summary

$\mathcal{L}cy$

time:2017 年 10 月 31 日

目录

1 present

1.1 时间

2017.10.9

1.2 题目

现有 n 个物品，每一种物品有无限个，价值是 p_i
现有 m 个人，每个人有一个数字 a_i
求有多少个人的数字可以用 n 个物品的价值凑出来
$n \leq 500$
$m \leq 3 * 10^5$
$0 < p_i \leq 10^4$
$0 \leq a_i \leq 4 * 10^7$

1.3 算法

建模 + 最短路

1.4 细节

令最小的 p_i 记为 p_0
假如有一种方案可以用 p_i 凑出 $k*p_0+x$
那么对于一个数字，如果满足 $k*p_0+x \le a_i$ 且 $a_i\%p_0=x$
那么这个数字就可以被凑出来
那么可以用最短路的方法跑出每一个 x 而言，最小的 $k*p_0+x$(这样才能凑出更多的数字)
枚举 x,y，当满足 $x+p_i\%p_0=y$ 时，x 向 y 连一条长度为 p_i 边
但这样连边可能连出 10^8 条边 (因为 x,y 的大小是与 p_0 相关，而 p_0 最大有 10^4)
但是 n 很小，所以直接可以枚举 x,p_i 这样就变成了 x 向 $x+p_i\%p_0$ 连一条长度为 p_i 的边
这样最短路后，节点 x 的距离就等于 $k*p_0+x$ 的最小值
现在讨论为什么只需要使用 p_0
因为如果 $k=0$ 是存在的，那么 x 是一个比 p_0 小的 p_i，不满足 p_0 的条件
所以 $k*p_0+x$ 中的 k 一定满足 $k>0$
所以每一个 x 都一定是从 p_0 出发经过若干条边 (或不动) 到达的
又因为 p_0 最小，余数最少，可以使点数最少，所以使用 p_0

1.5 出处

Test20171009

2 The Untended Antiquity

2.1 时间

2017.10.11

2.2 题目

给定一个 $n*m$ 的矩形
有 q 个操作：

1. 给定一个矩形，沿这个矩形的边缘放置平行于边的屏障 (位于两个矩形格子之间)
2. 将一个存在的屏障移除
3. 询问两个矩形格子上的点是否能够不出矩形且不碰触屏障的相互到达

保证对于任意时刻，所有同时存在的不同屏障不相交 (即既不点重合也不边重合)
$n, m \leq 2500$
$q \leq 10^5$

2.3 算法

二维树状数组 + 哈希

2.4 细节

一个放置屏障的操作就相当于在这个矩形上放置了一种颜色
如果是一个矩形套矩形就把小的矩形的颜色保留 (即面积小的矩形有优先权)
但是删除就非常不好处理了
因此可以考虑把覆盖操作换成加法操作
考虑给矩形随机加上一个值并记录下来，删除的时候减去这个值
这本质上是一种哈希，因为这样还是比较容易被卡掉，因此可以用双哈希来解决
二维树状数组可以实现
二维树状数组就相当于一维的写法套上一维的写法
询问修改的时候差分一下即可

2.5 出处

The Untended Antiquity

3 Huge Strings

3.1 时间

2017.10.11

3.2 题目

给定 n 个 01 字符串

执行 m 个操作，每个操作是将两个已经有的 01 字符串合并成一个新的 01 字符串

第 i 个操作是将 s_{a_i}, s_{b_i} 合并成一个新的串 s_{n+i}(操作编号从 1 开始)

每次操作后，你需要找到最大的 k 满足所有长度为 k 的 01 字符串都是新串的子串，如果不存在输出 0

$n, m \leq 100$

$1 \leq |s_i| \leq 100$

3.3 算法

结论

3.4 细节

这一题很巧妙，答案都不会超过 9

假设一个答案是 k，我们目前只考虑如何得到 2^k 种不同的 01 串

那么当我们合并两个串的时候，新生成的长度 k 的串只可能由两个串分别构成（即至少长度为 1）且总长度为 k

那么两个串合并后最多只有 $k-1$ 个新的长度为 k 串生成，所以最坏情况下只有 $n+m*k$ 种

如果 k 是合法的答案，那么就要满足关系式 $n+m*k >= 2^k$

解出来当 k=10 的时候刚好不满足不等关系，所以答案最大只有 9

所以我们只要记录每个串的前后 10 位的信息以及 9 位及以下的字符串是否在该串中出现，然后询问的时候暴力枚举答案并且判断皆可以了

因为串 001 和 01 是不同的，但是我们又是使用 bool 或者 bitset 判断，所以对于所有的长度为 k 的串我们在它们的前面或上 $1 << k$ 即可，并且不会对别的统计造成影响（因为每一种长度都这么做了）

3.5 出处

Huge Strings

4 starway

4.1 时间

2017.10.12

4.2 题目

有一个长方形通道 (顶点在 (0,0) 和 (n, m))
现有一人从左边任意一点进入，从右边任意一点走到终点
最左最右的距离为 n, 上下边界的距离为 m
长方形中有个 k 个 $Star$，每个 $Star$ 都有一个整点坐标，其大小忽略不记
求走到终点的路径上，距离所有 $Star$ 以及上下边界的最小距离的最大值
$n, m \leq 10^6$
$k \leq 6000$

