题目描述

题目背景：

小G出了一个签到题，但被小F随手加强了……

题目描述

这是一道交互题。

有一棵 $n(n \leq 1000)$ 个节点的树，所有边权值为 1，但你不知道哪两个点之间存在边。

但是你可以查询一个点到一个集合的距离和来寻找隐藏的边。

具体来说你可以使用 $ask(u, V)$ 其中 V 是一个集合（**元素两两不同**）,返回值为 $\sum_{x \in V} dist(u, x)$, 其中 $dist(x, y)$为树上x,y两点最短路的长度。你需要通过$answer(u, v)$来回答找到的所有边。

但询问的次数不能超过 A 次，且 $|V|$ 的总和不能超过 B。

实现细节

本题仅支持符合 C++ 11 标准及以上的 C++ 语言。

你需要在代码开头包含：

```
#include"tree.h"
```

你不需要实现主函数，你只需实现如下函数：

```
void solver(int n,int A,int B)
```

该函数会在每组测试点开始时调用一次。

你可以调用如下函数不超过 A次，v 的大小总和不能超过 B，同一次调用 v 中元素不能相同：

```
int ask(int u,vector<int>v)
```

表示询问点 u 到 v 中所有点距离之和。

得出答案后，你需要调用如下函数恰好 $n-1$ 次：

```
void answer(int u,int v)
```

表示存在一条 (u, v) 的树边。

输入在交互开始前已经确定，交互库不自适应。

输入格式：

第一行三个整数 n, A, B。

接下来 $n-1$ 行每行两个整数 u, v 表示一条树边。

输出格式：

若输入格式或询问格式有误，则会输出"Invalid"，并告知错误原因。

若询问次数超过A，则会输出"Too many queries"。

若询问的集合v的大小总和超过B，则会输出"The sum is too large"。

若返回的边错误，则会输出"Different tree"。

若答案正确，则会输出"Correct cnt=x sum=y"，其中x为询问总次数，y为v的大小总和。

样例输入

```
3 114 514
1 2
2 3
```

样例输出

```
Correct cnt=2 sum=3
```

交互过程

tree.cpp : ask(1,{2,3})

grader.cpp : 3

tree.cpp : ask(1,{2})

grader.cpp : 1

tree.cpp : answer(1,2)

tree.cpp : answer(2,3)

grader.cpp : Correct cnt=2 sum=3

数据范围

Subtask 1 (3分) : $n \leq 1000, A = 5 \times 10^5, B = 5 \times 10^5$。

Subtask 2 (17分) : $n \leq 100, A = 3 \times 10^3, B = 3 \times 10^4$。

Subtask 3 (20分) : $n \leq 1000, A = 5 \times 10^4, B = 3 \times 10^6$。

Subtask 4 (60分) : $n \leq 1000, A = 8500, B = 3 \times 10^5$。

时间限制$= 0.5s$。

空间限制$=1024MB$。

交互库使用的时间和内存很少，可以忽略。

题解

算法1 : subtask 1

直接询问每一个二元组的距离是否$=1$，可以知道每两个点之间是否存在边。

$cnt=sum=\frac{n\times(n-1)}{2}$。

期望得分：3。

算法2 : subtask 2,3

定义函数$Find(S)$来找到**连通块**S内的所有边。

考虑点分治来递归求解。

重心是到所有点距离和最小的点，然后找到重心之后就需要知道每个儿子的子树是啥：一个一个点确定是属于哪个儿子的子树，可以使用二分，因为到它对应的儿子距离是depth-1,其他都是depth+1。

操作次数是$O(n\log^2 n)$的，不过常数非常小。当$n\leq 1000$时，操作次数不太可能超过30000。同时$|v|$总和为$O(n^2)$级别，不会超过3×10^6。

期望得分：40 。

算法3 : subtask 1,2,3,4

本人最初在6月份的时候想到了这个题，但当时只想到了$n\log^2 n$的算法2，感谢ftq提供的$O(n\log n)$做法。

随便定个根，然后依次确定每两层中的边。

假设当前要确定集合A和集合B的边，已知集合A的深度为集合B深度-1。

将集合A平均随机分成两个集合A_L,A_R。

若B中元素i是A中元素j的儿子，则$ask(i,A_R)=ask(j,A_R)+|A_R|$，所以可以按照$ask(i,A_R)$分组再递归，可以证明组内个数不会超过$\frac{|A_r|}{2}$。

操作次数为$O(n\log n)$，通过dp可以算出最坏情况操作次数为9226，不过实际上很难卡满。

一些优化：

1. 如果是随机平分A集合是肯定卡不到9226的。
2. 随机选根。
3. 如果递归到$|A|=2$时，可以用一些其它方法，使得询问次数$<|B|$。可以参考https://codeforces.com/contest/1705/problem/F的做法。如果使用这个优化，可以减去几百次操作次数。
4. 如果递归到$|A|=2$时，可以用一些其它方法，使得询问次数$<|B|$。可以每12个分一组，对于一个组可以一次问出有几个属于A_0的儿子，如果等于0或者12可以直接结束，不然暴力询问组内。对于最劣情况优化比较明显。

使用这些优化可以很稳的在7500次内算出结果，甚至很少超过7000。如果不随机选根，只使用优化1,3/4，也很难构造出超过8000的数据。事实上，优化3和4是不必要的，只需要优化1和2基本就可稳定通过。A的限制实际上比std的次数多了一千多次，并不卡常数。

另外随机平分A集合的做法已经是正确的了，但同时存在确定性构造（来自djq）：

对于任意深度为 i 的点构成的集合 A，可以构造出A 的子集 S 使得$ask(i,S)$ 的众数不超过$(|A|+1)/2$。

递归地做，考虑当前树是 T，非空的子树依次有 $T_1,T_2,\ldots,T_k$ ，$size$ 递减，取最小的 i使得 T_{i+1} 到 T_k 的 $size$ 总和不超过 $|A|/2$我们在$T_1,T_2,\ldots,T_i$内递归构造，T_{i+1}到 T_k 子树里一个都不选。

总结

本题主要考察了选手分治与二分的运用，存在较多$O(n\log^2 n)$的做法，本人已经在准备题目的时候卡掉了所有已知的$O(n\log^2 n)$的做法，但由于数据范围太小，加上大部分做法都很难卡满，部分小常数的$n\log^2 n$的做法仍可通过。本人主要是希望选手找到询问次数尽量少的方法，并没有卡B的限制。

本题可通过的做法很多，欢迎大家分享。