12회 2번(2018-JAVA)

2. 신입생 오리엔테이션에 참석할 학생들에게 (1번부터 시작되는) 번호표를 주었다. 이 번호표는 학생들이 머무르는 방의 번호를 결정하고, (게임 등을 하기 위해) 같은 방을 쓰는 학생들에게도 내부적으로 일련번호를 주기 위해 쓰인다. 각 방은 15명씩 배정되고, 같은 방에 배정된 학생들의 경우는 번호표 순으로 방안에서의 번호가 결정된다.

학생에게 줄 번호표를 입력받아 그 학생의 방 번호와 방안에서의 번호를 결정해 주는 프로그램을 작성하시오.

[입력 형식]
- 번호표 n을 입력한다. ($1 \leq n \leq 1000$)

[출력 형식]
- 방 번호와 방안에서의 번호를 공백으로 구분하여 순서대로 출력한다.

[입력 예1]

```
7
```

[출력 예1]

```
1 7
```

[입력 예2]

```
30
```

[출력 예2]

```
2 15
```

총 15명씩 한 방에 배정된다.
1~15번 : 1번 방, 16~30번 : 2번 방 …

15, 30번과 같이 나누어 떨어지는 경우를 고려해야 함.

1~14번의 경우, 1번 방에 속하며 각자 숫자에 해당하는 번호를 받는다.
즉, 방 번호 = n/15 + 1, 방안에서의 번호 = n % 15

15번의 경우, 1번 방에 속하며 방안에서의 번호는 15번이다.
즉, 방 번호 = n/15, 방안에서의 번호 = 15

이것이 15마다 반복되므로
1. 15로 나누어 떨어지는 경우
2. 그렇지 않은 경우
를 구분하여 방 번호와 방안에서의 번호를 결정한다.

12회 2번(2018-JAVA)

자바 소스코드

2. 신입생 오리엔테이션에 참석할 학생들에게 (1번부터 시작되는) 번호표를 주었다. 이 번호표는 학생들이 머무르는 방의 번호를 결정하고, (게임 등을 하기 위해) 같은 방을 쓰는 학생들에게도 내부적으로 일련번호를 주기 위해 쓰인다. 각 방은 15명씩 배정되고, 같은 방에 배정된 학생들의 경우는 번호표 순으로 방안에서의 번호가 결정된다.

학생에게 줄 번호표를 입력받아 그 학생의 방 번호와 방안에서의 번호를 결정해 주는 프로그램을 작성하시오.

[입력 형식]
- 번호표 n을 입력한다. ($1 \leq n \leq 1000$)

[출력 형식]
- 방 번호와 방안에서의 번호를 공백으로 구분하여 순서대로 출력한다.

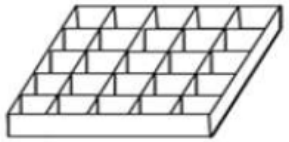
[입력 예1]	[출력 예1]
7	1 7

[입력 예2]	[출력 예2]
30	2 15

```java
package EPPER_2018_JAVA;
import java.util.*;
public class Q2_2018 {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        int n = input.nextInt();
        int number;//방안에서의 번호
        int room;//방 번호
        if(n%15 == 0) {
            room = n/15;
            number = 15;
        }//n == 15, 30, 45 ... 인 경우
        else {
            room = n/15 + 1;
            number = n%15;
        }//그 외의 경우
        System.out.println(room+" "+number);
    }
}
```

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

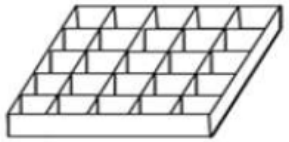
이 문제는 그래프 이론과 너비 우선 탐색(BFS)을 알아야 풀 수 있는 문제이다.

너비 우선 탐색(Breadth-First Search)
: 루트 노드(혹은 다른 임의의 노드)에서 시작해서 인접한 노드를 먼저 탐색하는 방법
시작 정점으로부터 가까운 정점을 먼저 방문하고 멀리 떨어져 있는 정점을 나중에 방문하는 순회 방법이다.
두 노드 사이의 최단 경로 혹은 임의의 경로를 찾고 싶을 때 이 방법을 선택한다.

