/\*

240515 깊이우선탐색, 너비우선탐색

\*/

# 깊이 우선 탐색, 인접 행렬 구현(깊이는 스택을 활용한다)

#include <stdio.h>

#include <stdlib.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

#define TRUE 1

#define FALSE 0

#define MAX\_VERTICES 50

typedef struct GraphType {

int n;// 정점의 개수

int adj\_mat[MAX\_VERTICES][MAX\_VERTICES];

} GraphType;

int visited[MAX\_VERTICES];

// 그래프 초기화

void init(GraphType\* g)

{

int r, c;

g->n = 0;

for (r = 0; r < MAX\_VERTICES; r++)

for (c = 0; c < MAX\_VERTICES; c++)

g->adj\_mat[r][c] = 0;

}

// 정점 삽입 연산

void insert\_vertex(GraphType\* g, int v)

{

if (((g->n) + 1) > MAX\_VERTICES) {

fprintf(stderr, "그래프: 정점의 개수 초과");

return;

}

g->n++;

}

// 간선 삽입 연산

void insert\_edge(GraphType\* g, int start, int end)

{

if (start >= g->n || end >= g->n) {

fprintf(stderr, "그래프: 정점 번호 오류");

return;

}

g->adj\_mat[start][end] = 1;

g->adj\_mat[end][start] = 1;

}

// 인접 행렬로 표현된 그래프에 대한 깊이 우선 탐색

void dfs\_mat(GraphType\* g, int v)

{

int w;

visited[v] = TRUE;// 정점 v의 방문 표시

printf("정점 %d -> ", v);// 방문한 정점 출력

for (w = 0; w < g->n; w++) // 인접 정점 탐색

if (g->adj\_mat[v][w] && !visited[w]) dfs\_mat(g, w); //정점 w에서 DFS 새로 시작

}

int main(void)

{

GraphType\* g;

g = (GraphType\*)malloc(sizeof(GraphType));

init(g);

for (int i = 0; i < 4; i++) insert\_vertex(g, i);

insert\_edge(g, 0, 1);

insert\_edge(g, 0, 2);

insert\_edge(g, 0, 3);

insert\_edge(g, 1, 2);

insert\_edge(g, 2, 3);

printf("깊이 우선 탐색\n");

dfs\_mat(g, 0);

printf("\n");

free(g);

return 0;

}

깊이 우선 탐색, 인접 리스트 구현

#include <stdio.h>

#include <stdlib.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

#define TRUE 1

#define FAlSE 0

#define MAX\_VERTICES 50

typedef struct GraphNode {

int vertex;

struct GraphNode\* link;

}GraphNode;

typedef struct GraphType {

int n;

GraphNode\* adj\_list[MAX\_VERTICES];

}GraphType;

int visited[MAX\_VERTICES];

void init(GraphType\* g) { // 그래프 초기화

int r;

g->n = 0;

for (r = 0; r < MAX\_VERTICES; r++) {

g->adj\_list[r] = NULL; // 전부 NULL로 초기화

}

}

//정점 삽입

void insert\_vertex(GraphType\* g, int v) {

if (((g->n) + 1) > MAX\_VERTICES) {

fprintf(stderr, "그래프: 정점의 개수 초과");

return;

}

g->n++;

}

//간선 삽입

void insert\_edge(GraphType\* g, int u, int v) {

GraphNode\* node;

if (u >= g->n || v >= g->n) {

fprintf(stderr, "그래프: 정점 번호 오류");

return;

}

node = (GraphNode\*)malloc(sizeof(GraphNode));

node->vertex = v;

node->link = g->adj\_list[u];

g->adj\_list[u] = node;

}

//인접 행렬로 표현된 그래프에 대한 깊이 탐색 그래프

void dfs\_list(GraphType\* g, int v) {

GraphNode\* w;

visited[v] = TRUE;

printf("정점 %d -> ", v);

for (w = g->adj\_list[v]; w; w = w->link) {

if (!visited[w->vertex]) //만약 w->vertex에 방문하지 않았다면,

dfs\_list(g, w->vertex); // 재귀함수로 한번 더 방문 프로세스를 진행(방문할 때까지 무한반복)

