빅데이터전공_20175328_이정현 DFS실습과제

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VERTEX 50
#define MAX_QUEUE MAX_VERTEX+1

int check[MAX_VERTEX];
typedef struct LIST {
    int test;
    struct LIST* link;
}List;//리스트생성
typedef struct GRAPH {
    int n;
    List* adj_list[MAX_VERTEX];
}Graph; //그래프 생성
Graph* Init_Graph(void);
void Vertex_Add(Graph* g);// 정점 생성 코드 생성
void Edge_Add(Graph* g, int a, int b);// 간선 추가 코드 생성
void Vertex_Delete(Graph* g, int a); // 정정 삭제 코드 생성
void Edge_Delete(Graph* g, int a, int b);// 그래프간 간선 삭제 코드 생성
void DFS(Graph *g, int n);// DFS 생성
void DFS_mat(Graph* g, int n);//DFS_mat 생성
void Queue_Add(int n); //큐 생성
int Queue_Delete(void); //그래프 동장 구현
int front = 0;
int rear = 0;
```

```
int queue[MAX_QUEUE]; //초기상태 설정
Graph* Init_Graph(void){
	Graph* g = (Graph*)malloc(sizeof(Graph));
	g->n = 0;
	for(int i = 0; i < MAX_VERTEX; i++)
		g->adj_list[i] = NULL;
	return g;
} //그래프의 사이즈 측정 후 null로 초기화 후 데이터 삽입
void Vertex_Add(Graph* g){
	g->adj_list[g->n] = (List*)malloc(sizeof(List));
	g->adj_list[g->n]->test = -1;
	g->adj_list[g->n]->link = NULL;
	g->n++;
}//정점 추가 코드
void Edge_Add(Graph* g, int a, int b){
	if(a >= g->n || b >= g->n){
		printf("범위 밖 입니다.₩n");
	}
	else{
		List* temp_a = (List*)malloc(sizeof(List)); //temp_a의 사이즈 삽입
		List* temp_b = (List*)malloc(sizeof(List)); //temp_b의 사이즈 삽입

		temp_a->test = a;
		temp_b->test = b;

		temp_b->link = g->adj_list[a]->link;
```

```
        g->adj_list[a]->link = temp_b;


        temp_a->link = g->adj_list[b]->link;
        g->adj_list[b]->link = temp_a;
    }
} //간선 추가 코드
void Vertex_Delete(Graph* g, int a){
    if(a >= g->n){
        printf("범위 밖 입니다.₩n");
    }
    else{
        List* circle;
        List* Rectangle;
        List* temp;
        for(int i = 0; i < g->n && i != a; i++){
            Rectangle = g->adj_list[i];
            circle = Rectangle->link;
            while(circle != NULL){
                temp = circle->link;
                if(circle->test == a){
                    Rectangle->link = circle->link;
                    free(circle);
                    break;
                }
                Rectangle = circle;
                circle = temp;
```

```
            }


        }
        Rectangle = g->adj_list[a];
        circle = g->adj_list[a]->link;
        while(circle != NULL){
            temp = circle->link;
            free(circle);
            circle = temp;
        }
        Rectangle->link = NULL;
    }
}//정점 삭제 코드
void Edge_Delete(Graph* g, int a, int b){
    if(a >= g->n){
        printf("범위 밖 입니다.₩n");
    }
    else{
        List* circle;
        List* Rectangle;
        List* temp;
        Rectangle = g->adj_list[a];
        circle = g->adj_list[a]->link;
        while(circle != NULL){
            temp = circle->link;
```

```
            if(circle->test == b){
                Rectangle->link = circle->link;
                free(circle);
                break;
            }
            Rectangle = circle;
            circle = temp;
        }
        Rectangle = g->adj_list[b];
        circle = g->adj_list[b]->link;
        while(circle != NULL){
            temp = circle->link;
            if(circle->test == a){
                Rectangle->link = circle->link;
                free(circle);
                break;
            }
            Rectangle = circle;
            circle = temp;
        }
    }
}//간선 삭제 코드
void DFS(Graph *g, int n){
    for(int i = 0; i < MAX_VERTEX; i++){
        check[i] = 0;
```

```
        }
        DFS_mat(g, n);
        printf("₩n");
} //DFS 그래프 DFS_mat을 받아 출력
void DFS_mat(Graph* g, int n){
        if(check[n] == 0){
                List* circle;
                List* temp;
                circle =    g->adj_list[n]->link;
                printf("Node %1d ", n);
                check[n] = 1;
                while(circle != NULL){
                        temp = circle->link;
                        if(check[circle->test] == 0)
                                DFS_mat(g, circle->test);
                        circle = temp;
                }
        }
}//DFS 구현부
void Queue_Add(int n){
        if((rear+1)%MAX_QUEUE == front){
                puts("Queue is full.");
        }
        else{
                rear = (rear+1)%MAX_QUEUE;
                queue[rear] = n;
```

```
    }

}//포화상태가 아니면 rear를 1증가 시키고 그 곳에 데이터를 삽입
int Queue_Delete(void){

    int temp = -1;
    if(front == rear){
        puts("Queue is empty.");
    }

    else{
        front = (front+1)%MAX_QUEUE;
        temp = queue[front];
    }
    return temp;
}//공백상태가 아니면 front를 1증가 시키고 그 곳에 데이터 삭제
int main(void) {

    Graph* g = Init_Graph();



    for(int i = 0; i < 10; i++){

        Vertex_Add(g);
```

```
}



    Edge_Add(g, 0, 1);

    Edge_Add(g, 1, 2);

    Edge_Add(g, 1, 3);

    Edge_Add(g, 3, 4);

    Edge_Add(g, 3, 5);

    Edge_Add(g, 5, 6);

    Edge_Add(g, 5, 7);

    Edge_Add(g, 7, 8);

    Edge_Add(g, 7, 9);

    printf("깊이 우선 탐색(DFS : Depth=first serach) 결과입니다.₩n");

    DFS(g, 0);

    return 0;


}//DFS 결과출력
```

실행결과 :