4.3 算法

最小生成树

4.4 细节

考虑现在假定的一个“答案”:x

那么不能走的区域就是矩形底端往上 x 距离部分的矩形，和顶端往下 x 距离部分的矩形，以及以所有的星星为圆心作半径为 x 的圆的范围

考虑矩形顶端底端分别为一个星星

若不能走的区域有重叠部分，则称这两个区域或者这两个星星连通，那么在这种情况下，如果顶端底端连通，则说明不存在路径可以通过，那么这个“答案”是不合法的

根据这个思想我们知道答案一定是某两个星星之间距离的一半，考虑每两个星星之间的边长度为距离的一半，假设某一条边，那么对于所有长度小于它的边我们都可以使用（即连通两个端点），这时候如果顶端底端刚好连通（即减小一点就不连通），就说明这条边是答案

我们考虑，如果这条边比答案大，那么就不是刚好连通，如果比答案小，那么就不连通，那么这条边恰好是最小且满足要求的

所以考虑 MST，MST 是找到一条最小且能够连通两个块的边，结合之前的讨论，这是同样的道理，故 MST 就可以帮助我们找到这条边，根据 MST 的性质，这条边就是 MST 上顶端到底端路径上的最长边

图论的构图不要因为它是题目的限制条件而单纯认为它在图中只能做限制条件而不能做点

4.5 出处

Test20171012

5 lost

5.1 时间

2017.10.12

5.2 题目

给定一棵以 1 为根的树

每个点上有一个权值 c_i，边权都是 1

对于任意一个点 u，它的一个祖先 v，v 对 u 的贡献为:

$\frac{c_v-c_u}{dist(u,v)}$

求除根外每一个点能够得到的最大贡献

$n \leq 5 * 10^5$

$c_i \leq 10^9$

5.3 算法

维护凸包

5.4 细节

$maximization \frac{c_v-c_u}{dist(u,v)} = -minimization \frac{c_u-c_v}{d_u-d_v}$

这就是一个斜率的式子

这种类似的东西可以用凸包维护

5.4.1 基本凸包的介绍

考虑让斜率最小化

考虑凸包上已经有三个点

按照横坐标从小到大分别是 a, b, c，之间的关系满足 $slope(a, b) < slope(b, c)$

考虑 b 是否对于所有横坐标大于它的点都没有影响 (因为加点的顺序是按照深度递增，相当于横坐标递增)

考虑一个在 c 右侧的点 x，如果 x 在 c 上方，那么一定有 $slope(c, x) < slope(b, x)$，如果 x 在 c 下方，那么一定有 $slope(a, x) < slope(b, x)$

因此如果是上述情况，b 对后面入队的点永远没有贡献，因此可以弹掉它

这样图像上来说，就维护出了一个上凸包

此时考虑凸包右边有一个点，发现它与凸包上从左到右的点的构成的直线的斜率是一个单峰函数，且是先上升后下降

因此维护出凸包后要用凸包上的点更新凸包右侧点的答案时，直接在凸包上二分即可

二分的时候用相邻的两个元素判断此时二分到的点位于凸包的上升还是下降区间

5.4.2 可持久化栈

对于栈内的每一个元素记录它在栈中的前一个元素即可

如果需要二分弹栈就使用倍增的方法，记录每一个元素它在栈中的前面第 2^i 个元素

5.5 出処

Test20171012

6 Points, Lines and Ready-made Titles

6.1 时间

2017.10.16

6.2 题目

给定平面上 n 个不同的整点

对于每个点你可以过它画一条平行于 x 轴的直线，或者一条平行于 y 轴的直线，或者什么都不做

几条重叠的直线相当于一条，求能得到多少个不同的图，输出模 10^9+7

保证点不同

$n \leq 10^5$

$-10^9 \leq x_i, y_i \leq 10^9$

6.3 算法

构图计算

6.4 细节

考虑每个点与它上下分别第一个 (或不存在) 横坐标相同的点或左右分别第一个 (或不存在) 纵坐标相同的点之间有一条连边

对于每个连通块单独考虑

令连通块里面不同的横线有 x 条，不同的竖线 y 条

如果这个连通块是一棵树那么这个连通块的图有 $2^{x+y}-1$ 种

否则这个连通块的图有 $2x+y$ 种

证明：

考虑连通块是树

那么当画了其中的一条横线时，必须要找一个点作为承担点 (即由它画出)

