목차

- 소요시간 : 1시간 미만
- 자료 참고 여부 : △ ('알고리즘 수업' 시 사용한 코드 참고)
- 체감 난이도 : 中

📌 5567번

- https://www.acmicpc.net/problem/5567

5567번: 결혼식

예제 1의 경우 2와 3은 상근이의 친구이다. 또, 3과 4는 친구이기 때문에, 4는 상근이의 친구의 친구이다. 5와 6은 친구도 아니고, 친구의 친구도 아니다. 따라서 2, 3, 4 3명의 친구를 결혼식에 초대

www.acmicpc.net

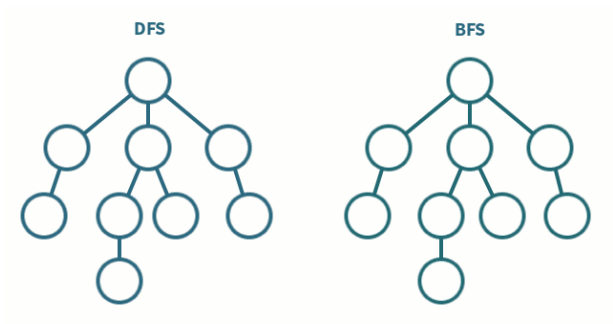
< 총평 >

- 오랜만에 초심자의 행운이 따랐다 .. 🍀

– 처음 문제를 이해하는데에 시간 조금 썼으나 문제 풀이에서는 빠르게 해결됐다 .. !

백준 | 5567번 결혼식

너비 우선 탐색 (BFS)



출처 : 나무위키 너비 우선 탐색 문서

- 루트 노드에서 시작해서 가까운 간선들을 먼저 방문하는 탐색법.
- 루트 노드와 바로 연결되어있는 간선의 탐색을 끝내면, 자식 노드 중 가장 먼저 입력된 노드의 자식 노드를 탐색한다. 이후 탐색 규칙 또한 자식 노드의 자식으로 가는 것이 아니라 자식 노드 중 두번째로 입력 받은 노드의 자식을 탐색한다.

📌코드 해석

```
typedef struct Graph {
    int** list;
    int  n;
}graph;


void  addedge(graph g ,int a, int b) {
    g.list[a][b] = 1;
    g.list[b][a] = 1;
}
```

- typedef struct Graph : 구조체를 구성하는 요소들을 포함하는 구조체
- void addedge : 그래프의 간선 정보가 주어졌을 때, 그래프 리스트 내에서 간선을 존재하게 만들어 주는 함수

```
void BFS(graph g, int s) {

    int size = g.n;

    int* q = (int*)malloc(sizeof(int) * size);
    int * visited = (int*) malloc(sizeof(int) * (size+1));

    int rear = 0, front = 0;

    // visited 초기화
    memset(visited, 0, sizeof(int) * (size + 1));


    visited[s] = 1;

    // printf("%d\n", s); // 시작점 세팅
    q[rear++] = s;
```

```
int friend = 0;

    // 상근이의 친구
    s = q[front++];
    for (int i = 0; i <= g.n; i++) {
            if (g.list[s][i] == 1 && !visited[i]) {
                visited[i] = 1;
                //printf("%d\n", i);
                q[rear++] = i;
                friend++;
            }
    }

    int f_rear = rear;

    for (int j = 1; j < f_rear; j ++) {
        s = q[j];
        for (int i = 0; i <= g.n; i++) {
            if (g.list[s][i] == 1 && !visited[i]) {
                visited[i] = 1;
                //printf("%d\n", i);
                q[rear++] = i;
                friend++;
            }
        }

    }

    printf("%d", friend);



}
```

- void BFS : 너비 우선 탐색을 하며 상근이의 친구, 친구의 친구 수를 세는 함수
  - 시작점 (상근이) 1부터 시작해서, 시작점과 바로 연결된 자식노드의 개수 먼저 센다.
  - 시작점과 연결된 자식의 노드 수는 rear 변수 안에 저장되게 되고, 앞선 for문이 끝난다면 f_rear라는 변수 안에 끝나는 순간의 rear 변수를 저장한다.
  - 친구의 친구, 즉 자식노드의 자식노드의 개수를 세기 위해 앞서 저장한 f_rear의 자식 노드 수 만큼 for문을 반복해준다. 친구의 친구를 세는 for문의 방법은 앞서 상근이의 친구를 셀 때 사용한 것과 같다.

```
int main() {

    graph g;

    int n, m, s;

    scanf("%d %d", &n, &m);
    s = 1;
    g.n = n;
    g.list = (int**)malloc(sizeof(int*) * (n + 1));

    for (int i = 0; i <= n; i++) {
        g.list[i] = (int*)malloc(sizeof(int) * (n + 1));
        memset(g.list[i], 0, sizeof(int) * (n + 1));
    }

    int a, b;
```

```
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &a, &b);
        addedge(g, a, b);
    }

    BFS(g, s);

}
```

- main 함수
  - 상근이의 동기 수(정점의 개수)와 리스트 길이(간선의 개수)를 입력 받은 뒤에 리스트를 초기화 해준다.
  - 상근이의 동기들 (간선의 정보)를 입력 받은 뒤, addedge 함수를 이용해서 리스트를 만든다.
  - BFS 함수에 그래프와 s(=1)을 넣어 함수를 실행시킨다.

📌 총평

- 수업에서 사용한 알고리즘에 조금의 변형을 주는거라 빠르게 풀렸다.

BFS, DFS 둘 다 작년 '이산수학 및 프로그래밍' 수업에서 배운 지식을 바탕으로 어렴풋하게 알고있었는데, 이 문제를 통해서 BFS에 대한 내용을 조금이나마 복기할 수 있었다.