第 *3-7* 讲*:* 图的遍历

姓名： 林凡琪　　**学号：** *211240042*

评分： ______　　**评阅：** ______

2022 年 *10* 月 *19* 日

请独立完成作业，不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

1　作业（必做部分）

题目 1 (TC 22.1-3)

解答：
对于邻接矩阵：$G = G^T$ 所以不用操作
对于邻接链表：

```
procedure GT($n$)
    Bdj[1..V];
    for $i = 1, i <= V; i++$ do
        for $j = 1; j <= V_o f_a dj[i]; j++$ do
            $Bdj[Adj[i][j]] \leftarrow i$
        end for
    end for
end procedure
```

时间复杂度：$O(|E| + |V|)$

题目 2 (TC 22.1-8)
Suppose that instead of a linked list, each array entry $Adj[u]$ is a hash table containing the vertices v for which $(u, v) \in E$. If all edge lookups are equally likely, what is the expected time to determine whether an edge is in the graph? What disadvantages does this scheme have? Suggest an alternate data structure for each edge list that solves these problems. Does your alternative have disadvantages compared to the hash table? 假设每个数组条目 $Adj[u]$ 不是链接列表，而是一个哈希表，其中包含顶点 v，其中 $(u\ v) inE$。如果所有边缘查找的可能性相等，则确定边缘是否在图形中的预期时间是多少？这个方案有什么缺点？为每个边缘列表建议一个替代数据结构来解决这些问题。与哈希表相比，您的替代方案是否有缺点？

解答：
期望查找时间是 $O(1)$，最差查找时间是 $O(|V|)$
替代方案：
首先把在散列表内的点都排序，然后用二分法查找，这样期望查找时间就是 $O(lg|V|)$。
缺点是期望查找时间变长，并且需要时间进行排序。

题目 3 (TC 22.2-3(”lines 5 and 14” 改为”line 18”))

解答:
将 18 行删除后，节点不再变成黑色。但是同样不会进入 line 13 的 if 判断，不会造成重复判断。
所以用两个颜色标明就可以。

题目 4 (TC 22.2-4)

解答:
因为边的数目变成了 V^2, 所以查询边的时间 $O(V^2)$
运行时间 $O(V + V^2) = O(V^2)$

题目 5 (TC 22.2-5)

解答:
(1) 证明：根据定理 22.5（广度优先算法的正确性）$v.d$ 是 s 到 v 的最短路径，所以不会因为在邻接链表里的顺序不同而改变。
(2) 证明：在图 22-3 可见，$t.d = x.d$，且 u 为他们俩共同拥有的直系后代。因为 t 的顺序比 x 前，所以在生成的广度优先树中，u 是 t 的后裔；倘若 x 的顺序比 t 早，那么 u 就会是 x 的后裔。

题目 6 (TC 22.3-6)
证明：再无向图中，根据深度优先搜索算法是先探索 (u, v) 还是先探索 (v,u) 来将边 (u,v) 分类为树边或者后向边，与根据分类列表中的 4 中类型的次序进行分类是等价的。

解答:
根据定理 22.10，无向图的每条边要么是树边，要么是后向边。首先假设 v 是通过探索边缘 (u, v) 首先发现的。然后根据定义，(u, v) 是树边。此外，(u, v) 必须在 (u, v) 之前被发现，因为一旦探索了 (u, v) ，就必然发现了 v。现在假设 v 不是首先被 (u, v) 发现的。然后它必须被 (r, v) 发现，$r \neq u$

如果 u 尚未被发现，那么如果首先探索 u v ，那么它必须是一个后向边，因为 v 是 u 的祖先。如果发现了 u，那么 u 是 v 的祖先，所以 (v, u) 是后向边。

题目 7 (TC 22.3-8)
Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, and if $u.d < v.d$ in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

解答:
Consider a graph with 3 vertices u, v, and w, and with edges (w, u), (u, w), and (w, v). Suppose that DFS first explores w, and that w's adjacency list has u before v. We next discover u. The only adjacent vertex is w, but w is already grey, so u finishes. Since v is not yet a descendant of u and u is finished, v can never be a descendant of uu.

题目 8 (TC 22.3-9)
Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, then any depth-first search must result in $v.d \leq u.f$.

