点分树（动态点分治）

by cxl

czc 在2023-5-20有少量增补

【前言】

点分治是一种树上分治算法，常用以处理树上路径相关信息的统计。在点分治的基础上加以变化，构造一颗支持快速修改的重构树，就成了点分树。

一【算法理解及复杂度分析】

前置芝士：需要有良好的 点分治 基础。

点分治的核心思想在于依据重心划分子连通块，其良好的性质保证了最多只会分治 $\log n$ 层。有了这一特性，便可使用各种暴力计算答案。

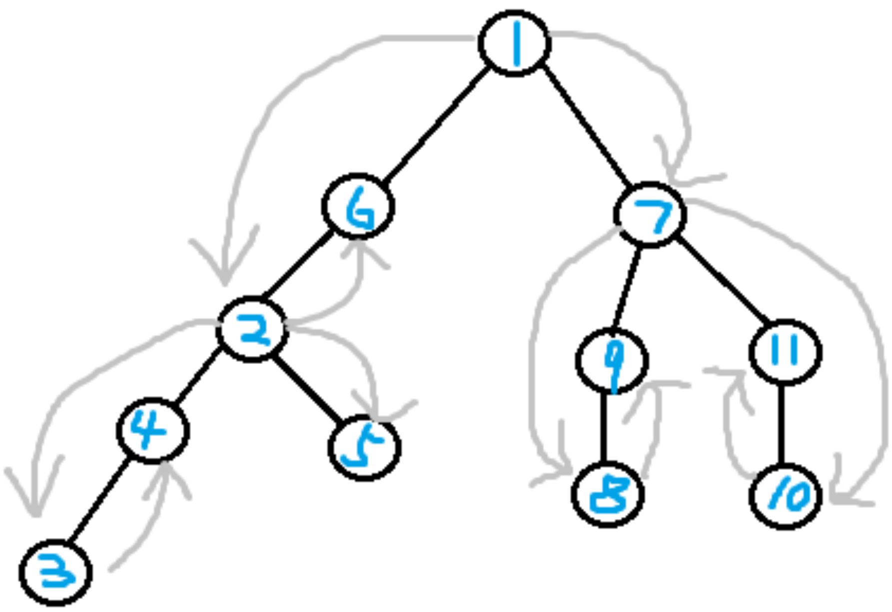
那么我们按照分治递归的顺序提一颗新树出来，易知树高是 $O(\log n)$ 的。

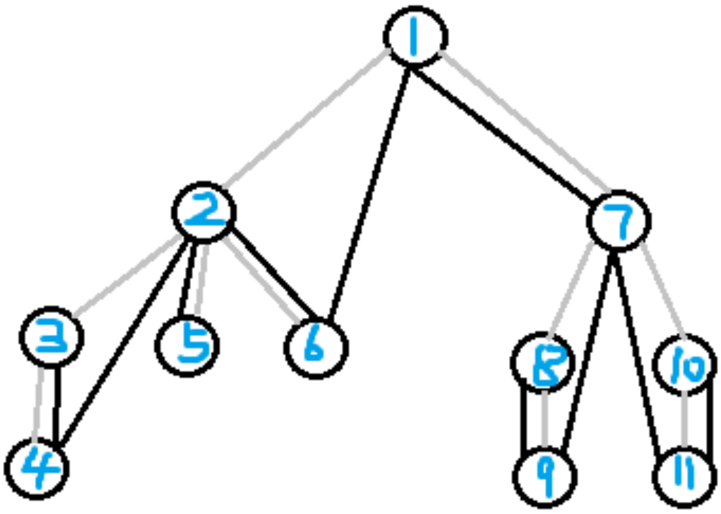
具体地说，对于每一个找到的重心，将上一层分治时的重心设为它的父亲，得到一颗大小不变、最多 $\log n$ 层的虚树（或者理解为重构树。亦可称点分树，意义一样）。

在这颗虚树上，奇妙的性质产生了：虚树上所有点的子树大小之和为 $O(\log n)$ 。

证明很简单：每个点会被从根到它的路径上最多 $\log n$ 个祖先所统计，因此总复杂度为 $\sum_{i=1}^{n}depth(i)=O(n\log n)$ 。

