树相关基本的定义、性质与问题

汪直方

宁波市镇海蛟川书院

2024 年 1 月 15 日

树论（tree theory），是图论的一个分支。它以树这一种特殊的图为研究对象。信息学竞赛中，题目通常会给定一棵树和一些其他参数，运用树的直径、重心、最近公共祖先、dfs 序等的性质，通过树形动规、树上差分、树分治等算法技巧，求解一个或多个这棵树上的问题。有时也会给定一些条件，求解关于满足这些条件的树集的问题。

“总是有出题人喜欢把序列上用线段树解决的题目出到树上，让选手强行写个树链剖分或树分治或某种动态树数据结构。

“总是有出题人喜欢把序列上用线段树解决的题目出到树上，让选手强行写个树链剖分或树分治或某种动态树数据结构。

这种行为已经很无趣了。

“总是有出题人喜欢把序列上用线段树解决的题目出到树上，让选手强行写个树链剖分或树分治或某种动态树数据结构。

这种行为已经很无趣了。

所以我们想让大家知道，不光可以放在静态树上，动态仙人掌也是可以的。”

“总是有出题人喜欢把序列上用线段树解决的题目出到树上，让选手强行写个树链剖分或树分治或某种动态树数据结构。

这种行为已经很无趣了。

所以我们想让大家知道，不光可以放在静态树上，动态仙人掌也是可以的。”

我们往往可以将序列问题看作链上的问题，在将其推广到仙人掌甚至广义串并联图的过程中树往往是关键的一环，树保留了链两点间存在唯一一条路径的性质，但使得祖先后代关系不再是全序打破了部分性质。

“总是有出题人喜欢把序列上用线段树解决的题目出到树上，让选手强行写个树链剖分或树分治或某种动态树数据结构。

这种行为已经很无趣了。

所以我们想让大家知道，不光可以放在静态树上，动态仙人掌也是可以的。”

我们往往可以将序列问题看作链上的问题，在将其推广到仙人掌甚至广义串并联图的过程中树往往是关键的一环，树保留了链两点间存在唯一一条路径的性质，但使得祖先后代关系不再是全序打破了部分性质。

由于在数据结构问题中，将树上的一些算法与技巧推广到仙人掌或更一般的图上经常比从链上推广到树上要来得平凡，树依然是图上数据结构问题中相当重要的研究对象。

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

显然我们可以把森林可以视为树的无序集合，对于多数问题，我们只需对每个连通块分别处理即可。

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

显然我们可以把森林可以视为树的无序集合，对于多数问题，我们只需对每个连通块分别处理即可。

Property 2.1

对于图 T，T 是树与以下结论等价：

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

显然我们可以把森林可以视为树的无序集合，对于多数问题，我们只需对每个连通块分别处理即可。

Property 2.1

对于图 T，T 是树与以下结论等价：

1. T 中任意两个顶点间存在唯一一条路径；

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

显然我们可以把森林可以视为树的无序集合，对于多数问题，我们只需对每个连通块分别处理即可。

Property 2.1

对于图 T，T 是树与以下结论等价：

1. T 中任意两个顶点间存在唯一一条路径；
2. T 是连通的且对任意边 $e \in E(T)$，$T - e$ 是不连通的；

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

显然我们可以把森林可以视为树的无序集合，对于多数问题，我们只需对每个连通块分别处理即可。

Property 2.1

对于图 T，T 是树与以下结论等价：

1. T 中任意两个顶点间存在唯一一条路径；
2. T 是连通的且对任意边 $e \in E(T)$，$T - e$ 是不连通的；
3. T 是无圈的且对其中任意两个不相邻的结点 $x, y \in V(T)$，$T + xy$ 包含圈。

Definition 2.1

无圈的图称为森林，而连通的森林称为树。

显然我们可以把森林可以视为树的无序集合，对于多数问题，我们只需对每个连通块分别处理即可。

Property 2.1

对于图 T，T 是树与以下结论等价：

1. T 中任意两个顶点间存在唯一一条路径；
2. T 是连通的且对任意边 $e \in E(T)$，$T - e$ 是不连通的；
3. T 是无圈的且对其中任意两个不相邻的结点 $x, y \in V(T)$，$T + xy$ 包含圈。

我们可以说明，一棵 $n \in \mathbb{N}^+$ 个结点的树有 $n-1$ 条边。在以下部分不引起歧义的前提下我们默认 n 为树的结点个数且 $n \geqslant 1$。

