分块与 k-d tree

By i207M
Graduated from SJZEZ
Studying @THU

Powered by Marp

替罪羊树

替罪羊树的思想就是暴力重构。看到哪个点的子树不平衡，就拍扁，重构，同时回收已经删除的节点。

每次操作以后检查操作路径，找到最高的满足

$$\max(size(son_L), size(son_R)) > \alpha * size(this)$$

的结点，重建整个子树。

一般α取$0.75, 0.8$这样比较大的数。

建树?

插入?

删除? 懒惰删除!

```
void setup(int &x,int l,int r)
{
    if(l>r)
    {
        x=0;
        return;
    }
    int mid((l+r)/2); x=cur[mid];
    val[x]=st[mid].fi,cnt[x]=st[mid].se;
    setup(ls,l,mid-1); setup(rs,mid+1,r);
    up(x);
}
```

https://www.luogu.com.cn/blog/i207M/ti-zui-yang-shu-shuo-ju-jie-gou-xue-xi-bi-ji

替罪羊树的插入删除复杂度是均摊$O(\log n)$，查询复杂度是严格$O(\log n)$.

替罪羊树的思想在很多地方都有应用，比方说动态点分树什么的。

替罪羊树的优点是：很好写；常数优秀；树高是严格log的；只有 push up 操作，没有 rotate 操作。

rqy

前Oler/过期jk/橘猫/thu求真书院（数学）在读

12 人赞同了该回答

发布于 2018-02-15 11:14，编辑于 2018-02-20 20:45

带插入删除修改的区间第k大。强制在线。我只会替罪羊树套splay。

（如果有更好的算法能不能告诉我哇）

upd：由于被块状链表打脸，我这里只讨论理论时间复杂度（手动滑稽）

k-d tree

kdt 可以快速地维护几何区间的信息。

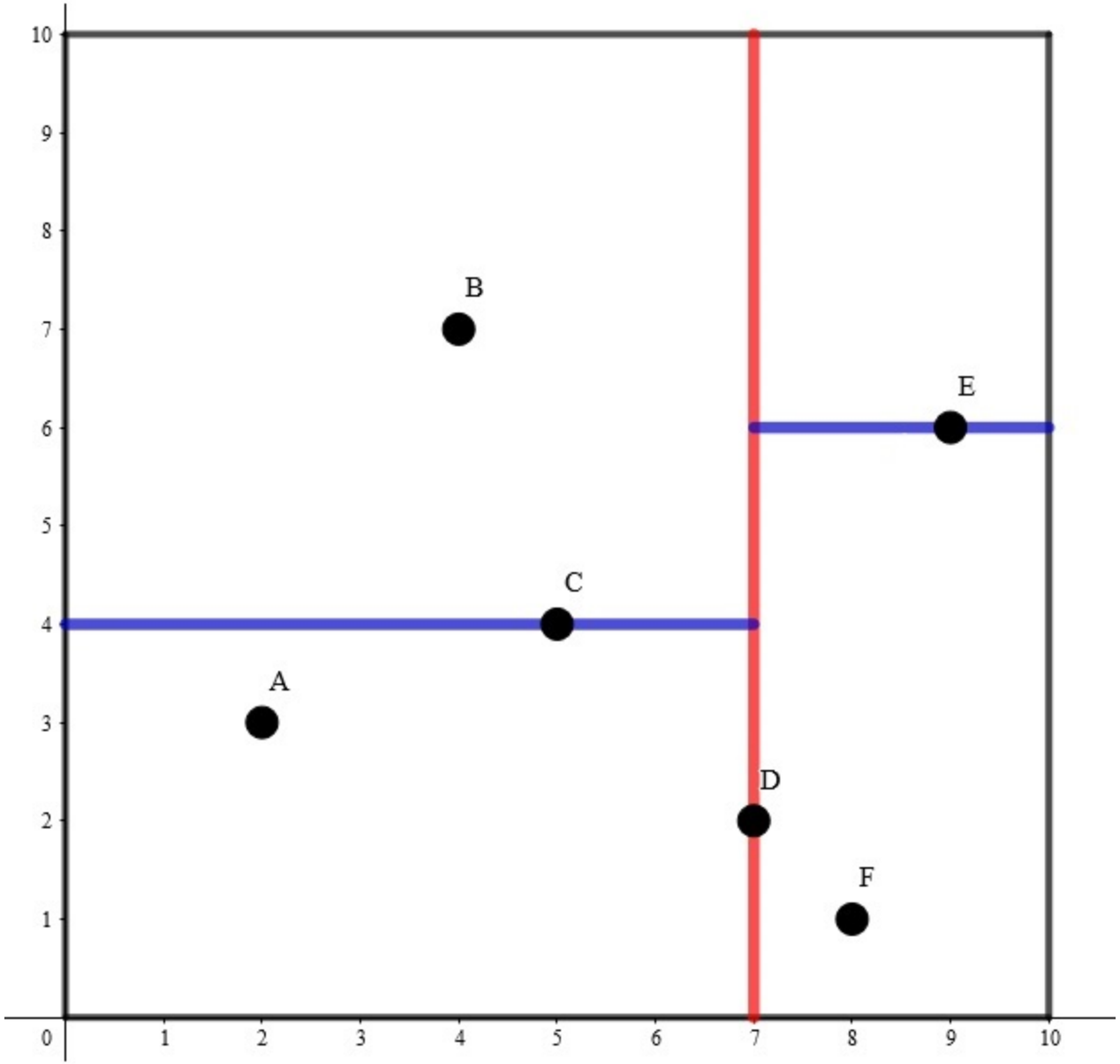
在 k 维空间中进行矩阵查询的复杂度是$O\left(n^{1-1/k}\right)$。
换句话说：$k=2$，还好；$k=3$，勉勉强强能用；$k>3$，算了吧。