如果要对某个点进行修改操作，直接在虚树上暴力跳父亲，改变 $O(\log n)$ 个祖先的虚子树信息即可。

> Q: 虚子树的信息与原树有啥关系呢?
> A: 点 x 在虚树上的子树集合就是原树中以 x 为重心（分治中心）时所囊括到的连通块。
> 如下图（黑边为原树，灰边为虚树，蓝色编号为分治的顺序）：
>
> 
>



> 以 1 为重心做分治时，所囊括的连通块为 ${1,2,3,4,5,6,7,8,9,10}$，删去该点后被划分为了 ${2,3,4,5,6},{7,8,9,10,11}$ 两个子连通块；
> 以 2 为重心做分治时，所囊括的连通块为 ${2,3,4,5,6}$，删去该点后被划分为了 ${3,4},{5},{6}$ 三个子连通块；
> 以 7 为重心做分治时，所囊括的连通块为 ${7,8,9,10,11}$，删去该点后被划分为了 ${8,9},{10,11}$ 两个子连通块；
> ……

那么能用它求什么呢?

比如我们要统计某个点 x 到其他所有点的距离之和，即 $sum_{y=1}^{n}dis(x,y)$ 。对于任意一个 y，首先在虚树上找到它与 x 的 lca（或者说囊括连通块同时包含 x,y 的所有虚树节点中深度最深的那一个），易知在以此点为重心划分子连通块时 x,y 会首次被分割开来，因此该点必定在原树的 x,y 路径上。

所以我们只需要在这些 lca 的虚子树中寻找 y 即可，此时记录虚子树信息的作用便显现出来了。

而对于一个 x，可能的 lca 最多存在 $log n$ 个，因此通常使用暴力枚举+简单容斥的方法来统计 y 的贡献。具体见下文 【 计算贡献】。

二【算法实现】

【模板】点分树 / 震波

【题目大意】 维护一颗带点权树，需要支持两种操作：修改 x 的点权，查询与点 x 距离不超过 K 的点权值之和。

1.【建点分树】

找到第一个重心 rt 后，先遍历整颗树得到 rt 的子树信息，然后删去 rt，得到若干个连通块（rt 的不同子树）。分别找到这些子树里的重心（在虚树上使其成为 rt 的儿子），然后递归处理这些子连通块。

实际上大体和点分治是相同的，只是将点分治中“计算答案”的操作改为了“统计虚子树信息”。

2.【计算贡献】

以下用 fa_i 表示点 i 在虚树上的父亲，$subtree(i)$ 为点 i 在虚树上的子树集合，$fatree(i)$ 为点 i 在虚树上的祖先集合，$dis(i,j)$为 i,j 两点在原树上的距离，A_i 为点 i 的点权。

设：

$$
\huge f_1(i,j)=\sum_{x \in subtree(i),dis(x,i) \leq j}A_x
$$

即虚树上 i 的子树中与 i 距离小于等于 j 的点权值之和；

为了除去某一个虚儿子子树的贡献，还需要：

$$
\huge f_2(i,j)=\sum_{x\in subtree(i),dis(x,fa_i)\leq j}Ax
$$

$

即虚树上 i 的子树中与 fa_i 距离小于等于 j 的点权值之和。

在一次查询 x,k 中，对于虚树上的一对父子节点$ (i,fa_i)$，$subtree(fa_i)-subtree(i)$ 对答案的贡献为

$$
\huge G(i,fa_i)=f_1(fa_i,k−dis(x,fa_i)) − f_2(i,k−dis(x,fa_i))
$$

则有

$$
\huge ans(x,k)=f_1(x,k)+\sum_{i\in fatree(x),fa_i\neq 0} G(i,fa_i)
$$

注意前面那个$f_1(x,k)$ 是因为容斥求和后 $subtree(x)$ 没有被统计进去，所以要单独拿出来算。

修改点权时同理，把所有对于 f_1,f_2 的查询操作换成修改就可以了。

3.【维护信息】

计算 dis 可以用树剖求 $text{LCA}$ 在线查询。

也可以用欧拉序 + $text{ST}$ 表。或者用在每次找完根后，dfs 遍历一下子树预处理 dis 数组。两者都是花费 $O(nlogn)$ 的时间省下了 $O(q\log^2 n)$ 的时间，但经实际测试，ST 表被树剖吊起来锤，预处理与树剖不相上下，emm...过于柯怕

维护 f_1,f_2 可以对每个节点开一棵动态开点线段树，下标为 dis（即 f_1,f_2 的第二维）。时间复杂度为 $O(n\log^2 n)$，空间按照 dis 范围的上界压缩一下可以做到 $O(n\log n)$（如果偷懒统一使用同一个上界 $[0,n]$ 可能会达到 $O(n\log^2 n)$。但线段树常数略大，可能会被卡（~~虽然我过子~~），换成树状数组会稳一点。

> Q: 树状数组咋动态开点啊?
>
> A: 当然是用动态分配内存的 $vector$ 啊!

