黎锦灏　　518021910771　　0330作业

1、G=(V,E)是一个无向图，每个顶点的度数都为偶数，设计线性时间算法，给 G 中每条边一个方向，使每个顶点的入度等于出度。（请先简单说明算法思想，再给出伪代码，然后证明其时间复杂性符合要求）

解：考虑类似于欧拉回路的算法，对全图进行dfs，得到一个dfs序列，在最后按dfs序前后添加新有向边即可。

伪代码如下：

```
#Algorithm: Graph1
Input: G=(V, E) (an undirected graph), n ,s(a vertex whose degree is not zero)
Output: Q (the sequence of dfs) , Directed_G=(V, E) (a directed graph in which
the in-degree of every vertex equals to its outdegree)
begin
    dfs(s)
    while Q is not empty do
            from:=Q.pop();
            if Q is not empty then
                to:=Q.front();
                add an edge (from , to) to Directed_G=(V, E)
end

Function dfs(u)
begin
     Q.push(u);
     if there exist an edge {u, v} then
        delete the edge {u, v} from G
        dfs(v)
end Function
```

时间复杂性：

　每一条边在dfs遍历时，会被访问两次（两端节点），同时每个节点也将会被访问到，故复杂度为：$O(|V|+|E|)$。

2、连连看游戏中用户可以把两个相同的图用线连到一起，如果连线拐的弯小于等于两个则表示可以消去。设计算法，判断指定的两个图形能否消去。如果是求两个图形间的最少转弯次数呢？

解：第一问、判断能否消去：

　考虑采用BFS广度优先算法，查找p1和p2之间可能存在的路径，从p1开始每个节点向四个方向拓展一格，将新的节点加入队列中，若不是障碍则前进，直到发现目标是p2。在BFS加每个节点进入队列时，记录上一次的移动方向和已经拐弯的次数，如果拐弯次数此时超过2次了则不再拓展。

```
Algorithm : BFS_Graph
Input : G[][] ( the state of the graph )
```

```
Output : flag

begin
    from p1 :
    4 directions( left right up down )
    if not barrier
       put p into queue
       log the direction
       the time of changing direction :=0

    // queue saves the nodes , the time of changing(cnt) and last direction(dir)
    while queue is not empty do
          p:=queue.front(); queue.pop();
          if p is p2 then :
               flag:=true;
               break;

          from p :
                4 directions( left right up down )
                if not barrier
                      log the direction
                      if p.dir == pi.dir then
                           p.cnt: = pi.cnt
                      else p.cnt: = pi.cnt+1

                      if p.cnt>2 then continue;
                      else then put pi into queue

    output flag;
end
```

第二问、求最少转弯次数:

在第一问的基础上，去掉2次的限制，每次抵达p2后记录当前转弯次数，并与当前最少次数作比较。若当前节点的转弯次数超过当前最少次数，则不必继续拓展。最后输出最少转弯次数即可。

3、证明任意连通无向图中必然存在一个点，删除该点不影响图的连通性。用线性时间找到这个点 。

证明：对任意连通无向图都可以进行DFS，得到一棵无向的DFS树，在这棵DFS树上删除一个叶子节点不会改变图的连通性（其余节点仍互相连通），而且树必然存在至少一个叶子节点，因此必然存在一个点，删除后不影响图的连通性。

具体算法：dfs遍历全图，找到一个叶子节点输出即可。由于每条边会被访问两次，故复杂度为 $O(|E|)$，即线性。

代码如下:

```
Algorithm : BFS_Graph
Input : G=(V,E)
Output : v (the point)

function dfs(int u)
begin
    vis[u]:=1;
```

```
    for all edges (u,v) do
         if(vis[v]) continue;
         dfs(v);
    if(all vis[v] == 1) then
          output u;
          exit(0);
end

begin
    dfs(1);
end
```

黎锦灏　518021910771　0401作业

1、对于给定的二叉树，求其最小深度，即从根节点到最近的叶子的距离

解：考虑bfs算法，设定根节点的深度为0，每次向子节点拓展，子节点深度=父节点深度+1，由于采用的是广度优先算法，故找到的第一个叶子节点的深度即最小深度。

代码如下：

```
Algorithm: Min_Dis_Tree
Input: T=(V, E) (a binary tree), root (The root of the tree)
Output: Min_dis
begin
    root.depth := 0
    put root in a queue: Q;
    while Q is not empty do
        w:=Q.front();
        Q.pop();
        if w->left is not NULL then
            w->left->depth := w->depth+1
            put w->left in Q

        if w->right is not NULL then
            w->right->depth := w.depth+1
            put w->right in Q

        if w->left is NULL and w->right is NULL then
            output w.depth
end
```

