11주차 과제

11단원 프로그래밍 프로젝트 1

20781015
이주민

계획 -->

(1)

함수 : 정점의 수 + 간선의 수 (입력) -> 무작위 그래프 발생

생선된 그래프를 화면에 출력 -> 파일에 저장

정점의 이름은 'A'부터 시작하여 하나씩 증가

void randomGraph (int numVtx, int numEdge);

---

(2)

(1)에서 만든 그래프의 연결 성분 탐색을 통해,
연결된 성분 개수 구하기

---

(3)

(1)을 보완하여, 모든 정점이 연결된 '연결 그래프'를 발생시키는 함수 구현.

(2)와 같이 연결 성분 탐색 이용해서, 연결된 성분 개수가 1되는지 확인.

---

(1)+(2)

main()함수(11.3) :
정점의 수와 간성의 수 입력해서 무작위 생성 함수로 보냄.
그래프 출력해서 보여줌
연결 성분 테스트 그래프로 보냄
연결 성분 결과 보여줌

[AdjMatGraph.h] (11.1) :
인접행렬을 이용한 그래프 클래스 프로그램에
무작위 생성 함수를 추가해줌
(11.3) : 파일 입출력 함수 추가
----------(1)
[SrchAMGraph.h] :
(11.11) (11.13)
----------(2)

<main>

```
#include "ConnectedComponentGraph.h"
void main() {
	ConnectedComponentGraph g;

	int numVtx, numEdge;

	printf("정점의 개수 : ");
	cin >> numVtx;
	printf("간선의 개수 : ");
	cin >> numEdge;

	g.randomGraph(numVtx, numEdge);
	printf("그래프 출력\n");
	g.display();
	g.store("graph.txt");
	printf("\n");
	printf("연결 성분 테스트 그래프\n");
	g.resetVisited();
	g.findConnectedComponent();

	return 0;
}
```

<AdjMatGraph.h>

```
#include <cstdio>
#define MAX_VTXS        256 //표현 가능한 최대 정점의 개수

class AdjMatGraph {
protected :
        int size; //정점의 개수
        char vertices[MAX_VTXS]; //정점의 이름
        int adj[MAX_VTXS]; //인접 행렬

public :
        AdjMatGraph() { reset(); }
        char getVertex(int i) { return vertices[i]; }
        int getEdge(int i, int j) { return adj[i][j]; }
        void setEdge(int i, int j, int val) { adj[i][j] = val; }

        bool isEmpty() { return size == 0; }
        bool isFull() { return size >= MAX_VTXS; }

        //그래프 초기화 ==> 공백 상태의 그래프
        void reset() {
                size = 0;
                for (int i = 0; i < MAX_VTXS; i++)
                        for (int j = 0; j < MAX_VTXS; j++)
                                setEdge(i, j, 0);
        }

        //정점 삽입 연산
        void insertVertex(char name) {
                if (!isFull()) vertices[size++] = name;
                else printf("Error : 그래프 정점 개수 초과 \n");
        }

        //간선 삽입 연상 : 무방향 그래프의 경우
        void insertEdge(int u, int v) {
                setEdge(u, v, 1);
                setEdge(v, u, 1); //방향 그래프에서는 삭제 (<u,v>만 존재)
        }

        //무작위 그래프
        void randomGraph(int numVtx, int numEdge) {
                srand(time(0));
                int u, v;
```

```
		for (int ii = 0; i < numEdge; i++) {
			u = rand() % numVtx;
			v = rand() % numVtx;
			if (getEdge(u, v) || u == v) {
				i--;
				continue;
			}
		}
		insertEdge(u, v);
}

//그래프 정보 출력
void display(FILE* fp = stdout) {
		fprintf(fp, "%d\n", size); //정점의 개수 출력
		for (int i = 0; i < size; i++) { //각 행의 정보 출력
			fprintf(fp, "%d\n", size); //정점의 이름 출력
			for (int j = 0; j < size; j++) //간선 정보 추력
				fprintf(fp, " 3%d", getEdge(i, j));
			fprintf(fp, "\n");
		}
}

//파일 입력 함수
void load(char* filename) {
		FILE* fp = fopen(filename, "r");
		if (fp != NULL) {
			int n;
			fscanf(fp, "%d", &n); //정점의 전체 개수
			for (int i = 0; i < n; i++) {
				char str[80];
				fscanf(fp, "%s", str); //정점의 이름
				insertVertex(str[0]); //정점 삽입

				for (int j = 0; j < n; j++) {
					int val;
					fscanf(fp, "%d", &val); //간선 정보
					if (val != 0) //간선이 있으면
						insertEdge(i,j) //간선 삽입
				}
			}
			fclose(fp);
		}
}
```

```
        //파일 저장 함수
        void store(char* filename) {
                FILE* fp = fopen(filename, "w");
                if (fp != NULL) {
                        display(fp);
                        fclose(fp);
                }
        }
};
```

<SrchAMGraph.h>

```
#include "AdListGraph.h"
#include "CircularQueue.h"

class SrchAMGraph : public AdjMatGraph {
private :
	bool visited[MAX_VTXS]; //정점 방문 정보

public :
	void resetVisited() { //모든 정점을 방문하지 않았다고 설정
		for (int i = 0; i < size; i++)
			visited[i] = false;
	}
	bool isLinked(int u, int v) { return getEdge(u, v) != 0; }

	//깊이 우선 탐색 함수
	void DFS(int v) {
		visited[v] = true; //현재 정점을 방문함
		printf("%c ", getVertex(v)); //정점의 이름 출력

		for (int w = 0; w < size; w++)
			if (isLinked(v, w) && visited[w] == false)
				DFS(w); //연결 + 방문x => 순환호출로 방문
	}

	//인접 행렬로 표현된 그래프에 대한 너비우선탐색 연산
	void BFS(int v) {
		visited[v] = true; //현재 정점을 방문함
		printf("%c ", getVertex(v)); //정점의 이름 출력

		CircularQueue que;
		que.enqueue(v); //시작 정점을 큐에 저장

		while (!que.isEmpty()) {
			int v = que.dequeue(); //큐에 정점 추출
			for (int w=0;w<size;w++) //인접 정점 탐색
				if (isLinked(v, w) && visited[w] == false) {
					visited[w] = true; //방문 표시
					printf("%c ", getVerTex(w)); //정점의 이름 출력
					que.enqueue(w); //방문한 정점을 큐에 저장
				}
		}
	}
```

```
};
```

<ConnectedComponentGraph.h>

```
#include "SrchAMGraph.h"
class ConnectedComponentGraph : public SrchAMGraph {
private :
	int label[MAX_VTXS]; //정점의 색상 필드 추가

public : //깊피 우선 탐색
	void labelDFS(int v, int color) {
		visited[v] = true; //현재 정점을 방문함
		label[v] = color; //현재 정점의 색상
		for (int w = 0; w < size; w++)
			if (isLinked(v, w) && visited[w] == false)
				labelDFS(w, color);
	}

	//그래프의 연결 성분 검출 함수
	void findConnectedComponent() {
		int count = 0; //연결 성분의 수
		for (int i = 0; i < size; i++) //방문하지 않았으면
			if (visited[i] == false)
				labelDFS(i, ++count);
		print("그래프 연결성분 개수 == %d \n", count);
		for (int i = 0; i < size; i++)
			print("%c = %d ", getVertex(i), label[i]);
		print("\n");
	}
};
```