}

}

int main()

{

GraphType\* g;

g = (GraphType\*)malloc(sizeof(GraphType));

init(g);

for (int i = 0; i < 4; i++) insert\_vertex(g, i); // 정점 4개 생성

insert\_edge(g, 0, 1); // 0과 1 사이 간선 연결

insert\_edge(g, 0, 2); // 0과 2 사이 간선 연결

insert\_edge(g, 0, 3); // 0과 3 사이 간선 연결

insert\_edge(g, 1, 2); // 1과 2 사이 간선 연결

insert\_edge(g, 2, 3); // 2과 3 사이 간선 연결

printf("깊이 우선 탐색\n");

dfs\_list(g, 0);

printf("\n");

free(g); // 그래프 메모리 정리

}

너비 우선 탐색.

#include <stdio.h>

#include <stdlib.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

#define TRUE 1

#define FALSE 0

#define MAX\_QUEUE\_SIZE 10

typedef int element;

typedef struct { // 큐 타입

element queue[MAX\_QUEUE\_SIZE];

int front, rear;

} QueueType;

// 오류 함수

void error(const char\* message)

{

fprintf(stderr, "%s\n", message);

exit(1);

}

// 공백 상태 검출 함수

void queue\_init(QueueType\* q)

{

q->front = q->rear = 0;

}

// 공백 상태 검출 함수

int is\_empty(QueueType\* q)

{

return (q->front == q->rear);

}

// 포화 상태 검출 함수

int is\_full(QueueType\* q)

{

return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);

}

// 삽입 함수

void enqueue(QueueType\* q, element item)

{

if (is\_full(q))

error("큐가 포화상태입니다");

q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;

q->queue[q->rear] = item;

}

// 삭제 함수

element dequeue(QueueType\* q)

{

if (is\_empty(q))

error("큐가 공백상태입니다");

q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;

return q->queue[q->front];

}

#define MAX\_VERTICES 50

typedef struct GraphType {

int n; // 정점의 개수

int adj\_mat[MAX\_VERTICES][MAX\_VERTICES];

} GraphType;

int visited[MAX\_VERTICES];

// 그래프 초기화

void graph\_init(GraphType\* g)

{

int r, c;

g->n = 0;

for (r = 0; r < MAX\_VERTICES; r++)

for (c = 0; c < MAX\_VERTICES; c++)

g->adj\_mat[r][c] = 0;

}

// 정점 삽입 연산

void insert\_vertex(GraphType\* g, int v)

{

if (((g->n) + 1) > MAX\_VERTICES) {

fprintf(stderr, "그래프: 정점의 개수 초과");

return;

}

g->n++;

}

// 간선 삽입 연산

void insert\_edge(GraphType\* g, int start, int end)

{

if (start >= g->n || end >= g->n) {

fprintf(stderr, "그래프: 정점 번호 오류");

return;

}

g->adj\_mat[start][end] = 1;

g->adj\_mat[end][start] = 1;

}

void bfs\_mat(GraphType\* g, int v)

{

int w;

QueueType q;

queue\_init(&q); // 큐 초기화

visited[v] = TRUE; // 정점 v 방문 표시

printf("%d 방문 -> ", v);

enqueue(&q, v); // 시작 정점을 큐에 저장

while (!is\_empty(&q)) {

v = dequeue(&q); // 큐에 정점 추출

for (w = 0; w < g->n; w++) // 인접 정점 탐색

if (g->adj\_mat[v][w] && !visited[w]) {

visited[w] = TRUE; // 방문 표시

printf("%d 방문 -> ", w);

enqueue(&q, w); // 방문한 정점을 큐에 저장

}

}

}

int main(void)

