2023 Algorithm Study 9Week

sookyoung0620 · 방금 전 · 비공개

그래프 구현, 순회

- DFS 깊이 우선 탐색
  - 간선을 따라 다음 정점 방문
  - 더 이상 탐색할 간선이 없으면 역추적하여 탐색하지 않은 간선이 있는지 확인
  - 탐색 가능한 간선이 있으면 다시 간선을 따라 다음 정점으로 이동
  - 모든 정점을 탐색할 때까지 반복
- BFS 너비 우선 탐색
  - 처음에 레벨 0에 있는 한 정점에서 시작
  - 먼저 레벨 1의 모든 정점 방문
  - 그다음 레벨2의 모든 정점 방문
  - 레벨을 늘리면서 모든 레벨에 있는 노드들을 방문할 때까지 반복

> **문제**
> 상근이는 자신의 결혼식에 학교 동기 중 자신의 친구와 친구의 친구를 초대하기로 했다. 상근이의 동기는 모두 N명이고, 이 학생들의 학번은 모두 1부터 N까지이다. 상근이의 학번은 1이다.
> 상근이는 동기들의 친구 관계를 모두 조사한 리스트를 가지고 있다. 이 리스트를 바탕으로 결혼식에 초대할 사람의 수를 구하는 프로그램을 작성하시오.

- 문제를 보고 BFS 너비 우선 탐색 알고리즘을 선택하는 것이 나을 것이라고 판단.

  1. 상근이를 레벨 0으로 생각

  2. 친구관계에서 a,b 둘 중 하나가 1일 때 나머지 하나를 레벨 1로 생각

  3. a,b 둘 중 하나가 레벨 1일 때 나머지 하나를 레벨 2로 생각

  4. 레벨이 1,2 일때 카운트해서 출력하기

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

```
#define MAX 501

int adj[MAX][MAX]; // 인접행렬
int level[MAX];    // 레벨 저장 배열

typedef struct Queue {
    int save[MAX];
    int front;
    int rear;
} Queue;

void bfs(int start, int n);

int main() {
    int n, m;
    scanf("%d", &n);
    scanf("%d", &m);


    for (int i = 0; i < m; i++) {
        int v1, v2;
        scanf("%d %d", &v1, &v2);
        adj[v1][v2] = 1;
        adj[v2][v1] = 1;
    }
    bfs(1, n);

    int count = 0;
    for (int i = 2; i <= n; i++) {
        if (level[i] == 1 || level[i] == 2) { // 레벨이 1 또는 2일 때
            count++;
        }
    }
    printf("%d\n", count);

    return 0;
}
```

```
void bfs(int start, int n) {
    int current, next;
    Queue q;
    int level[MAX] = { -1 };
    q.front = q.rear = -1;

    level[start] = 0; // 상근이의 레벨은 0

    q.save[++q.rear] = start;

    while (q.front < q.rear) {
        current = q.save[++q.front];

        for (int i = 1; i <= n; i++) {
            if (adj[current][i] == 1 && level[i] == -1) {
                level[i] = level[current] + 1; // 이전 노드의 레벨 + 1로 설정
                q.save[++q.rear] = i;
            }
        }
    }
}
```

- 시도 1 : 레벨이 제대로 저장되지 않음... current가 1일 때 나머지가 레벨 2가 되도록 저장해야겠음 (예시 출력은 잘 됨.. 하지만 레벨 1이 저장 안 되는 경우를 발견)

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

#define MAX 1001

int adj[MAX][MAX]; // 인접행렬
int level[MAX];    // 레벨 저장 배열

typedef struct Queue {
    int save[MAX];
    int front;
    int rear;
} Queue;

void bfs(int start, int n);

int main() {
    int n, m;

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

    for (int i = 1; i <= m; i++) {
        int v1, v2;
        scanf("%d %d", &v1, &v2);
        adj[v1][v2] = 1;
        adj[v2][v1] = 1;
    }
```

```
        bfs(1, n);


    return 0;
}

void bfs(int start, int n) {
    int current, next;
    Queue q;

    q.front = q.rear = -1;

    memset(level, -1, sizeof(level)); // 레벨 초기화
    level[start] = 0; // 상근이의 레벨은 0

    q.save[++q.rear] = start;

    while (q.front < q.rear) {
        current = q.save[++q.front];

        for (int i = 1; i <= n; i++) {
            if (adj[current][i] == 1 && level[i] == -1) {
                if (current == 1 || i == 1) {
                    level[i] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                }
                q.save[++q.rear] = i;
            }
        }
    }


    for (int i = 1; i <= n; i++) {
        if (level[i] == 1) { // 레벨이 1인 학생 출력
            printf("%d ", i);
        }
    }


}
```

시도 2. 레벨 1 저장은 잘 되는 거 같음!