4.【实现细节】

- 子连通块大小不要直接用 size(to)（这都是从点分治那里遗留下来的问题了，但依旧有人不重视）
- 一般不需要把虚树的实际形态建出来，但如果能用到的话，注意不要和原树搞混。
- 树状数组大小要设置得当。设 $now=|subtree(i)$$|$，由于 i 在$subtree(i)$ 中为重心，所以 $f_1(i,j)$ 中 j 的值域为 $[0\sim \frac{now}{2}]$， $f_2(i,j)$ 中 j 的值域为 $[1\sim now]$ 。由于下标可以为 0，在树状数组内部还需要统一向后移一位。也就是说 f_1,f_2 的大小要分别设置为$\frac{now}{2}+1$ 和$ now+1$ 。
- 如果预处理了每个点在虚树上的祖先，修改 / 查询 中跳父亲时需要倒序枚举（具体见代码）。
- 关于卡常：点分树做题经常会用到$ \text{vector}, \text{set}, \text{multiset}, \text{priority queue} $之类的东西，如果您嫌弃它们太慢且不愿开 O2，自备几个能代替 $\text{STL} $的模板吧....

5.【Code】

（线段树的代码就不放了）

【在线查询 Dis】

```
1 https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/在线查询Dis.cpp
```

【预处理 Dis】

```
1 https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/预处理Dis.cpp
```

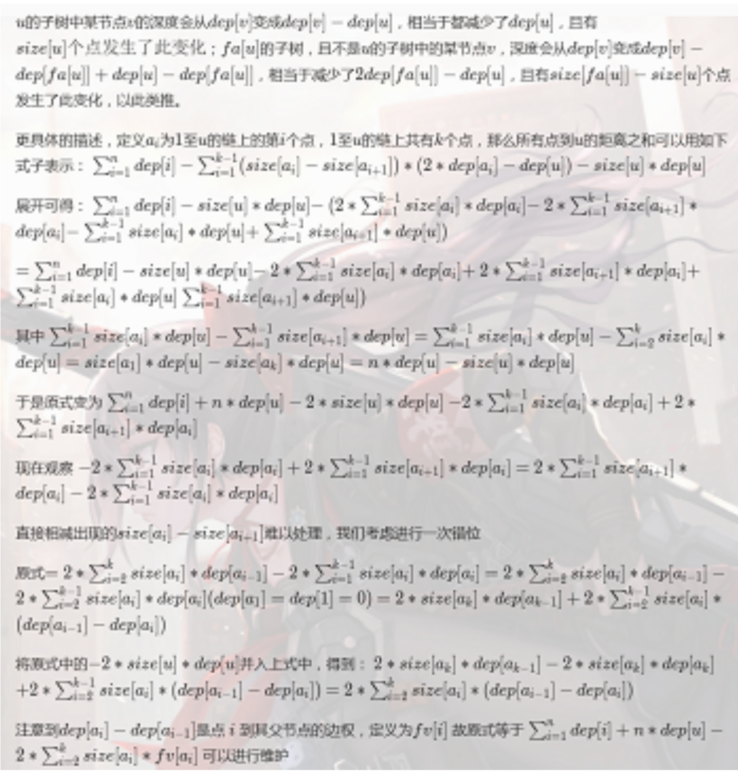
三【常见套路、经典例题】

（注： 后面的题都不再用文字具体阐述 f_1,f_2 含义，且只放核心代码）

1.【开店】

【题目大意】 维护一颗带点权、边权树，每次给出 x,l,r，查询$\large \sum_{l\leqslant A_y \leqslant r}dis(x,y)$其中 A_y 为 y 的点权。

说起这道题，想起一张图（来自 $ \text{hychyc} $ 巨佬的主席树做法）



u的子树中某节点v的深度会从$dep[v]$变成$dep[v]-dep[u]$，相当于都减少了$dep[u]$，且有$size[u]$个点发生了此变化；$fa[u]$的子树，且不是u的子树中的某节点v，深度会从$dep[v]$变成$dep[v]-dep[fa[u]]+dep[u]-dep[fa[u]]$，相当于减少了$2dep[fa[u]]-dep[u]$，且有$size[fa[u]]-size[u]$个点发生了此变化，以此类推。