此时该横线上的其它点可以任意地选择画竖线或者不画

而这个点所在的竖线如果想要画出来就必须再找一个该竖线上的其它的点作为承担点

以此下去总会到一个点导致它的一个方向的线是画不出来的

但是与此同时别的点所管辖的线都是有方案将其画出的

因此这个连通块不能同时画出所有的横线与竖线，但其它的图都可以画出来

所以方案数为 $2^{x+y}-1$

考虑连通块不是树

那么必定存在一个正方形的环

那么对于环的四个角上的点，我们可以找到一种简单的方案使得画出这四个角管辖的两个横线和两个竖线，此时别的点就可以选择画另一个方向或者不画

因此此时是可以将所有的横线竖线同时画出

所以方案数是 2^{x+y}

6.5 出处

Points, Lines and Ready-made Titles

7 题目名称

7.1 时间

2017.10.17

7.2 题目

一个 n 个点 m 条边的无重边无自环的无向图

点有点权, 边有边权

定义一条路径的权值为路径经过的点权的最大值乘边权最大值

求任意两点间的权值最小的路径的权值

$n \leq 500$，$m \leq n * (n-1)/2$

点权和边权 $\leq 10^9$

7.3 算法

Floyd

7.4 细节

因为有两个 max，所以可以考虑用一种方法限制一个，再用另一种方法弄出另一个

考虑从小到大枚举点权（这样可以遍历所有点路径的最大点权的情况）

将所有点权小于枚举的点权的点保留，并且对应地求出每一个点权下保留点之间的路径边权的最小值

比较好想的是用最小生成树实现，但是只有 50 分

考虑枚举点权的时候，每次增加点权，就相当于将一个点加入图中

此时的变化只有这一个点，所以能被更新的所有信息都是该点能更新的信息

考虑到 Floyd 就是相当于将点以此插入图中，更新图中的路径，所以直接上 Floyd

外层枚举 k 的时候按点权大小即可。

7.5 出处

Test20171017

8 segment

8.1 时间

2017.10.17

8.2 题目

现有一颗线段树维护了三种操作：

1. 区间加上一个数 x
2. 区间赋值为一个数 x
3. 求区间和

求把所有的操作随机打乱，每个询问的期望输出
随机打乱指 $m!$ 种操作排列的出现概率相同
题目保证询问都在最后且不参与随机打乱
$n, m, q, a_i \leq 10^5$
1 操作中 $x \leq 100$
2 操作中 $x \leq 10^5$

8.3 算法

数学

8.4 细节

考虑每一个点的最终的期望值（因为不同点之间没有依赖关系）

先考虑存在覆盖操作的情况（覆盖了当前考虑的点的覆盖操作）

此时考虑只有覆盖操作，那么起作用的只有最后一个操作，而每个操作出现在最后的概率是相同的，因此覆盖操作对期望值的贡献是 $\frac{\sum cover_i}{k}$，cover 表示覆盖操作的值，k 表示覆盖操作的个数

再把加法操作考虑进来，把加法操作的位置考虑是插入到覆盖操作之间的，那么有 k+1 个位置允许插入，但只有出现在最后一个覆盖操作后才有贡献

又因为对于每个加法操作而言，出现在最后一个位置的概率为 $\frac{1}{k}$，所以加法操作的总贡献为 $\frac{\sum add_i}{k+1}$，add 表示加法操作的值

如果没有覆盖操作就要把初始值考虑为一个加法操作。

8.5 出处

Test20171017

9 Delivery Club

9.1 时间

2017.10.17

9.2 题目

现有两个人，一个站在点 $s1$ 上，一个人站在点 $s2$ 上
现有 n 个点需要严格按顺序访问，第 i 个访问的点位于 x_i 上
每次访问可以由两个人中的任意一个去访问，访问结束时去访问的人在对应的 x_i 上
当一个人去访问一个点时，另一个人在原地不动
要求最小化，从初始到访问完第 n 个点的过程中，这两个人的最大距离
求这个值
$n \leq 10^5$
$0 \leq s1, s2, x_i \leq 10^9$

9.3 算法

二分答案

9.4 细节

二分答案
判定一个答案 x 是否可行
用一个 set 维护当前状态下可以到达的点的位置
首先加入合法的 $s1$，$s2$(即对于 x_1 是合法的)
考虑下一个走到的点是 $i(i \in [1, n])$
对于 set 中与 i 相距 $> x$ 的点全部删掉
如果集合为空则无法走到该点，即 x 是不可行的
否则加入点 i，继续上述操作直至结束 n 个点 (x 可行) 或集合在中途为空
考虑为什么这样维护的集合是正确的：
因为现在二分答案 x
那么对于所有两个人相距超过 x 的情况全部要删除
现在有一个人要走到 i，那么另一个待在的位置一定不能与 x_i 相距超过 x
删点操作就是维护这样的一种情况
所以现在只要讨论为什么 set 中只要存在元素即可对于当前合法
以下考虑的情况都是在此次二分的答案是合法的基础之上
根据上面的删点操作，set 中任意两点之间的距离都是 $\leq x$ 的
先考虑一次删点操作进行之前的 set
因为至少有一个元素来自之前的 set，同时有一个新的元素 (即上一次访问到的点) 加入 set
所以对于任意完成加点后的合法时刻，set 中一定至少含有两个元素
那么可以让一个人到当前新加的元素上面 (因为要求到达这个点)，另一个人的位置暂时随意 (即可以选择非新加元素的任意一个元素上)
此次删点操作操作后至少会剩余一个元素

如果这个元素是上一轮加入的，那么可以让上一轮走到该元素上的人不动，另一个人走到当前新加入的元素上

如果这个元素不是上一轮加入的，那么可以让位置随意的那个人走到这个元素上，让走到上一轮元素上的人走到这一轮的元素上

这样做就可以保证两个人在移动过程中的距离不会超过 x

因此只要 set 里面存在元素就一定可以当前合法 (即走到 i 的时候仍然合法)

9.5 出处

Delivery Club

10 Royal Questions

10.1 时间

2017.10.18

10.2 题目

现有 n 个王子和 m 个公主
每个公主喜欢两个不同的王子，对于第 i 个公主，喜欢的王子为 a_i, b_i
每个公主有一个聘礼 c_i，当这个公主出嫁的时候，你会获得这份聘礼
但是公主仅能嫁给她喜欢的王子，每一个王子可以娶任意一个公主
一个公主至多只能嫁给一个王子，一个王子至多可以娶一个公主
现在由你安排婚事使得获利最大
$n, m \leq 2 * 10^5$
$a_i, b_i \leq n, a_i \neq b_i, c_i \leq 10^4$