但是对于随机数据，期望的查询复杂度是$O(\log n)$，因此可以用来骗分。

kdt 本质上是以棵二叉搜索树，树上的点与空间中的点一一对应。

构建一棵 k-D Tree，步骤如下：

1. 若当前空间中只有一个点，返回这个点。
2. 选择一个维度，将当前空间按照这个维度分成两个空间。
3. 选择切割点：在选择的维度上选择一个点（一般是中点），这一维度上的值小于这个点的归入一个空间（左子树），其余的归入另一个空间（右子树）。
4. 将选择的点作为这棵子树的根节点，递归对分出的两个空间构建左右子树，维护子树的信息。


10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
A
B
C
D
E
F

对于维度的选择，如果是 2 维、3 维，一般就轮流来；高维情况，可能要根据极差和方差来选择最优秀的一维进行分割。

于是树高就是$O(\log n)$.

回顾快排的思想，`nth_element`可以在期望$O(n)$的时间内找到中位数，并使得左边比中位数小，右边比中位数大。

这样，建树的复杂度就是$O(n \log n)$。

插入/删除依靠替罪羊树来实现。

那么查询呢?

k 近邻查询

P1429 平面最近点对（加强版）

这是一个玄学的复杂度，最劣是$O(n)$的，但是在随机数据下是$O(\log n)$，一般也跑不满。

~~因此是一个优秀的骗分算法~~

这种查询本质上就是在 kdt 上爆搜+剪枝。

我们可以维护一个子树中的所有结点在每一维上的坐标的最小值和最大值。

假设当前已经找到的最近点对的距离是 ans，如果查询点到子树内所有点都包含在内的长方形的 **最近** 距离大于等于 ans，则在这个子树内一定没有答案，搜索时不进入这个子树。

此外，还可以使用一种启发式搜索的方法，即若一个结点的两个子树都有可能包含答案，先在与查询点距离最近的一个子树中搜索答案。可以认为，**查询点到子树对应的长方形的最近距离就是此题的估价函数。**

```
double calc(int u)
{
    double x=max(0.0,q[0]-mx[u][0])+max(0.0,mn[u][0]-q[0]),y=max(0.0,q[1]-mx[u][1])+max(0.0,mn[u][1]-q[1]);
    return x*x+y*y;
}
void query(int l,int r,int d)
{
    if(l>r) return;
    int mid=(l+r)>>1;
    if(qid!=mid) ans=min(ans,dis(q,p[mid]));
    double dl=calc(ls[mid]),dr=calc(rs[mid]);
    if(q[d]<p[mid][d])
    {
        if(dl<ans) query(l,mid-1,d^1);
        if(dr<ans) query(mid+1,r,d^1);
    }
    else
    {
        if(dr<ans) query(mid+1,r,d^1);
        if(dl<ans) query(l,mid-1,d^1);
    }
}
```

