贪心法 - 实验3报告

李雨轩

2204112913

计算机2205

一、题目描述

设T为一带权树，树中的每个边的权都为整数。又设S为T的一个顶点的子集，从T中删除S中的所有结点，则得到一个森林，记为T/S。如果T/S中所有树从根到叶子节点的路径长度都不超过d，则称T/S是一个d森林。设计一个算法求T的最小顶点集合S，使T/S为一个d森林。

二、问题分析

我们需要考虑如何找到最小的顶点集合S，使得删除S后的树T/S成为一个d森林。这是一个优化问题，我们需要找到一种方法来最小化S的大小。

因此，我们考虑采用贪心策略。其核心思想是尽早发现并处理距离超出阈值d的部分，从而最小化顶点集合S的大小。通过自底向上搜索和及时删除节点，我们可以有效地控制整体距离，以确保得到一个满足条件的树T/S。

这种贪心策略好比是修剪一棵树，我们从树的末梢开始修剪，然后逐渐向根部移动。

我们首先从最外层的叶子节点开始，因为它们离树的顶端最远。这些叶子节点可能延伸得很远，因此是我们首要考虑的对象。然后，我们逐渐向上检查每个节点，查看它及其子树的总距离是否超出了我们设置的限制。

如果一个节点及其子树的总距离超出了限制，我们就会将这个节点从树中移除。这样做的好处是，我们及时发现了距离过长的部分，并且通过移除这些部分，我们可以确保整个树都符合我们的要求。

接着我们继续向上检查并修剪树，直到我们无法再找到需要移除的节点，或者直到我们到达了树的顶部。这种逐步修剪的方法确保了我们尽早发现和处理了距离过长的部分，从而有效地控制了整个树的长度。

三、算法设计

1. 贪心策略

对于dTree问题，我们采取以下的贪心策略：

1. **从叶节点开始往上搜索：** 首先从叶节点（即没有子节点的节点）开始搜索。
2. **判断节点往下的最大距离是否大于阈值d：** 对于每个搜索到的节点，我们计算它往下的最大距离。如果这个距离已经大于给定的阈值d，说明这个节点及其子树的距离已经超过了限制，需要进行删除操作。
3. **删除操作：** 如果某个节点的往下的最大距离已经大于阈值d，那么删除这个节点。这样做的目的是为了防止整体距离超过限制，此时立刻删除这个节点通常比往后再删除多个节点更有效率。
4. **往上继续搜索：** 在删除节点之后，我们继续向上搜索，寻找下一个需要删除的节点。这个过程持续进行，直到找不到任何需要删除的节点，或者搜索到根节点为止。

2. 证明贪心策略的正确性

2.1 基础情况

考虑最简单的情况，即树 T 只包含一个节点，且该节点的权重为零（即没有边）。

- 如果这个单一节点的权重不超过阈值 d，那么按照贪心策略，我们不需要删除任何节点，因为没有路径超过 d。这是最优的，因为删除节点数量为零，满足条件。
- 如果这个单一节点的权重超过阈值 d（虽然这种情况在只有一个节点且无边的树中不可能发生，但理论上考虑），那么我们需要删除这个节点，这同样是最优的，因为没有其它方式可以减少路径长度。

2.2 归纳假设

假设对于所有包含 n 个节点的树，当节点的最大路径长度超过 d 时，使用贪心策略删除具有超过阈值路径的最深节点能达到一个全局最优解，即路径长度不超过 d 且删除的节点数最少。

2.3 归纳步骤

现在，我们需要证明当树 T 有 $n+1$ 个节点时，这个策略仍然有效。

- 考虑树 T 有一个根节点和若干子树。如果根节点及其子树中的任何节点的最大路径长度超过 d，我们按照贪心策略选择最深的那个节点进行删除。
- 当删除了一个或多个这样的节点后，剩余的每个子树都变成了新的树，每棵树的节点数少于 $n+1$。
- 根据归纳假设，每个这样的子问题（子树）都可以使用相同的策略达到最优解。即在这些子树中，所有从根到叶的路径都不超过 d，且没有额外的节点被删除。
- 删除操作保证了在这些子树和树 T 中，没有路径超过 d，因为任何超过 d 的路径都会导致其最深节点的删除。

2.4 结论

由于删除操作是针对最深的超阈值节点进行的，它最小化了对树的影响，同时保证了路径长度的限制。归纳步骤证明了这种策略在从较小的子树扩展到更大的树时仍然保持有效。因此，通过归纳，我们可以断言贪心策略对所有大小的树都有效，满足问题的要求。

