成就工匠的必备素质

zjq

注意事项

- 由于本人水平有限，选择的内容可能太水，像两位林姓大佬CC，PTY这一类的队爷可以提(gu)前(gu)离(gu)场，切勿大喊“这些东西我幼儿园就会啦！”等类似言语，以免影响进入梦乡的童鞋。

- 由于本人不像你们一样家财万贯，此次未能准备奖品，但请各位同学踊跃发言。

- 如果在之前的同样难度的讲课中，收获不大或者信号不良的同学，这次讲课务必慎重考虑。

- 若内容有不恰当之处，请各位批评指正。

先来一道《小学生数据结构题》劝退

- N*M的房间，四面都有墙，初始时4连通，有Q次操作：
- Build (x , y , w) 在(x , y)房间w方向筑墙，墙有两面，双向无法越墙。
- Remove (x , y , w)拆掉(x , y)房间w方向的墙。
- Create (x1 , y1 , w1 , x2 , y2 , w2)表示建一对传送门，从(x1 , y1)以w1方向进去，会从(x2 , y2)以w2方向出来，反之也可以。
- Destroy (x1 , y1 , w1 , x2 , y2 , w2)表示原有连通(x1 , y1)w1方向和(x2 , y2)w2方向出来的传送门被拆除。
- Query (x , y , w)问从(x , y)房间以w方向开始跑，直到撞墙一共经过多少房间，如果无限多，则输出”INF”
- 保证不重复在同一个地方同一方向建墙，不拆没墙的地方，不拆有传送门的墙，不在没墙的地方建传送门，不在有传送门的地方建传送门，拆传送门则必有传送门，N,M,Q<=200000

你要证明你比小学生更强！

- 如果N*M很小，不难想到~~暴力~~将每一个格子的上下、左右两种方向都视作一个点，然后能够一步互相到达的两个点挨在一起，建平衡树。
- 对于拆墙和传送门操作就将两个平衡树合并，如果成了环(两个在同一颗平衡树上)，就打个环的标记。
- 建墙和拆传送门操作就相当于分裂一棵平衡树，或者是取消环标记分裂重组一下。
- 询问的话就是问一个点左边或是右边有多少个点。
- 当N,M比较大的时候，就一开始用一个点表示一行或一列，然后动态开点。
- 复杂度O(Q log (N + M + Q))。

内容简述

- ~~用树状数组维护01背包？~~
- AAA树ABF树？~~又不会考——我也不会~~

- 讲师太累了懒得讲：
- 全局平衡二叉树
- 区间最值操作及历史版本查询问题

- 讲解内容：
- 可追溯化数据结构
- 数据结构的简单应用

全局平衡二叉树

- 也是基于树链剖分
- 对一条重链，将每一个点附上轻儿子的size和的权，找这一条链最中间的那个点，也就是去除这个点之后两边的权值和尽量均匀，然后将这个点取出来作为根，然后将分出的两条链作为儿子，做同样操作递归下去，就相当于对这条链分治。用这样形成的二叉树来维护信息。
- 因为从这棵二叉树往父亲走，和跳重链，size都会至少翻倍，那么这个复杂度是O(n log n)的。
- 这种做法的常数要比LCT小，但代码复杂度要稍微比普通的树链剖分套数据结构大，如果需要强行将二次方log的复杂度降为一次方，这种方法是个很好的选择。

区间最值操作

- 要求支持：
- 1.区间对x取min取max
- 2.单点修改
- 3.区间查询最大值、最小值或求和

- O(n log n)的复杂度实现

区间最值操作

- 由于没办法知道每一个位置是否比操作的数要小，区间取最值无法用普通的懒标记处理，那我们要用特殊一点的标记。
- 那么我们将引用如下方法：吉司机线段树(其一)

- 我们很容易能够臆想到，如果这个区间的最大值比操作的x大，那么就暴力改下去，否则就不改。但很显然，可以不断对一个更小的取min，你每一次都要走下去，显然这个要爆。
- 按照吉老师的说法，我们记最大值和严格次大值，如果修改值x大于严格次大值，那么就打标记或者不动它，这个复杂度是O(n log n)的。

