23 春季基础算法 B Contest05

南京大学 陈佳庚

1254068357@qq.com

2023 年 3 月 25 日

题目概览

1. 二叉树游走
2. 验证 BFS 序列
3. 目录归档
4. MAS 的平衡多叉树

我认为的难度: $\mathrm{T1} = \mathrm{T4} < \mathrm{T2} \ll \mathrm{T3}$

二叉树游走

题目大意

一颗以 1 为根节点的无穷大满二叉树，给定初始节点 X 以及长度为 N 的操作序列，问最终位于哪个节点。
($1 \leq N \leq 10^6$，保证最终答案不超过 10^{18})

二叉树游走

题目大意

一颗以 1 为根节点的无穷大满二叉树，给定初始节点 X 以及长度为 N 的操作序列，问最终位于哪个节点。
($1 \leq N \leq 10^6$，保证最终答案不超过 10^{18})

- 首先很容易想到可以暴力模拟，依次读入操作并更新当前节点即可。

二叉树游走

题目大意

一颗以 1 为根节点的无穷大满二叉树，给定初始节点 X 以及长度为 N 的操作序列，问最终位于哪个节点。
($1 \leq N \leq 10^6$，保证最终答案不超过 10^{18})

- 首先很容易想到可以暴力模拟，依次读入操作并更新当前节点即可。
- 然而答案虽然保证最终是在 long long 范围内，中间运算过程却可能溢出。

二叉树游走

题目大意

一颗以 1 为根节点的无穷大满二叉树，给定初始节点 X 以及长度为 N 的操作序列，问最终位于哪个节点。
($1 \leq N \leq 10^6$，保证最终答案不超过 10^{18})

- 首先很容易想到可以暴力模拟，依次读入操作并更新当前节点即可。
- 然而答案虽然保证最终是在 long long 范围内，中间运算过程却可能溢出。
- ~~一种无脑的解决方法：实现高精度的加法、乘法、除法。~~

二叉树游走

题目大意

一颗以 1 为根节点的无穷大满二叉树，给定初始节点 X 以及长度为 N 的操作序列，问最终位于哪个节点。
($1 \leq N \leq 10^6$，保证最终答案不超过 10^{18})

- 首先很容易想到可以暴力模拟，依次读入操作并更新当前节点即可。
- 然而答案虽然保证最终是在 long long 范围内，中间运算过程却可能溢出。
- ~~一种无脑的解决方法：实现高精度的加法、乘法、除法。~~
- 作为第一题显然不应该投入很多时间在代码实现上。

二叉树游走

题目大意

一颗以 1 为根节点的无穷大满二叉树，给定初始节点 X 以及长度为 N 的操作序列，问最终位于哪个节点。
($1 \leq N \leq 10^6$，保证最终答案不超过 10^{18})

- 首先很容易想到可以暴力模拟，依次读入操作并更新当前节点即可。
- 然而答案虽然保证最终是在 long long 范围内，中间运算过程却可能溢出。
- ~~一种无脑的解决方法：实现高精度的加法、乘法、除法。~~
- 作为第一题显然不应该投入很多时间在代码实现上。
- 如何仿照前述暴力的代码思路成功 AC 呢？

二叉树游走

- 我们能用的性质只有答案不会超出 long long 范围，也就是说即使中间节点号溢出了，最终仍然会回退回来。

二叉树游走

- 我们能用的性质只有答案不会超出 long long 范围，也就是说即使中间节点号溢出了，最终仍然会回退回来。
- 假如我们在一次 L 操作或者 R 操作后面紧跟了一次 U 操作，那么是不是相当于原地不动？

二叉树游走

- 我们能用的性质只有答案不会超出 long long 范围，也就是说即使中间节点号溢出了，最终仍然会回退回来。
- 假如我们在一次 L 操作或者 R 操作后面紧跟了一次 U 操作，那么是不是相当于原地不动？
- 那么我们可以将这种原地不动的操作去除，只保留真正有效的操作，最后再进行答案的计算。

二叉树游走

- 我们能用的性质只有答案不会超出 long long 范围，也就是说即使中间节点号溢出了，最终仍然会回退回来。
- 假如我们在一次 L 操作或者 R 操作后面紧跟了一次 U 操作，那么是不是相当于原地不动？
- 那么我们可以将这种原地不动的操作去除，只保留真正有效的操作，最后再进行答案的计算。
- 使用栈维护当前操作序列，假如当前操作为 U 且栈顶为 L/R，那么弹出栈顶，否则将当前操作入栈。