10.3 算法

贪心 + 并查集

10.4 细节

考虑从大往小选公主，能选就选。
证明：
如果当前能选但不选，说明她可以匹配的王子要被后面小的公主选，但这显然是不优的，因为她可以直接替换一个小的公主
如果当前不能选，那么说明之前更大的已经选了，无影响
实现方法：
考虑公主为带权边，王子为点
考虑图如果连成了一棵树，那么任意能且仅能一个点可以空闲出来（即不和边匹配）
当前考虑当前公主 (边) 和她喜欢的两个王子 (点)
如果连接两个点所在的联通块中至少有一个是树，那么这个边一定可以得到匹配，否则不行
因此从大往小连边并判断当前边是否能够得到匹配即可

10.5 出处

Royal Questions

11 Fire

11.1 时间

2017.10.19

11.2 题目

现有一场火灾发生

有 n 个物品，每个物品有营救需要花费的时间 a_i，被烧毁的时间 b_i(营救完成的时间必须 $< b_i$)，营救该物品得到的价值 c_i

现在从 0 时刻开始

你可以选择营救哪些物品

求可以得到的最大价值

$n \leq 100$

$a_i \leq 20$

$b_i \leq 2000$

$c_i \leq 20$

11.3 算法

排序 + DP

11.4 细节

最终的灭火方案这样一定不会差：

从 b_i 小的开始营救

因此可以根据 b_i 排序

然后用 $f_{i,j}$ 表示考虑前 i 个物品到 j 时一共可以挽救的最大价值

排序使得 DP 不存在后效性了

11.5 出处

Fire

12 Buy Low Sell High

12.1 时间

2017.10.19

12.2 题目

你已经知道了一种货物接下来 n 天的价格
你希望最大化利润，但是每天只想进行一个货物的交易
也就是说，每天你可以买一个货物，或者卖出一个货物，或者什么都不做
一开始你没有货物，第 n 天结束的时候你希望也没有货物，但是希望得到最大的利润
求最大的利润
$2 \leq n \leq 3 * 10^5$
$1 \leq$ 价格 $(p_i) \leq 10^6$

12.3 算法

堆 + 思维

12.4 细节

0 = -x + x
此题以用一个大根堆
从 1 到 n 考虑当前的操作
先插入两个-p_i，再取出堆顶元素
插入两个分别表示中转和购入
如果当前的货物进行卖操作不优，就会导致堆顶取出-p_i，此时堆中还有一个-p_i 表示当前的货物可以充当买入的货物
如果当前的货物进行卖操作暂时较优，就会堆顶取出一个-p_i 更小的货物作为买入，此时-p_i 会留下两个
如果当前的货物在堆中留下了两个-p_i
那么当后面的货物使用了它的第一个-p_i，表示货物 i 与之前货物 x(-p_x 被 i 取出) 进行的买卖操作变成了货物 j(-p_i 被 j 取出) 和 x 的买卖操作
如果使用了它的第二个 p_i，说明后面的货物将这个 p_i 作为买入的货物
因此这样维护的堆可以求出最大利润

12.5 出处

Buy Low Sell High

13 叶则杨的题

13.1 时间

2017.10.19

13.2 题目

有一个 DAG.

这个 DAG 有 n 层, 第一层只有一个源点, 最后一层只有一个汇点, 剩下的 n 层每一层都有 k 个节点.

定义取反操作:

给出 i, 取反 i 层与 $i+1$ 层, 即把原本从 $(i,\ k_1\ )$ 连到 $(i+1, k_2)$ 的边, 变成从 (i, k_2) 连到 $(i+1, k_1)$.

题目中会给出 m 个操作, 分成以下几种格式:

1. 给出 i, 以及 k 2 个数, 第 $(j-1)\times k+t$ 个整数表示 (i, j) 到 $(i+1, t)$ 有没有边. 把 i 到 $i+1$ 层的边修改成这 k_2 确定的状况.

2-1. 询问起点到终点的方案数.

2-2. 给出 l, r, 询问将 l 层到 r 层所有的边取反后起点到终点的方案数.(询问后不用修改, 也就是说这个操作不会改变原图.)

