자료구조설계

제 출 일	2017.09.14.
과 제 번 호	02
분 반	01
학 과	컴퓨터공학과
학 번	201602038
이 름	이 미 진

1. 구현 내용 설명

1) public Graph(String[] args)

```
public Graph(String[] args){
        // 생성자
        // 인자로 받은 args 저장
        // 배열 a 초기화
        size = args.length;
        vertices = new String[size];
        a = new Node[size];
        int i;
        for (i=0; i<args.length; i++){
                vertices[i] = args[i];
                a[i] = new Node(i);
        }
}
```

- Graph 생성자에서는 args를 인자로 받고, 받은 인자의 길이를 size 변수에 저장한다. 그 후 정점들을 저장할 배열인 vertices와 각 정점당 하나의 리스트를 가지게 하기 위한 배열 a를 size만큼 정의한다.
for문을 통해 args의 길이만큼 인덱스를 1씩 증가시키며 vertices 배열에 args 인자를, a 배열엔 새로운 노드들을 집어넣는다.

2) public void add(String v, String w)

```
public void add(String v, String w){
        // 간선 추가
        // 정점 v의 리스트에 w 추가, 정점 w의 리스트에 v 추가
        a[index(v)].add(index(w));
        a[index(w)].add(index(v));
}
```

- 간선을 추가하는 메소드로, Graph 클래스 내의 index 메소드와 add를 통해 정점 v의 리스트에 w를 추가하고 정점 w의 리스트에 v를 추가하는 기능을 수행한다.

3) public String toString()

```
public String toString(){
        // 그래프 프린트
        Node e;
        if(size == 0) return "{}"; // 사이즈가 0이면 배열에 아무것도
들어가있지 않으므로 {} 만 출력
        StringBuffer buf = new StringBuffer();
        for( int i=0; i<size; i++ ) {
                if( i == 0 ) {    // 인덱스가 0일때
                        buf.append("{"+vertices[0]+":");
                }
                else {   // 인덱스가 0이 아닐때
                        buf.append(","+vertices[i]+":");
                }
                for( e = a[i].next; e != null; e = e.next ) {       // 인접한
정점 buf에 저장
                        buf.append(vertices[e.data]);
                }
        }
        return buf+"}";
```

- 만들어진 그래프를 프린트하는 메소드로, size가 0일 땐 배열이 empty 상태이므로 {}

만 출력한다.
배열에 값이 하나라도 들어있다면 StrinBuffer 클래스를 통해 값을 저장할 buf를 생성하고,
for문을 통해 인덱스 0부터 size 만큼 인덱스를 증가시키면서 인덱스가 0일땐 append
메소드를 사용해 그래프의 시작 형태인 “{”+vertices[0]+“:”를 buf에 저장한다.
인덱스가 0이 아닐 땐 “,”+vertices[i]+“:”를 buf에 저장해 그래프를 형성한다.
각 조건에 맞는 문장이 실행된 후에는 인접한 정점들을 buf에 저장한다.
모든 반복이 끝나면 buf+“}”를 return 해주면서 그래프를 프린트한다.

4) **private int** index(String v)

```
        private int index(String v){
                // vertices 배열에서 정점 v의 인덱스 반환
                for(int i = 0; i<size; i++){
                        if( vertices[i] == v ){
                                return i;
                        }
                }
                return 0;
        }
```

- vertices 배열에서 정점 v의 인덱스를 반환하는 메소드이고, size만큼 인덱스를 증가시키면서 vertices 배열의 인덱스별 값과 v가 같으면 인덱스 I를 return 해주고, 반복이 끝나면 return 0 로 종료시킨다.

5) **private void** calc_degree()

```
        public void calc_degree(){
                Node e;
                System.out.println("정점 인접한 정점 수");
                for(int i = 0; i<size; i++) {
                        int edges = 0;   // 초기화
                        System.out.print(" " + vertices[i] + "          ");   //
인덱스를 증가시키며 정점을 출력
                        for( e = a[i].next; e != null; e = e.next){// 간선 수를 통해
인접한 정점의 수를 구함
                                ++edges;          // 간선 증가
                        }
                        System.out.println(edges);        // 간선 수 출력
                }
        }
```

-정점과 인접한 정점의 수를 프린트해주는 메소드로, for문을 통해 size만큼 인덱스 I를 증가시킴과 동시에 간선 edges 변수를 0으로 초기화 시킨다.
그 후 vertices 배열 안의 정점들을 출력하고 다른 for문에서 인접한 정점이 null이 아닐 때까지 edges를 1씩 증가시켜 인접한 정점의 수를 구한다.
반복이 끝나면 edges를 출력한다.

6) **private class** Node

```
        private class Node {     // 노드 클래스
                private int data;
                private Node next;

                Node() {// data와 next 둘다 받지 않는 생성자
                        this.data = 0;
                        this.next = null;
```

```
                }

                Node(int data){  // data만 받는 생성자
                        this.data = data;
                        this.next = null;
                }
                public Node(int data, Node next){       // data와 next 모두 받는
생성자
                        this.data = data;
                        this.next = next;
                }

                public void add(int data) {       // data를 입력받아 리스트에 추가
                        this.next = new Node(data, next);
                }

        }
```

- Node 클래스에서 data 변수에 정점의 인덱스를 저장하고, next 노드를 통해 노드들간의 연결을 한다.

인자를 아무것도 받지 않는 생성자, data만 받는 생성자, data와 next 모두 받는 생성자를 작성하고, add 메소드를 이용해 data를 입력받으면 data를 리스트에 추가시킨다.

2. 실행 결과 화면

```
{A:EB,B:FCA,C:HDB,D:HC,E:FA,F:GEB,G:F,H:DC}

정점        인접한 정점 수
 A              2
 B              3
 C              3
 D              2
 E              2
 F              3
 G              1
 H              2
```