[알고리즘] 그래프 순회

imhyun · 방금 전 · 비공개

백준 알고리즘

문제

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

> 첫째 줄의 동기의 수를 정점으로, 두번째 줄에서 주어진 리스트를 통해 간선들로 이어나간다. 너비우선탐색인 BFS를 이용해서 정점 1을 기준으로 친구와 친구의 친구 즉 Level이 2인 지점까지 찾아 최종 동기의 수를 출력하면 된다.

이론

그래프 구현

간선리스트 구조

- 정점 리스트와 간선 리스트의 구조
  - 정점 리스트 : 정점 노드들에 대한 포인터의 리스트
  - 간선 리스트 : 간선 노드들에 대한 포인터의 리스트
- 노드
  - 정점 노드 : 원소
  - 간선 노드 : 원소, 시점, 종점 노드로 구성

인접리스트 구조

- 간선리스트 구조 + 부착 리스트
    - 부착리스트 : 각 정점들의 부착 간선들을 표시

그래프 순회 (BFS)

필요한 것들

- 큐 : 방문 차례의 기록을 목적으로 이용.
- 배열 : 방문 정보의 기록을 목적으로 이용. **즉 연결된 곳을 큐에 저장한 후, 차례로 dequeue하며 너비를 기준으로 접근한다.**

코드

첫 시도 (인접리스트)

첫 시도는 배열을 이용한 인접리스트로 접근하였다. 그래프를 구현한 후, adjacentvertices를 이용해 1번의 친구들을 구한 후, 각 친구들에게서 1번이 아니며 1번의 친구가 아닌 애들을 더하며 구하고자 하였다. 하지만, adjacentvertices를 구하며 판단하며 더하는 과정이 복잡해 너비우선탐색 BFS를 활용하기로 하였다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#pragma warning (disable:4996)

typedef struct edge {
    int str; //정점의 인덱스
    int fin;
}edge;

typedef struct incid_node {
    int idx_E;
    struct incid_node* next;
}incid_node;

typedef struct vertex {
    incid_node* incid;
}vertex;

incid_node* makeVNode();

void makeV(vertex* f, int n);

void makeEdges(edge *,int,int,int);

void addLast(vertex* v, int i);
```

```
void makeIncidence(vertex* vertices, int v1, int v2, int i);

int main(void)
{
    //===================동기 수와 정점 형성=========================
    int n;

    scanf("%d", &n);

    //정점 배열 형성
    vertex* vertices = (vertex*)malloc(sizeof(vertex)*n);

    //vertex 구성
    makeV(vertices, n);

    //=====================리스트의 길이==================
    int m;
    int v1, v2;

    scanf("%d", &m);

    //간선 배열 형성
    edge* edges = (edge*)malloc(sizeof(edge) * m);

    for (int i = 0; i < m; i++) {
        scanf("%d %d", &v1, &v2);
        makeEdges(edges,v1,v2,i);
        makeIncidence(vertices, v1, v2, i);
    }

    //===================초대하는 동기수=============================
    printnum(vertices);
}

incid_node* makeVNode() {
    incid_node* node = (incid_node*)malloc(sizeof(incid_node));

    node->idx_E = -1;
    node->next = NULL;

    return node;
}

void makeV(vertex * vertices,int n) {

    for (int i = 0; i < n; i++) {
        vertices[i].incid = makeVNode();
    }
}

void makeEdges(edge * edges, int v1, int v2,int i) {
    //정점 연결 설정 (배열의 인덱스와 통일하기 위해 -1)
    edges[i].str = v1-1;
    edges[i].fin = v2-1;

}
```

```
void addLast(vertex *v, int i) {
    incid_node* p = v->incid;

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

    p->next = makeVNode();
    p->next->idx_E = i;
}

void makeIncidence(vertex * vertices, int v1, int v2, int i) {
    addLast(&vertices[v1-1], i);
    addLast(&vertices[v2-1], i);
}

int listlen(vertex v) {
    incid_node* p = v.incid->next;

    int total = 0;

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

    return total;
}

void printnum(vertex* vertices) {
    int total = 0;

    //============친구===========
    total += listlen(vertices[0]);