解答:
Consider a graph with 3 vertices u, v, and w, and with edges (w, u), (u, w), and (w, v).
$u.f = 3 < 4 = v.d$

题目 9 (TC 22.3-12)
证明：我们可以再无向图 G 上使用深度优先搜索来获得图 G 的连通分量，并且深度优先森林所包含的树的棵数与 F 的连通分量数量相同。更准确地说，请给出如何修改深度优先搜索来让其给每个节点赋予一个介于 1 与 k 之间的整数值 $v.cc$，这里 k 是 G 的连通分量数，使得 $u.cc = v.cc$ 当且仅当节点 u 和节点 v 处于同一个连通分量中。

解答:
串页了（3-12）

题目 10 (TC 22.4-2)

解答:
The algorithm works as follows. The attribute u.pathsu.paths of node u tells the number of simple paths from u to v, where we assume that v is fixed throughout the entire process. First of all, a topo sort should be conducted and list the vertex between u, v as $\{v[1], v[2], \dots, v[k-1]\}$. To count the number of paths, we should construct a solution from vv to uu. Let's call u as v[0] and v as v[k], to avoid overlapping subproblem, the number of paths between v_k and u should be remembered and used as k decrease to 0. Only in this way can we solve the problem in $\Theta(V+E)$ An bottom-up iterative version is possible only if the graph uses adjacency matrix so whether v is adjacency to u can be determined in O(1) time. But building a adjacency matrix would cost $\Theta(|V|^2)$, so never mind.

```
SIMPLE-PATHS(G, u, v)
  TOPO-SORT(G)
  let {v[1], v[2]..v[k - 1]} be the vertex between u and v
  v[0] = u
  v[k] = v
  for j = 0 to k - 1
      DP[j] = ∞
  DP[k] = 1
```

```
procedure DFS-CC($G$)
    for each vertex $u \in G.V$ do
        $u.color = WHITE$
        $u.\pi = NIL$
    end for
    $time = 0$
    cc = 1
    for each vertex $u \in G.V$ do
        if $u.color == WHITE$ then
            u.cc = cc
            cc = cc + 1
            DFS-VISIT-CC(G,u)
        end if
    end for
end procedure
procedure DFS-VISIT-CC($G, u$)
    time = time + 1
    u.d = tiime
    u.color = GRAY
    for each vertex $v \in G.Adj[u]$ do
        if v.color ==WHITE then
            v.cc = u.CC
            $v.\pi = u$
            DFS-VISIT-CC(G,v)
        end if
    end for
    u.color = BLACK
    time = time + 1
    u.f = time
end procedure
```

```
return SIMPLE-PATHS-AID(G, DP, 0)

SIMPLE-PATHS-AID(G, DP, i)
if i > k
    return 0
else if DP[i] != ∞
    return DP[i]
else
   DP[i] = 0
   for v[m] in G.adj[v[i]] and 0 < m   k
        DP[i] += SIMPLE-PATHS-AID(G, DP, m)
   return DP[i]
```

题目 11 (TC 22.4-3)
Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in O(V) time, independent of |E|.

Algorithm 1: cycle

解答:

```
procedure CYCLE(G)
    if |G.E| ≥ |G.V| then
        return yes
    end if
    for $v \in G.V$ do
        if !visited[v] then
            p ← dfs(v)                          ▷ 返回是否有返祖边
            if p == 1 then
                return yes
            end if
        end if
    end for
    return no
end procedure
```

题目 12 (TC 22.5-5)
Give an $O(V + E)$-time algorithm to compute the component graph of a directed graph $G = (V, E)$. Make sure that there is at most one edge between two vertices in the component graph your algorithm produces.

解答:
(在下面)

---

2 作业 (选做部分)

题目 1 (TC 22.5-7)

Algorithm 2: scc

```
procedure SCC(G)
    for u ∈ G.V do
        for (u,v) ∈ G.E do
            if u.scc $\neq$ v.scc then
                if hashmap.find(u,v) then
                    continue
                end if
                addedge(u, v)
                hashmap.insert((u, v))
            end if
        end for
    end for
end procedure
```

解答：

3 Open Topics

Open Topics 1 (Tarjan's Algorithms)
[参考资料：R. Tarjan, "Depth-first search and linear graph algorithms," 12th Annual Symposium on Switching and Automata Theory (swat 1971), East Lansing, MI, USA, 1971, pp. 114-121, doi: 10.1109/SWAT.1971.10.]

Open Topics 2 (带边标记的 DFS 算法)
写出带边标记的 DFS 算法并证明算法的正确性

4 反馈