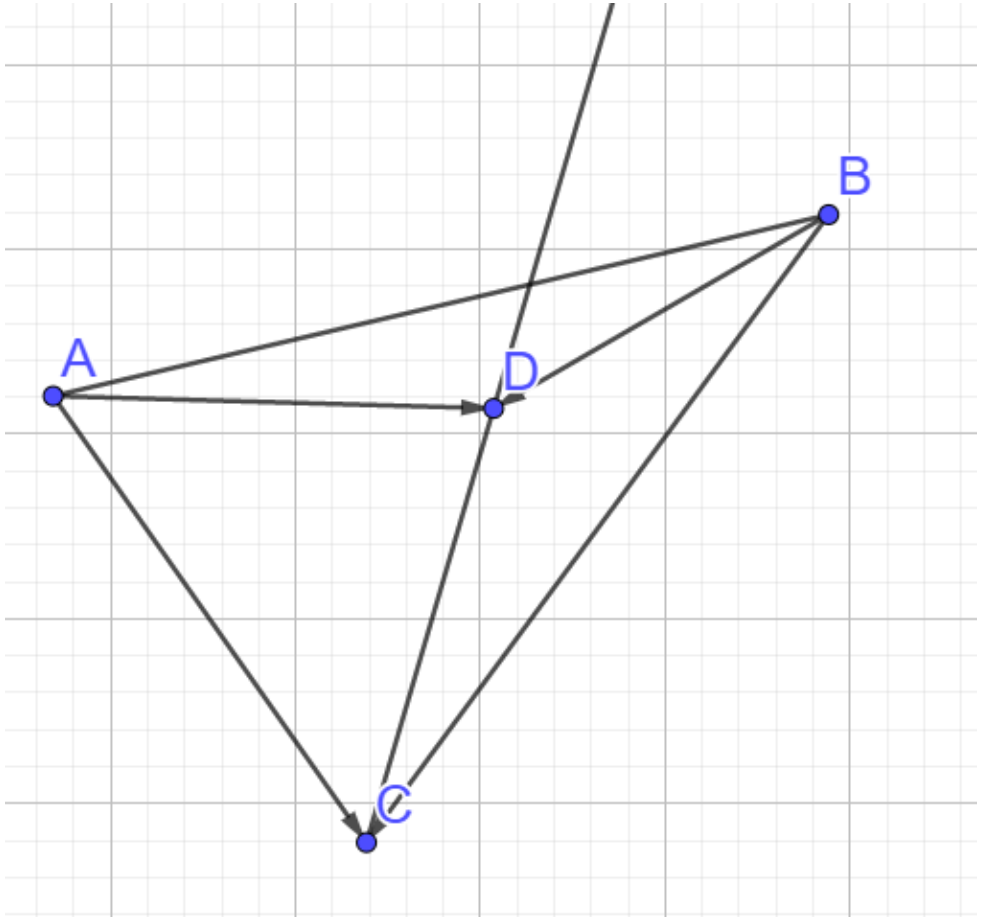
- 有N条龙，第i条龙坐标为(A_i, B_i)，属于第C_i个部落
- 有两个人类的村庄$(D_1, E_1), (D_2, E_2)$
- 接下来有Q次询问，每次询问给出F, G，表示所有属于F的龙向属于G的龙作射线，问与线段$(D_1, E_1) - (D_2, E_2)$的交点个数
- 不存在三点共线