更具体的描述，定义a_i为1至u的链上的第i个点，1至u的链上共有k个点，那么所有点到u的距离之和可以用如下式子表示：$\sum_{i=1}^{n}dep[i]-\sum_{i=1}^{k-1}(size[a_i]-size[a_{i+1}])*(2*dep[a_i]-dep[u])-size[u]*dep[u]$

展开可得：$\sum_{i=1}^{n}dep[i]-size[u]*dep[u]-(2*\sum_{i=1}^{k-1}size[a_i]*dep[a_i]-2*\sum_{i=1}^{k-1}size[a_{i+1}]*dep[a_i]-\sum_{i=1}^{k-1}size[a_i]*dep[u]+\sum_{i=1}^{k-1}size[a_{i+1}]*dep[u])$

$=\sum_{i=1}^{n}dep[i]-size[u]*dep[u]-2*\sum_{i=1}^{k-1}size[a_i]*dep[a_i]+2*\sum_{i=1}^{k-1}size[a_{i+1}]*dep[a_i]+\sum_{i=1}^{k-1}size[a_i]*dep[u]\sum_{i=1}^{k-1}size[a_{i+1}]*dep[u])$

其中$\sum_{i=1}^{k-1}size[a_i]*dep[u]-\sum_{i=1}^{k-1}size[a_{i+1}]*dep[u]=\sum_{i=1}^{k-1}size[a_i]*dep[u]-\sum_{i=2}^{k}size[a_i]*dep[u]=size[a_1]*dep[u]-size[a_k]*dep[u]=n*dep[u]-size[u]*dep[u]$

于是原式变为$\sum_{i=1}^{n}dep[i]+n*dep[u]-2*size[u]*dep[u]-2*\sum_{i=1}^{k-1}size[a_i]*dep[a_i]+2*\sum_{i=1}^{k-1}size[a_{i+1}]*dep[a_i]$

现在观察$-2*\sum_{i=1}^{k-1}size[a_i]*dep[a_i]+2*\sum_{i=1}^{k-1}size[a_{i+1}]*dep[a_i]=2*\sum_{i=1}^{k-1}size[a_{i+1}]*dep[a_i]-2*\sum_{i=1}^{k-1}size[a_i]*dep[a_i]$

直接相减出现的$size[a_i]-size[a_{i+1}]$难以处理，我们考虑进行一次错位

原式$=2*\sum_{i=2}^{k}size[a_i]*dep[a_{i-1}]-2*\sum_{i=1}^{k-1}size[a_i]*dep[a_i]=2*\sum_{i=2}^{k}size[a_i]*dep[a_{i-1}]-2*\sum_{i=2}^{k-1}size[a_i]*dep[a_i](dep[a_1]=dep[1]=0)=2*size[a_k]*dep[a_{k-1}]+2*\sum_{i=2}^{k-1}size[a_i]*(dep[a_{i-1}]-dep[a_i])$

将原式中的$-2*size[u]*dep[u]$并入上式中，得到：$2*size[a_k]*dep[a_{k-1}]-2*size[a_k]*dep[a_k]+2*\sum_{i=2}^{k-1}size[a_i]*(dep[a_{i-1}]-dep[a_i])=2*\sum_{i=2}^{k}size[a_i]*(dep[a_{i-1}]-dep[a_i])$

注意到$dep[a_i]-dep[a_{i-1}]$是点 i 到其父节点的边权，定义为$fv[i]$ 故原式等于 $\sum_{i=1}^{n}dep[i]+n*dep[u]-2*\sum_{i=2}^{k}size[a_i]*fv[a_i]$ 可以进行维护

（窝还是老老实实打点分树吧...）

1.【分析】

统计贡献还是和模板题一样的套路，设：

$$
\huge f_1(i,j)=\sum_{x\in subtree(i),A_x\leqslant j}dis(x,i) \\ \huge f_2(i,j)=\sum_{x\in subtree(i),A_x\leqslant j}dis(x,fa_i)
$$