输出答案时对 998244353 取模

$n \leq 10^5$

$k \leq 5$

$m \leq 10^4$

13.3 算法

线段树 + 矩阵

13.4 细节

矩阵转置相关计算：$A^T * B^T = (B * A)^T$

其中，A 为矩阵，A^T 为 A 的转置矩阵

邻接矩阵的性质：对于一个邻接矩阵 A，A^x 表示经过 x 条边后的邻接矩阵

比如此题，i 到 $i+1$ 之间的邻接矩阵为 A，$i+1$ 到 $i+2$ 之间的邻接矩阵为 B

$A * B$ 表示的矩阵为 i 各点到 $i+2$ 各点之间的方案数

用线段树维护这些矩阵的乘积

第 i 个叶子节点保存的信息为 i 到 $i+1$ 之间的邻接矩阵

合并信息的时候，令左儿子的矩阵为 A，右儿子的矩阵为 B

则当前节点的矩阵为 $A * B$

第一个操作直接到叶子节点修改

第二个操作直接取根节点的矩阵信息

第三个操作相当于求

$\prod_{i=1}^{l-1} A * \prod_{i=l}^{r} A^T * \prod_{i=r+1}^{n} A$

那么可以维护两颗线段树

一颗维护 $\prod_{i=l}^{r} A$，一颗维护 $\prod_{i=r}^{l} A$
根据转置的计算，询问的时候分别取出相乘得到答案

13.5 出处

share1003

14 胡旭林的题

14.1 时间

2017.10.19

14.2 题目

这里有数量分别为 A,B,C 的三堆石头, 每次操作将 A 堆石头的数量各分一半到 B,C 堆, 对于 B,C 也是如此操作,

求至少操作多少次后使得其中有一堆的石头数量为奇数.

$A, B, C \leq 10^9$

14.3 算法

模拟

14.4 细节

模拟可行的证明:

操作最多 log 次:

一次操作后，三堆石头的数量为

$\frac{B+C}{2}, \frac{A+C}{2}, \frac{A+B}{2}$

假设 $A \geq B \geq C$

那么操作后最大的是 $\frac{A+B}{2}$，最小的是 $\frac{B+C}{2}$

操作前最大值-最小值 $=A-C$

操作后最大值-最小值 $=\frac{A-C}{2}$

因此每次操作后最大最小值的差值都会小一倍

当 $A=B=C$ 时，无论怎么操作三堆的石头数量都不会改变了

结合每次最大最小值的差值的变化，操作最多有 log 次

14.5 出处

share1004

15 彭力的题

15.1 时间

2017.10.19

15.2 题目

初始有一个长度为 n 的序列, 进行 m 次操作

- 修改一个位置上的数
- 查询 l, r 区间内每个值出现的长度之和

一个值在某个区间内出现的长度定义为这个值最后一次出现的位置与第一次出现的位置的差

$1 \leq n, m \leq 10^5$

15.3 算法

思维题

15.4 细节

假设已经统计好了每个区间的答案

考虑一次修改对哪些区间的答案造成了影响

假设这一次修改的位置是 x，它的数值由 a 改变成了 b

与 a 相同的上一个数字是 $l1$，下一个数字是 $r1$

与 b 相同的上一个数字是 $l2$，下一个数字是 $r2$

那么修改了 x 位置的值后

左端点在 $[l1+1, x]$ 右端点在 $[r1+1, n]$ 的区间的 a 的第一次出现的位置发生了变化，从 x 变成了 $r1$，这些区间的答案需要 $+x-r1$

左端点在 $[1, l1-1]$ 右端点在 $[x, r1-1]$ 的区间的 a 的最后一次出现的位置发生了变化，从 x 变成了 $l1$，这些区间的答案需要 $-x+l1$

左端点在 $[l1+1, x]$ 右端点在 $[r1+1, n]$ 的区间的 b 的第一次出现的位置发生了变化，从 $r2$ 变成了 x，这些区间的答案需要 $+r2-x$

左端点在 $[1, l1-1]$ 右端点在 $[x, r1-1]$ 的区间的 b 的最后一次出现的位置发生了变化，从 $l2$ 变成了 x，这些区间的答案需要 $-l2+x$

把一个区间看做一个点

一次修改就相当于一个矩形的加减操作

询问的时候直接查询单点信息

初始化的时候直接将原序列当做是数值一个个插入的过程

在考虑到 n 的范围，用线段树套线段树或 cdq 分治解决

15.5 出处

share1005

16 杨曜嘉的题

16.1 时间

2017.10.19

16.2 题目

给出 n 个点 m 条边的图, 现把点和边分组, 每条边只能和相邻两点之一分在一组, 点可以单独一组, 问分组方案数

答案取模 10^7

$n, m \leq 10^5$

16.3 算法

结论

16.4 细节

考虑每一个连通块

假设边数为 e，点数为 v

如果 $e > v$ 那么无解

如果 $e = v$ 那么是一颗基环外向树，答案为环上的点数

如果是一颗树的话，答案为 v

原因很好想

16.5 出处

share1006

17 彭贻豪的题

17.1 时间

2017.10.19

17.2 题目

给定平面上 n 个点的坐标
求第 k 大的斜率
$n \leq 10^5$
$k \leq n * (n-1)/2$

17.3 算法

二分答案

17.4 细节

二分一个斜率 $slope$
判断就是数有多少个斜率 $\geq slope$
根据个数与 k 的大小关系继续二分
考虑如果得到个数
对于一个点 b 而言，所有横坐标大于它的点与它构成的直线的斜率大于 $slope$ 的 a 满足

$$\frac{y_a - y_b}{x_a - x_b} \geq slope$$

$$y_a - y_b \geq slope * (x_a - x_b)$$

$$y_a - slope * x_a \geq y_b - slope * x_b$$

令 $c_i = y_i - slope * x_i$
那么就是要求有多少对 i, j 满足 $i > j$ 且 $c_i \geq c_j$
二维数点解决

17.5 出处

share1007

18 彭贻豪的题

18.1 时间

2017.10.19

18.2 题目

n 个点，m 条边的 DAG，点，边都有各自的权值
定义一条路径的权值为路径经过的点权的最大值乘边权最大值
一组询问，询问两个点之间最小路径的权值
$n, m \leq 10^5$