```
> while (q.front < q.rear) {
        current = q.save[++q.front];
        for (int i = 1; i <= n; i++) {
            if (adj[current][i] == 1) {
                if (current == 1 && level[i] == -1) {
                    level[i] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                }
                else if (i == 1 &&level[current] == -1) {
                    level[current] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                }
                q.save[++q.rear] = i;
            }
```

```
            }
        }
```

시도 3. 런타임 에러 (Outofbounds)

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

#define MAX 501

int adj[MAX][MAX]; // 인접행렬
int level[MAX];    // 레벨 저장 배열

typedef struct Queue {
    int save[MAX];
    int front;
    int rear;
} Queue;

void bfs(int start, int n);

int main() {
    int n, m;

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

    for (int i = 1; i <= m; i++) {
        int v1, v2;
        scanf("%d %d", &v1, &v2);
        adj[v1][v2] = 1;
        adj[v2][v1] = 1;
    }

    bfs(1, n);
    return 0;
}

void bfs(int start, int n) {
    int current, next;
    int count = 0;
    Queue q;

    q.front = q.rear = -1;

    int level[MAX] = { -1 };
    level[start] = 0; // 상근이의 레벨은 0

    q.save[++q.rear] = start;

    while (q.front < q.rear) {
        current = q.save[++q.front];

        for (int i = 1; i <= n; i++) {
```

```
            if (adj[current][i] == 1) {
                if (current == 1 && level[i] == -1) {
                    level[i] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                }
                else if (i == 1 &&level[current] == -1) {
                    level[current] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                }
                if (current != 1 && i != 1) {//레벨1과 연결된 것 2로 저장
                    if (level[current] == 1 && level[i] == -1) {
                        level[i] = 2;
                    }
                    else if (level[i] == 1 && level[current] == -1) {
                        level[current] = 2;
                    }
                }

                q.save[++q.rear] = i;
            }
        }
    }


    for (int i = 1; i <= n; i++) {
        if (level[i] == 1||level[i] == 2) { // 레벨이 1인 학생 출력
            count++;
        }
    }
    printf("%d", count);


}
```

정답

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

#define MAX 501

int adj[MAX][MAX]; // 인접행렬
int level[MAX];    // 레벨 저장 배열

typedef struct Queue {
    int save[MAX];
    int front;
    int rear;
} Queue;

void bfs(int start, int n);

int main() {
    int n, m;
```

```
    scanf("%d %d", &n, &m);



    for (int i = 0; i < m; i++) {
        int v1, v2;
        scanf("%d %d", &v1, &v2);
        adj[v1][v2] = 1;
        adj[v2][v1] = 1;
    }

    bfs(1, n);
    return 0;
}

void bfs(int start, int n) {
    int current, next;
    int count = 0;
    Queue q;

    q.front = q.rear = -1;


    level[start] = 0; // 상근이의 레벨은 0

    q.save[++q.rear] = start;

    while (q.front < q.rear) {
        current = q.save[++q.front];

        for (int i = 1; i <= n; i++) {
            if (adj[current][i] == 1) {
                if (current == 1 && level[i] == NULL) {
                    level[i] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                    q.save[++q.rear] = i;
                }
                else if (i == 1 && level[current] == NULL) {
                    level[current] = 1; // 상근이와 직접적으로 연결된 경우 레벨 1로 설정
                    q.save[++q.rear] = current;
                }
                else if (current != 1 && i != 1) {
                    if (level[current] == 1 && level[i] == NULL) {
                        level[i] = 2;
                        q.save[++q.rear] = i;
                    }
                    else if (level[i] == 1 && level[current] == NULL) {
                        level[current] = 2;
                        q.save[++q.rear] = current;
                    }
                }
            }
        }
    }

    for (int i = 1; i <= n; i++) {
        if (level[i] == 1 || level[i] == 2) { // 레벨이 1이나 2인 학생 카운트
            count++;
```

```
            }
        }
        printf("%d", count);
    }
```

윤수경

이전 포스트
2023 Algorithm Study 8Week

0개의 댓글

댓글을 작성하세요

댓글 작성