BFS는 방문한 노드들을 차례로 저장한 후 꺼낼 수 있는 자료구조인 큐(Queue)를 사용한다.
즉, 선입선출(FIFO) 원칙으로 탐색한다.

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

하루가 지나면 익은 토마토(1)를 기준으로 상하좌우의 익지 않은 토마토(0)가 익은 토마토로 바뀐다.

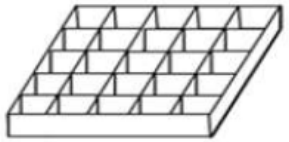
즉, 이 때 bfs를 수행하는데, 익은 토마토를 큐에 삽입하고 해당 토마토의 상하좌우에 위치한 익지 않은 토마토를 익은 토마토로 바꿔준다.

그 후, 익은 토마토를 다시 큐에 삽입하는 방식으로 바꿀 수 있는 모든 토마토를 반복적으로 바꿔준다.

가능한 탐색이 모두 끝나면 0인 토마토가 남아있는 지 확인해야 한다. 이를 위해 입력 받을 때 0인 토마토의 수를 미리 세고, 토마토가 익을 때마다 그 개수에서 차감한다.

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

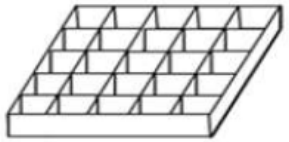
```
14
```

토마토가 익는데 걸린 날짜를 측정해야 하는데,
이 부분은 bfs를 수행하면서, 토마토 배열을 수정하는 방식으로 측정될 수 있다.

우선, 초기 토마토 배열에 익은 토마토에는 1이 저장되어 있다.
이것을 기준으로, 처음부터 익어 있는 토마토의 상하좌우이면서, 익지 않은 토마토(0)이면 해당 값에 +1을 해주는 방식으로 day를 측정할 수 있다.
그렇다면 첫 토마토의 상하좌우에는 2가 저장되게 될 것이다.
첫째 날 익은 토마토에는 2가 저장되고 또 다시 큐에 삽입되어 탐색을 반복한다. 그렇다면 둘째 날 익은 토마토에는 3이 저장될 것이다.
이런 방식으로 큐가 비어 있게 될 때까지 반복해서 토마토 배열의 값을 업데이트한다.

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

첫번째 입력 예를 통해 조금 더 직접적으로 이해해보자.

초기 상태
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1

Day 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 2
0 0 0 0 2 1

Day 2
0 0 0 0 0 0
0 0 0 0 0 3
0 0 0 0 3 2
0 0 0 3 2 1

Day 3
0 0 0 0 0 4
0 0 0 0 4 3
0 0 0 4 3 2
0 0 4 3 2 1

Day 4
0 0 0 0 5 4
0 0 0 5 4 3
0 0 5 4 3 2
0 5 4 3 2 1

Day 5
0 0 0 6 5 4
0 0 6 5 4 3
0 6 5 4 3 2
6 5 4 3 2 1

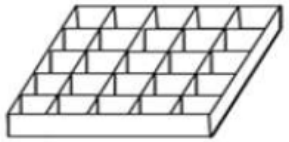
Day 6
0 0 7 6 5 4
0 7 6 5 4 3
7 6 5 4 3 2
6 5 4 3 2 1

Day 7
0 8 7 6 5 4
8 7 6 5 4 3
7 6 5 4 3 2
6 5 4 3 2 1

Day 8
9 8 7 6 5 4
8 7 6 5 4 3
7 6 5 4 3 2
6 5 4 3 2 1

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

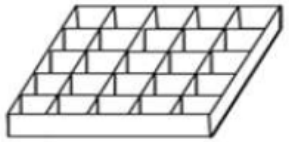
이런 식으로 day를 측정하는데, 마지막으로 pop 한 큐의 값이 최종 day 값이 된다.
그런데 첫째 날 익은 토마토에 2가 저장되었다.
초기값이 1이 아니기 때문에 최종 day 값은 -1 한 값이 된다.
즉, (마지막 pop한 토마토 배열의 값 - 1) 이 최소 일수가 된다.