18.3 算法

LCT

18.4 细节

从小到大枚举点权并依次将点权小于枚举值的点加入
用 LCT 动态维护询问的两个点之间最大边权
加入新边的时候如果与原图构成了环，就将环上的最大边权删掉

18.5 出处

share1007

19 permutation

19.1 时间

2017.10.21

19.2 题目

你有一个长度为 n 的排列 P 与一个正整数 K

你可以进行如下操作若干次使得排列的字典序尽量小

对于两个满足 $|i-j| \geq K$ 且 $|P_i - P_j| = 1$ 的下标 i 与 j, 交换 P_i 与 P_j

$n \leq 10^5$

19.3 算法

思维

19.4 细节

考虑得到这个排列的位置序列，第 i 个元素 b_i 表示在原排列中数值为 i 的数字位于原排列中第 b_i 个位置

求原排列的最小字典序就相当于位置排列的最小字典序

此时可以交换的是相邻的两个元素且满足 b_i 之差 $\geq K$

那么对于一个元素，它往左有一些元素会限制它不能继续交换

然后就相当于一个依赖关系，就可以连边然后用堆拓扑求得

但是这样有 $O(n^2)$ 条边，不过对于一个元素 x 事实上只需要连接一条边

考虑小于 x 且限制 x 的元素

因为是需要字典序尽量小，所以小于 x 且限制了 x 的元素在没有这样的限制条件下也是处在 x 的前面，所以 x 与小于它且限制它的元素之间不需要连边

考虑大于 x 且限制 x 的元素

如果 $j < i < x$ 且 $b_j, b_i \geq b_x$ 且 j, i 都限制了 x，那么 i 一定会限制 j

所以对于每一个 x 只需要将它向往左第一个大于它且限制它的元素之间连边

19.5 出处

Test20171021

20 tree

20.1 时间

2017.10.21

20.2 题目

给定一颗 n 个点的树, 树边带权, 试求一个排列 P, 使下式的值最大

$$\sum_{i=1}^{n-1} maxflow(P_i, P_{i+1})$$

其中 $maxflow(s,t)$ 表示从点 s 到点 t 之间的最大流, 即从 s 到 t 的路径上最小的边权

20.3 算法

结论

20.4 细节

答案就是边权之和

证明:

首先答案最大是边权之和

考虑边权从大到小来确定最终的排列

最终的排列中，每个元素最多有两个相邻的元素

如果将相邻元素之间相互连边且不成环，那么就等价于每个元素的度最多为 2

下面，边指上述相邻元素之间所连的边，原边指题中的边

考虑一条原边连接的是 x, y

如果 x, y 的度都不是 2

那么可以在 x, y 之间连一条边，表示它们在最终排列里相邻，此时这条原边的权值会统计到答案中

如果 x, y 的度至少有一个是 2

那么对于这两个中度数为 2 的点，可以沿着连边一直走，找到一个端点

假设分别找到的为 l, r(如果度数不是 2 那么找到的就是自己)

那么如果将 l, r 之间连边，此时这条原边的权值依然会统计到答案中，因为它是当前最小的权值，尽管连接的是 l, r，但是最终也导致了 l, r 之间的最大流为这条原边的权值

因此每一条边的权值都可以统计进去

20.5 出处

Test20171021

21 permutation

21.1 时间

2017.10.27

21.2 题目

给定长度为 n 的排列 A

定义

$$c(l,r) = max_{i=l}^{r} A_i$$

m 个询问，每次询问给出 l, r，询问有多少个 $x \in [l+1, r]$ 满足

$$A_{l-1} < c(l, x-1) < A_x$$

或者

$$A_x < c(l, x-1) < A_{l-1}$$

$n \leq 2 * 10^5$

$m \leq 1666666$

21.3 算法

21.3.1 算法 1

考虑满足条件的区间

21.3.2 算法 2

考虑一个询问的区间是如何得到的

21.4 细节

21.4.1 细节 1

考虑每个点作为最大值对哪些区间有贡献

令 i 左边第一个大于 A_i 的位置是 $l-1$，左边第一个大于 A_i 的位置是 $r+1$

考虑第一种条件

此时满足条件的区间为: 左端点在 $[l+1, i]$，右端点在 $r+1$

考虑第二种条件

此时满足条件的区间为: 左端点在 l，右端点在 $[i+1, r]$

上述满足条件的区间在询问时会造成 1 的贡献

此时考虑是平面上的问题，把区间 $[l, r]$ 视为平面上的点 (l, r)

那么就相当于给上述满足条件的区间所在平面上的位置标记上 1，表示这个位置对应的区间是满足条件的

那么对于一个询问而言，就相当于询问左端点在 l，右端点在 $[l+1, r]$ 的这些区间中满足条件的区间的个数

转化到平面上来说，就是询问一个线段上有多少个点标记了 1

所以可以直接用扫描线 + 树状数组统计即可解决

21.4.2 细节 2

考虑每个询问的答案是如何得到的

令 $l-1$ 往右第一个大于 A_{l-1} 的位置是 i(即 $i=r_{l-1}$)，i 往右第一个大于 A_i 的位置是 r_i，对于 r_i 也有一个 r_{r_i}