{

GraphType\* g;

g = (GraphType\*)malloc(sizeof(GraphType));

graph\_init(g);

for (int i = 0; i < 6; i++)

insert\_vertex(g, i);

insert\_edge(g, 0, 2);

insert\_edge(g, 2, 1);

insert\_edge(g, 2, 3);

insert\_edge(g, 0, 4);

insert\_edge(g, 4, 5);

insert\_edge(g, 1, 5);

printf("너비 우선 탐색\n");

bfs\_mat(g, 0);

printf("\n");

free(g);

return 0;

}

너비 우선 탐색 리스트 구현(너비탐색은 큐를 이용한다)

#include <stdio.h>

#include <stdlib.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

#define TRUE 1

#define FALSE 0

#define MAX\_QUEUE\_SIZE 10

typedef int element;

typedef struct {

element queue[MAX\_QUEUE\_SIZE];

int front, rear;

}QueueType;

//오류 함수

void error(const char\* message) {

fprintf(stderr, "%s\n", message);

exit(1);

}

//초기화 함수

void queue\_init(QueueType\* q) {

q->front = q->rear = 0;

}

//공백 상태 검출 함수

int is\_empty(QueueType\* q) {

return (q->front == q->rear);

}

//포화 상태 검출 함수

int is\_full(QueueType\* q) {

return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);

}

//삽입 함수

void enqueue(QueueType\* q, element item) {

if (is\_full(q))

error("큐가 포화상태입니다\n");

q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;

q->queue[q->rear] = item;

}

//삭제 함수

element dequeue(QueueType\* q) {

if (is\_empty(q))

error("큐가 공백상태입니다\n");

q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;

return q->queue[q->front];

}

#define MAX\_VERTICES 50

typedef struct GraphNode {

int vertex;

struct GraphNode\* link;

}GraphNode;

typedef struct GraphType {

int n;

GraphNode\* adj\_list[MAX\_VERTICES];

}GraphType;

void graph\_init(GraphType\* g) {

int v;

g->n = 0;

for (v = 0; v < MAX\_VERTICES; v++) {

g->adj\_list[v] = NULL;

}

}

void insert\_vertex(GraphType\* g, int v) {

if (((g->n) + 1) > MAX\_VERTICES) {

fprintf(stderr, "그래프: 정점의 개수 초과");

return;

}

g->n++;

}

void insert\_edge(GraphType\* g, int u, int v) {

GraphNode\* node;

if (u >= g->n || v >= g->n) {

fprintf(stderr, "그래프: 정점 번호 오류");

return;

}

node = (GraphNode\*)malloc(sizeof(GraphNode));

node->vertex = v;

node->link = g->adj\_list[u];

g->adj\_list[u] = node;

}

int visited[MAX\_VERTICES];

void bfs\_list(GraphType\* g, int v) {

GraphNode\* w;

QueueType q;

queue\_init(&q);//큐 초기화

visited[v] = TRUE;//정점 v 방문 표시

printf("%d 방문 -> ", v);

enqueue(&q, v);//시작정점을 큐에 저장

while (!is\_empty(&q)) {

v = dequeue(&q);//큐에 저장된 정점 선택

for (w = g->adj\_list[v]; w; w = w->link) {//인점 정점 탐색

if (!visited[w->vertex]) {//미방문 정점 탐색

visited[w->vertex] = TRUE;//방문 표시

printf("%d 방문 -> ", w->vertex);

enqueue(&q, w->vertex);//정점을 큐에 삽입

}

}

}

}

int main(void) {

GraphType\* g;

g = (GraphType\*)malloc(sizeof(GraphType));

graph\_init(g);

for (int i = 0; i < 6; i++) {

insert\_vertex(g, i);

}

insert\_edge(g, 0, 2);

insert\_edge(g, 2, 1);

insert\_edge(g, 2, 3);

insert\_edge(g, 0, 4);

insert\_edge(g, 4, 5);

insert\_edge(g, 1, 5);

printf("너비 우선 탐색\n");

bfs\_list(g, 0);

printf("\n");

free(g);

}