1. (**실습**: 그래프 표현, 인접행렬) 다음은 인접행렬에 기반한 방향그래프 표현을 구현한 예시 코드이다. 그래프의 총 노드 수는 변수 V에, 간선 집합은 문자열 변수 input에 저장되어 있다고 가정한다. input에 저장된 문자열 "0 1 0 3 1 2 ..."는 노드 0에서 노드 1로의 간선, 노드 0에서 노드 3으로의 간선, 노드 1에서 노드 2로의 간선 등이 존재함을 의미한다. 아래 코드를 입력하고 실행하면서 인접행렬 그래프 표현 구현법을 학습하시오.

   - **실습**: 아래 코드는 그림 1의 방향그래프(directed graph)를 인접행렬로 표현한 후 그 행렬을 출력한 것이다. 아래 코드가 그림 2의 무방향그래프를 인접행렬로 표현하도록 아래 코드를 수정하시오.

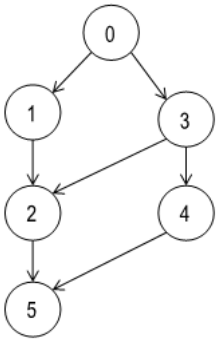


그림 1. 방향그래프

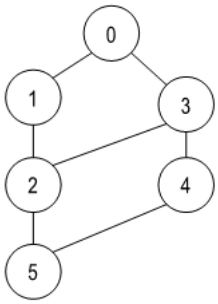


그림 2. 무방향그래프

```
public class Test {
	public static void main(String[] args) {
		int		V=6; // 그래프 내 총 노드 개수 (노드 번호 0 ~ 5)
		String	input="0 1 0 3 1 2 3 2 3 4 2 5 4 5";
		int		adjMat[][]=new int[V][V];
		String	s[]=input.split("\\s+");
		for (int i = 0; i < s.length; i+=2){
			int		v1=Integer.parseInt(s[i]);
			int		v2=Integer.parseInt(s[i+1]);
			adjMat[v1][v2]=1;
		}
		for (int i = 0; i < adjMat.length; i++) {
			for (int j = 0; j < adjMat[i].length; j++) {
				System.out.print(adjMat[i][j]);
			}
			System.out.println();
		}
	}
}
```

2. (**실습**: 그래프 표현, 인접리스트) 다음은 인접리스트에 기반한 방향그래프 표현을 구현한 예시 코드이다. 그래프의 총 노드 수는 변수 V에, 간선 집합은 문자열 변수 input에 저장되어 있다고 가정한다. input에 저장된 문자열 "0 1 0 3 1 2 ..."는 노드 0에서 노드 1로의 간선, 노드 0에서 노드 3으로의 간선, 노드 1에서 노드 2로의 간선 등이 존재함을 의미한다. 아래 코드를 입력하고 실행하면서 인접리스트 그래프 표현 구현법을 학습하시오.

- **실습**: 아래 코드는 그림 1의 방향그래프(directed graph)를 인접리스트로 표현한 후 그 인접리스트를 출력한 것이다. 아래 코드가 그림 2의 무방향그래프를 인접리스트로 표현하도록 아래 코드를 수정하시오.