区间最值操作

- 但是这怎么看都很假，怎么证呢？
- 吉老师有一种很好的证法，但是这里用高手林立的证明方法：

设势能函数$\phi(x)$为线段树上每一个点对应的区间不同数的种数。

一开始势能是$O(n \log n)$的。

每一次单点修改会使势能增加$O(\log n)$。

每一次取min会定位到$O(\log n)$个区间，若在某个区间上，不动或者直接打标记，不耗费复杂度；如果走下去，就说明最大值和严格次大值都会被取min为同一值，时间复杂度贡献$O(1)$，势能至少会减少$O(1)$，那么总复杂度是$O(n \log n + m \log n)$的。

区间最值操作

- 如果加上区间加，赋值？
- 上述分析就不适用了，钠碘酸呢？

- 其实用上述方法也是可行的，只是复杂度升到O(n log^2 n)
- 这里给大家一点思考，就不证明了，有兴趣的可以参考吉如一2016的论文。

区间最值操作

例题：

- 支持区间取min
- 区间加
- 区间求gcd

- 数据范围50000

区间最值操作

- 又加又gcd的，乍看不好处理，如果将其转换一下，gcd(a1,a2,a3,…,an)=gcd(a1,a2-a1,a3-a2,…,an-a[n-1])，那么区间加也可以变成单点操作，那么需要维护的量只有最大值，严格次大值，除最大值之外差分的gcd，这样复杂度就是O(n log^3 n)的。

区间最值操作

- 这种吉司机线段树不仅可以处理区间取min，也可以处理区间取and\or。
- 具体做法就是记录该区间所有数的and和or。
- 假设我们现在要区间对x取and，那么如果区间内所有数and x都不变，那么就不搜下去，如果存在x的非一位，区间内所有数原本在这一位上不相同，那么就往下暴力更改。
- 势能分析的方法就是：设势能$\phi(x)$为线段树上所有节点对应区间上的数的不全相同的二进制位的总数，那么原本的势能是$O(n*\log n*\log A)$的，每次往下走会使势能减少至少$O(1)$，时间复杂度就是$O(n*\log n*\log A)$。

历史版本查询

- 这个也是吉老师的，美其名曰吉司机线段树(其二)

- 历史版本查询问题无非就是求历史最值与历史和。

历史版本查询

- 主要形式是:
- 已知{an}
- 令bi=ai
- 支持对a区间加、赋值
- 单点询问bk的值
- 每次操作之后对b进行更新bi=max(bi,ai)

- O(n log n)

历史版本查询

- 这是一种可以用懒标记处理的问题
- 具体做法如下
- 如果不考虑区间赋值的话，一般要记录区间加标记tag_add，同时如果有b的更新，还要记录max_add，表示标记最大加了多少，下传显然，时间复杂度当然是O(n log n)的

历史版本查询

- 也有无法用懒标记的：
- 给定一个序列{an}，起初bi=ai
- 要求支持：
- 区间对x取max
- 区间加x
- 查询区间bi的min
- 每次操作都对b作更新bi=min(bi,ai)
- n<=300000

历史版本查询

- 这道题也是用区间最值操作的办法，将最小值和非最小值分开标记，相当于记一下对于最小值，它能到达最小是多少，非最小值最少被加上多少，时间复杂度O(n log n)

历史版本查询

- 对于历史版本求和，就相当于打上+k*ai+B的标记
- 总之具体问题具体分析

可追溯化数据结构

- 部分可追溯化：

维护一个操作序列，支持在某个位置插入一个操作，或删除一个操作， 以及对按顺序执行这些操作后的状态进行一些询问。

- 完全可追溯化：

在部分可追溯化的基础上还支持对操作序列的任意一个前缀进行询问

可追溯化数据结构——并查集1

- 支持insert(t , union(x , y))在第t个操作之后加入union(x , y)操作

- 部分可追溯化：

询问x,y是否连通

- 完全可追溯化：

询问在第t时刻，x,y是否连通

可追溯化数据结构——并查集1

- 部分可追溯化：

由于union操作满足交换律，因此直接并查集即可

- 完全可追溯化：

其实也很简单，将一条边加入的时间看做关键字，用lct维护最小生成树，查询就找lct上路径中边的最晚加入时间

可追溯化数据结构——并查集2

- 1. insert(t , union(x , y))在第t个操作之后加入union(x , y)操作
- 2. delete(t) 删除第t个操作

- 部分可追溯化：

询问x,y是否连通

- 点数5000，操作数500000

可追溯化数据结构——并查集2

- 做法：
- 这就是动态图的连通性问题。
- 对于每条边都有它的存在时间区间，那么按时间建线段树，将加入的边下放到线段树上，离线跑整棵线段树，因为要有撤回，所以并查集用按秩合并。