二叉树游走

- 我们能用的性质只有答案不会超出 long long 范围，也就是说即使中间节点号溢出了，最终仍然会回退回来。
- 假如我们在一次 L 操作或者 R 操作后面紧跟了一次 U 操作，那么是不是相当于原地不动？
- 那么我们可以将这种原地不动的操作去除，只保留真正有效的操作，最后再进行答案的计算。
- 使用栈维护当前操作序列，假如当前操作为 U 且栈顶为 L/R，那么弹出栈顶，否则将当前操作入栈。
- 最后从栈底扫描到栈顶，按照暴力的思路去模拟节点的变动即可，可以证明任意时刻答案均不会超出 long long。

二叉树游走

- 我们能用的性质只有答案不会超出 long long 范围，也就是说即使中间节点号溢出了，最终仍然会回退回来。
- 假如我们在一次 L 操作或者 R 操作后面紧跟了一次 U 操作，那么是不是相当于原地不动？
- 那么我们可以将这种原地不动的操作去除，只保留真正有效的操作，最后再进行答案的计算。
- 使用栈维护当前操作序列，假如当前操作为 U 且栈顶为 L/R，那么弹出栈顶，否则将当前操作入栈。
- 最后从栈底扫描到栈顶，按照暴力的思路去模拟节点的变动即可，可以证明任意时刻答案均不会超出 long long。
- 时间复杂度是 $O(N)$。

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

- 问题的关键在于如何判断一个树上的 BFS 序是合法的 BFS 序？

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

- 问题的关键在于如何判断一个树上的 BFS 序是合法的 BFS 序？
- 首先其肯定得是 1 到 n 的一个排列，对于所有合法的 BFS 序其具有什么性质呢？

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

- 问题的关键在于如何判断一个树上的 BFS 序是合法的 BFS 序？
- 首先其肯定得是 1 到 n 的一个排列，对于所有合法的 BFS 序其具有什么性质呢？
- 从 1 号节点出发 BFS 能得到一颗 BFS 树。BFS 树上有什么信息？

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

- 问题的关键在于如何判断一个树上的 BFS 序是合法的 BFS 序？
- 首先其肯定得是 1 到 n 的一个排列，对于所有合法的 BFS 序其具有什么性质呢？
- 从 1 号节点出发 BFS 能得到一颗 BFS 树。BFS 树上有什么信息？
- 一个合法的 BFS 序一定得满足 BFS 树上的层序关系:BFS 树上的深度较高的节点在 BFS 序中一定较后。也就是说将每个点的深度按照 BFS 序排列所得的序列一定是升序的。

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

- 问题的关键在于如何判断一个树上的 BFS 序是合法的 BFS 序？
- 首先其肯定得是 1 到 n 的一个排列，对于所有合法的 BFS 序其具有什么性质呢？
- 从 1 号节点出发 BFS 能得到一颗 BFS 树。BFS 树上有什么信息？
- 一个合法的 BFS 序一定得满足 BFS 树上的层序关系:BFS 树上的深度较高的节点在 BFS 序中一定较后。也就是说将每个点的深度按照 BFS 序排列所得的序列一定是升序的。
- 那么我们可以预处理出从 1 号节点出发所得的 BFS 树，计算所有点在 BFS 树上的深度，将其按照给定的 BFS 序排列检查是否是升序的即可。

验证 BFS 序列

题目大意

给定一颗 n 个点的树和若干个 BFS 序，分别判断每个 BFS 序是否是一个从 1 号点出发的合法 BFS 序。

$1 \leq n \leq 10^5, 1 \leq \sum |B| \leq 10^6$

- 问题的关键在于如何判断一个树上的 BFS 序是合法的 BFS 序？
- 首先其肯定得是 1 到 n 的一个排列，对于所有合法的 BFS 序其具有什么性质呢？
- 从 1 号节点出发 BFS 能得到一颗 BFS 树。BFS 树上有什么信息？
- 一个合法的 BFS 序一定得满足 BFS 树上的层序关系:BFS 树上的深度较高的节点在 BFS 序中一定较后。也就是说将每个点的深度按照 BFS 序排列所得的序列一定是升序的。
- 那么我们可以预处理出从 1 号节点出发所得的 BFS 树，计算所有点在 BFS 树上的深度，将其按照给定的 BFS 序排列检查是否是升序的即可。
- 单次询问的时间复杂度是 $O(N)$。