在某些问题中，将树的一个结点做特殊的处理可以让问题简化。我们称这个结点为树的根，而固定了根的树称为有根树，反之称为无根树。

在某些问题中，将树的一个结点做特殊的处理可以让问题简化。我们称这个结点为树的根，而固定了根的树称为有根树，反之称为无根树。

Definition 2.2

设有根树 T 的根为 r，对任意 $x, y \in V(T)$，若 x 在 T 中 r, y 之间的路径上，则称 $x \leqslant_T y$，于是 $\leqslant_T$ 是 $V(T)$ 上的一个偏序关系，称为由有根树 T 的树序。为了叙述方便，令 $x <_T y$ 当且仅当 $(x \leqslant_T y) \wedge (x \neq y)$。

在某些问题中，将树的一个结点做特殊的处理可以让问题简化。我们称这个结点为树的根，而固定了根的树称为有根树，反之称为无根树。

Definition 2.2

设有根树 T 的根为 r，对任意 $x, y \in V(T)$，若 x 在 T 中 r, y 之间的路径上，则称 $x \leqslant_T y$，于是 $\leqslant_T$ 是 $V(T)$ 上的一个偏序关系，称为由有根树 T 的树序。为了叙述方便，令 $x <_T y$ 当且仅当 $(x \leqslant_T y) \wedge (x \neq y)$。

Definition 2.3

对于一棵有根树 T 与结点 $u \in V(T)$，若 u 非根结点，则称其父亲 $\mathrm{par}(u)$ 为满足 $x <_T u, \nexists y \in T, (x <_T y) \wedge (y <_T u)$ 的唯一结点 x，而 u 称为 $\mathrm{par}(u)$ 的儿子。对于一个结点 u，我们令 $\mathrm{son}(u) = \{v | \mathrm{par}(v) = u\}$，$u$ 的祖先结点集 $\mathrm{anc}(u) = \{x | x \leqslant_T u\}$，后代结点集为 $\{x | u \leqslant_T x\}$。

在某些问题中，将树的一个结点做特殊的处理可以让问题简化。我们称这个结点为树的根，而固定了根的树称为有根树，反之称为无根树。

Definition 2.2

设有根树 T 的根为 r，对任意 $x, y \in V(T)$，若 x 在 T 中 r, y 之间的路径上，则称 $x \leqslant_T y$，于是 $\leqslant_T$ 是 $V(T)$ 上的一个偏序关系，称为由有根树 T 的树序。为了叙述方便，令 $x <_T y$ 当且仅当 $(x \leqslant_T y) \wedge (x \neq y)$。

Definition 2.3

对于一棵有根树 T 与结点 $u \in V(T)$，若 u 非根结点，则称其父亲 $\mathrm{par}(u)$ 为满足 $x <_T u, \nexists y \in T, (x <_T y) \wedge (y <_T u)$ 的唯一结点 x，而 u 称为 $\mathrm{par}(u)$ 的儿子。对于一个结点 u，我们令 $\mathrm{son}(u) = \{v | \mathrm{par}(v) = u\}$，$u$ 的祖先结点集 $\mathrm{anc}(u) = \{x | x \leqslant_T u\}$，后代结点集为 $\{x | u \leqslant_T x\}$。

