算法讨论班第 14 期—郭清沛

2015 年 11 月 27 日

1.Populating Next Right Pointers in Each Node

Given a binary tree

```
struct TreeLinkNode {
  TreeLinkNode *left;
  TreeLinkNode *right;
  TreeLinkNode *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.

Initially, all next pointers are set to NULL.

Note:

You may only use constant extra space.

You may assume that it is a perfect binary tree (ie, all leaves are at the same level, and every parent has two children).

For example,

Given the following perfect binary tree,

```
        1
      /  \
     2    3
    / \  /  \
   4  5  6  7
```

After calling your function, the tree should look like:

```
        1 -> NULL
      /  \
     2 -> 3 -> NULL
    / \  / \
   4->5->6->7 -> NULL
```

解题思路：

直观上看题目要将每一层的节点相互连接起来，这让我们想到广度优先遍历二叉树结点，然后将每一层的节点连接，但是 BFS 需要用队列来存储每一层的节点，并不是常数空间。

仔细观察，需要连接的只有两部分：1）同一父节点的兄弟之间　2）不同父节点的兄弟之间

1）处理简单：遍历到父节点时将其左孩子的 next 指向右孩子

2）处理下个节点时，需要保存同层的上个节点，以便将上个节点的右孩子指向当前结点的左孩子

代码：

```
class Solution {
public:
    void connect(TreeLinkNode *root) {
        TreeLinkNode * curNode = root;
        TreeLinkNode * levelNode, *lastVisitNode;
        while (curNode&&curNode->left)//循环处理，直到处理完所有有孩子的层
        {
            levelNode = curNode;//保存当前层的第一个节点，用于向下层深入
            lastVisitNode = NULL;
            while (curNode)//循环处理，直到处理完当前层的所有节点
            {
                if (lastVisitNode)//将上个节点的右孩子指向当前结点的左孩子
                    lastVisitNode->right->next = curNode->left;
                curNode->left->next = curNode->right;//当前节点的左孩子指向右孩子
                lastVisitNode = curNode;
                curNode = curNode->next;
            }
            curNode = levelNode->left;
        }
    }
};
```

2.Word Ladder

Given two words (beginWord and endWord), and a dictionary's word list, find the length of shortest transformation sequence from beginWord to endWord,such that

1)Only one letter can be changed at a time

2)Each intermediate word must exist in the word list

For example,

Given:

beginWord = "hit"

endWord = "cog"

wordList = ["hot","dot","dog","lot","log"]

Return

[

["hit","hot","dot","dog","cog"],

["hit","hot","lot","log","cog"]

]

Note:

All words have the same length.

All words contain only lowercase alphabetic characters.

解题思路：

正常思路应该怎样做？？

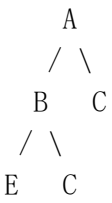
1.必须用 BFS，如果用 DFS 必定会超时

BFS：用队列存储当前层的节点，当访问当前层的某个节点时，把它的子节点入队

这样就能保证：访问完当前层之后才会访问当前层所有节点的子节点

2.为甚么要用 BFS：

这基于本题的一个很重要的条件：找到最短路径，而不是所有路径（有的长路径也可能到达，但我们不要）~



注意：如果一个节点在上层出现过，那么这个节点不可能在下层出现！！！(不仅仅只是路径上的前面出现过在该路径之后不能出现——这样也就意味着不仅是树的深度不会超过 dict.size()+2，而是树的所有节点数目都不会超过 dict.size()+2)

如上图：如果节点 C 在上层出现过，再在 B 下面层出现时，假设节点 C 可达 end,那么 A-B-C 路径一定会比 A-C 那条路径长

有种 Best Effort 的感觉：尽可能的往前迈，尽可能提前到达

寻找最短路径——dijkstra 算法

3.基于以上结论：我们可以知道：如果可达，最多经过 dict.size()+2 层可达

----------127----------

4.如果 BFS 已经找到了最短路径的层数 finalLevel，那么裁剪树根到 finalLevel 的每一层节点构成子树，在这棵子树中继续寻找其它的可能最短路径

5.已知各层的子树，寻找其它路径时，为了继续修剪树枝，从 end 逆序往上寻找，用 dfs(如果从上往下，那么又会有很多无用的分支最后可能到达不了 end)

核心代码：

```
63          int finalLevel = dict.size() + 2;
64          while (level < finalLevel && levelSize)
65          {
66              string curChoice = choices.front();
67              choices.pop();
68              levelWord.push_back(curChoice);
69              levelSize--;
70              for (auto i = 0; i < curChoice.size(); i++)
71              {
72                  string tmp = curChoice;
73                  for (char j = 'a'; j <= 'z'; j++)
74                  {
75                      tmp[i] = j;
76                      if (tmp == end)
77                          finalLevel = level + 1;
78                      if (dict.find(tmp) != dict.end())
79                      {
80                          choices.push(tmp);
81                          dict.erase(tmp);
82                      }
83                  }
84              }
85              if (levelSize == 0)
86              {
87                  levelInfo.push_back(levelWord);
88                  levelWord.clear();
89                  level += 1;
90                  levelSize = choices.size();
91              }
92          }
```

全部代码见附件

下周由李玥佩主讲：

主题：树和图的遍历

题目：105，106，133