탐색을 수행하면서,
1) 모든 토마토가 바뀐 경우
2) 모든 토마토가 바뀌지 않은 경우

두 경우가 존재하는데, 이 부분을 구분하기 위해 초기 입력 때 익어야 하는 토마토의 수를 세야 한다고 말했다.
토마토가 익어가면서 해당 값을 차감하고, 탐색이 끝나고 그 값이 0이라면, 모든 토마토가 바뀐 것이므로 1)에 해당한다. 따라서 BFS 함수는 (마지막 pop한 토마토 배열의 값 - 1)을 return 한다.
그 값이 0이 아니라면 모든 토마토가 바뀐 것이 아니므로 모두 익지 못하는 상황에 해당해 -1를 return 한다.

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

로직 전체를 입력 예시 1번으로 함께 확인해보자.

초기 상태, cnt: 23
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1

Day 1, cnt: 21
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 2
0 0 0 0 2 1

Day 2, cnt: 18
0 0 0 0 0 0
0 0 0 0 0 3
0 0 0 0 3 2
0 0 0 3 2 1

Day 3, cnt : 14
0 0 0 0 0 4
0 0 0 0 4 3
0 0 0 4 3 2
0 0 4 3 2 1

Day 4, cnt: 9
0 0 0 0 5 4
0 0 0 5 4 3
0 0 5 4 3 2
0 5 4 3 2 1

Day 5, cnt: 6
0 0 0 6 5 4
0 0 6 5 4 3
0 6 5 4 3 2
6 5 4 3 2 1

Day 6, cnt: 3
0 0 7 6 5 4
0 7 6 5 4 3
7 6 5 4 3 2
6 5 4 3 2 1

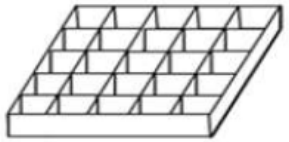
Day 7, cnt: 1
0 8 7 6 5 4
8 7 6 5 4 3
7 6 5 4 3 2
6 5 4 3 2 1

Day 8, cnt: 0
9 8 7 6 5 4
8 7 6 5 4 3
7 6 5 4 3 2
6 5 4 3 2 1

BFS 탐색을 완료한 후 cnt가 0이므로 모든 토마토가 익었다.
이때, 마지막으로 pop 한 토마토는 (0, 0)에 위치한 토마토이므로 9-1, 즉 8을 return 한다.

14회 8번(2019-JAVA)

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

Queue 기능 구현 부분 소스

이 코드는 C언어로 작성되었기 때문에
Queue 기능을 직접 구현했다.
Java, C++ 등을 사용한다면
내장된 기능을 사용하면 된다.

C 소스코드

```c
#include <stdio.h>
#include <malloc.h>//동적할당을 위한 헤더

typedef struct n {
    int x;
    int y;
}q;//좌표를 저장하기 위한 struct

q queue[1000000];//bfs를 수행하기 위한 queue
int front = 0;
int rear = 0;

void insert(int _x, int _y) {
    queue[rear].x = _x;
    queue[rear].y = _y;
    rear = rear + 1;
}

q pop() {
    q temp = queue[front];
    front++;
    return temp;
}

int isEmpty() {
    if (rear == front) return 0;
    return 1;
}
//queue 기능 구현
```

14회 8번(2019-JAVA)