视语境有时祖先与后代指将上述 $\leqslant_T$ 改为 $<_T$ 后的集合，课件后面部分以上述定义为默认。

Definition 2.4

对于一棵有根树，称结点 u 含自身的后代结点集的导出子图为结点 u 的子树 $\mathrm{subtree}(u)$，称没有儿子的结点为叶结点。

Definition 2.4

对于一棵有根树，称结点 u 含自身的后代结点集的导出子图为结点 u 的子树 $\mathrm{subtree}(u)$，称没有儿子的结点为叶结点。

一般我们不用子树称呼一棵树的作为树的子图，而简单地用连通子图进行指代。

Definition 2.4

对于一棵有根树，称结点 u 含自身的后代结点集的导出子图为结点 u 的子树 $\mathrm{subtree}(u)$，称没有儿子的结点为叶结点。

一般我们不用子树称呼一棵树的作为树的子图，而简单地用连通子图进行指代。

Definition 2.5

对于一棵有根树，两个结点 u, v 的距离 $\mathrm{dist}(u, v)$ 为 u, v 间唯一简单路径的长度，称一棵树所有点对距离的最大值为它的直径，称到其它结点的距离最大值最小的结点为中心，称中心到其它结点的距离最大值为树的半径。

Definition 2.4

对于一棵有根树，称结点 u 含自身的后代结点集的导出子图为结点 u 的子树 $\mathrm{subtree}(u)$，称没有儿子的结点为叶结点。

一般我们不用子树称呼一棵树的作为树的子图，而简单地用连通子图进行指代。

Definition 2.5

对于一棵有根树，两个结点 u, v 的距离 $\mathrm{dist}(u, v)$ 为 u, v 间唯一简单路径的长度，称一棵树所有点对距离的最大值为它的直径，称到其它结点的距离最大值最小的结点为中心，称中心到其它结点的距离最大值为树的半径。

对于边带非负权的树，我们可以类似地定义其带权距离、带权直径、带权中心与带权半径。

Definition 2.6

对于一棵根为 r 的有根树，结点 u 的深度 $\mathrm{dep}(u) = \mathrm{dist}(u, r) + \mathrm{dep}(r)$，根的深度一般为 1 或 0，称 $\{u | \mathrm{dep}(u) = d\}$ 为深度为 d 的一层。一个结点的高度为其到叶结点的距离加上叶结点高度的最大值，叶结点的高度一般为 1 或 0，树的高度为其根结点的高度。

Definition 2.6

对于一棵根为 r 的有根树，结点 u 的深度 $\mathrm{dep}(u) = \mathrm{dist}(u, r) + \mathrm{dep}(r)$，根的深度一般为 1 或 0，称 $\{u | \mathrm{dep}(u) = d\}$ 为深度为 d 的一层。一个结点的高度为其到叶结点的距离加上叶结点高度的最大值，叶结点的高度一般为 1 或 0，树的高度为其根结点的高度。

Definition 2.7

对于一棵有根树，我们称一个结点集 S 的最近公共祖先 $\mathrm{lca}(S)$ 为 $\bigcap_{u \in S} \mathrm{anc}(u)$ 中深度最大的一个结点。

Definition 2.6

对于一棵根为 r 的有根树，结点 u 的深度 $\mathrm{dep}(u) = \mathrm{dist}(u, r) + \mathrm{dep}(r)$，根的深度一般为 1 或 0，称 $\{u | \mathrm{dep}(u) = d\}$ 为深度为 d 的一层。一个结点的高度为其到叶结点的距离加上叶结点高度的最大值，叶结点的高度一般为 1 或 0，树的高度为其根结点的高度。

Definition 2.7

对于一棵有根树，我们称一个结点集 S 的最近公共祖先 $\mathrm{lca}(S)$ 为 $\bigcap_{u \in S} \mathrm{anc}(u)$ 中深度最大的一个结点。

类似 min 与 max，我们常将 $\mathrm{lca}\{u_1, u_2, \ldots, u_k\}$ 记为 $\mathrm{lca}(u_1, u_2, \ldots, u_k)$。

