ST 表、并查集

黎伟诺

7.18.2024

ST 表

- ST 表（Sparse Table，稀疏表）是用于解决可重复贡献问题的数据结构。
- 和 RMQ（Range Minimum/Maximum Query）问题联系紧密
- 除 RMQ 以外，还有其它的「可重复贡献问题」。例如「区间按位与」、「区间按位或」、「区间 GCD」，ST 表都能高效地解决。

令 $mx_{i,j}$ 为 $[i, i+2^j-1]$ 区间内的最大值，那么初始化为 $mx_{i,0}=a_i$，可以通过 $mx_{i,j}=\max(mx_{i,j-1}, mx_{i+2^{j-1},j-1})$ 递推。

```
int query(int l,int r) {
    int x = l,ret = 0;
    for(int i = __lg(r - l + 1); i >= 0; i--)
        if(x + (1 << i) - 1 <= r) {
            ret = max(ret, mx[x][i]);
            x += 1 << i;
        }
    return ret;
}
```

ST 表

但是注意到最大值重复贡献之后不会对答案产生影响，于是我们可以直接找到两端极大的被 $[l, r]$ 包含的区间，从而在 $\Theta(1)$ 时间内回答问题。

```
int query(int l,int r) {
    int k = __lg(r - l + 1);
    return max(mx[l][k], mx[r - (1 << k) + 1][k]);
}
```

Triangle: The Data Structure

P6648

大小为 m 的一个三角形由 m 行组成，第 i 行包含 i 个元素。
并且，这些行必须排为等边三角形的形状。
比如说，以下是一个 $m=4$ 的三角形。

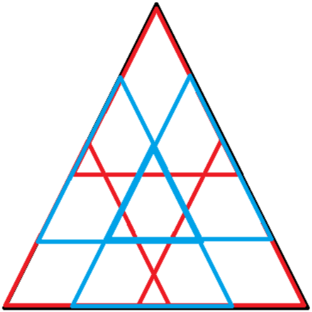
每个三角形还包含子三角形。
比如说上面这个三角形，包含：

- 10 个大小为 1 的三角形。
- 6 个大小为 2 的三角形。
- 3 个大小为 3 的三角形。

注意，每个三角形都是自身的子三角形。
现在给定一个大小为 n 的三角形，求对于每个大小为 k 的子三角形，子三角形内几个数的最大值的和。

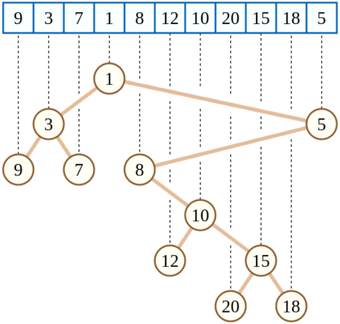
空间限制：128MB。$1 \leq k \leq n \leq 3000$

很容易想到 ST 表，但是空间会爆
注意到询问的大小固定，我们可以对 st 表滚动
有多种 ST 表的建法，有的需要把二维下标映射成一维下标（利用三角形空间减半）
下图这种合并方法可以只用建一个 st 表



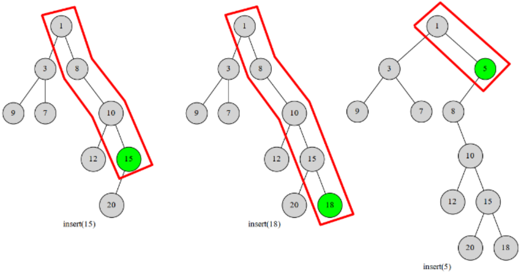
笛卡尔树

每次把数组的最小值拿出来，然后连接左右区间形成的树
可以用 RMQ 模拟来做到 $O(n\log n)$ 建树



笛卡尔树

当然也可以使用单调栈来构建，从左往右做
维护笛卡尔树的最右链



章节划分

P7244

天依决定了 n 个素材，它们将**依次**在作文中被叙写。其中，第 i 个素材的立意特征值是 a_i。

但天依发现她构思的大作实在是太长啦，所以她想把它们划分为**恰好 k 个**章节，每个章节包含一段**连续且非空的**素材。假设第 i 个章节包含素材 $[l_i, r_i]$，天依将选取立意特征值最大的素材来升华，得到该章节的立意值 b_i，满足 $b_i = \max\limits_{i \in [l_i, r_i]} \{a_i\}$。

最后，整篇作文的凝练度为每个章节立意值的**最大公约数**，即 $\gcd\limits_{i \in [1,k]} \{b_i\}$。

天依当然希望**最大化**作文的凝练度，那么凝练度的最大值是多少呢?

