数据结构

By i207M
From SJZEZ

数据结构是算法中非常重要的一部分，这一点大家肯定深有体会。

从简单到复杂，咱们再来过一遍数据结构。

每个算法后会跟一道例题，最后会讲一些数据结构综合应用的题。

PS：由于习惯，请自行`#define NOIP CSPS`

前缀与差分

简单又困难的一个算法，在很多时候它的应用让人意想不到。

POI2017 Kontenery

给定数组，每次操作为：每隔d个$+k$。

在线询问每个位置的数字。$n, q \leq 10^5$

一道水题。

对d分块，大的暴力，小的差分一下。

[LNOI2014]LCA

给出一个n个节点的有根树。设$dep[i]$表示点i的深度，$LCA(i,j)$表示i与j的最近公共祖先。 有q次询问，每次询问给出$l\ r\ z$，求$\sum_{l\le i\le r} dep[LCA(i,z)]$

$n \le 10^5$

一道非常经典的题。

发现区间的贡献可减，于是可以差分前缀和。

于是我们离线所有询问，按r排序。同时考虑转化 $\sum dep$ ，其实就是添加一个点的时候，到根的链上+1，询问就是求到根的路径和。树剖即可。

差分小总结

遇到区间询问值贡献可减的情况，必转化为前缀和相减。这样我们要处理的问题往往就减少一维或者从“支持插入、删除”变成“只需支持插入”。这往往对复杂度有巨大的影响。

树状数组

应该是大家接触到的第一个“高级数据结构”吧。

码量小，速度快，在很多题目中有奇效。

速度快：$\log$跑不满（一次i位置修改会运算`__builtin_popcount(i)`次，均摊$\log/2$）

比方说10^6的树剖+树状数组能跑过；

线段树套线段树太慢了，树状数组套线段树还凑活。（经常出现只有后者能过的情况）

进阶：

二维树状数组

树状数组区间加，区间求和

二维树状数组矩形加，矩形求和（P4514 上帝造题的七分钟）

[SDOI2009]虔诚的墓主人

P2154

对于一个空地(x,y)，如果上下左右分别有u，d，l，r棵常青树，且$u,d,l,r \geq k$，那么这块墓地的虔诚度为 $C(u,k)C(d,k)C(l,k)C(r,k)$ 。

可以从下往上扫描，对于每一行从左往右扫描（当然是排序后扫描点），对于连续的没有格点的部分， $C(l,k)C(r,k)$ 是固定不变的，而 $C(u,k)C(d,k)$ 的区间和可以树状数组维护，单点加，区间和；

线段树/主席树

超强数据结构，应用甚广。

为什么这两棵树一起说？因为它们之间的区别一般也就在动态开点上。

线段树的优越之处不仅仅是区间加，区间乘；其优越之处在于将分治结构显示的建立出来，可以在左右区间递归的维护信息，如果信息合并的复杂度可以接受，那么就可以使用线段树。

使用线段树的要求：

- 信息能否在每个点存下
- 信息是否可以快速合并
- 标记能否下放

线段树一定要找一个自己熟悉、喜欢的写法，然后一直写这种写法。

线段树优化常数：

当询问完全在一个区间时，直接`return`，不要再合并1次。

CF221D Little Elephant and Array

给定一个数列，有若干次询问，每次给定 $[l,r]$，问有多少个 x，在 $[l,r]$ 中出现恰好 x 次。$n,q\leq 10^5,Max\leq 10^9$

EOJ上有强制在线版本，题号找不到了。

用线段树扫描线的思想，维护每个后缀的答案。比方说现在扫描到了r，那么线段树中$[i,r]$的权值和就是答案。

为了方便统计，可以对每个权值开一个vector；

强制在线？主席树！

```
void prework()
{
    for(ri i=1;i<=n;++i) v[i].pb(0);
    for(ri i=1,t;i<=n;++i)
    {
        tre[i]=tre[i-1];
        if(a[i]>n||a[i]<=0) continue;
        v[a[i]].pb(i);
        t=v[a[i]].size()-1;
        if(t>=a[i])
        {
            if(t>a[i]) upd(tre[i],tre[i],1,n,v[a[i]][t-a[i]-1]+1,v[a[i]][t-a[i]],-1);
            upd(tre[i],tre[i],1,n,v[a[i]][t-a[i]]+1,v[a[i]][t-a[i]+1],1);
        }
    }
}
```

线段树分治、线段树扫描线、线段树合并、线段树...

They are important.

