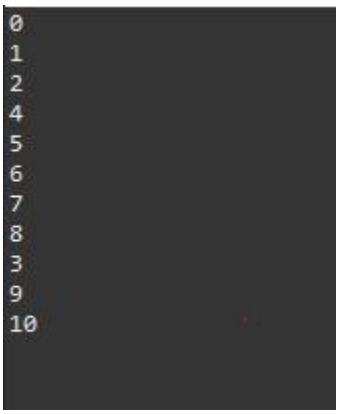
Algorithm
- deep first search -

제출일자	2020.11.13
과제 번호	10
이　　름	강인한
학　　번	201701969

실행결과

case 1

case 2



풀이과정 : 깊이 우선 탐색을 하기 위해서 data를 인접 리스트로 구현하고 스택을 사용하여 반복하였다. 먼저 각각의 edge에 따라 i번째 index에 i와 연결된 모든 vertex가 들어가도록 arraylist를 작성하였고 stack을 이용해 vertex를 저장하였다. stack에 저장하였다면 arraylist에 존재하는 vertex를 삭제하였고 더 이상 추가할 vertex가 없을 때는 pop을 이용하여 전의 상태로 돌아가 추가할 vertex가 있는지 확인하였다 이러한 반복을 통해 stack이 아무것도 존재하지 않을 때까지 반복한다면 모든 노드를 깊이에 따른 모든 vertex에 접근할 수 있게 된다.

코드 설명 : 파일에서 vertex와 edge를 읽어 저장한 뒤, edge에 크기만큼 반복하며 arraylist에 저장한다. 이런 과정을 통해 arraylist의 i번째 index에는 vertex i 와 연결된 모든 vertex가 저장된다. 시작점인 0을 먼저 stack에 넣고 arraylist에 존재하는 모든 0은 삭제된다.

-위의 설명처럼 stack에 들어간 값을 arraylist에서 삭제해주는 과정

```
while (true) {
    for (int i = 0; i < ver; i++) {  //stack에 들어간 숫자를 제거해주는 과정
        for (int j = 0; j < arr.get(i).size(); j++) {
            if (idx == arr.get(i).get(j)) {
                arr.get(i).remove(j);
            }
        }
    }
```

arraylist의 idx번째 index가 비어있어 stack에 들어갈 수 없다면 이 idx보다 아래 존재하는 stack에 접근하기 위해 pop을 사용하여 idx의 값을 삭제한다.

idx번째 index가 비어있지 않다면 idx index 값 중에서 첫 index인 0번째 값을 stack에 추가한다. 추가로 stack에 들어갔다는 것은 vertex에 접근했다는 것이므로 출력해야 한다.

-위의 설명처럼 조건을 나눠 필요한 작업을 진행한다.

```
if (arr.get(idx).isEmpty()) { //arraylist의 idx번째 index가 이미 다 stack에 들어가 더이상 넣을것이 없다면 pop
    sck.pop();
} else {  //stack에 넣을것이 있다면 push
    sck.push(arr.get(idx).get(0));
    System.out.println(sck.peek());
}
```

위의 과정을 통해 모든 값이 다 비어있어 스택이 빈 값이 된다면 전체의 while문을 종료해준다.
추가로 index를 stack에 top부분으로 바꿔주는 작업을 해야한다.

```
if (sck.isEmpty()) {  //stack이 다 비워지면 while문 종료
    break;
}
idx = sck.peek();  //index를 바꿔주는 과정
```

느낀 점 : linkedlist를 구현만 해보았지 사용한 적이 없었는데 이번 과제를 통해 더 어려운 arraylist를 잘 알게 돼 만족스럽습니다. 또 arraylist와 stack을 이용해 어떻게 공존해서 사용하는지 고민을 많이 했습니다. 과제에서는 이미 접근했던 vertex를 다시 출력하지 않기 위해 pop을 사용해 이전의 top으로 돌아가는 것이 핵심인 것 같습니다.