C 소스코드

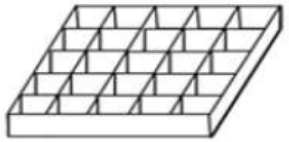
```c
int solution(int sizeX, int sizeY, int **arr) {
    int vectX[4] = { 1,-1,0,0 };
    int vectY[4] = { 0,0,-1,1 };
    //상하좌우 좌표 이동을 위한 X,Y vertor 배열

    int cnt = 0;//바뀌어야 하는 토마토의 개수
    for (int i = 0; i < sizeX; i++) {
        for (int j = 0; j < sizeY; j++) {
            if (arr[i][j] == 0) cnt++; //바뀌어야 하는 토마토의 개수
            else if (arr[i][j] == 1) insert(i, j);//1인 토마토는 큐에 삽입
        }
    }

    int x, y;
    while (isEmpty()) {
        q temp = pop();//1인 토마토
        x = temp.x;
        y = temp.y;

        for (int i = 0; i < 4; i++) {
            int nextX = x + vectX[i];
            int nextY = y + vectY[i];
            if (nextX >= 0 && nextX < sizeX && nextY >= 0 && nextY < sizeY) {//적정 범위 내의 좌표인지 체크
                if (arr[nextX][nextY] == 0) {
                    arr[nextX][nextY] = arr[x][y] + 1;
                    //map[x][y]는 1인 토마토이고, day를 측정해야 하기 때문에 +1을 함.
                    //day3에 바뀐 토마토가 바꾸는 토마토는 day4에 바뀌는 것이기 때문.
                    insert(nextX, nextY);//다시 큐에 삽입
                    cnt--;//0인 토마토의 개수
                }
            }
        }
    }
    if (cnt == 0) return arr[x][y] - 1;//첫째날에 바뀐 토마토도 1+1이라 2의 값을 가지고 있으므로 -1을 해줘야 며칠이 걸렸는지를 표시하는 것.
    return -1;
}
```

그래프 탐색을 위한 BFS를 진행하는 solution 함수 소스코드

14회 8번(2019-JAVA)

C 소스코드

8. 모든 잘 익은 과일은 잘 익도록 도와주는 역할을 하는 가스인 에틸렌을 방출한다. 다음 $N \times M$의 칸으로 나누어진 상자에 토마토를 보관한다. 토마토 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있다.



보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토의 인접한 곳은 왼쪽, 오른쪽, 앞, 뒤 네 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지, 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자 모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하시오. 단, 상자의 일부 칸에는 토마토가 없을 수도 있다.

[입력 형식]

- 첫째 줄에 상자의 크기를 나타내는 두 정수 N, M을 입력한다. $(2 \le N, M \le 1000)$
- 둘째 줄부터는 하나의 상자에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 상자에 담긴 토마토의 정보가 주어진다. 하나의 줄에는 상자 가로 줄에 들어 있는 토마토의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다.

[출력 형식]

- 상자 안에 있는 토마토들이 모두 익을 때까지의 최소 일수를 출력한다. 모두 익은 상태로 주어졌을 때는 0을 출력한다. 토마토가 모두 익지 못하는 상황이면 -1을 출력한다.

[입력 예1]

```
6 4
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예2]

```
6 4
0 -1 0 0 0 0
-1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
```

[입력 예3]

```
5 5
-1 1 0 0 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 -1 -1 -1 0
0 0 0 0 0
```

[출력 예1]

```
8
```

[출력 예2]

```
-1
```

[출력 예3]

```
14
```

```c
int main() {
    int sizeX, sizeY;//가로 세로 크기
    scanf("%d%d", &sizeY, &sizeX);
    int **map = (int **)malloc(sizeof(int *)*sizeY);//토마토 배열
    for (int i = 0; i < sizeY; i++) {
        map[i] = (int *)malloc(sizeof(int) * sizeX);
    }
    printf("%d₩t%d₩n", sizeY, sizeX);
    for (int i = 0; i < sizeX; i++) {
        for (int j = 0; j < sizeY; j++) {
            scanf("%d", &map[i][j]);
        }
    }

    printf("%d₩n", solution(sizeX, sizeY, map));
}
```

main 함수 소스코드
입출력 및 solution 함수 호출

13회 9번(2019-C)

9. (10점) 이화는 길을 걷고 있는데, 어느 날 산신령이 나타나서 길에 돈을 일렬로 놓으며 "돈을 마음껏 주워가라. 단, 연속해서 3개의 돈을 가질 수 없다."라고 말하였다. 이화가 최대 주울 수 있는 돈의 액수를 구하는 프로그램을 작성하시오.

다음과 같이 길에 돈이 있을 경우, 최대 주울 수 있는 돈은 37원이 된다.

5	7	10	1	2	10	10	8

[입력 형식]
- 첫 번째 줄에 입력받을 돈의 개수 n을 입력한다. $(1 \leq n \leq 30000)$
- 두 번째 줄에 n개의 돈의 액수 M_i를 공백으로 구분하여 입력한다. $(1 \leq M_i \leq 10)$

[출력 형식]
- 주울 수 있는 돈의 최대 금액을 출력한다.

[입력 예1]