- $N, Q \leq 30000$

Dragon 2

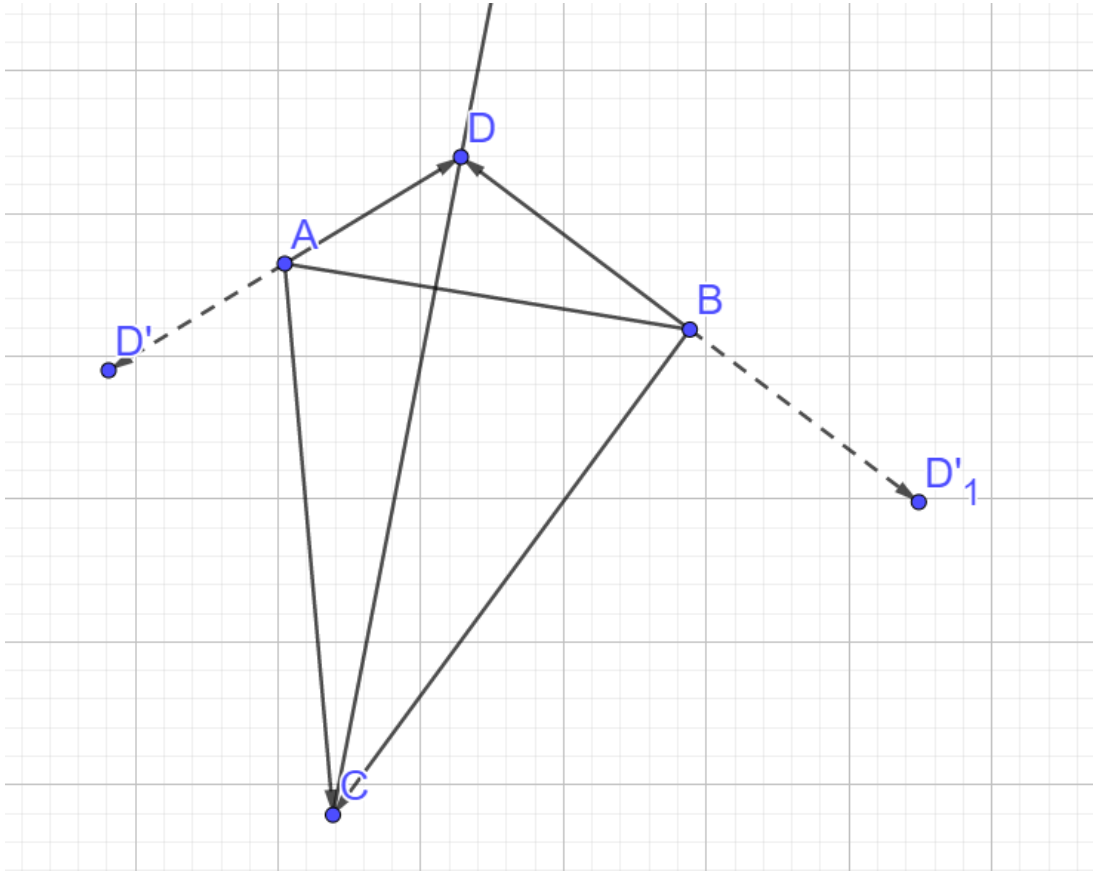
- 我们将龙抽象为点
- 首先观察一下两个点所成的射线和线段相交有什么性质

Dragon 2



- 首先来看这种情况，即C, D都在$\overrightarrow{AB}$的右侧
- 对所有的点P，我们把向量$\overrightarrow{AP}$排序，排序的关键字是$\overrightarrow{AB}$绕A点逆时针旋转得到$\overrightarrow{AP}$所需要的角度，我们假设排序后点P对应的向量的排名为a_P
- 对B同理得到b_P
- 那么在左图的情况下，射线CD与线段AB相交的条件为$a_D > a_C$且$b_D < b_C$

Dragon 2



- 再来看这种情况，即C在$\overrightarrow{AB}$的右侧，D在$\overrightarrow{AB}$的左侧
- 对所有的点P，作关于点A的对称点P'，对所有的$\overrightarrow{AP}$和$\overrightarrow{AP'}$ 按照和之前一样的方法排序
- 假设点P对应的向量$\overrightarrow{AP'}$排名为c_i，对B同理得到d_i
- 那么在这种情况下相交的条件为$c_D < a_C$且$d_D > b_C$

Dragon 2

- 另外两种情况和上面的两种情况对称，不再赘述
- 这样我们就将射线和线段的相交判定转化为二维偏序的判定
- 每次询问我们就是做一次二维数点

- 我们对所有包含点数$> N^{\frac{1}{3}}$的部落，预处理以这个部落作为F和G的答案
- 使用一定的分块技巧可以做到$O(N\sqrt{N} + N^{\frac{5}{3}})$的时间复杂度
- 其余询问我们暴力枚举，复杂度为$O(QN^{\frac{2}{3}})$

Dragon 2

- 我们可以对所有的部落预处理插入点的部分，但是对于点数$<\sqrt{N}$的部落我们不处理和它有关询问（即只修改，不询问）。同时我们将分块中插入点的过程可持久化，这样询问的时候暴力的复杂度可以由平方降为线性。
- 时间复杂度为$O(N\sqrt{N})$

- 由于没有强制在线，可以不需要可持久化，复杂度不变。

City

- 给定一棵点数为N的有根树，树的深度≤ 18。你需要给树上的每个节点一个编码。Q次询问时给你两个节点X,Y的编码，让你判断下列条件是否满足：
- 0）X为Y的祖先
- 1）Y为X的祖先
- 2）上述两个条件都不满足