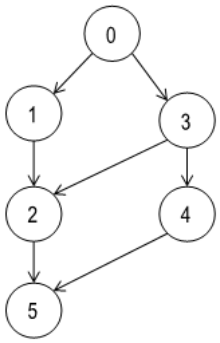


그림 1. 방향그래프

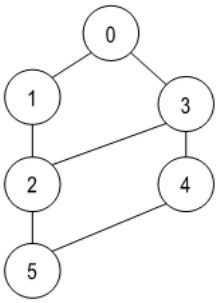


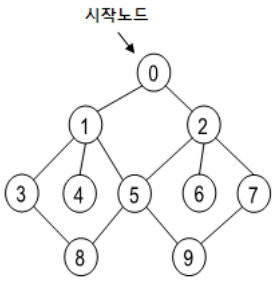
그림 2. 무방향그래프

```
public class Test {
	public static void main(String[] args) {
		int		V=6; // 그래프 내 총 노드 개수 (노드 번호 0 ~ 5)
		String	input="0 1 0 3 1 2 3 2 3 4 2 5 4 5";
		LinkedList<Integer>	adjList[]=new LinkedList[V];
		for (int i = 0; i < adjList.length; i++){
			adjList[i]=new LinkedList<>();
		}
		String	s[]=input.split("\\s+");
		for (int i = 0; i < s.length; i+=2){
			int		v1=Integer.parseInt(s[i]);
			int		v2=Integer.parseInt(s[i+1]);
			adjList[v1].add(v2);
		}
		for (int i = 0; i < adjList.length; i++){
			System.out.println("node "+i+" => "+adjList[i]);
		}
	}
}
```

3. (**실습**: 인접리스트 그래프 표현 및 dfs 방문) 다음은 인접리스트에 기반한 무방향그래프 표현을 사용하여 그래프 내 연결요소들을 dfs 방문하는 코드이다. 그래프의 총 노드 수는 변수 V에, 간선 집합은 문자열 변수 input에 저장되어 있다고 가정한다. input에 저장된 문자열 "0 1 0 2 1 3 ..." 는 노드 0에서 노드 1로의 간선, 노드 0에서 노드 2으로의 간선, 노드 1에서 노드 3으로의 간선 등이 존재함을 의미한다. 아래 코드를 입력하고 실행하면서 인접리스트 그래프 표현 및 dfs 탐색법을 학습하시오.

   A. 깊이우선탐색 (dfs, depth first search): 현재 노드를 방문(표시) 및 처리하고 현재 노드의 미방문 인접노드에 대해 깊이우선탐색(dfs)을 다시 적용한다.

   B. **실습**: 아래 코드의 그래프에서 0번 노드가 시작노드이고 7번 노드가 목표노드라고 할 때 dfs 탐색 중 목표노드에 도달하면 탐색을 종료하도록 아래 코드를 수정하시오.



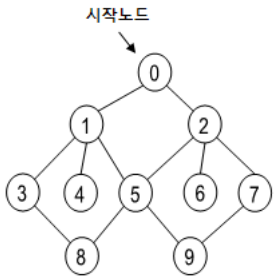

```
public class Test {
    public static void main(String[] args) {
        int       V=10; // 그래프 내 총 노드 개수 (노드 번호 0 ~ 9)
        String    input="0 1 0 2 1 3 1 4 1 5 2 5 2 6 2 7 3 8 5 8 5 9 7 9";

        LinkedList<Integer>   adjList[]=new LinkedList[V];
        for (int i = 0; i < adjList.length; i++) adjList[i]=new LinkedList<>();
        String    s[]=input.split("\\s+"); // String          s[]=input.split(" ");
        for (int i = 0; i < s.length; i+=2){
                int       v1=Integer.parseInt(s[i]);
                int       v2=Integer.parseInt(s[i+1]);
                adjList[v1].add(v2); // 무방향그래프로 처리
                adjList[v2].add(v1); // 무방향그래프로 처리
        }
        for (int i = 0; i < adjList.length; i++) System.out.println("node "+i+" => "+adjList[i]);
        dfsVisit(adjList);
    }
    private static void dfsVisit(LinkedList<Integer>[] adjList) {
        boolean  visited[]=new boolean[adjList.length]; // adjList.length = 그래프 내 총 노드 개수
        dfs(adjList, visited, 0); // 시작노드부터 dfs 방문
    }
    private static void dfs(LinkedList<Integer>[] adjList, boolean[] visited, int v) {
        visited[v]=true; // 현재 노드 v 방문 표시
        System.out.print(v+" "); // 현재 노드 v에 대한 처리
        for (Integer w : adjList[v]) { // 현재 노드 v의 인접 노드 w에 대해
                if(visited[w]==false) { // w가 미방문 인접 노드라면
                        dfs(adjList, visited, w); // 미방문 인접노드 w에 대해 dfs 재귀호출
                }
        }
    }
}
```

