《通道建设》 解题报告

November 2023

题面

题目描述

这是一道交互题。

一场叛乱后，上层政府任命你担任这座城市的市长，负责重建。你需要在城市里修建 N 条通道。

这是一座有着 $2N$ 个依次编号为 $1,2,\ldots,2N$ 街区的大城市，它们由 $2N-1$ 条双向道路连通。换言之，城市的街区和道路构成一棵树。

所有通道都是单向的，你需要选定恰好 N 个街区作为入口，其余 N 个街区作为出口。N 条通道需要不重不漏地覆盖所有入口和出口。同时为了避免上层政府质疑通道的修建方案，不能存在两条通道 $(a,b),(c,d)$，其中 a,c 为入口，b,d 为出口，满足 $\mathrm{dis}(a,b)<\mathrm{dis}(a,d)<\mathrm{dis}(c,d)$。此处 $\mathrm{dis}(x,y)$ 定义为街区 x 到街区 y 的距离，即最少经过的道路数。

然而叛军烧毁了很多资料，导致你只拥有一份年代久远的城市地图。现在的城市相较于你手上的地图描绘的城市早已经过了许多次扩建。你的地图只有现在的城市的街区 $1\ldots N$ 和当时连通它们的 $N-1$ 条道路。但扩建不会破坏城市的布局：对扩建前的两条恰有一个公共街区的道路 $(a,b),(b,c)$，扩建后 a 到 b 的最短路径和 b 到 c 的最短路径依旧只有一个公共街区 b。

由于你暂时人手不足，你只有以下两种渠道收集信息：

1. 给定一系列街区 A 和一系列街区 B（需要满足 $|B|>1$），对 A 中每个街区 x，你可以知道离 x 最远的在所有路径 $(x,y)(y\in B)$ 上的街区（即以 x 为根时 B 中街区的 LCA）是否为一个给定点 P。
2. 给出两个街区 x,y，你可以知道 $\mathrm{dis}(x,y)$。

请完成修建通道的任务。

实现细节

你不需要，也不应当实现主函数。

你应当引入头文件 `passageconstruction.h`。

你应当实现如下函数：

```
std::vector<std::pair<int,int>> ConstructPassages(int N, const std::vector<std::pair<int,int>>
&E);
```

N 的含义如题目所述，E 为你的地图上的道路，你需要返回恰好 N 个`pair`，代表一组合法的修建方案。一个`pair` 代表一条通道，其成员 `first` 为入口，`second` 为出口。特别地，如果无解，请返回一个空`vector`。

可调用函数：

```
std::vector<int> QueryLCA(const std::vector<int> &A, const std::vector<int> &B, int P);
```

该函数代表一次操作 1，A,B,P 的含义如上文所述。保证返回`vector` 的大小与`A.size()`相同，值只包含 0 和 1，第 k 项（从 0 开始）为 1 当且仅当以 A_k 为根时 B 中街区的 LCA 为 P。

```
int GetDistance(int x,int y);
```

查询距离，x,y 含义如上文所述。

```
int getSubtaskID();
```

获取子任务编号。在样例交互库中该函数恒返回 0。

保证城市在交互开始时已经确定，即交互库不自适应。

样例交互库

下发了 `sample_grader.cpp` 和 `passageconstruction.h`，你可以使用命令 `g++ sample_grader.cpp <your source file> -o passageconstruction -O2 -std=c++14` 生成可执行文件。

注意评测时交互库与样例交互库有所不同。

输入格式

第一行一个数 N，含义如题目所述。

接下来 $2N-1$ 行，每行两个数，代表城市的一条道路。注意 $1,2,\ldots,N$ 需要满足前文所述的条件。

注意样例交互库不会检查输入文件是否合法，输入不合法时其行为未定义。

输出格式

交互库会在运行过程正常结束且答案正确时返回 0，答案错误或交互错误时返回 1。

当答案错误或交互错误时的提示信息如下：

`Index out of range`：街区编号不为 $[1,2N]$ 内的整数。

`Invalid construction plan`：返回方案格式有误。

`Condition not satisfied`：返回方案不满足上文的条件。

`Invalid type 1 query`：执行操作 1 时未满足 $|B|>1$ 的条件。

当答案正确时的提示信息如下：

`Accepted with a+b operations, sum of size(s)=c+d`。其中 a,b 分别为操作 1 和操作 2 的次数，$c=\sum|A|,d=\sum|B|$。

当无解时的提示信息如下：

`Reported no solution with a+b operations, sum of size(s)=c+d`。a,b,c,d 含义同上。

数据范围

记 C_1 为操作 1 次数，C_2 为操作 2 次数，$S_1=\sum|A|,S_2=\sum|B|$。

子任务编号	分值	$N \leq$	C_1 上限	C_2 上限	S_1 上限	S_2 上限	特殊性质
1	3	5	100	100	1×10^4	1×10^4	
2	6	100	2×10^4	1×10^4	2×10^5	2×10^5	
3	8	1000	1×10^5	4000	2×10^5	2×10^5	A
4	9	1000	1×10^5	4000	2×10^5	2×10^5	BC
5	11	1000	5×10^4	4000	2×10^5	1×10^5	B
6	12	1000	2.5×10^4	4000	2×10^5	1×10^5	C
7	14	1000	2.5×10^4	4000	2×10^5	1×10^5	
8	10	5000	5×10^4	1×10^5	5×10^5	2×10^5	
9	27	5000	1.5×10^4	1×10^4	2×10^5	5×10^4	