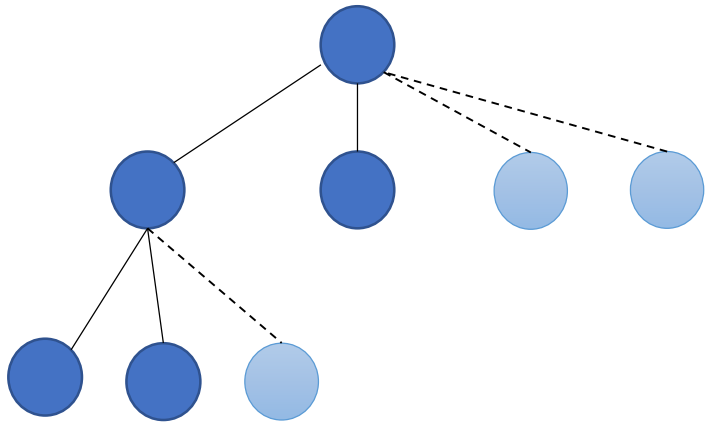
- $N,Q \leq 250000$，你的编码大小不能超过2^{28}

City

- 判断两个点是否具有祖先-后代关系，通常使用括号序列。即在dfs过程中，记录一个点入栈和出栈的时间。假设点x的入栈和出栈时间记为(L_x, R_x)
- 直接记录的话需要传$O(N^2)$的信息量，需要37~38位
- 我们发现L_x, R_x都不容易压缩，那么我们考虑压缩$R_x - L_x + 1$
- 定义一个常量ρ，和一个序列$A_i = \lceil \rho^i \rceil$，并对$A_i$去重
- 那么如果$R_x - L_x + 1$不在序列A中我们就添加虚点使它满足条件

City

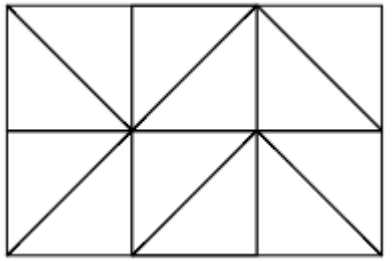
- 右图是一个$\rho = 2$的例子
- 显然加入虚点不会影响祖先-儿子判定



- 并且容易归纳证明：
- 若一棵树的总点数为N，深度为d，那么新的图的点数$\leq N \times \rho^d$
- 而$R_i - L_i + 1$可以被压缩到$\log_\rho(N \times \rho^d)$ 的状态数
- 总的状态数为$N\rho^d(d + \log_\rho N)$，取$\rho = 1.03 \sim 1.05$均可

以下是JOIsc2016的一些好题

Sandwich



- 有摆放成形如右图的$2 \times R \times C$个三明治
- 一个三角形代表一个三明治
- 一个三明治可以被拿走，需要满足下列两个条件之一：
- 1）这个三明治直角边对应的两个三明治都被拿走了（一开始就存在视作被拿走了）
- 2）这个三明治斜边对应的三明治被拿走了
- 对于每一个三明治，求出取走这个三明治至少要拿走多少三明治

- $R, C \leq 300$

Sandwich

- 一个显然的结论是：
- 如果某一步拿走了一块三明治且它斜边对应的三明治还没有被拿走，那么下一步一定拿走他斜边对应的三明治（除非当前三明治就是最终需要取走的三明治）
- 如果不这么做，那么拿走当前这块三明治不能对后续操作产生影响，拿走这块三明治没有意义
- 那么我们将拿三明治的过程倒过来，直接对每一块三明治dfs就能得出答案
- 不妨假设每个方格内先被取走的是直角边位于方格上边界的三明治，那么我们直接在同一列中上一行的dfs基础上求解，时间复杂度降为$O(N^3)$

Solitaire

- 给定一个3行N列的棋盘，上面一开始有若干棋子和空格，每次你可以在某一个空格放上棋子当且仅当它上下都有棋子或左右都有棋子。（超过边界算作没有棋子）
- 问有多少种不同的放棋子的顺序使得最后棋盘被填满

- $N \leq 2000$

Solitaire

- 对于第一行和最后一行的空格，要放棋子只能是左右都有棋子
- 因此第一行和最后一行不能有两个连续的空格

- 接下来考虑第二行的空格，有两种方法：
- 1）将上下两个空格（如果是空格）变成棋子，然后在这一格放棋子
- 2）在上下两格不全为棋子的前提下，左右两个都是棋子，然后在这一格放棋子
- 这样和原问题等价

Solitaire

- 显然如果第二行某一格为棋子，它会将原问题分成左右两个独立的子问题
- 对于每一个子问题，我们考虑dp，令$dp_{i,j,k}$表示从左到右考虑到i，其中i这个棋子是在第j个放的，k表示i这个棋子是用哪一种方法放的
- 用组合数处理转移，最后将所有子问题合并即可。

- 总时间复杂度显然为$O(N^2)$

祝大家在ZJOI中取得好成绩！

GL & HF!

一些链接

- JOIsc2017官方题目与题解地址：
- https://www.ioi-jp.org/camp/2017/2017-sp-tasks/index.html

- JOIsc2017在atcoder上的题面和评测地址：
- https://joisc2017.contest.atcoder.jp/

- JOIsc2018 Online Contest报名地址：
- https://cms.ioi-jp.org/joi-sp-2018/index-en.html