在查询点 x 的答案时，上面只算了合法点到祖先的距离，漏掉了 x 到 fa_i 这一段，所以还要记点数：

$$
\huge g_1(i,j)=\sum_{x\in subtree(i),A_x\leqslant j}
$$

（这里就不需要设 g_2 了，直接相减即可）。

查询时差分一下，只算权值小于等于 k 的距离之和。

则有

$$
\huge ans(x,k)=f_1(x,k)+\sum_{i\in fatree(x),fa_i\neq 0} G(i,fa_i)
$$

其中

$$
\large G(i,fa_i)=f_1(fa_i,k)-f_2(i,k)+dis(x,fa_i)\times (g_1(fa_i,k)-g_1(i,k))
$$

由于这题没有修改操作，所以直接在建虚树时开个 $text{vector}$ 预排序，顺便求出前缀和，查询时二分位置即可。

另外，为减小常数可以把柿子拆开，在一次 k 查询中对于虚树上每个祖先只使用一次二分。

空间复杂度：$O(n\log n)$。

时间复杂度：$O((n+q)\log^2 n) $ 。

2.【Code】

```
https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/开店.cpp
```

2.【幻想乡战略游戏】

【题目大意】 维护一颗带点权、边权树（树上点的度数不超过 20）。现有若干次修改点权的操作，每次操作结束后您需要选出一个核心点 x 使得 Fx 最小，其中 $F(x)=\sum_{i=1}^{n}dis(x,i)\times A_i$，$A_i$ 为 i 的点权。

1.【分析】

这题关键在于分析性质~~猜结论~~。

假设当前核心为 x，将 x 设为树的根，定义 $ S_A(i) $ 为 i 子树内所有点的点权之和。

对于 x 的任意一个儿子节点 y，若将核心改为 y，那么$\Delta F_{x \to y} =F(y)-F(x)=(S_A(x)-2S_A(y))\times dis(x,y)$。选 y 更优当且仅当满足 $\Delta F_{x \to y}<0$即 $S_A(x)<2S_A(y)$，易知对于一个 x 满足该式的 y 最多只存在一个。

于是一个经过优化的暴力就产生了：每次询问从根开始，不断进入比它更优的儿子，如果找不到更优则说明本身已是最优。

但很多人只讲到这些就开始扯点分树信息维护了，实际上是不严谨的，我们还需要证明："在以 x 为根时，若 y 没有 x 优秀（表现为 $Delta F_{x to y}>0$），则 y 子树里的点也一定没有 x 优秀"。

> 设$ y$ 的儿子为$ y'$，考虑还是在以 $x $为根的前提下记算贡献：
> $\Delta F_{y \to y'}=(S_A(y)-2S_A(y')+S_A(x)-S_A(y))\times dis(y,y')=(S_A(x)-2S_A(y'))\times dis(y,y')$
> 又因为$ S_A(y)\geqslant S_A(y')$（这题 $A_i $应该是不为负的，所以能得出这个关系式）
> 若$ \Delta F_{x \to y}>0 则 S_A(x)>2S_A(y)\geqslant 2S_A(y')$
> 因此$ \Delta F_{y \to y'}>0,\Delta F_{x \to y'}=\Delta F_{x \to y}+\Delta F_{y \to y'}>0 $。
> 证毕。

暴力单次询问复杂度 $O(20depth)$，而点分树有着 $depth=O(\log n) $的天然优势，我们可以把跳儿子的过程转移到点分树上进行。具体的说，从第一个重心开始，枚举它在原树上的儿子，若找到了比他更优的点，则跳到该子树（或者说子连通块）的重心。容易证明这样做一定不会错过最优解。

现在的问题只剩下快速计算$ F(x) $了，按照套路，设：

$f_1(i)=\sum_{x\in subtree(i)}dis(x,i)\times A_x$

$f_2(i)=\sum_{x\in subtree(i)}dis(x,fa_i)\times A_x$

$g_1(i)=\sum_{x\in subtree(i)}A_x$

则有$F(x)=f_1(x)+\sum_{i\in fatree(x),fa_i\neq 0} G(i,fa_i)$

其中$G(i,fa_i)=f_1(fa_i)-f_2(i)+dis(x,fa_i)\times (g_1(fa_i)-g_1(i))。$

由于这题的 $f_1,f_2,g_1 $ 都只有一维，直接用数组存下来就好了。

空间复杂度：$O(n\log n) $。

时间复杂度：$O(n\log n+20q\log^2n)$。

2.【Code】

```
https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/幻想乡战略游戏.cpp
```