可追溯化数据结构——队列

- 1. insert(t , enqueue(x))，在t操作后加入将x元素入队的操作
- 2. insert(t , dequeue( ))，在t操作后加入出队的操作
- 3. delete(t)删除t操作

- 部分可追溯化：

queue_front( )询问将要弹出的元素

queue_back( )询问最后一个加入的元素

- 完全可追溯化：

queue_front(t)询问前t个操作后将要弹出的元素

queue_back(t)询问前t个操作后最后一个加入的元素

可追溯化数据结构——队列

- 很容易发现其实队列也是满足交换律的(前提是不会出现空队列出队的情况)。

- 部分可追溯化：

先不考虑dequeue

用链表维护不出队的整个队列，顺便用指针记下将出的位置。

- 完全可追溯化：

同样的原理，将链表改成两个平衡树，一个平衡树按时间装下dequeue操作，用于查询某时刻已经进行了多少次dequeue，另一个用于查询元素。

可追溯化数据结构——栈

- 1. insert(t , push(x)) 在第 t 次操作后插入将元素 x 入栈操作。
- 2. insert(t , pop( )) 在第 t 次操作后插入出栈操作。
- 3. delete(t) 删除第 t 次操作。

- 完全可追溯化：

query(t, top()) 询问 t 时刻后的栈顶元素。

可追溯化数据结构——栈

- 求得的答案一定满足，那个位置以后的push和pop次数一样多，那么我们要找最后一个满足后缀和为1的，只需要用平衡树维护即可。
- 维护后缀和的max , min，然后在平衡树上二分一下。

可追溯化数据结构——双端队列

- 1. insert(t , pushl(x)) 在第 t 次操作后插入将元素 x 从左端加入双端队列操作。
- 2. insert(t , pushr(x)) 在第 t 次操作后插入将元素 x 从右端加入双端队列操作。
- 3. insert(t , popl( )) 在第 t 次操作后插入弹出左端第一个数操作。
- 4. insert(t , popr( )) 在第 t 次操作后插入弹出右端第一个数操作。
- 5. delete(t) 删除第 t 次操作。

- 完全可追溯化：
- 1. query(t , left( )) 询问 t 时刻后双端队列中最左端元素的值。
- 2. query(t , right( )) 询问 t 时刻后双端队列中最右端元素的值。

可追溯化数据结构——双端队列

- 需要求i和a[i]。
- 这就是双头栈，和栈的处理方式类似，维护两棵平衡树 Tl 和 Tr，Tl 按时间存下 pushL(x) 操作和 popL() 操作，权值分别为 +1 和 −1，Tr 一样。求 t 时刻的 i 的值只用在 Tl 或 Tr 中求一下前缀和就行了。
- a[i] 的值肯定是最后一次 pushL(x) 后 L = i 的操作或者最后一次 pushR(x) 后 R = i 的操作。这两个都可以在平衡树上二分出后缀和为 k 的点，取两者较晚的一次操作即可。

可追溯化数据结构——堆

- 1. insert(t , insert(k))在第 t 次操作后插入将元素 k 入堆操作。
- 2. insert(t , delete_min( ))在第 t 次操作后插入弹出最小元素的操作。
- 3. delete(t) 删除第 t 次操作。