3. 算法描述

该算法旨在找到一个顶点的最小子集 S，从带权树 T 中删除这些顶点后，得到的森林 T/S 中所有从根到叶的路径的权重和不超过给定的阈值 d。实现这一目标的方法是通过一种改进的深度优先搜索（DFS）算法，其在遍历树的过程中动态地决定是否需要删除某个节点来保证路径长度的约束。

1. **初始化**：
   - 输入总节点数和路径长度阈值。
   - 构建树的邻接表表示，并初始化每个节点的删除状态。
2. **深度优先搜索 (DFS) 函数**：
   - 从根节点开始，递归地遍历每个节点。
   - 对于当前节点 v，遍历所有子节点 u：
     - 计算从 v 到 u 的距离，包括边的权重。
     - 如果子节点 u 的路径长度加上当前边权超过阈值，并且当前节点 v 还未被删除，则将 v 标记为删除，并增加删除计数。
   - 如果当前节点 v 被标记为删除，则返回一个很大的负数表示不需要进一步计算这个分支（因为 v 已经删除，其子树路径无需考虑）。
   - 否则，返回 v 的子树中的最大路径长度。
3. **输出结果**：
   - 输出删除节点的数量。

四、算法实现：源代码

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;
// 算法要求：设计一个算法求 T 的最小顶点集合 S，使 T/S 为一个 d 森林。

class dTree
{
private:
    int _numNodes, _threshold, _answer;
    vector<vector<pair<int, int>>> _tree;
    vector<bool> _state;
    const int INF = 1e9; // 用于表示无穷大

public:
    dTree(int nodes, int threshold) : _numNodes(nodes), _threshold(threshold), _answer(0)
    {
        _tree.resize(_numNodes);
        _state.resize(_numNodes, false);
        for (int i = 0; i < _numNodes; i++)
        {
```

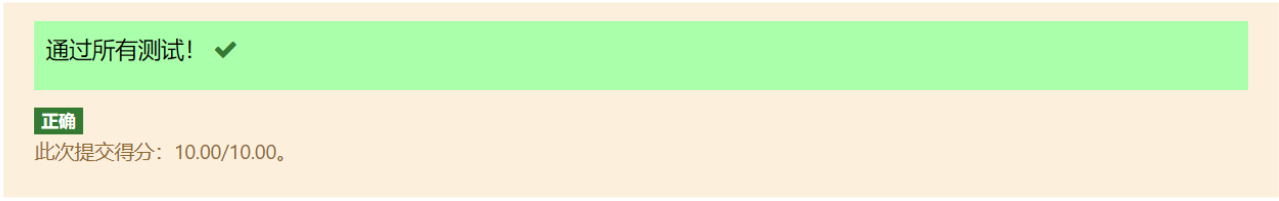
```
            int branches;
            cin >> branches;
            while (branches--)
            {
                int node, w;
                cin >> node >> w;
                _tree[i].emplace_back(make_pair(node, w));
            }
        }
    }

    int dfs(int node)
    {
        int result = 0;
        for (auto it = _tree[node].begin(); it != _tree[node].end(); ++it)
        {
            int neighbor = it->first;
            int weight = it->second;
            int distance = dfs(neighbor);
            if (!_state[node])
                distance = max(distance + weight, 0);
            if (distance > _threshold && !_state[node])
            {
                _answer++;
                _state[node] = true;
                continue;
            }
            result = max(result, distance);
        }
        if (_state[node])
            return -INF;
        else
            return result;
    }

    void solution()
    {
        dfs(0); // 从根节点开始
        cout << _answer << endl;
    }
};

int main()
{
    int n, d; // n为顶点个数，d为路径长度
    cin >> n >> d;
    dTree dt(n, d); // 构建与初始化树
    dt.solution();  // 通过solution函数输出结果
    return 0;
}
```

五、运行结果：Moodle平台测试用例结果




```
rouge ~/XJTUALGO/Labs  main ± g++ dTree.cpp -o dTree && ./dTree
5 5
3 1 4 2 9 3 2
0
1 4 5
0
0
1
rouge ~/XJTUALGO/Labs  main ±
```

六、实验过程说明

本次实验中，我设计并实现了一个基于贪心策略的算法，用于解决带权树问题中的最小顶点集合求解。该算法通过动态地删除超出路径长度阈值的节点，使得剩余树的所有从根到叶子节点的路径长度都不超过给定阈值。具体来说，算法采用深度优先搜索（DFS）遍历树的方式，不断检查并删除节点，直到满足条件为止。

我们证明了贪心策略的正确性，并给出了算法的实现思路和具体步骤。通过Moodle平台的测试用例，验证了算法的正确性。