特殊性质 A：城市的每个街区至多与两条道路直接连接。

特殊性质 B：城市的每个街区至多与三条道路直接连接。

特殊性质 C：保证你拥有的地图上的所有道路仍然存在。

评分方式：在一个子任务内，如果 C_1, C_2, S_1, S_2 均 $\leq$ 对应上限，该子任务得满分。

否则设对应上限分别为 $C_1^{Lim}, C_2^{Lim}, S_1^{Lim}, S_2^{Lim}$，令 $v = \max(\frac{C_1}{C_1^{Lim}}, \frac{C_2}{C_2^{Lim}}, \frac{S_1}{S_1^{Lim}}, \frac{S_2}{S_2^{Lim}})$。如果 $v > 2$ 则该子任务得 0 分，否则该子任务得分为该子任务满分分值$\times(1.4 - 0.5v)^2$。

保证交互库运行时间在给定数据范围内 ≤ 0.1s，空间 ≤ 64MiB。子任务 $1 \sim 7$ 的时间限制为 1s，其余子任务的时间限制为 2.4s。

题意简述

有一棵 $2N$ 个的点的无根树，满足点集 $\{1,2,3,\ldots,N\}$ 的虚树的点集为其自身。

给出该点集的虚树边集并提供两种操作：

1. 给出集合 A,B 和点 P，对 A 中的每个点，定其为根后得知 B 中所有点的 LCA 是否为 P。
2. 询问两个点的距离。

试找到 $2N$ 个点的一个匹配，满足对于不存在一对匹配 $(a,b),(c,d)$，有 $\operatorname{dis}(a,b)<\operatorname{dis}(a,d)<\operatorname{dis}(c,d)$。此处匹配定义为 N 个元素均为 $[1,2N]$ 内的整数且没有元素重复的有序对。

交互库不自适应。

$N\leq 5000$，要求操作 1 次数在 $\mathcal{O}(N)$ 级别，操作 2 次数 $\leq 2N$。

同时 $|A|,|B|$ 需要满足 $\sum|A|=\mathcal{O}(N\log N),\sum|B|=\mathcal{O}(N\log N)$。

解题过程

算法一

询问出所有点之间的距离后可以得知树的形态，枚举所有匹配并检查。共执行 $\frac{N(N-1)}{2}$ 次操作 2，时间复杂度在指数级别。

算法二

结论1

所有树均存在合法修建方案。

证明1

若所有通道端点之间的距离均 ≤ 2，则该方案显然合法。构造该方案可以从叶子向根贪心，每次尽可能地在未匹配的孩子结点之间互相匹配。

于是得知树的形态后执行上述贪心过程，共执行 $N(2N-1)$ 次操作 2，时间复杂度 $\Theta(N^2)$。

算法三

本文之后的内容中均视原树为以 1 为根的有根树。

考虑如何更加高效地询问出树的形态。先询问出所有点的深度，随后按照深度顺序增量构造虚树。对于当前增量构造的点 x 和一个度数 ≥ 2 的点 u，在操作 1 中定 x 为根，询问 u 与其所有邻接点的LCA：如果LCA为 u 则直接将 x 与 u 连边；为 x 则说明 x 应当放在 u 的一条边上，可以枚举邻接点 v 并判定以 x 为根时 $\operatorname{LCA}(u,v)=x$ 是否成立；否则应当往以LCA为根的子树内递归。点分治可以在 $\mathcal{O}(N\log N)$ 次操作 1 之内解决树为二叉树时的问题。

略微改进询问形式。对虚树上两个邻接点 a,b 和一个不在虚树上的点 c，如果 c 在边 (a,b) 上或者定 a 为根时在以 b 为根的子树内，则认为 c 在 a 朝 b 的方向上，该方向记作 $a\to b$。

设 u 邻接点集合的一个子集为 V。若 $|V|=1$，设$V=\{v\}$，则询问以 x 为根时 $\operatorname{LCA}(u,v)$ 是否为 u 就可以判断 x 在不在 $u\to v$ 的方向上。$|V|\geq 2$ 时定 x 为根，询问 $\operatorname{LCA}(V)$ 是否为 u。若为 u 则说明 x 在 u 朝某一个 $v\in V$ 的方向上。找到 x 所在方向 $u\to w$ 后可以用一次操作 1 判断 x 是否在边 (u,w) 上。于是可以用二分代替枚举使得点分治能够解决非二叉树时的问题，操作 1 次数为 $\mathcal{O}(N\log^2 N)$。