$1 \le k \le n \le 10^5$ 且 $1 \le a_i \le 10^6$

- 答案必然是全局最大值的约数。并且假如能划分成 $\geq k$ 个段，那么相邻合并使得段数 $= k$ 后答案不会下降
- 枚举这个约数 x 。希望求出序列划分最多的段数，使得每一段最大值都是 x 的倍数
- 设区间 $[l, r]$ 的最多段数为 $f(l, r)$，区间最大值位于 mid。
- case1：最大值被 x 整除，可以直接贪心取 a_mid 作为单独的一段，$f(l, r) = f(l, mid-1) + 1 + f(mid+1, r)$
- case2：$l > 1$ 时，$l-1$ 一定是某一段的 mid，$[l, mid]$ 这一段可以直接往左合并，这时 $f(mid+1, r) \to f(l, r)$，同理 $r < n$，所以 $f(l, r) = \max([l > 1]f(mid+1, r), [r < n]f(l, mid-1))$

Delivering Balls

P7974

给定一个长度为 N 的序列 H 和 Q 次询问。

第 i 次询问中，你初始在第 S_i 列 H_{S_i} 行，想要到第 T_i 列第 H_{T_i} 行。

你可以进行若干次移动。每次移动你可以选择以下两种参数:

- 列 -1，列不变，列 $+1$。
- 行 -1，行不变，行 $+1$。

如果你选择行 -1，消耗 1 体力，如果你选择行不变，消耗 2 体力，如果你选择行 $+1$，消耗 4 体力。

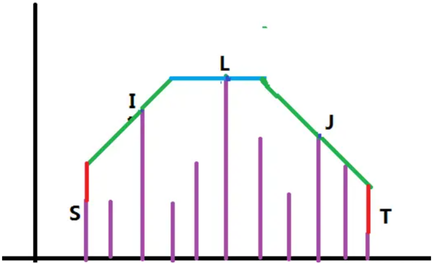
你需要保证每次移动后，你的列数 x 在 $[1, N]$ 之间，且你的行数 y 不小于 H_x。

对于每个询问，你需要求出你消耗体力的最小值。

Delivering Balls

P7974

赛博爬山



需要维护区间的 $a = \max\{h_i\}$ ， $b = \max\{h_i - i\}$ ， $c = \max\{h_i + i\}$
答案为 $a - 4 * h_s - 4 * h_t + 2(b + c)$

序列

P3246

给定长度为 n 的序列：$a_1, a_2, \cdots, a_n$，记为 $a[1\colon n]$。类似地，$a[l\colon r]$（$1 \le l \le r \le N$）是指序列：$a_l, a_{l+1}, \cdots, a_{r-1}, a_r$。若 $1 \le l \le s \le t \le r \le n$，则称 $a[s\colon t]$ 是 $a[l\colon r]$ 的子序列。

现在有 q 个询问，每个询问给定两个数 l 和 r，$1 \le l \le r \le n$，求 $a[l\colon r]$ 的不同子序列的最小值之和。例如，给定序列 $5, 2, 4, 1, 3$，询问给定的两个数为 1 和 3，那么 $a[1\colon 3]$ 有 6 个子序列 $a[1\colon 1], a[2\colon 2], a[3\colon 3], a[1\colon 2], a[2\colon 3], a[1\colon 3]$，这 6 个子序列的最小值之和为 $5 + 2 + 4 + 2 + 2 + 2 = 17$。

序列

P3246

假设我们已经有 st 表能求区间最小值

每个点 x 求出了最大的 $y < x$ 满足 $a[y] \le a[x]$，记作 pre_x

每个点 x 求出了最小的 $y > x$ 满足 $a[y] \le a[x]$，记作 nxt_x。（用单调栈求）

对于 $[l, r]$ 区间，假设最小值位置为 x，那么这里的贡献就是 $a[x] * (x - l + 1) * (r - x + 1)$，接下来只需要计算 $[x + 1, r]$ 和 $[l, x - 1]$ 区间的贡献

序列

P3246

枚举右端点 R，我们需要计算 $[x+1, R], [x+2, R], \ldots, [R, R]$ 这些区间的贡献。设 $f_R = \min[1, R] + \min[2, R] + \min[R, R]$，那么 $f_R = f_{pre_R} + (R - pre_R)a_R$。

$[x+1, R], [x+2, R], \ldots, [R, R]$ 的贡献很容易知道为 $f_R - f_x$，所以 $[x+1, r]$ 为 $\sum_{R=x+1}^{r} f_R - f_x$，可以用前缀和维护。$[l, x-1]$ 区间的贡献也可以对称处理