目录归档

题目大意

给定若干目录，实现目录的树状结构的重建工作。

- 思路很简单，构造出目录树，dfs 目录树输出答案即可。

目录归档

题目大意

给定若干目录，实现目录的树状结构的重建工作。

- 思路很简单，构造出目录树，dfs 目录树输出答案即可。
- 几个小注意点: 需要按字典序输出，先输出子目录再输出子文件，缩进。

目录归档

题目大意

给定若干目录，实现目录的树状结构的重建工作。

- 思路很简单，构造出目录树，dfs 目录树输出答案即可。
- 几个小注意点: 需要按字典序输出，先输出子目录再输出子文件，缩进。
- 难点在于如何实现构造目录树的代码。

目录归档

- 使用链表维护目录树，将儿子分为两类: 目录、文件。对于每个儿子存储两个信息: 该儿子的对应字符串 (用于根据字典序遍历)，该儿子对应节点的指针 (用于递归构建目录树)。

目录归档

- 使用链表维护目录树，将儿子分为两类: 目录、文件。对于每个儿子存储两个信息: 该儿子的对应字符串 (用于根据字典序遍历)，该儿子对应节点的指针 (用于递归构建目录树)。
- 对于每个输入从根节点开始递归构建，首先从字符串中获得当前儿子的名字与当前儿子的类型，检查是否已经存在，若不存在则构建一个新的儿子节点。同时处理出传给儿子的新输入字符串，递归访问该儿子构建目录树。

目录归档

- 使用链表维护目录树，将儿子分为两类: 目录、文件。对于每个儿子存储两个信息: 该儿子的对应字符串 (用于根据字典序遍历)，该儿子对应节点的指针 (用于递归构建目录树)。
- 对于每个输入从根节点开始递归构建，首先从字符串中获得当前儿子的名字与当前儿子的类型，检查是否已经存在，若不存在则构建一个新的儿子节点。同时处理出传给儿子的新输入字符串，递归访问该儿子构建目录树。
- 递归时若输入字符串为空说明当前节点为叶子，直接返回。

目录归档

- 使用链表维护目录树，将儿子分为两类: 目录、文件。对于每个儿子存储两个信息: 该儿子的对应字符串 (用于根据字典序遍历)，该儿子对应节点的指针 (用于递归构建目录树)。
- 对于每个输入从根节点开始递归构建，首先从字符串中获得当前儿子的名字与当前儿子的类型，检查是否已经存在，若不存在则构建一个新的儿子节点。同时处理出传给儿子的新输入字符串，递归访问该儿子构建目录树。
- 递归时若输入字符串为空说明当前节点为叶子，直接返回。
- 输出时 DFS 目录树，记录 DFS 的深度来输出缩进，同时将两类儿子按照名字字典序排序，递归访问儿子即可。

目录归档

- 使用链表维护目录树，将儿子分为两类: 目录、文件。对于每个儿子存储两个信息: 该儿子的对应字符串 (用于根据字典序遍历)，该儿子对应节点的指针 (用于递归构建目录树)。
- 对于每个输入从根节点开始递归构建，首先从字符串中获得当前儿子的名字与当前儿子的类型，检查是否已经存在，若不存在则构建一个新的儿子节点。同时处理出传给儿子的新输入字符串，递归访问该儿子构建目录树。
- 递归时若输入字符串为空说明当前节点为叶子，直接返回。
- 输出时 DFS 目录树，记录 DFS 的深度来输出缩进，同时将两类儿子按照名字字典序排序，递归访问儿子即可。
- 时间复杂度为 $O(N \times 255)$。

MAS 的平衡多叉树

题目大意

给定若干颗以 1 为根的多叉树，判断其是否是平衡的。
$t \leq 10, n \leq 55$

- 和判断二叉平衡树的方法非常类似，可以递归判断同时记录子树深度。

MAS 的平衡多叉树

题目大意

给定若干颗以 1 为根的多叉树，判断其是否是平衡的。
$t \leq 10, n \leq 55$

- 和判断二叉平衡树的方法非常类似，可以递归判断同时记录子树深度。
- 首先递归访问所有子树判断其是否平衡，若都平衡则检查深度最大的子树和深度最小的子树的深度差值。

MAS 的平衡多叉树

题目大意

给定若干颗以 1 为根的多叉树，判断其是否是平衡的。
$t \leq 10, n \leq 55$

- 和判断二叉平衡树的方法非常类似，可以递归判断同时记录子树深度。
- 首先递归访问所有子树判断其是否平衡，若都平衡则检查深度最大的子树和深度最小的子树的深度差值。
- 时间复杂度为 $O(n)$。

The End

谢谢大家