```
8
5 7 10 1 2 10 10 8
```

[출력 예1]

```
37
```

[입력 예2]

```
8
1 2 3 4 5 6 7 8
```

[출력 예2]

```
27
```

동적 계획법, 즉 다이나믹 프로그래밍(DP)을 이용하여 해결하는 문제.

동적계획법이란?

분할 정복(divide-and-conquer)과 유사.

작은 문제들의 답을 테이블에 저장해 놓은 후 필요할 때 테이블의 값을 이용함.

1. 재귀적 속성
2. 작은 문제들에 대한 답을 구해 테이블에 저장
3. 테이블 내의 값을 이용

대표적으로 피보나치 수열이 동적 계획법에 해당함.

13회 9번(2019-C)

9. (10점) 이화는 길을 걷고 있는데, 어느 날 산신령이 나타나서 길에 돈을 일렬로 놓으며 "돈을 마음껏 주워가라. 단, 연속해서 3개의 돈을 가질 수 없다."라고 말하였다. 이화가 최대 주울 수 있는 돈의 액수를 구하는 프로그램을 작성하시오.

다음과 같이 길에 돈이 있을 경우, 최대 주울 수 있는 돈은 37원이 된다.

5	7	10	1	2	10	10	8

[입력 형식]
- 첫 번째 줄에 입력받을 돈의 개수 n을 입력한다. $(1 \leq n \leq 30000)$
- 두 번째 줄에 n개의 돈의 액수 M_i를 공백으로 구분하여 입력한다. $(1 \leq M_i \leq 10)$

[출력 형식]
- 주울 수 있는 돈의 최대 금액을 출력한다.

[입력 예1]

```
8
5 7 10 1 2 10 10 8
```

[출력 예1]

```
37
```

[입력 예2]

```
8
1 2 3 4 5 6 7 8
```

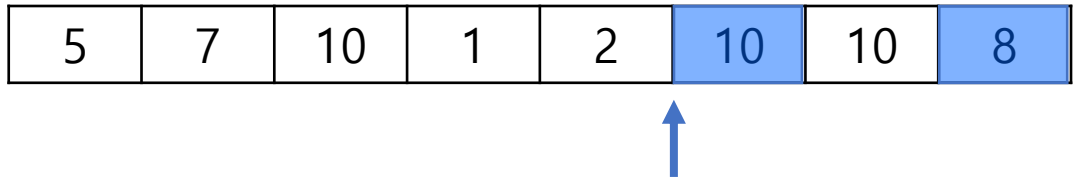
[출력 예2]

```
27
```

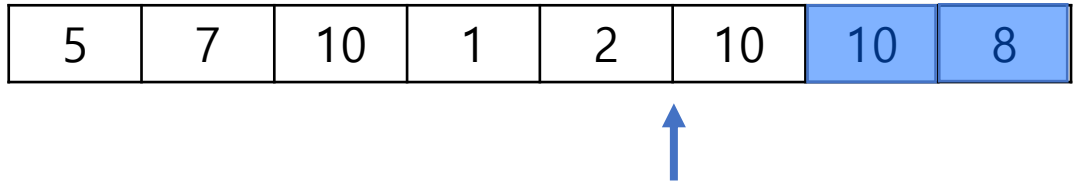
문제에서 연속해서 3개의 돈을 가질 수 없다는 조건이 있다.

그렇다면 마지막 돈을 기준으로 있을 수 있는 경우의 수는 총 세가지이다.

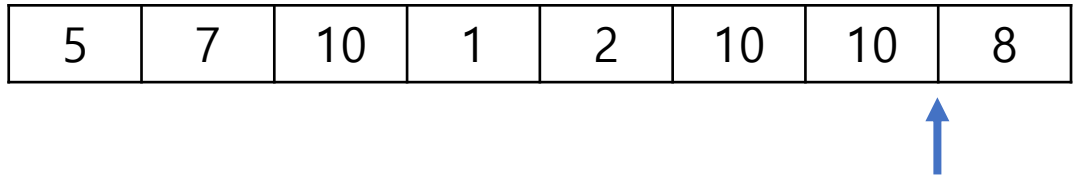
1. N-2번째 + N번째



2. N-1번째 + N번째



3. N번째에 돈을 줍지 않은 경우



이 속성을 이용하여 재귀적으로 작은 문제들에 대한 답을 구해 최종 값을 구할 수 있다.

13회 9번(2019-C)

9. (10점) 이화는 길을 걷고 있는데, 어느 날 산신령이 나타나서 길에 돈을 일렬로 놓으며 "돈을 마음껏 주워가라. 단, 연속해서 3개의 돈을 가질 수 없다."라고 말하였다. 이화가 최대 주울 수 있는 돈의 액수를 구하는 프로그램을 작성하시오.

다음과 같이 길에 돈이 있을 경우, 최대 주울 수 있는 돈은 37원이 된다.

5	7	10	1	2	10	10	8

[입력 형식]
- 첫 번째 줄에 입력받을 돈의 개수 n을 입력한다. $(1 \leq n \leq 30000)$
- 두 번째 줄에 n개의 돈의 액수 M_i를 공백으로 구분하여 입력한다. $(1 \leq M_i \leq 10)$

[출력 형식]
- 주울 수 있는 돈의 최대 금액을 출력한다.

[입력 예1]