Definition 2.8

称每个结点最多含有 k 个儿子的有根树为 k 叉树。一般以 k 叉树称呼一棵树时往往会区分这 k 个儿子，特别地，称二叉树一个结点的第一个儿子为左儿子，第二个儿子为右儿子，它们的子树分别为左子树与右子树。定义高度为 0 的满 k 叉树为空树，高度为 $h(h \in \mathbb{N}^+)$ 的满 k 叉树为满足根结点 k 个儿子的子树均为高度为 $h-1$ 的满 k 叉树的 k 叉树。定义高度为 0 的完全 k 叉树为空树，高度为 $h(h \in \mathbb{N}^+)$ 的完全 k 叉树为满足存在 $i \in [1,k] \cap \mathbb{N}$，其第 i 个儿子的子树为高度为 $h-1$ 的完全 k 叉树，第 $j \in [1,i) \cap \mathbb{N}$ 个儿子的子树为高度为 $h-1$ 的满 k 叉树，第 $j \in (i,k] \cap \mathbb{N}$ 个儿子的子树为高度为 $h-2$ 的满 k 叉树。

Definition 2.8

称每个结点最多含有 k 个儿子的有根树为 k 叉树。一般以 k 叉树称呼一棵树时往往会区分这 k 个儿子，特别地，称二叉树一个结点的第一个儿子为左儿子，第二个儿子为右儿子，它们的子树分别为左子树与右子树。定义高度为 0 的满 k 叉树为空树，高度为 $h(h \in \mathbb{N}^+)$ 的满 k 叉树为满足根结点 k 个儿子的子树均为高度为 $h-1$ 的满 k 叉树的 k 叉树。定义高度为 0 的完全 k 叉树为空树，高度为 $h(h \in \mathbb{N}^+)$ 的完全 k 叉树为满足存在 $i \in [1,k] \cap \mathbb{N}$，其第 i 个儿子的子树为高度为 $h-1$ 的完全 k 叉树，第 $j \in [1,i) \cap \mathbb{N}$ 个儿子的子树为高度为 $h-1$ 的满 k 叉树，第 $j \in (i,k] \cap \mathbb{N}$ 个儿子的子树为高度为 $h-2$ 的满 k 叉树。

由于翻译原因，一些材料将满 k 叉树称为完全 k 叉树。

Definition 2.8

称每个结点最多含有 k 个儿子的有根树为 k 叉树。一般以 k 叉树称呼一棵树时往往会区分这 k 个儿子，特别地，称二叉树一个结点的第一个儿子为左儿子，第二个儿子为右儿子，它们的子树分别为左子树与右子树。定义高度为 0 的满 k 叉树为空树，高度为 $h(h \in \mathbb{N}^+)$ 的满 k 叉树为满足根结点 k 个儿子的子树均为高度为 $h-1$ 的满 k 叉树的 k 叉树。定义高度为 0 的完全 k 叉树为空树，高度为 $h(h \in \mathbb{N}^+)$ 的完全 k 叉树为满足存在 $i \in [1,k] \cap \mathbb{N}$，其第 i 个儿子的子树为高度为 $h-1$ 的完全 k 叉树，第 $j \in [1,i) \cap \mathbb{N}$ 个儿子的子树为高度为 $h-1$ 的满 k 叉树，第 $j \in (i,k] \cap \mathbb{N}$ 个儿子的子树为高度为 $h-2$ 的满 k 叉树。

由于翻译原因，一些材料将满 k 叉树称为完全 k 叉树。

对于 n 个结点的完全 k 叉树，我们可以方便地用 $[1,n] \cap \mathbb{N}$ 将点重标号：根结点重标号为 1，标号为 i 的第 j 个儿子标号为 $(i-1)k+j+1$。也可用 $[0,n) \cap \mathbb{N}$ 将点重标号：根结点重标号为 0，标号为 i 的第 j 个儿子标号为 $ik+j$。

Definition 2.9

对于一棵树，一个结点 u 的度数 $\deg(u)$ 为以 u 为端点的边数。

Definition 2.9

对于一棵树，一个结点 u 的度数 $\deg(u)$ 为以 u 为端点的边数。

Definition 2.10

度数不超过 2 的树称为链，最大度为结点个数减一的树称为菊花。对于有根树，我们通常用度数最小的结点作为链的根，度数最大的结点作为菊花的根。

Problem (树的存储)

给定一棵 b 个结点的树的边集，q 次查询：给定点 a_i，求与结点 a_i 相邻的所有结点。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(\deg(a_i))$。

