Tree(Da Shui Ti)

【问题描述】

给定一棵 N 个节点的树，每个点有一个权值，有 M 个询问(a,b,c)若 a 为 1，回答 b 到 c 路径上的最小权值，若 a 为 2，回答 b 到 c 路径上的最大权值，若 a 为 3，回答 b 到 c 路径上的所有权值的中位数，k 个数的中位数定义为从 0 到 k-1 编号从小到大排序后 k/2 号的那个数。

【输入格式】

第一行两个整数 N，M

第二行 N 个整数，v[1]~v[n]，代表每个节点的权值。

接下来 N-1 行每行两个整数 x，y 代表 x 和 y 有一条边

最后 M 行每行三个整数 a，b，c，表示一组询问。

【输出格式】

共 M 行，每行一个整数，代表询问的答案。

【样例输入】

5 3
1 3 2 4 5
1 2
2 4
4 3
4 5
1 1 5
2 1 3
3 1 5

【样例输出】

1
4
4

【数据范围】

共 20 个数据

数据 1~3　N,M<=1000
数据 4~6　N,M<=5000
数据 7~10　N,M<=10000
数据 11~18　N,M<=30000
数据 19~20　N,M<=100000

Solution：块状树的话也可以做。函数式线段树的做法比较容易，对于每个结点建立出从根到这个点所有权值的权值线段树。无论询问最大最小还是中位数我们都可以用询问第 k 小解决。我们找到 x，y 两点的 lca，利用它们的三棵线段树进行二分即可。

Missing on the tree

时限：1.5s，内存限制：256M

【问题描述】
你有一棵以 1 号节点为根的有根树。。。
【输入格式】
第一行一个数 n 表示节点个数
第 2 到 n 行，每行一个数 x，第 i 行的 x 表示 i 节点的父亲为 x
接下来一个数 m，表示操作与询问个数的总和
接下来 m 行，每行的开头有 1 个数 p
若 p 为 0，则接下来有 2 个数 x，y，表示将 x 号节点的权值加上 y
若 p 为 1，则接下来有 3 个数 x，y，z，表示将 x 号节点到 y 号节点的这条链上的每个节点的权值加上 z
若 p 为 2，则接下来有 2 个数 x，y，表示将以 x 号节点为根的子树的每个节点的权值加上 z
若 p 为 3，则接下来有 2 个数 x，y，表示询问 x 号节点到 y 号节点的这条链上每个节点的权值和，你需要输出这个和
若 p 为 4，则接下来有 2 个数 x，y，表示询问 x 号节点到 y 号节点的这条链上每个节点的权值的最大值，你需要输出这个最大值
若 p 为 5，则接下来有 2 个数 x，y，表示将 x 号节点的父亲改为 y 号节点
【输出格式】
对于每个 3 与 4 操作，输出 1 行，表示答案
【样例输入】
3
1
1
2
1 2 3 -1
3 2 3
【样例输出】
-3
【数据范围】
保证任何时候每个节点的权值的绝对值小于 40000
4%的数据 n,m<=5
另外 12%的数据 n,m<=10000
另外 40%的数据，没有 5 操作
另外 20%的数据，没有 2 操作
100%的数据 N<=50000，m<=100000

Solution：Link-Cut Tree，对于 0、1、3、4、5 的操作都是 lct 的基本操作，不再赘述。对于第二个操作，对一棵子树加上一个值，我们可以在这棵子树的根结点打上一个标记，对于某一条链链形成的 splay（我们默认左子树比较浅），在更新每个结点的时候我们看左子树是否有标记，有的话右子树以及当前这个结点用来更新答案的数据也进行相应的改变，并且更新完后这个标记再放到本身这个结点上。

Catch The Penguins

(ctps.cpp/c/pas)

TimeLimit：2s　MemoryLimit：256M

【问题描述】

Xyz 带着他的教徒们乘着科考船一路破冰来到了南极大陆，发现这里有许许多多的企鹅。邪恶的 Xyz 想要抓很多企鹅回去开动物园，当宠物玩。但动物保护协会很快赶来，他必须尽快行动！