2、设 G 是有向非循环图，其所有路径最多含 k 条边，设计线性时间算法，将所有顶点分为k+1 组，每一组中任意两个点之间不存在路径

解：考虑对G这一有向非循环图做拓扑排序，每次取出入度为0的点v，将v及其出边从图中删除，并减少其出边顶点的入度，若入度减少为0则加入队列。重复上述操作直至遍历全图。

根据拓扑排序的性质，记录初始入度为0的点属于第1组，之后每个节点入度减少为0时，记录它的Group等于前驱节点的Group+1。注意到，此时分在同一Group的节点入度同时取到0，故他们之间不存在路径。由于路径最多含k条边，即一条路径上最多包含k+1个点，所以分组数不会超过k+1。

代码如下：

```
Algorithm: Topo_group
Input: G=(V, E) (a directed acyclic graph) , v[i] (vertex: indegree,group)
Output: Group information of Vertexes
begin
    for i:=1 to n do
        if v[i].Indegree == 0 then
            v[i].group := 1;
            put v[i] in queue : Q;

    while Q is not empty do
        from := Q.front();
        Q.pop();

        for all edges (from, to) do
            to.indegree := to.indegree-1;
            if to.indegree == 0 then
                to.group := from.group+1;
                put to in Q;

end
```

3、给定连通无向图 G，以及 3 条边 a,b,c，在线性时间内判断 G 中是否存在一个包含 a 和 b但不含 c 的闭链

解：考虑若以c的一个端点为根节点，dfs遍历图G得到双连通树，考察a、b、c中节点的High值。若遍历时先访问到b，则说明a在b的子树中，此时若存在包含a和b但不含c的闭链，则需要满足的条件是：

$$b < High[a] < c$$

即a的后退边位于b和c之间，此时满足题目要求，该闭链存在。

同理可得到若先访问a的情况，判定条件改成 $a < High[b] < c$ 即可。

只需要对连通G遍历一次，故时间复杂度为$O(|E|)$

代码如下：

```
Algorithm:Chain
Input: G=(V, E)(undirected graph), a, b, c, v(the root), va(the vertex of a) ,
vb(the vertex of b) , n(|G|)
Output: flag (exists a required chain or not)

begin
    cnt := 0;
    dfs(v);
```

```
    if(High[va]>dfs[vb] && High[va]<dfs[v]) flag := true;
    else if(High[b]>dfs[va] && High[va]<dfs[v]) flag :=true;
    else flag := false;

    output flag;
end

function dfs(int u)
begin
    cnt := cnt+1;
    DFS[u] := cnt;
    High[u] := cnt;
    for every edge(u,v) do
       if(DFS[v] == 0) then
            dfs(v);
            High[u] := min(High[u],High[v]);
       else then
            High[u] := min(High[u],DFS[v]);
end
```

4、设计线性时间算法求树的最大匹配

解：对于一个节点v来说，v要么和某一个子节点选入匹配集合，要么不选入匹配，可以用动态规划分析这个问题，设 $f[v]$ 表示以v为根的子树，选取了v 和某一个儿子节点的边的最大匹配， $g[v]$ 表示v未选入匹配的最大匹配，则有：

$$g[v] = \sum max(g[son], f[son]);$$

对于v和某个儿子节点的边加入匹配的情况，只需枚举其余儿子节点不匹配+当前边匹配即可：

$$f[v] = max(g[v] - max(g[son], f[son]) + g[son] + 1);$$

任取一个节点做根，做dfs遍历整棵树，遍历的过程统计f和g即可，最后对根的f和g取max即所求。

代码如下：

```
Algorithm: Max_Match
Input: G=(V, E), root
Output: Max_Match
begin
    DFS(root);
    output max(f[root],g[root]);
end

function DFS(u)
begin
    f[node] := 0;
    g[node] :=0;
    for all edges (u, v) do
        DFS(v);
        g[u] := g[u]+max{f[v], g[v]};

    for all edges (u, v) do
```

```
        f[u]=max{f[u], g[u]-max(f[v], g[v])+g[v]+1}
end
```