算法设计与分析 LECTURE 5

第六周作业-solution

LECTURER: 杨启哲 LAST MODIFIED: 2023 年 12 月 23 日

1. (教材习题 3.24) 假设 $Find$ 和 $Union$ 操作是利用按秩合并的方式进行的，请给出一个长度为 n 的 $Find$ 和 $Union$ 序列，使得其需要 $\Theta(n\log n)$ 的时间。这里假定元素集合是 $\{1,2,\ldots,n\}$.

   解答. 这个实例的构造在于要让树的高度尽可能平均的增加，所以序列如下：

   (1) $Union(1,2), Union(3,4), \ldots, Union(n-1,n)$
   (2) $Union(1,3), Union(5,7), \ldots, Union(n-3,n-1)$
   (3) $Union(1,5), Union(9,13), \ldots, Union(n-7,n-3)$
   (4) $\ldots$
   (5) $Union(1,1+2^{k-1}),,\ldots, Union(n-2^k+1, n-2^{k-1}+1)$
   (6) $Find(1), Find(1+2^{k-1}), \ldots, Find(n-2^{k-2}+1)$

   由于每次 $Union$ 操作都是按秩合并，所以每次合并的两个集合的秩都是相同的，从而每次合并后的秩都是原来的秩加一，因此上述第 k 轮后每次合并后的树的高度为 k，因此在第 k 次操作过后，所以每次 $Union$ 操作的时间复杂度都是 $\Theta(\log k)$，所以总的时间复杂度是 $\Theta(n\log n)$. 同时不难验证，这是一个 n 次的 $Find$ 和 $Union$ 序列 □

   Remark 0.1

   事实上，如果题目要求 $O(n)$ 的操作序列，叙述可以简单一些，但现在为了严格构出一个 n 轮的操作序列，所以需要仔细计算次数。事实上我们只需要保证在 $O(\log n)$ 次数的 $Find$ 操作需要执行某个 n 的倍数次就可以了，哪怕这个倍数可能是个很小的分数。

2. (教材习题 8.8) 假定在无向图 G 上应用算法 DFS，给出一个算法，把 G 的边分为树边或者回边。

   解答. 只要注意到在 DFS 的过程中，每次访问一个节点时，如果发现这个节点已经被访问过了，那么这条边就是回边，否则就是树边。算法流程如下：

   标记回边与树边

   输入: 图 G 和开始的节点 v
   输出: G 的边被分为树边或者回边

   ```
   DFS(G, v, p = NULL)
   function DFS(G, v, p)
       visited[v] ← true
       for u ∈ G.adj[v] do
           if visited[u] and u ≠ p then
               type[v, u] ← 回边
           else if not visited[u] then
   ```

```
            type[v, u] ← 树边
            DFS(G, u, v)
        end if
    end for
end function
```

算法仅仅是在 DFS 的过程中同步添加标注，所以时间复杂度依旧为 $O(|V|+|E|)$. □

3. (教材习题 8.12) 请给出一个 $O(n)$ 时间的算法，确定在 n 个顶点的连通无向图中是否包含回路。

解答. 判定一个无向图是否包含回路，只需要判定在 DFS 的过程中是否存在回边即可。而判定回边的过程在上一题中已经给出了，只需要确定该点是否被访问过并且不是上一个节点即可。

同时注意到如果一个无向图有至少 $|V|$ 条边，那么一定存在回路，因此我们只需要判断 $|E|$ 是否大于 $|V|$ 即可。结合两者，算法流程如下：

(1) 判断图的边数是否大于节点数，如果是则返回 True，否则继续。
(2) 从任意一个节点开始 DFS，如果发现回边，则返回 True，否则返回 False.

由于 DFS 的时间复杂度为 $O(|V|+|E|)$，而算法进行 DFS 时一定有 $|E|<|V|$，从而算法时间复杂性为 $O(|V|)=O(n)$。 □