算法四

结论2

称入口构成的集合为左部点，出口构成的集合为右部点。则无论如何划分左部点和右部点，均存在合法修建方案。

证明2

考虑稳定婚姻问题[1]。令左部点的偏好顺序为右部点与自己之间的距离升序，右部点的偏好顺序为左部点与自己之间的距离降序，由稳定婚姻问题即知一定有解。于是只要 N^2 次操作 2 就能解决问题。

算法五

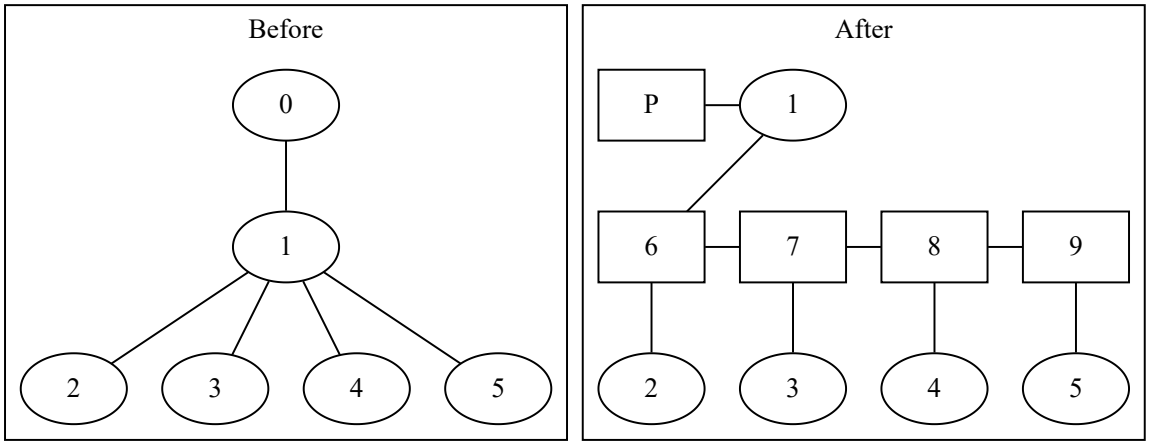
钦定 $\{1,2,\ldots,N\}$ 为左部点，$\{N+1,N+2,\ldots,2N\}$ 为右部点，则无需知道树的具体形态，只要知道右部点在虚树上的位置。位置只有两种可能：挂在一个点上或者在一条边上。知道位置后对于符合前者的右部点询问到悬挂点的距离，对符合后者的右部点询问到边两端的距离。同时虚树边权可以由挂在边上且到某一端点距离为 1 的点计算而得（若无即为 1），因此知道右部点在虚树上的位置后就可以计算出左部点和右部点之间的所有距离。

观察到用于确定树形态的点分治同样可以用于定位，只需把其中的二分改为整体二分。操作 1 次数为 $\mathcal{O}(N\log N)$，$\sum|A|=\mathcal{O}(N\log^2 N),\sum|B|=\mathcal{O}(N\log N)$。

解法

先将多叉树转二叉树，令原树上挂在边上的右部点挂在孩子和虚点的边上。

某个结点（重编号后设为 1）及其邻接点转二叉树的过程的示例如下图：



如图，左图结点深度从上到下递增；右图方框点为新增的点，P 为 0 的孩子虚点，1 为根时 $0,P$ 不存在。

称新增的点为虚点，原树上的点为实点，虚点的原点为其最近的实点祖先。

新二叉树中的所有方向都可以在原树中找到对应方向，如上图中的边 $(6,2)$ 对应原树中的边 $(1,2)$，$(7,3)$ 对于原树中的边 $(1,3)$。点分治过程中 $6\to7$ 方向对应原树中 $\{1\to3,1\to4,1\to5\}$ 三个方向，同理 $8\to7$ 方向对应挂在 1 上的点，$1\to0$ 方向上的点（若 1 不为根）和原树中 $\{1\to2,1\to3\}$ 两个方向。

于是该二叉树上也可以点分治，但如果每次询问所有对应方向则 $\sum|B|$ 在 $\mathcal{O}(n^2)$ 级别。

观察到某些方向已经被提前排除，例如点分治取 8 为分治重心后再选取 6 为分治重心时不用询问 $1\to4,1\to5$ 方向。形式化地，设当前结点为 p，以 p 为原点的虚点序列为 V，满足 V 中元素按照在新二叉树上深度从小到大排序，则 V 中相邻元素存在边连接。在分治重心 V_x 为虚点时要剥离朝虚孩子方向的点，只要找到最小的 $y>x$ 满足 V_y 已经成为过分治重心（若无则为 $|V|+1$），询问哪些点在朝 $V_{x+1\ldots y-1}$ 之间的方向上。由于点分

树深度为 $\mathcal{O}(\log N)$ 级别， $\sum |B| = \sum |V| \mathcal{O}(\log N) = \mathcal{O}(N \log N)$。在朝上方向上的点可以通过剥离实孩子方向和虚孩子方向的点得出。

参考资料

[1] Stable marriage problem - Wikipedia