HW5_singly connected problem

B10532027 四電資二 林科廷

程式摘要:

此程式由 C 語言所寫成，核心的觀念為 DFS，將所有的 node 做 travalsal,並對所有 node 做 DFS 並標記顏色(白=未 DFS,灰=DFS 未結束,黑=DFS 結束的 Node),並在做完 DFS 後,判斷是否有 EDGE 不是 forward edge 或是 cross edge,若有,則輸出 NO

```
1. #include<stdlib.h>
2. #include<stdio.h>
3.
4. bool ans = true;  //最後要輸出的答案
5.
6. struct node {  //NODE 的結構
7.     int color;
8.     node *parent;
9.     node **adjacent;
10.      int adjacent_cnt;
11.  };
12.
13.  void DFS(node *nodes, node* v) {  //遞迴做 DFS 並標記顏色
14.      v->color = 1;
15.      for (int i = 0; i < v->adjacent_cnt; i++) {
16.          if (v->adjacent[i]->color == 0) {
17.              v->adjacent[i]->parent = v;
18.              DFS(nodes, v->adjacent[i]);
19.          }
20.          else if (v->adjacent[i]->color==2) {  //判斷是否是 forward edge
21.              ans = false;
22.              return;
23.          }
24.      }
25.      v->color = 2;
```

```
}
int main() {


    int n;
    scanf("%d", &n);

    int m = 1;
    while (m<=n) {
        ans = true;
        int v;
        scanf("%d", &v);
        node *nodes = new node[v];

        for (int i = 0; i < v; i++) {
            nodes[i].adjacent = new node*[10];
        }

        int e;
        scanf("%d", &e);

        for (int i = 0; i < e; i++) {  //讀測資
            int from;
            int to;
            scanf("%d", &from);
            scanf("%d", &to);
            nodes[from].adjacent[nodes[from].adjacent_cnt] = &nodes[to];
            nodes[from].adjacent_cnt++;
        }

        for (int i = 0; i < v && ans; i++) {   //node travalsal
            for (int j = 0; j < v; j++) {  //初始化
                nodes[j].color = 0;
                nodes[j].parent = NULL;
            }
            DFS(nodes, &nodes[i]);
```

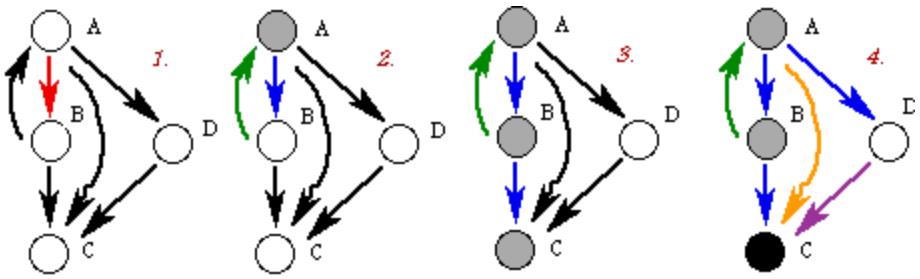
```
62.             }
63.
64.             if(ans)
65.                 printf("%d YES\n", m);
66.             else
67.                 printf("%d NO\n", m);
68.
69.             m++;
70.         }
71.
72.         return 0;
73.     }
```

Pseudo code:

```
//遞迴做 DFS 並標記顏色
void DFS(node *nodes, node* v) {
        v->color = 1;
        for (int i = 0; i < v->adjacent_cnt; i++) {
                if (v->adjacent[i]->color == 0) {
                        v->adjacent[i]->parent = v;
                        DFS(nodes, v->adjacent[i]);
                }
                else if (v->adjacent[i]->color==2) {    //判斷是否是 forward edge
                        ans = false;    //不允許 forward edge
                        return;
                }
        }
.       v->color = 2;
}

//node travalsal
for (int i = 0; i < v && ans; i++) {
    for (int j = 0; j < v; j++) {    //初始化
            nodes[j].color = 0;
            nodes[j].parent = NULL;
    }
    DFS(nodes, &nodes[i]);
}
```

圖解:



如圖,

黑色的 edge 是還沒使用過的 edge;

藍色的 edge 是還指向 child node 的 edge;

綠色的 edge 是還指向 parent node 的 edge;

橘色的 edge 是還指向 子代 node 的 edge;(simple path 中 不允許出現)