https://www.luogu.com.cn/blog/i207M/k-d-tree-xue-xi-bi-ji

P4357 [CQOI2016]K 远点对

给定平面上的 n 个点 (x_i, y_i)，求欧几里得距离下的第 k 远无序点对之间的距离。

$n \leq 100000, 1 \leq k \leq 100, 0 \leq x_i, y_i < 2^{31}$

和上一道题类似，从最近点对变成了 k 远点对，估价函数改成了查询点到子树对应的长方形区域的最远距离。用一个小根堆来维护当前找到的前 k 远点对之间的距离，如果当前找到的点对距离大于堆顶，则弹出堆顶并插入这个距离，同样的，使用堆顶的距离来剪枝。

矩形查询

P4148

在一个初始值全为 0 的 $n \times n$ 的二维矩阵上，进行 q 次操作，每次操作为以下两种之一：

1. `1 x y A`：将坐标 (x, y) 上的数加上 A。
2. `2 x1 y1 x2 y2`：输出以 $(x1, y1)$ 为左下角，$(x2, y2)$ 为右上角的矩形内（包括矩形边界）的数字和。

强制在线。内存限制 `20M`。保证答案及所有过程量在 `int` 范围内。

$1 \leq n \leq 500000, 1 \leq q \leq 200000$

在查询矩形区域内的所有点的权值和时，仍然需要记录子树内每一维度上的坐标的最大值和最小值。如果当前子树对应的矩形与所求矩形没有交点，则不继续搜索其子树；如果当前子树对应的矩形完全包含在所求矩形内，返回当前子树内所有点的权值和；否则，判断当前点是否在所求矩形内，更新答案并递归在左右子树中查找答案。

k 维 kdt 的矩形查询的最坏时间复杂度是 $O\left(n^{1-\frac{1}{k}}\right)$ 的。

kdt 在 OI 中的应用频率比较低，虽然它在工业上广泛应用。

kdt 的出现场合，一般是查询二维矩形。相比于树套树，最大的优势在于内存占用是线性的。

比较遗憾的是，对于可以离线的情况，kdt 一般会被 cdq 分治、整体二分暴打。

分块

AND 序列

给出一个序列，求出一个最长的子序列，使得对于子序列的每一项，与上它前面的每一项都等于它本身；$n, a_i <= 2^{17}$

我们有两种做法：

1.在尝试求一个数的值时，我们可以枚举他前面的那个数，$f[i]$表示以 i 结尾的最长序列；修改$O(1)$，查询$O(2^{17})$；

2.在求完一个数的值时，我们可以用它更新数组$g[i]$表示，一个数字如果是 i，它的最长序列长度；修改$O(2^{17})$，查询$O(1)$；

我们可以将这两个糅合起来；前 9 位用第一种，后 9 位用第二种，这样就均摊复杂度了。

复杂度平衡

借助均值不等式来平衡复杂度是一个十分常用的技巧。
例如双向搜索、根号分类讨论等。

P3645 [APIO2015]雅加达的摩天楼

https://www.luogu.com.cn/problem/P3645

也不需要显式建图，直接 BFS 转移即可。

状态 (i,j) 表示，当前在第 i 个点，当前的 doge 跳跃能力为 j。

当 $j \le \sqrt{n}$ 时，只有 $n\sqrt{n}$ 个状态。

当 $j > \sqrt{n}$ 时，只有 $m\sqrt{n}$ 个状态（最多有 m 只 doge，每只 doge 只有 $\frac{n}{j} < \sqrt{n}$ 个可行位置）。

(i,j) 可以转移到 $(i-j,j)(i-j \ge 0)$ 和 $(i+j,j)(i+j < n)$，同时对于第一次访问到的 i，把初始在 i 的所有 doge 加入队列。

状态判重时使用 `std::set` 会 TLE，可以用 hash 或者 `std::bitset`。

洛谷

分块基本操作

区间待修改最大值/最小值/和（不用线段树）
区间修改、询问区间内超过 k 的数字个数

单点修改、区间第 k 大?
P2617 Dynamic Rankings

暴力做法是先二分答案，再分块查询排名，$O(n\sqrt{n}\log n\log V)$.

有没有$O(n\sqrt{n})$做法?