- 部分可追溯化：

询问堆顶元素

- 完全可追溯化：

询问执行前t个操作后的堆顶元素

可追溯化数据结构——堆

- 部分可追溯化：
- 先定义：

1. Q_t为t次操作之后堆内的元素，Q_{now}为最终堆内的元素
2. t被称作桥，当且仅当$Q_t \subseteq Q_{now}$

- 考虑一个insert(t , k)操作对Q_{now}的影响：

相当于给加入了$\max\{k, k' | k'在t后删除\}$，就是代替原来被删除的最大的那个

可追溯化数据结构——堆

- 对于某一时刻t，其之前最近一个桥t′，有$\max\{k'|k'在t后删除\} = \max\{k' \notin Q_{now}|k'在t'后插入\}$

证明可以反证：如果最大值k′在t之后正确性显然；否则假设在t之前，比k小没被删除的最小值为k′，那么如果不存在，那么不会影响到答案；否则，有t′以后在k′插入之前比k′小的都被删掉了，那么k′插入的位置将是一个桥，与题设相悖。

- 考虑一个delete(t)的影响，t之后最近一个桥t′，Q_{now}将要删除 $\min\{k' \in Q_{now}|k'在t'前插入\}$，证明类似。
- 因此我们可以用一颗平衡树维护Q_{now}，一棵按时间的平衡树维护

$$\max\{k' \notin Q_{now}|k' 在该时间区间内插入\}$$
$$\min\{k' \in Q_{now}|k'在该时间区间内插入\}$$

- 对于桥的维护，显然如果将$x \in Q_{now}$视作0，$x \notin Q_{now}$视作1，delete视作-1，那么桥必然是前缀和=0的位置，再用一颗平衡树维护就好了

可追溯化数据结构——堆

- 完全可追溯化：
- 一种很容易的想到的做法就是分块，对每个块做部分可追溯化，但复杂度要乘上$O(\sqrt{n})$
- 用一棵替罪羊树维护操作区间形成的Q_{now}及Q_{del}，插入操作就是往log n个区间做一遍部分可追溯化。考虑求答案：在合并的时候，假设$Q_3 = Q_1 \cup Q_2$，那么：

$$Q_{3,now} = Q_{2,now} \cup \max -A\{Q_{1,now} \cup Q_{2,del}\}$$
$$Q_{3,del} = Q_{1,del} \cup \min -D\{Q_{1,now} \cup Q_{2,del}\}$$

其中$A = |Q_{1,now}|, D = |Q_{2,del}|$

可追溯化数据结构——堆

- 直接合并的复杂度太大，考虑将平衡树直接串起来(没有实际连上)
- 我们要找到$Q_{1,now} \cup Q_{2,del}$的第A小，这里后面GetSplitKey仔细讲
- 这样原本Q_1 Q_2都是k棵平衡树，分裂合并之后就会变成3k，总平衡树个数将会是$O\left(3^{log_2 k}\right) = O\left(k^{log_2 3}\right) \approx O(k^{1.58})$
- 但事实上，由于$Q_{*,now} = \cup Q_{i,now} \cup Q_{i,del}$，最终只用2k个平衡树就可以表示出$Q_*$，总数也就k个，因此复杂度将是$O(\log n \cdot k \log k)$，替罪羊树中k是O(log n)级别的，复杂度是$O(\log^2 n \log\log n)$

GetSplitKey({ T1 , T2 , … , Tk } , s)

- 如果平衡树总大小N小于常数C，直接排序找。
- 如果s<N/2，令s=N-s，将平衡树翻转。
- 找到每棵平衡树Ti最左的子树T_{mi}，使$|T_{mi}| \in (|T_i|/256, |T_i|/4)$，并且该子树代表的区间是(-∞,mi)。
- 找到最大的mj，使得$\sum_{mi \le mj} |T_i| \le N/4$，对于不大于mj的mi，$T_i' = T_i - T_{mi}$，其余$T_i' = T_i$，s'=s-$\sum_{mi \le mj} |T_{mi}|$；
- 求GetSplitKey({ T1' , T2' , … , Tk' } , s')
- 它保证了删掉的子树中一定没有答案，因为比mj大的mi对应|Ti|中，比mi大的最少3/4，总数比mj大的也不会少于3/4*3/4=9/16。
- 因此一层的复杂度是$O(k)$因为每一个平衡树往下搜索的次数大约是2，层数最少为$O(log_{16/15} n)$，最多$O(log_{1024/1023} n)$，常数有点大。

可追溯化数据结构——堆

- 较详细的题解详见

Erik D. Demaine , Tim Kaler, Quanquan Liu, Aaron Sidford, and Adam Yedidia. Polylogarithmic Fully Retroactive Priority Queues via Hierarchical Checkpointing