区间覆盖问题

CF 1175E

有 n 个区间，每个区间 $[l, r]$ 满足 $1 \le l \le r \le 10^5$。有 m 次询问，每次询问给出 $[l, r]$，问覆盖 l 到 r 所有位置最少需要几个区间？

区间覆盖问题

CF 1175E

如果询问是 $[1, p]$ 怎么做？
有这样一个贪心，假设当前我们已经覆盖了 $[1, x]$（初始 $x = 0$）
那么下一个区间一定会选择满足 $l \le x + 1$ 的区间中 r 最大的一个区间。
那么如果要询问 $[a, b]$，那么我们第一个区间一定会选择 $l \le a$ 中 r 最大的，其实相当于我们已经覆盖了 $[1, a]$。

区间覆盖问题
CF 1175E

注意到如果我们已经覆盖了 $[1, x]$，那么选择的下一个区间是一定的。
所以我们可以预处理 $to_{i,j}$ 代表当前已经覆盖了 $[1, i]$，然后再选择 2^j 个区间之后最大覆盖到哪里。
那么递推关系就是 $to_{i,j} = to_{to_{i,j-1},j-1}$。对于一个询问，我们可以在 $O(\log n)$ 的时间内求出答案。

并查集

并查集是一种用于管理元素所属集合的数据结构
一般实现为一个森林，其中每棵树表示一个集合，树中的节点表示对应集合中的元素。
并查集支持两种操作：

- 合并（Union）：合并两个元素所属集合（合并对应的树）
- 查询（Find）：查询某个元素所属集合（查询对应的树的根节点），这可以用于判断两个元素是否属于同一集合

初始化：

```
for (int i=1;i<=n;i++) fa[i]=i;
```

查询：

```
int getfa(int x){return x==fa[x]?x:getfa(x);}
```

并查集

路径压缩:

```
int getfa(int x){return x==fa[x]?x:fa[x]=getfa(x);}
```

合并:

```
int unite(int x,int y){
    int fax=getfa(x);
    int fay=getfa(y);
    if (fax!=fay) fa[fax]=fay;
}
```

并查集

启发式合并：
合并时，选择哪棵树的根节点作为新树的根节点会影响未来操作的复杂度。
我们可以将节点较少或深度较小的树连到另一棵，以免发生退化。

```
int unite(int x,int y){
    int fax=getfa(x);
    int fay=getfa(y);
    if (fax!=fay){
        if (sz[fax]>sz[fay]) swap(fax,fay);
        fa[fax]=fay;
        sz[fay]+=fax;
    }
}
```

这样树的高度最多是 $O(\log n)$。

并查集

删除：
要删除一个叶子节点，我们可以将其父亲设为自己。为了保证要删除的元素都是叶子，我们可以预先为每个节点制作副本，并将其副本作为父亲。
移动：
与删除类似，通过以副本作为父亲，保证要移动的元素都是叶子。

并查集

时间复杂度：

- 单独使用启发式合并或者路径压缩的最坏复杂度都是 $O(\log n)$。
- 同时使用路径压缩和启发式合并之后，并查集的每个操作平均时间仅为 $O(\alpha(n))$，其中 α 为阿克曼函数的反函数，其增长极其缓慢
- 也就是说其单次操作的平均运行时间可以认为是一个很小的常数（比如说 4）
- 实战中仅仅写路径压缩，期望速度就已经很快了

实现类似并查集的数据结构，支持以下操作：

- 合并两个元素所属集合
- 移动单个元素
- 查询某个元素所属集合的大小及元素和

关押罪犯

P1525

S 城现有两座监狱，一共关押着 N 名罪犯，编号分别为 $1 \sim N$。他们之间的关系自然也极不和谐。很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。我们用“怨气值”（一个正整数值）来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 c 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并造成影响力为 c 的冲突事件。

每年年末，警察局会将本年内监狱中的所有冲突事件按影响力从大到小排成一个列表，然后上报到 S 城 Z 市长那里。公务繁忙的 Z 市长只会去看列表中的第一个事件的影响力，如果影响很坏，他就会考虑撤换警察局长。

在详细考察了 N 名罪犯间的矛盾关系后，警察局长觉得压力巨大。他准备将罪犯们在两座监狱内重新分配，以求产生的冲突事件影响力都较小，从而保住自己的乌纱帽。假设只要处于同一监狱内的某两个罪犯间有仇恨，那么他们一定会在每年的某个时候发生摩擦。