我们把南极大陆看成一个三维直角坐标系。

有 N 只企鹅，每只企鹅会在一定的时刻的出现，第 i 只企鹅在 Ai 时刻出现在坐标为(Bi,Ci,Di)的地方。

Xyz 要在某一时刻在某一地方(X,Y,Z)撒一张大网，将(0,0,0)到(X,Y,Z)这个大长方体里的企鹅全都网进去捕捉回家(还没出现的企鹅就不会被捉进去了)。

为了快准狠而且保证不铺张浪费网，Xyz 想知道不同时间不同地点撒网能抓到几个企鹅(这样的询问有 Q 个)。然后他再行动。

【输入格式】

第一行一个整数 N 表示企鹅个数。

第二行到 N+1 行每行四个实数(Ai,Bi,Ci,Di)，表示企鹅的出现时间和位置

第 N+2 行一个整数 Q 表示询问个数。

接下来 Q 行每行四个实数(T,X,Y,Z)，表示询问的时间和位置。

【输出格式】

输出共 Q 行，每行一个整数，回答每个询问能抓到几个企鹅。

【样例输入】

```
1
0 0 0 0
2
1 1 1 1.0
1 1 1 -1
```

【样例输出】

```
1
0
```

【数据范围】

共 20 个数据

数据 1~3　N,Q<=1000

数据 4~6　N,Q<=5000

数据 7~10　N,Q<=10000

数据 11~14　N<=30000,Q<=10000

数据 15~18　N<=10000,Q<=30000

数据 19~20　N,Q<=30000

【算法 0】

对于每一个询问，一个 for 循环枚举每只企鹅，判断四维坐标是否都小于询问的坐标。

也就是直接暴力枚举

时间复杂度 O(n^2)

【算法 0 优化】
把企鹅分别按四维坐标排序，对于每个询问，每一维在排好序的这四个数组里二分得出这一维小于询问坐标的企鹅的个数，取其中最小的进行枚举（常数优化）。
【算法 1】
先离散化。
离线回答询问。
1. 分别将询问和企鹅按照第一维坐标排序，然后按排好的顺序一个一个处理询问。
2. 对于每一个询问，加入所有第一维比它小的企鹅并按照第二维排序，为了保证复杂度，我们每加入 sqrt(n)个企鹅，就暴力进行一次排序，当没有到 sqrt(n)时，不到 sqrt(n)的那部分就不进行排序而暴力询问（这部分总复杂度 O(nsqrt(n))）。
3. 当进行一次暴力排序时，设已经加入排序的企鹅的个数为 M，我们再把它们分成 sqrt(M)块，每块里面按照第三维坐标排序，排好序后把序列按第四维维护前缀的权值线段树（用函数式线段树）。
4. 进行询问。对于“整块的第二维坐标”都比“当前询问第二维坐标”小的那些块，我们在块中二分出询问第三维坐标所对应的位置，再在那棵函数式线段树上询问第四维比它小的企鹅的个数。对于不是“整块的第二维坐标”都比“当前询问第二维坐标”小的那些块，我们仍然只需要暴力询问即可。

总复杂度 O(nsqrt(n)log(n))
【算法 2】
同样的，还是离散化。
还是离线回答询问。
先将询问和企鹅全都放在一起，按第一维坐标排序，那么只有前面的企鹅可能为在它之后的询问增加答案。从第一维排好序的顺序开始，我们再按照第二维坐标进行类似归并排序的分治算法：
1. 假设排好序后的编号序列为 id[L]~id[R]，其中有的是企鹅，有的是询问，令 MID 为其终点。
2. 递归处理 L~MID，MID+1~R 部分的答案。（当然，当 L，R 相同时就可以直接返回）
3. 我们发现我们还没有处理的情况就只有左半部分（L~MID）的企鹅对右半部分（MID+1~R）的询问的影响，这时左右两部分都已经按第二维坐标排好序（为什么已经排好序见后），而它们按第一维排的顺序已经不需要了，我们把左半部分和右半部分按第二维坐标进行归并（有序线性表合并），合并成新的表时，如果添加的是左边的企鹅，就把它三四两维形成的点插入一个“集合”，如果添加的是右边的询问，就在“集合”里询问有多少点在以它三四两维形成的点的左下方，把答案累加到这个询问里。
4. 至于这个“集合”的实现，我们可以用树状数组（或线段树）套平衡树。

总时间复杂度 O(nlog^3(n))