    //==========친구의 친구==========


    printf("%d\n", total);
}
```

두 번째 시도 (BFS)

BFS 코드에서 친구의 친구까지만 저장하고 count하는 함수로 변형했다. 친구의 친구를 판단하는 기준은 type 변수를 사용했다.type는 처음 상근이의 친구 최종 인덱스이며, dequeue할 때 사용되는 front가 type와 같다면 친구의 친구까지 count할 수 있다고 판단했다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#pragma warning (disable:4996)

//==========사용된 자료구조===========
#define MAX_SIZE 500
```

```
typedef struct { // 큐 타입
    int queue[MAX_SIZE];
    int  front, rear;
}QueueType;

typedef struct GraphType {
    int n; //정점 수
    int adj_matrix[MAX_SIZE][MAX_SIZE];
}GraphType;

int visited[MAX_SIZE];

int cnt = 0;
//===============큐 ADT===================
void init_Queue(QueueType* q)
{
    q->front = q->rear = 0;
}

void enqueue(QueueType* q, int item)
{
    q->rear++;
    q->queue[q->rear] = item;
}

int dequeue(QueueType* q)
{
    q->front++;
    return q->queue[q->front];
}

//==============인접행렬======================

void init_Graph(GraphType* g) {
    int i, j;
    g->n;
    for (i = 0; i < MAX_SIZE; i++)
        for (j = 0; j < MAX_SIZE; j++)
            g->adj_matrix[i][j] = 0; //인접행렬
}

void insert_vertex(GraphType* g, int v) {
    if (((g->n) + 1) > MAX_SIZE) {
        return;
    }
    g->n++;
}

void insert_edge(GraphType* g, int start, int end) {
    g->adj_matrix[start][end] = 1;
    g->adj_matrix[end][start] = 1;
}

//=============그래프 순회================
void BFS(GraphType* g, int v) {
    int w;

    //방문 차례 표현하기 위한 큐 구현
```

```
        QueueType q;
        init_Queue(&q);
        enqueue(&q, v);

        //방문 여부 기록하기 위한 배열
        visited[v] = 1;


        while (q.front!=q.rear) {
            v = dequeue(&q);        // 큐에 정점 추출

            for (w = 0; w < g->n; w++)  // 인접 정점 탐색
                if (g->adj_matrix[v][w] && !visited[w]) { //연결되어 있고 방문한 적이 없다면.
                    visited[w] = 1;    // 방문 표시
                    enqueue(&q, w);     // 방문한 정점을 큐에 저장
                }
        }
    }


    void printNum(GraphType* g) {
        int w;

        //방문 차례 표현하기 위한 큐 구현 - 우선 상근이부터 접근
        QueueType q;
        init_Queue(&q);
        enqueue(&q, 0);

        //방문 여부 기록하기 위한 배열 - 우선 상근이부터 접근
        visited[0] = 1;

        int v;
        int type = -1; //친구의 친구까지 표현하기 위해

        while ((q.front != q.rear)&&(type!=q.front)) { //친구의 친구까지만
            v = dequeue(&q);        // 큐에 정점 추출

            for (w = 0; w < g->n; w++)  // 인접 정점 탐색
                if (g->adj_matrix[v][w] && !visited[w]) { //연결되어 있고 방문한 적이 없다면.
                    visited[w] = 1;    // 방문 표시
                    enqueue(&q, w); // 방문한 정점을 큐에 저장
                    cnt++;
                }

            if (type == -1)
                type = q.rear;

        }
    }

    int main(void)
    {
        GraphType* g;
        g = (GraphType*)malloc(sizeof(GraphType));

        //정점의 수
        int N;
        scanf("%d", &N);
```

```
        g->n = N;

        //간선을 만들어줄 리스트
        int M;
        scanf("%d", &M);

        int v1, v2;

        for (int i = 0; i < M; i++) {
            scanf("%d %d", &v1, &v2);
            insert_edge(g, v1, v2);
        }

        printNum(g);
        printf("%d\n", cnt);
        free(g);
    }
```

박시현

이전 포스트
[백준]탐색트리

0개의 댓글

댓글을 작성하세요

댓글 작성