所以可以将数列分成$\sqrt{n}$块，同时将权值分块，令 val[i][j] 表示前 i 块权值在第 j 个值域的数的个数，w[i][j] 表示前 i 块权值为 j 的数的个数。修改暴力跑一遍前缀和即可，询问先早答案暴力从小到大枚举在哪一个值域里，然后在值域里暴力枚举答案即可。

区间修改、区间第 k 大?

数列分块入门 6

给出一个长为 n 的数列，以及 n 个操作，操作涉及单点插入，单点询问。

根号重构

根号重构围绕着修改操作，每隔若干次修改后暴力重构数据结构。

2019 牛客多校 8-D Distance

$n \times m \times h \leq 10^5$ 的三维空间内有 $q \leq 10^5$ 次操作，分两种：

1. 给点(x,y,z)打标记；
2. 询问离位置 A(x,y,z)最近的标记点到 A 的最近曼哈顿距离。

k-d tree 就别想过了，复杂度太高。

如果所有的询问操作都在打标记后，那么从所有的标记点开始做一次多源 bfs 求最短路，则询问点的 dis 就是到离它最近的点的曼哈顿距离。

我们每$\sqrt{q}$次，暴力跑一遍 bfs 就行了。

还有一种做法是讨论曼哈顿距离的三种情况，然后用三维树状数组维护，但是跑起来比分块慢。

题

可能会混入一些奇奇怪怪的东西。

CODECHEF FNCS

一个长度为 n 数组以及 n 个区间$[l_i, r_i]$，$f(i)$定义为区间$[l_i, r_i]$的和。
两种操作，一个是单点改值，另一个是查询$\sum_{i=l}^{r} f(i)$

将 n 个区间分块，预处理出数组中的每一个数对每一个块的影响个数，维护每个块$f(i)$总和，然后改值的时候对所有的$\sqrt{n}$个块暴力更新块的$f(i)$总和，然后查询的时候对于整块$O(1)$得到答案，非整块用树状数组暴力查询。

如何去掉 log?

块套块。
很巧妙的复杂度平衡。

能量石

给出一张图，每个点有点权。
支持加边、删边，修改点权，询问有连边的点对的点权和的最大值。
$n,m \leq 200000$

对度数分类讨论。

P4117 [Ynoi2018] 五彩斑斓的世界

一个长为 n 的序列 a，有 m 次操作

1. 把区间 $[l,r]$ 中大于 x 的数减去 x。
2. 查询区间 $[l,r]$ 中 x 的出现次数。

对于 100% 的数据，$1\le n\le 10^6$，$1\le m\le 5\times 10^5$，$1\le l\le r\le n$，$0\le a_i,x\le 10^5+1$。

这种“把大于 x 的数减去 x”非常难办。但是这也有一个好处，就是我们每修改一次，值域就粗略的缩小 1.

具体的，我们这样实现：

对每个点建立并查集，把权值的修改变成并查集的修改。

- 当块内的最大值$\geq 2v$，我们反着做，把$\leq v$的+v，然后整体-v。
- 否则就照着题意做。

在零散的修改时，直接暴力重构整个块。

旅行

给出一棵树，
每次询问中，给出一条链 s~t，每天在链上走 k 步然后在到达的那个点住下来。
每个点的宾馆有一个价格，求每次询问的住店的总花费。
n,m ≤ 10^5

预处理$\leq \sqrt{n}$的步数。

CF802O April Fools' Problem

https://www.luogu.com.cn/problem/CF802O

线段树模拟网络流，或者 wqs 二分。

P4475 巧克力王国

对于每一块巧克力，我们设 x 和 y 为其牛奶和可可的含量。由于每个人对于甜的程度都有自己的评判标准，所以每个人都有两个参数 a 和 b，分别为他自己为牛奶和可可定义的权重，因此牛奶和可可含量分别为 x 和 y 的巧克力对于他的甜味程度即为 $ax+by$。而每个人又有一个甜味限度 c，所有甜味程度大于等于 c 的巧克力他都无法接受。每块巧克力都有一个美味值 h。