平衡树

数据结构的老大。

有很多形式，一个个分析一下。

Treap

应该是大家学的第一种平衡树吧。

优点？没啥优点，速度不是最快的；树的结构也不能改变。

优点就是能帮大家理解笛卡尔树（

Splay

很多人在用的平衡树。

优点？

可以控制树的形态，通过旋转操作解决序列问题。

中序插入的时间复杂度$O(n+\log n)$，然而这并没有什么用...因为她常数太大了。

写LCT只能用Splay...这可能是我唯一一种写Splay的情况。

缺点？

慢！

时刻记得旋转到根，不然TLE。

不能可持久化。

~~难以调试~~

讲道理fhq Treap（非旋Treap）吊打Splay好吗！

非旋Treap

（以下简称Treap）

在各种方面（除LCT），Treap都吊打Splay。（个人观点）

先说缺点：

比较慢，但是比Splay要快。

优点：

操作简单，不管干什么事，直接split+merge就好了。

不用惦记着标记下放。

不用惦记着时间复杂度。（说的就是你，Splay到根）

可以可持久化。

可以和各种算法结合。

树的结构不会轻易改变。（可以支持一边查一边找）

代码简单，思路自然。

替罪羊树

很值得学习。

原因是速度很快。在卡常的题目中有奇效。

最大的优点是不需要旋转，不会频繁的pushup。

因此，她和点分治/KD-Tree/其他大型数据结构可以很方便的结合起来。

AVL，SBT，红黑树...

没啥大用，看兴趣学吧。

[NOI2005]维护数列

用你喜欢的平衡树写法过了它，你的基础平衡树就过关了。

~~超长调试警告~~

STL

一定要善用。

priority_queue，set

OI中最常用的、好用的数据结构。

你说慢？

开O2后强无敌。

~~况且NOIP2018都用上i7 8700了，在乎什么常数（~~

~~考场上怒写multiset+三分两只log过了赛道修建。~~

但是priority_queue，set更多的时候是作为辅助工具出现的。解题的关键是找到适合的比较策略。

[NOI2010]超级钢琴

P2048

重点是把区间不重不漏地找出来。

(a, l, r, x)表示左端点为a，右端点所属范围在$[l, r]$，此范围内最优解为x处。

前缀和，ST表查区间最值，用堆维护最优解。

vector

STL老祖宗，小数据奇快。

LCT，CDQ分治...以及很多高级数据结构们

它们远远高于NOIP难度，暂且不表（

数据结构题目汇总

小试牛刀！

[GXOI/GZOI2019]旧词

P5305

就是前一道题的加强版。由于每个点的增量$d^k-\left(d-1\right)^k$是固定的，我们只需要多维护一个标记即可。

CF1129D Isolation

一道比较难的例题。

给定数列A，求划分A的方案数，使得每个区间内只出现一次的数的数量$\leq K$

很明显要DP，设$dp[i]$表示前缀i划分方案数。

转移就是把$dp[i] = \sum_{j<i}[count(j+1,i) <= K]dp[j]$

我们运用扫描线的思想很容易将"后缀内只出现一次的数的数量"统计出来。然后看看我们需要支持什么操作：区间加，查询区间$\leq K$的数的权值和。

貌似除了$O(n\sqrt{n}\log n)$之外没有好方法？

注意到区间加这个事情，其实我们每次只是前缀加，于是我们可以用更聪明的办法解决：差分。单点修改可以暴力重构解决。对于每块维护$g[i]$表示

$$\sum_{j}[sum[j+1,r] \leq i]f[j]$$

对每个块打标记，就可以从后往前扫每个块，$O(1)$查询了。

[POI2016]Korale

有n个带标号的珠子，第i个珠子的价值为$a[i]$。现在你可以选择若干个珠子组成项链（也可以一个都不选），项链的价值为所有珠子的价值和。现在给所有可能的项链排序，先按权值从小到大排序，对于权值相同的，根据所用珠子集合的标号的字典序从小到大排序。请输出第k小的项链的价值，以及所用的珠子集合。

$n \leq 1000000, k \leq \min(2^n, 1000000)$

套路

对于这种，求出第（前）k个序列（要求是无序的），n比较大，而k远远不到真实的上界（2^N），而是有一个严格的上界限制在10^6左右，这一定是有用意的，对，**它就是让我们一一求出来前k个；**

这道题类似“超级钢琴”。那道题要求输出子串和的第k大，我们采用的方法就是：每次取出当前最优解，然后只更新能够由它产生的，最优的解，再插进堆里，同时保证不会漏掉任何解；这样我们取出的第k个就是答案，这道题同理：

首先对于空集特判一下，然后**模拟DFS的过程**：我们是从左到右枚举每个数，可能选也可能不选，那么我们模拟这个过程，分两种情况：

1.加入一个数，那么显然是加入右面最小的数。

2.将最后一个数换成一个数，那么显然是将当前数换成刚好比当前数大的那个数，这里采用超级钢琴的做法，维护当前数的选择区间[l,r]，假设当前数是x，那么在[l,x)和(x,r]中寻找新的最小数，替换当前数。

但是如何保证字典序最小呢？首先首位必须最小，那么我们**将DFS的顺序反过来**即可；然后记录一下每个状态是由哪个状态得到的，在取出一个状态的时候顺便记录一个这个状态的排名（在相同的和的序列的排名），比较时比较上一个状态的排名即可。 -- by CQZhangyu

[POI2015]PUS

P3588

线段树优化建图+拓扑排序

总结一下

对于NOIP来说，最重要的是掌握好比较简单的数据结构的灵活应用，比方说

前缀与差分，树状数组，线段树...

以及**STL**的善用。可以加快平常做题速度；并且我们这届经常在大考中用STL，也并没有什么锅出现。

出题不会出裸数据结构，思维是最重要的。

多加练习吧！

祝大家CSPS取得好成绩