3.【小清新数据结构题】

【题目大意】 维护一颗带点权树，需要支持两种操作：修改 x 的点权，查询以点 x 为根时的$\large \sum_{i=1}^{n}(\sum_{j\in sub(i)}A_j)^2$,其中$A_j$ 为 j 的点权，$sub(i) $ 为点 i 子树内的节点集合。

1.【分析】

一看就知道是个~~丧心病狂~~拆柿子题。

上公式（以 x 为根）：

设 $sum=\sum_{i=1}^{n}A_i，S(i)=\sum_{j\in sub(i)}A_j$

$\sum_{i=1}^{n}S_i=\sum_{i=1}^{n}A_i(dis(i,x)+1)=sum+\sum_{i=1}^{n}dis(i,x)\times A_i（$（每个点会在自己以及它的 dis 个祖先处被统计到）

设 $F=\sum_{i=1}^{n}dis(i,x)\times A_i$（用与 [幻想乡战略游戏] 同样的方法可以求得）

由于 $\sum_{i=1}^{n}S_i(sum-S_i) $ 始终为一个定值（对于每条边 x,y，两边的连通块点权之和乘起来然后再求和），我们可以先$O(n) $预处理出来，设为 tmp 。

则 $ans(x)=\sum_{i=1}^{n}S_i^2=sum\sum_{i=1}^{n}S_i-tmp=sum(sum+F)-tmp $

空间复杂度：$O(n\log n) $。

时间复杂度：$O((n+q)\log n) $

2.【Code】

代码就不放了，毕竟和上一道差不多，只是多了个 dfs 预处理 tmp。

4.【成都七中】

【题目大意】 由一颗树，树上每个节点有一种颜色，每次查询给出 l,r,x，求保留树上编号在 $[l,r]$ 内的点，x 所在联通块中颜色种类数。

1.【分析】

这题比较难想。

先建出点分树，对于一次查询 l,r,x，在点分树上 x 的祖先中找到深度最小的点 pa，且满足 x 只经过编号 $[l,r]$ 内的点在原树上能到达 pa 。记一下每个点 i 到虚树祖先的路径上所经过的节点编号最小/大值就可以轻松求得（分别记为分别记为 $d_{min}(i,j) 和 d_{max}(i,j)$。

分析可知：x 只经过编号 $[l,r]$ 内的点所在的连通块被完全包含在了$subtree(pa)$ 中（虚子树）。我们把本次询问放到 pa 节点处，最后再统一离线处理。

枚举虚树上的点 rt，处理该节点处的询问时，对于任意一个 l,r,x，满足$l\leqslant d_{min}(i,rt) $且$d_{max}(i,rt)\leqslant r $的 i 即为与 x 在同一连通块内的点，现需要统计这些点的颜色种类。

显然是个偏序问题，把询问和节点信息放一起按 l 排序，指针从右往左扫，同时记录每种颜色节点右端点最小的位置，再开一棵树状数组维护每个位置上的数量，便可直接查询了。

空间复杂度：$O(nlog n)$ 。

时间复杂度：$O(nlog^2 n)$ 。

2.【Code】

```
https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/成都七中.cpp
```

5.【Iqea】

【题目大意】 二维平面上给出若干个点的坐标，在这些点处打好地基（保证为一个四连通块，且不会出现封闭的空地）。有两种操作：在 x,y 处建造商店，询问离 x,y 最近的商店与 x,y 的距离大小。两点距离定义为只经过有地基的点的最短路径长度，相邻两个格子距离为 1 。保证每次给出的坐标处一定有地基。