☺随便来几道NOIP题☺

【BZOJ3681】Arietta

- 在有根树上第i个点上权值为Hi。
- 你有m个选择次数，第i次能选择Ti个点，所选的点要在Di的子树中，权值范围应是[Li,Ri]。
- 问最多多少个点曾被m次选择中选过。
- n , m <= 10000

【BZOJ3681】Arietta

- 很简单？
- 跑网络流，图很好建。
- 边数很多$O(n^2)$怎么办？
- 用dfs序来处理子树问题！
- 但是连边不可以作差，区间的dfs序没有办法实现。
- 可持久化线段树合并优化连边？
- 树链剖分，保留重儿子的可持久化线段树，加入轻儿子节点，也行。
- 边数减少为$O(n\log n)$~$O(n\ log^2\ n)$

【JZOJ6153】Tree

- 一颗树，节点度数不大于16，现要求选择若干条路径，两两无公共边，且所有边都被覆盖，求满足下列要求的方案数：

1. 选择的条数要最少；
2. 满足1的情况下要让选择的路径长度最大值最小。

n<=100000

2.5s

【JZOJ6153】Tree

- 前两个问是真NOIP题，第三个问先想普通DP f[i][j] 表示点i的子树伸下一条长为j的链的方案数。求出i所有儿子的f，然后用g[s]求出匹配完状态为s的儿子的方案数，然后枚举不伸出还是伸出哪一个儿子，利用g来求f的方案数，求g需要不断优化，有用的状态总数不会很多，但是求f要$O(n^2)$？
- 考虑长链剖分，和轻重链剖分类似，令重儿子为最深的儿子，然后对重链按深度建线段树，如果伸出来的是重链，就直接对f打个区间乘标记，如果伸出的是轻链，那就将系数乘到那一棵线段树上，然后加给f。
- 时间复杂度O(n log n + 可过)，空间复杂度O(n)。

Codechef MXMN

- 给两个图，求$\sum_{i<j} f_1(i,j) * f_2(i,j)$
- 其中f1(x,y)表示在图1中x,y所有简单路径中权值最小的值，f2(x,y)表示在图2中x,y所有简单路径中权值最小的值。
- 一条路径的权值定义为经过边的最大值。

- n,m<=200000

- 1s

Codechef MXMN

- 两边先弄出生成树，然后都搞个kruskal重构树，重构树有个优美的性质就是点权由下往上递增，那么两点之间的f就是重构树中lca的点权。
- 然后对一边的重构树建点分树，将另一棵树的每一个点都放着对应点在点分树上到根的路径，然后对另一棵树做点分树合并，两棵点分子树的点可以在点分树的lca处(x)计算答案，点分树合并的方法和线段树合并类似，如果对于两棵点分树，如果都不为空，那么就往下跑，否则取其中一个不为空的代替掉另一个。
- 要记着点分树上x的哪一个儿子是从原来朝上的边过来的，也就是说这个儿子要记下子树中的点在重构树中和x的lca的权值的和，其他儿子只需要记下个数，那么答案相当于，两个往下走的数量之积乘上x的权值，加上往上走的权值和乘上往下走的数量。
- 和线段树合并的复杂度类似，O(n log n)

Codechef SONATR

- 一棵树，每个点有权值1~n，每种权值都存在，m次操作，每次交换两个点的权值，并询问最大的L，权值为1~L的点形成一条路径。

- n<=500000

- 3s

Codechef SONATR

- 我们采用如下方式判断“连续路径”：
- 对于权值1~L的点，有一个值v，如果点出现了，则v++，如果同时它的最小儿子也出现了，则v--
- 如果1~L在一条祖孙后代链上，如果v=1，那么这些点连续
- 如果1~L不在一条祖孙后代链上，那么合法必然有lca的两个孩子的权值<=L，其次v=2。
- 那么对1~n开线段树，存v，每个点维护个set，存最小儿子的权值，交换就在对应线段树上改，这个还是相当容易维护的，换点的时候就在线段树改一下。
- 时间复杂度O(n)

谢谢大家！

- 若你能将上述每一道题目的解法各在1个小时之内完成，你此时必定是一位大国工匠！