[알고리즘] 방향그래프

imhyun · 방금 전

c언어 백준 알고리즘

문제

https://www.acmicpc.net/problem/9466

> 이 문제는 모든 학생들이 프로젝트를 함께하고 싶은 학생을 선택한 후, 프로젝트 **팀에 속하지 못한 학생들의 수**를 구하는 문제이다. (이 때 단 한 명만을 선택하며, 자기 자신을 선택하는 것도 가능하다.)
> 팀이 만들어지는 규칙은 **자기자신을 선택하거나, 싸이클이 돌게끔 선택**하면 팀이 만들어진다. 따라서 이 문제는 **방향 그래프를 구현하고, 루프와 싸이클을 판단해 나머지 학생 수를 판단**하면 되는 문제이다.

아이디어

이 문제는 **방향 그래프를 구현하고, 루프와 싸이클을 판단해 나머지 학생 수를 판단**하면 되는 문제이다.

먼저 방향 그래프를 구현해보자. 지난번 인접 Matrix를 통해 구현한 방향 그래프에서 단순한 차이점은, 배열 구성시에 방향을 고려해 인덱스를 넣어주면 되는 것이다. 하지만 이번 문제에서는 간선이 오직 N개만 존재하기에 N^2의 배열을 사용하는 것은 과도한 공간 낭비이다. 따라서 인접리스트를 통해 구하기로 했다. 이때 인접리스트는 진출리스트와 진입리스트로 구분해야 하지만, 진출리스트의 경우 이미 순서대로 학생을 선택하기 때문에 진입리스트만 구현해보겠다.

또한, 자기 자신을 선택하거나, 싸이클이 돈다면 팀이 만들어진다. 이는 방향 그래프 내에서 자기자신을 가르키는 루프나 싸이클을 찾아내면 된다. Loop의 경우는 진입리스트에 속한 간선배열 인덱스가 자기 자신을 가르키는 것면 된다. 싸이클은 DFS를 통해서 찾아낼 수 있다.

코드

```
#include <stdio.h>
#include <stdlib.h>
#pragma warning (disable:4996)

typedef struct Edge {
    //연결된 정점
    int start;
    int end;

    //상태
    char state;
}Edge;

typedef struct Incid_Node {
    struct Incid_Node* next;
    int edge_idx;
}Incid_Node;

typedef struct Vertex {
    struct Incid_Node* In_list;
    int id;

    //상태
    char state;
}Vertex;

typedef struct Graph {
    Vertex* vertex;
    Edge* edge;
}Graph;

Incid_Node* makeIncidNode(int n) {
    Incid_Node* node = (Incid_Node*)malloc(sizeof(Incid_Node));

    node->next = NULL;
    node->edge_idx = n;

    return node;
}

void init_Vertex(Vertex* v,int n) {
    for (int i = 0; i < n; i++) {
        v[i].id = i + 1;
        v[i].In_list = makeIncidNode(-1);
    }
}
```

```
void init_Graph(Graph* graph,int n) {
    graph->vertex = (Vertex*)malloc(sizeof(Vertex) * n);
    graph->edge = (Edge*)malloc(sizeof(Edge) * n);

    init_Vertex(graph->vertex,n);
}

void add_Node(Vertex* vertex, int from, int to) {
    Incid_Node* p = vertex[to].In_list;

    Incid_Node* node = makeIncidNode(from);


    for (; p->next != NULL; p = p->next)
        ;

    node->next = p->next;
    p->next = node;

}

void modify_Edge(Graph* graph, int from, int to) {
    //================간선 리스트 변경=================
    graph->edge[from - 1].start = from - 1;
    graph->edge[from- 1].end = to - 1;

    //================인접 리스트 변경=================
    add_Node(graph->vertex,from-1,to-1);

}

//==============멤버수 파악==================
int opposite(Graph *graph, int idx) {
    return graph->edge[idx].end;
}

int rDFS(Graph* graph, int idx,int start,int sum) {
    graph->vertex[idx].state = 'V';//Visited라는 뜻으로, 방문했음을 의미.

    int w=opposite(graph, idx);
    int result = sum;

    if (graph->edge[idx].state == 'F') {
        if (graph->vertex[w].state == 'F') {
            graph->edge[idx].state = 'T';//방문한 간선

            result++;
            sum=rDFS(graph, w,start,result);
        }

        else
            graph->edge[idx].state = 'B';//Bac
    }

    if (start == w)
        return result;
    else
```

```
        return sum;
}

int DFS(Graph* graph,int N) {
    int result=0;
    for (int i = 0; i < N; i++)
        graph->vertex[i].state = 'F'; //방문하지 않은 상태

    for (int i = 0; i < N; i++)
        graph->edge[i].state = 'F'; //방문하지 않은 상태

    for (int i = 0; i < N; i++) {
        if (graph->vertex[i].state == 'F')
            result=rDFS(graph, i,i,result);
    }

    return result;
}

int isLoop(Graph *graph,int idx) {
    Incid_Node* p = graph->vertex[idx].In_list->next;

    for (; p != NULL; p = p->next) {
        if (p->edge_idx == idx) {
            return 1;
        }
    }

    return 0;
}

void printNonMem(Graph* graph,int N) {
    int member=0;

    //루프 판단
    for (int i = 0; i < N; i++) {
        member += isLoop(graph, i);
    }

    //싸이클 판단
    member += DFS(graph, N);

    printf("%d\n", N - member);
}


int main(void)
{

    //=================그래프 설정==================
    Graph* graph = (Graph*)malloc(sizeof(Graph));

    int T, N;
    int choice;

    scanf("%d", &T);

    for (int i = 0; i < T; i++) {
```

```
        scanf("%d", &N);
        init_Graph(graph,N);

        for (int j = 0; j < N; j++) {
            scanf("%d", &choice);
            modify_Edge(graph, j+ 1, choice);
        }

        printNonMem(graph,N);
    }

}```
```

박시현

이전 포스트
[알고리즘] 그래프 순회

0개의 댓글

댓글을 작성하세요

댓글 작성