假设将 $r_i, r_{r_i}, r_{r_{r_i}}, ...$ 这样的位置加入到一个序列中，直到某个值的 r_x 大于询问的右端点

那么满足第一个条件的区间的右端点可以选的位置即这个序列中所有元素值

那么对于这样的情况，我们可以 $O(1)$ 询问出区间内最大值的位置 (假设它大于 A_{l-1})，然后再看从 $l-1$ 往右跳多少个 r_x 可以跳到这个最大值的位置，这种情况的答案就是跳的次数减一，可以用类似于跳跃的后缀和的形式预处理出来然后 $O(1)$ 询问

此时第二个满足条件的区间的右端点一定是在 $[l, i-1]$

假设 $[l, i-1]$ 中有一个 x，它往右第一个大于它的位置是 r_x

那么当右端点取 $[x+1, r_x-1]$ 的时候满足第二种条件

那么对于每个点 i，我们可以预处理出 $[i+1, r_i-1]$ 之间有多少个数

同样用类似于跳跃的后缀和的形式预处理出来然后 $O(1)$ 询问

记得要特殊判断一下区间最大值小于 A_{l-1} 的情况

21.5 出处

Test20171026

22 segment

22.1 时间

2017.10.27

22.2 题目

现有一颗管辖长度为 n 的序列的线段树，给定每个节点管辖的区间的左右端点以及区间中间值(不一定为左右端点的平均值)

现进行一次询问，询问区间和，询问时只会走到与询问区间有交的线段树节点

对于每个节点而言，如果询问的区间完全包含线段树节点的区间，那么就不会进行 $pushdown$ 操作

否则就会进行一次 $pushdown$ 操作

现有 m 个询问，求出每个询问进行了多少次 $pushdown$ 操作

$n, m \leq 3 * 10^5$

22.3 算法

22.3.1 算法 1

将区间问题考虑为平面问题

22.3.2 算法 2

考虑怎么统计

22.4 细节

22.4.1 细节 1

考虑线段树上最多有 $2 * n - 1$ 个节点，其中有 n 个节点的区间的左右端点相同

把剩下的 $n - 1$ 个节点看做区间，那么会进行 $pushdown$ 操作的节点的区间满足它与询问区间有交且不被完全包含

转化成平面问题就是询问 (l, r) 的左上，(l, l) 到 (l, r) 的左侧，(l, r) 到 (r, r) 的上侧有多少个点所代表的区间在线段树上出现过

然后用扫描线 + 树状数组统计即可解决

22.4.2 细节 2

下面的端点指线段树上出现过的所有左右端点除了左右端点相同情况下的两个端点

先统计有多少个端点不在询问区间内

再减去右端点在询问左端点左侧，左端点在询问右端点右侧的两倍

此时就统计了所有与询问区间相交但不包含询问区间的区间 1 次，但是统计了包含询问区间的区间 2 次

这些被统计了 2 次的区间是在询问区间左右端点的 lca 往上链上

相当与减去 lca 的深度即可，但是如果 lca 的左端点或右端点与询问区间的端点重合的时候，这些重合的区间只被统计了一次

所以在询问的时候要特判，并且预先处理出左端点为一个值的深度最小的节点，同理预处理出右端点的情况

这样对于有重合的情况只要直接跳到那个最上面的父亲的位置，然后减去它的深度

当然如果左右端点同时重合，此时直接往上面跳一次就变成了有一个端点重合或者没有端点重合的情况了，都是上面讨论过的情况

这样就可以统计答案了

22.5 出处

Test20171026

23 game

23.1 时间

2017.10.27

23.2 题目

现有一个 $n*m$ 的矩形，左上角为 $(1,1)$，右下角为 (n,m)，位置 (i,j) 的权值为 $A*i+B*j$

每个点有两个状态，一个是正面朝上，一个是翻面朝上，初始时部分点正面朝上，其它点反面朝上

现在可以进行若干次操作，每次操作可以选择一行或一列，将选中的这一行或这一列上的点的状态改变成另一种状态

求最大权值和

$n \leq 6*10^5, m \leq 10, |A|, |B| \leq 10^4$

23.3 算法

枚举 + 二分

23.4 细节

首先明确的是每个行或列要么被操作一次，要么不被操作

因为 m 非常小，所以很多行的情况是相同的，考虑枚举翻转哪些列，再单独考虑每一种状态下的所有行是否操作

我们可以计算出一种状态下，每一行在什么情况下操作比不操作更优

发现这是具有单调性的，即一种状态下选择操作的行要么是从最小编号的行开始一直到连续的某个行结束，要么是从最大编号的行开始一直到连续的某个行结束

然后根据在什么情况下操作比不操作优这个问题的解来确定哪些行操作哪些行不操作

这样就可以统计出权值之和

注意统计权值的时候分别统计行和列的，因为分开之后可以累加一些值

23.5 出处

Test20171026

24 toyuq

24.1 时间

2017.10.28

24.2 题目

给定一棵树，每条边都有一个长度 c

每个点上都可能有一个资源，你到达一个有资源的点后可以获取 w_i 个资源，获取后资源消失

有些点上可能有敌人，如果你到达一个有敌人的点后必须先花 t_i 秒的时间解决它，解决后消失

如果有资源的点上有敌人，必须先解决敌人

一开始你可以选择任意一个点出发，你有 T 秒的时间去尽可能地获取资源

求最大能获取多少资源

$n, T \leq 300, 0 \leq w_i, t_i, c \leq 10^6$

24.3 算法

树型 DP

24.4 细节

用 $f_{i,j}$ 表示进入 i 为根的子树并且回到 i

$g_{i,j}$ 表示进入 i 为根的字数并且不回到 i

$h_{i,j}$ 表示从 i 子树中出发经过 i 并且回到 i 为根的子树

它们表示在各自条件下用 j 秒可以获取的最大资源

记 son 为 i 的一个儿子，c 为 i 到 son 的路的长度

$f_{i,j} = max\{f_{i,j-k-2*c} + f_{son,k}\}$ 表示先在 i 的子树中 (不包括 son) 走 $j-k-2*c$ 秒并且回到 i，再用 c 秒走到 son，再用 k 秒在 son 中获取资源并且回到 son，最后用 c 秒走到 i