1.【分析】

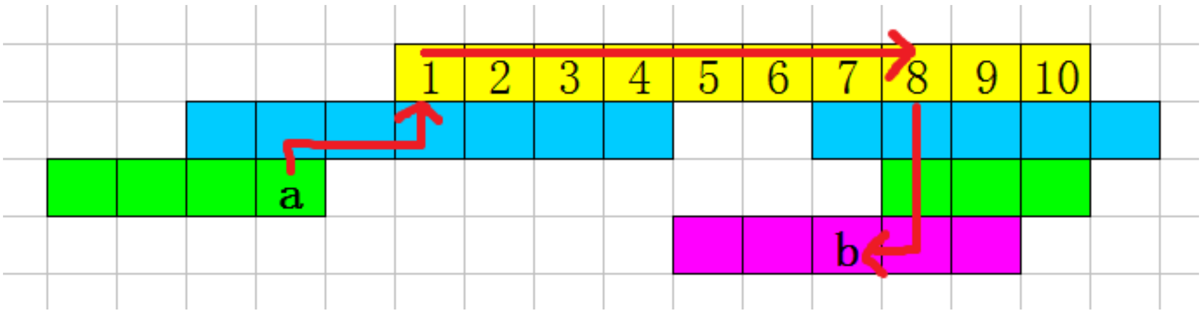
一只神蒟。

首先是非常巧妙的建图：每一行分开看，把同一行的若干个联通块分别缩成点，然后向四周相邻的点连边，由于没有封闭的空地，这样连出来一定是棵树。

支持加入关键点，查询距离最近的关键点，建点分树?

似乎还没做完：两点之间的最短距离要如何在树上表示呢?

看图：



任选$ a,b $路径上一个缩成点的小连通块 $m $作为中介（如图中黄色框），先分别求出$ a,b $移动到中介的最短路径$ d_a(m),d_b(m)$，并记录它们到达中介时的纵坐标 $y_a(m),y_b(m)$（在图中表现为 1 号和 8 号位置），则 $dis(a,b)=d_a(m)+d_b(m)+|y_a(m)-y_b(m)|$，即 $\min\{d_a(m)+d_b(m)+y_a(m)-y_b(m),d_a(m)+d_b(m)+y_b(m)-y_a(m)\}$ 。

现在点分树就可以轻松维护答案了，设：

$f_1(i,j)=\sum_{x\in subtree(i),y_x(i)\leqslant j}d_x(i)-y_x(i)$

$g_1(i,j)=\sum_{x\in subtree(i),y_x(i)\geqslant j}d_x(i)+y_x(i)$

则有 $ans(x,k)=\min_{i\in fatree(x)}\min\{d_x(i)+y_x(i)+f_1(i,y_x(i)),d_x(i)-y_x(i)+g_1(i,y_x(i))\}$

二维的 f_1,g_1 用树状数组维护。

空间复杂度：$O(n\log n)$ 。

时间复杂度：$O(n\log n+q\log^2 n) $。

2.【Code】

```
https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/Iqea.cpp
```

6.【大毒瘤】

[紫荆花之恋 $text{[WC2014] [P3920]}$

四【总结】

模板及例题 1.1,1.2 都是靠数据结构维护贡献，

例 2 则换成了$ \text{STL}$ 。套娃行为所导致的的码量增加以及大常数都是值得关注的问题。

例 3,4 主要是按照容斥套路推式子，相对比较好掌握，但细节较多，要注意统计贡献时补充不漏。

例 5 难点在于利用点分树的特殊性质转离线。点分树各种神奇的特性，对应到不同的题上就会有各种神奇的解法。如果没有强大的瞎蒙猜结论能力，就多刷刷题吧，见多识广总没有坏处的。

例 6 不光要会神仙建图，还要想办法用点分树能维护的东西来表示两点距离。这道题有效地提醒了我们：点分树有着不错的灵活性，并非只有那个一成不变的模板，所以不要死记硬背啊...

> Q: 话说点分树能搞可持久化吗?
> A: 虽然听起来比较毒瘤，但不瞒您说，还真可以（具体见 [可持久化点分树]$https://www.cnblogs.com/Khada-Jhin/p/10175454.html$）
>
> CZC: NOI之前不建议掌握这个技巧，性价比太低，并且偏门。

五 【练习题目】

UOJ 题号	题目名	short-mark
2343	「模板」点分治 「POJ2114」Boatherds	不带修点分治 =k
1114	「模板」「BZOJ1468」Tree	<k
2428	「IOI2011」Race	给一棵树，每条边有权。求一条简单路径，权值和等于 k，且边的数量最小。
2342	聪聪可可	相对模版有点变化
2958	「BZOJ3730」震波	带修
7177	atm的树	二分答案后又是一只震波
7175	[BJOI2017]树的难题	
6792	「HNOI2015」开店	
6802	[2010国家集训队]Crash的旅游计划	有出题人题解pdf，在资源包中（国家集训队作业08）
2957	「ZJOJ2015」幻想乡战略游戏	
2959	「BZOJ4372」烁烁的游戏	和模板题震波类似，用一个数据结构进行维护
5833	小清新数据结构题	
5830	[Ynoi2011] 成都七中	
6803	「CF936E」 Iqea	挑战题
7176	模式字符串	挑战题