现在我们想知道对于每个人，他所能接受的巧克力的美味值之和为多少。

对于 100%的数据，$1<=n,m<=50000,-10^9<=a_i,b_i,x_i,y_i<=10^9$。

首先用 KDtree 维护矩阵内的权值总和以及分裂点权值，之后在查询的时候把矩阵的四个点都 check 一遍。

如果都满足就直接加上总权值，如果都不满足就 return，否则 check 分裂点更新答案之后递归左右分裂区间。

铃铛计数问题

给一棵树，点带权，size(i)表示 i 号点子树中节点的权值和，多组询问修改点权或求 $\sum_{i=l}^{r} size(i)$。

和 CODECHEF FNCS 这道题本质相同。

对原编号分块。设$f_{i,r}$表示点 i 对块 r 的贡献，这个可以 dp 出来，从父亲那里转移。这样我们大块询问和大块修改就可以完成了。

考虑小块询问和小块修改，考虑放到 dfs 序上，就成了单点改，区间询问，然后再分块就可以了。

P4135 作诗

给定 n 个不大于 c 的正整数 $a_1 \dots a_n$ 和 m 组询问，每次问 $[l, r]$ 中有多少个数出现正偶数次。强制在线。

对于 100% 的数据，$1 \leq n, c, m \leq 10^5$，$0 \leq a_i \leq c$，$1 \leq l, r \leq n$。

$cnt[i][j]$ 表示第 i 块开始到结尾，j 的出现次数；

$f[i][j]$ 表示 i 块开头到 j 块末尾的答案。

其实这道题的难度在于预处理。

开灯

有 n 盏灯，每个灯有颜色
每次同时翻转某种颜色的所有灯的状态
求极长的开着灯的连续段的个数
n, m ≤ 10^5

转化为颜色之间连边，求边两边状态不同的数的个数；
对度数分类讨论。

P4785 [BalticOI 2016 Day2]交换

https://www.luogu.com.cn/problem/P4785

每个点可以取的值只有 log 种。

P3591 [POI2015] ODW

给定一棵 n 个点的树，树上每条边的长度都为 1，第 i 个点的权值为 a_i。

Byteasar 想要走遍这整棵树，他会按照某个 1 到 n 的全排列 b 走 $n-1$ 次，第 i 次他会从 b_i 点走到 b_{i+1} 点，并且这一次的步伐大小为 c_i。

对于一次行走，假设起点为 x，终点为 y，步伐为 k，那么 Byteasar 会从 x 开始，每步往前走 k 条边，数据保证了每次行走的距离是 k 的倍数。

请帮助 Byteasar 统计出每一次行走时经过的所有点的权值和。

$2 \leq n \leq 50000$

考虑我们让所有$k > \sqrt{n}$的暴力跳，预处理所有$k \leq \sqrt{n}$的情况即$sum[i][j]$表示i往上每j级祖先选一个点，一直到根节点选的点的情况和，查询就类似树上差分就好了

那么这样暴力跳的复杂度是$O(n\sqrt{n}\log n)$，预处理是$O(n\sqrt{n})$

对时间进行优化，可以使用长链剖分，每次$O(1)$查询k级祖先，这样我们的总复杂度就可以做到$O(n\log n + n\sqrt{n})$了

所以说这是一道长链剖分模板题/doge

P4547 [THUWC2017]随机二分图

https://www.luogu.com.cn/problem/P4547

Map 状压记搜。

为了防止重复统计，每次只能转移 lowbit 的出边。

对于边组如何处理?

先来考虑第二类边组，如果我们强行认为这两个边都是独立的话，第一条边单独出现在匹配方案中的概率是 50%，第二条边单独出现在匹配方案中的概率是 50%，两条边同时出现在匹配方案中的概率是 25%。

而真实情况是第一条边,第一条边单独出现在匹配方案中的概率是 50%，同时出现在匹配方案的概率也是 50%，出现这种情况是因为我们只关心出现在匹配方案中的边是否出现而不关心这张图里到底出现什么边。

如果这两个边的左部点相同那么我们直接认为两条边相互之间独立，否则我们把左部点较大的边挂在左部点较小的边上，如果选择了那个左部点较小的边我们有 25%的概率立即选择另一条边(前提是你能同时删掉这 4 个点)，这样的话我们概率就对上了

所以我们多加一条+-25%的附加边，在考虑选择那条边之后，立即考虑它的附加边。

P3826 [NOI2017] 蔬菜

https://www.luogu.com.cn/problem/P3826

大家的网络流水平如何?

Thank you

祝大家 CSP/NOIP 取得好成绩！

如果你比较强，可以尝试以下几个题目：

P4119 [Ynoi2018] 未来日记，P3769 [CH 弱省胡策 R2]TATT