$g_{i,j} = max\{g_{i,j-k-2*c} + f_{son,k}\}$ 表示从 i 用 c 秒走到 son，然后用 k 秒在 son 中获取资源并且回到 son，然后用 c 秒走到 i，最后用 $j-k-2*c$ 秒走到除 son 子树外任意一个不是 i 的点上

$g_{i,j} = max\{f_{i,j-k-c} + g_{son,k}\}$ 表示从 i 出发花 $j-k-c$ 的时间获取部分子树中的资源，然后用 c 秒走到 son，最后用 k 秒在 son 中获取资源并且走到一个不是 son 的位置

$h_{i,j} = max\{h_{i,j-k-2*c} + f_{son,k}\}$ 表示从 i 子树中的某位置出发用 $j-k-2*c$ 秒获取资源然后走到 i 再用 k 秒在 son 中获取资源并且回到 son，最后用 c 秒回到 i

$h_{i,j} = max\{f_{i,j-k-2*c} + h_{son,k}\}$ 表示从 son 子树中的某位置出发用 k 秒获取资源然后走到 son 再用 c 秒走到 i 再用 $j-k-2*c$ 秒在除 son 外的其它子树中获取资源，最后用 c 秒回到 son

$h_{i,j} = max\{g_{i,j-k-c} + g_{son,k}\}$ 表示用 $j-k-c$ 秒走到 i 然后用 c 秒走到 son 最后走到一个不是 son 的位置

先处理儿子再更新即可

24.5 出处

Test20171028

25 题目名称

25.1 时间

25.2 题目

25.3 算法

25.4 细节

25.5 出处

26 题目名称

26.1 时间

26.2 题目

26.3 算法

26.4 细节

26.5 出处

27 题目名称

27.1 时间

27.2 题目

27.3 算法

27.4 细节

27.5 出处

28 题目名称

28.1 时间

28.2 题目

28.3 算法

28.4 细节

28.5 出处

29 题目名称

29.1 时间

29.2 题目

29.3 算法

29.4 细节

29.5 出处

30 题目名称

30.1 时间

30.2 题目

30.3 算法

30.4 细节

30.5 出处

31 题目名称

31.1 时间

31.2 题目

31.3 算法

31.4 细节

31.5 出处

32 附录

- 对于区间问题可以试着将区间端点串联起来，首尾各是源汇，然后对于一个区间信息可以将端点相连，限制源点的流量
- 学会构造流量平衡的等式
- 最多一共有 N-1 个不同的 s-t 最小割
- 对于连边的时候如果 a 连向 b,c，b 连向 c, 那么 $a->c$ 这条边可以去掉
- $F_x = f_x + g_x * h_x$ f_x 应当乘上一个单位多项式，单位多项式的每一项都是（1.0,0.0）
- 卷积形式的要求：$F_j = \sum_{i=0}^{n} f_x * g_y$ 其中 n 中含有 j，$x + y = n$
- 狄利克雷卷积的形式：$(f * g)(n) = \sum_{d|n} f(d) * g(\frac{n}{d})$
- 两个积性函数的狄利克雷卷积形式为积性函数
- 学会单独考虑两个点之间的情况
- 注意异或的运算法则
- 二进制相关的题目仔细考虑题目所给条件的二进制下的意义，考虑二进制的每一位求解
- 线段树合并信息的时候可以考虑再往下递归得到信息，但是这样会多一个 log
- 斜率式可以考虑凸包
- 当一个操作不好打的时候可以想办法换一个比较简单的操作加上一点算法变成等价的操作
- 多重哈希可以降低冲突带来的影响率
- 学会先猜后证
- 考虑计数问题的时候可以从别的计数方向出发
- 学会用插入以及等效的方法考虑问题
- 精度最好设置到 10^{-9}，再就很容易运算出错
- 某些情况下排序可以取消 DP 的后效性
- sort 的比较函数要求：比较函数不能使得存在：$a < b \&\& b < a$
- 利用重复询问的特点
- 注意 tarjan 的应用 (判断点是否在环上)
- 期望题可以考虑二分一个变量的值来做
- 考虑生成两个可以相互抵消的操作
- 矩阵转置相关计算：$A^T * B^T = (B * A)^T$
- 其中，A 为矩阵，A^T 为 A 的转置矩阵

- 邻接矩阵的性质：对于一个邻接矩阵 A，A^x 表示经过 x 条边后的邻接矩阵
- 图可以转化成 dfs 树来做
- 用函数形式代替和式，当递归出现相同函数形式时可以考虑递归求解
- 将函数相邻两项做差