Problem (树的存储)

给定一棵 b 个结点的树的边集，q 次查询：给定点 a_i，求与结点 a_i 相邻的所有结点。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(\deg(a_i))$。

Solution

我们可以用 `vector`，链式前向星或前向星对每个结点存储相邻的边。`vector` 空间较大，预处理稍慢于链式前向星，链式前向星在遍历时空间连续性略差，前向星预处理慢于前两者。

Problem (树的存储)

给定一棵 b 个结点的树的边集，q 次查询：给定点 a_i，求与结点 a_i 相邻的所有结点。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(\deg(a_i))$。

Solution

我们可以用 `vector`，链式前向星或前向星对每个结点存储相邻的边。`vector` 空间较大，预处理稍慢于链式前向星，链式前向星在遍历时空间连续性略差，前向星预处理慢于前两者。

对于给定每个结点父亲的情况，我们还可以对每个结点存储其父亲和儿子，儿子集合可以同样使用上述方法记录，也可使用左儿子右兄弟，与链式前向星实质等效但时空常数略小。

Problem (树的存储)

给定一棵 b 个结点的树的边集，q 次查询：给定点 a_i，求与结点 a_i 相邻的所有结点。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(\deg(a_i))$。

Solution

我们可以用 `vector`，链式前向星或前向星对每个结点存储相邻的边。`vector` 空间较大，预处理稍慢于链式前向星，链式前向星在遍历时空间连续性略差，前向星预处理慢于前两者。

对于给定每个结点父亲的情况，我们还可以对每个结点存储其父亲和儿子，儿子集合可以同样使用上述方法记录，也可使用左儿子右兄弟，与链式前向星实质等效但时空常数略小。

对于部分题目，结点按拓扑序编号且对每个结点记录了父亲结点编号，我们无需记录儿子结点。

下面描述的问题均已有解法，暂不在此处具体介绍，后面的一些问题可能会在介绍以下问题解法前用到以下问题的解法。

下面描述的问题均已有解法，暂不在此处具体介绍，后面的一些问题可能会在介绍以下问题解法前用到以下问题的解法。

Problem

给定一棵 n 个结点的树，求整棵树的半径、直径、重心与每个结点的深度、高度、度数、子树大小。

要求时间复杂度 $O(n)$。

下面描述的问题均已有解法，暂不在此处具体介绍，后面的一些问题可能会在介绍以下问题解法前用到以下问题的解法。

Problem

给定一棵 n 个结点的树，求整棵树的半径、直径、重心与每个结点的深度、高度、度数、子树大小。

要求时间复杂度 $O(n)$。

Problem (最近公共祖先)

给定一棵 n 个结点的树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(1)$。

Problem (距离)

给定一棵 n 个结点的树，q 次查询：给定结点 u, v，求 $\mathrm{dist}(u, v)$。
强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(1)$。

Problem (距离)

给定一棵 n 个结点的树，q 次查询：给定结点 u, v，求 $\mathrm{dist}(u, v)$。
强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(1)$。

Definition 3.1 (k 级祖先)

对于一棵有根树中的结点 u 与非负整数 k，称 u 的 k 级祖先 $\mathrm{par}_k(u)$ 为 par 的 k 次复合幂在点 u 的值，即结点 u 沿着到根的简单路径移动 k 条边后到达的结点。

Problem (距离)

给定一棵 n 个结点的树，q 次查询：给定结点 u, v，求 $\mathrm{dist}(u, v)$。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(1)$。

Definition 3.1 (k 级祖先)

对于一棵有根树中的结点 u 与非负整数 k，称 u 的 k 级祖先 $\mathrm{par}_k(u)$ 为 par 的 k 次复合幂在点 u 的值，即结点 u 沿着到根的简单路径移动 k 条边后到达的结点。

Problem (k 级祖先)

给定一棵 n 个结点的树，q 次查询：给定结点 u 与整数 k，求 $\mathrm{par}_k(u)$。

强制在线，要求时间复杂度预处理 $O(n)$，单次查询 $O(1)$。