那么，应如何分配罪犯，才能使 Z 市长看到的那个冲突事件的影响力最小？这个最小值是多少？

关押罪犯

P1525

按照权值从大到小排序，我们需要让权值大的尽量分配在两个不同的监狱里。依次加边，直到冲突了停止

给每个点 a 添加一个额外的点 a' 代表 a 虚拟的敌人

枚举到边 a, b，我们需要把 a 和 b' 合并，a' 与 b 合并

海滩防御

P1783

WLP 同学最近迷上了一款网络联机对战游戏（终于知道为毛 JOHNKRAM 每天刷洛谷效率那么低了），但是他却为了这个游戏很苦恼，因为他在海边的造船厂和仓库总是被敌方派人偷袭。于是，WLP 动用了他那丰满且充实的大脑（或许更偏向前者），想出了一个好主意，他把海滩分成垂直于海岸线的若干列，在其中的几列上放置几个信号塔，试图来监视整个海滩。然而，WLP 是一个非常心急的人，他把信号塔建好后才发现还需给信号塔供能，它们才能投入使用（这不是废话么），它们都有一个工作半径，一个圆形区域里的所有敌人都逃不过它们的监视，不过，WLP 发现，敌人们非常狡猾，除非他将道路完全封死，否则 WLP 的敌人可以走过一条任意弯曲的路（不一定走整点，但是不会出第 0 列和第 N 列构成的边界）来偷他的东西。

于是，WLP 就思考了：到底需要给每个信号塔多大的工作半径，才能将从海滩到内地的路径完全封死呢？他再次动用了他那丰满且充实的大脑，想了一堂数学课，终于，还是没想出来。于是，他向 LZZ 神犇求助（额...... CSUNSHINE 的身份是不是暴露了）。

终于，在 WLP：" %^!*@#!*(*^!*#@$^&（此处省略无数卖萌场景）"的哀求下，LZZ 神犇写了一个程序，在一秒内就解决了问题。但是，邪恶的 LZZ 神犇决定要将这个难题共享给无数无辜的 OIer，所以，现在轮到你了。

输入格式

第一行两个整数 N 和 M，表示海滩被 WLP 分成的列数 $0, 1, 2, \cdots, N$ 和信号塔个数。

第 2 至第 $M+1$ 行，每行两个数 X_i，Y_i 表示 $1, 2, 3, \cdots, M$ 号信号塔所在的列数和离开海滩的距离。

对于 100% 的数据：$1 \le M \le 800$，$1 \le N \le 1000$，$1 \le X_i \le N$，$1 \le Y_i \le 100,000$。

海滩防御

P1783

预处理一下各个防御塔之间的距离放入结构体里，再给每个防御塔连一条和左边界，右边界的边。
两个防御塔之间要连接只需要一半距离的攻击范围即可，但与边界间就只能靠防御塔自己一个
然后按距离从小到大排序，依次连边，每次检查左右边界是否连接即可。

萌萌哒

P3295

一个长度为 n 的大数，用 $S_1S_2S_3\cdots S_n$表示，其中 S_i 表示数的第 i 位, S_1 是数的最高位。告诉你一些限制条件，每个条件表示为四个数，l_1,r_1,l_2,r_2，即两个长度相同的区间，表示子串 $S_{l_1}S_{l_1+1}S_{l_1+2}\cdots S_{r_1}$ 与 $S_{l_2}S_{l_2+1}S_{l_2+2}\cdots S_{r_2}$ 完全相同。

比如 $n=6$ 时，某限制条件 $l_1=1,r_1=3,l_2=4,r_2=6$，那么 123123，351351 均满足条件，但是 12012，131141 不满足条件，前者数的长度不为 6，后者第二位与第五位不同。问满足以上所有条件的数有多少个。

首先发现肯定要并查集，因为 $l = r$ 的情况基本需要这个

发现这个区间连线可以看作两组彼此有交的区间连线的并。（比如 $[1, 5]$ 连 $[6, 10]$ 相当于 $[1, 4]$ 连 $[6, 9]$ 以及 $[2, 5]$ 连 $[7, 10]$）

想到可以借助 RMQ 的思想解决这个问题，每次连线先拆成两个长度为 2 的整数次幂的操作，在并查集将 $st[i][K]$ 节点和 $st[j][K]$ 节点连一起

做完所有操作之后，按层数从高往低，$st[i][K]$ 属于 $fa[st[i][K]]$，所以往下要变成 $st[i][K-1]$ 的两个长度为 2^{K-1} 的区间去连 $fa[st[i][K]]$ 对应的两个区间

最终 $st[i][0]$ 形成了 num 个等价类，答案就是 $9\times 10^{num-1}$