4. (**실습**: 인접리스트 그래프 표현 및 bfs 방문) 다음은 인접리스트에 기반한 무방향그래프 표현을 사용하여 그래프 내 연결요소들을 bfs 방문하는 코드이다. 그래프의 총 노드 수는 변수 V에, 간선 집합은 문자열 변수 input에 저장되어 있다고 가정한다. input에 저장된 문자열 "0 1 0 2 1 3 ..." 는 노드 0에서 노드 1로의 간선, 노드 0에서 노드 2으로의 간선, 노드 1에서 노드 3으로의 간선 등이 존재함을 의미한다. 아래 코드를 입력하고 실행하면서 인접리스트 그래프 표현 및 bfs 탐색법을 학습하시오.

   A. 너비우선탐색 (bfs, breadth first search): 최초 시작 노드를 방문 표시 후 큐에 삽입한 후, 공백 큐가 아닌 동안 "큐에서 노드 추출 후, 추출 노드의 인접 노드 중 미방문 노드를 방문 표시 후 큐에 삽입하는 작업"을 반복 수행한다.

   B. **실습**: 아래 코드의 그래프에서 0번 노드가 시작노드이고 7번 노드가 목표노드라고 할 때 dfs 탐색 중 목표노드에 도달하면 탐색을 종료하도록 아래 코드를 수정하시오.




```
public class Test {
    public static void main(String[] args) {
        int         V=10; // 그래프 내 총 노드 개수 (노드 번호 0 ~ 9)
        String      input="0 1 0 2 1 3 1 4 1 5 2 5 2 6 2 7 3 8 5 8 5 9 7 9";

        LinkedList<Integer>    adjList[]=new LinkedList[V];
        for (int i = 0; i < adjList.length; i++) adjList[i]=new LinkedList<>();
        String      s[]=input.split("\\s+"); // String          s[]=input.split(" ");
        for (int i = 0; i < s.length; i+=2){
                int         v1=Integer.parseInt(s[i]);
                int         v2=Integer.parseInt(s[i+1]);
                adjList[v1].add(v2); // 무방향그래프로 처리
                adjList[v2].add(v1); // 무방향그래프로 처리
        }
        for (int i = 0; i < adjList.length; i++) System.out.println("node "+i+" => "+adjList[i]);
        bfsVisit(adjList);
    }
    private static void bfsVisit(LinkedList<Integer>[] adjList) {
        boolean  visited[]=new boolean[adjList.length]; // adjList.length = 그래프 내 총 노드 개수
        LinkedList<Integer>    queue=new LinkedList<>();
        visited[0]=true; // 시작노드를 방문표시
        queue.addLast(0); // 시작노드를 큐에 삽입
        while(queue.isEmpty()==false){ // 공백 큐가 아닌 동안
                int v=queue.removeFirst(); // 큐에서 노드 추출
                System.out.print(v+" "); // 큐에서 추출된 노드 v에 대한 처리
                for (Integer w : adjList[v]) { // 큐에서 추출된 노드 v의 인접노드 w에 대해
                        if(visited[w]==false){ // w가 미방문 인접노드라면
                                visited[w]=true; // 미방문 인접노드 w를 방문표시
                                queue.addLast(w); // 미방문 인접노드 w를 큐에 삽입
                        }
                }
        }
    }
}
```

References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- 김윤명. (2008). 뇌를 자극하는 Java 프로그래밍. 한빛미디어.
- 남궁성. 자바의 정석. 도우출판.
- 김윤명. (2010). 뇌를 자극하는 JSP & Servlet. 한빛미디어.