```
8
5 7 10 1 2 10 10 8
```

[출력 예1]

```
37
```

[입력 예2]

```
8
1 2 3 4 5 6 7 8
```

[출력 예2]

```
27
```

1. N-2번째 + N번째

5	7	10	1	2	10	10	8

여기서 N-2번째에 돈을 주웠다는 것은 결국 N-2번째까지의 최댓값을 의미하므로
N-2번째까지의 최댓값에 N번째 돈을 더한다.
즉, DP[n] = DP[n-2] + money[n]

2. N-1번째 + N번째

5	7	10	1	2	10	10	8

연속해서 3번을 주울 수 없기 때문에 N-2번째의 돈은 주울 수 없다.
N-3번째까지의 최댓값에 N-1번째, N번째 돈을 더한다.
즉, DP[n] = DP[n-3] + money[n-1] + money[n]

3. N번째에 돈을 줍지 않은 경우

5	7	10	1	2	10	10	8

N번째 돈을 줍지 않았으므로 그 전까지의 최댓값을 취한다.
즉, DP[n] = DP[n-1]

13회 9번(2019-C)

9. (10점) 이화는 길을 걷고 있는데, 어느 날 산신령이 나타나서 길에 돈을 일렬로 놓으며 "돈을 마음껏 주워가라. 단, 연속해서 3개의 돈을 가질 수 없다."라고 말하였다. 이화가 최대 주울 수 있는 돈의 액수를 구하는 프로그램을 작성하시오.

다음과 같이 길에 돈이 있을 경우, 최대 주울 수 있는 돈은 37원이 된다.

5	7	10	1	2	10	10	8

[입력 형식]
- 첫 번째 줄에 입력받을 돈의 개수 n을 입력한다. $(1 \leq n \leq 30000)$
- 두 번째 줄에 n개의 돈의 액수 M_i를 공백으로 구분하여 입력한다. $(1 \leq M_i \leq 10)$

[출력 형식]
- 주울 수 있는 돈의 최대 금액을 출력한다.

[입력 예1]

```
8
5 7 10 1 2 10 10 8
```

[출력 예1]

```
37
```

[입력 예2]

```
8
1 2 3 4 5 6 7 8
```

[출력 예2]

```
27
```

세 가지 경우의 수 중, 최댓값을 DP배열에 저장하고 이후의 계산에서 이용한다.
최종적으로 DP[n]에 주울 수 있는 최대 돈의 액수가 저장된다.

단, n이 3 미만일 경우 점화식이 해당되지 않기 때문에 초기화가 필요하다.
(편의를 위해 1번째 index부터 값을 저장함)

DP[1] = money[1]
DP[2] = money[1] + money[2]

연속해서 세 번 줍는 경우가 없기 때문에 모든 돈을 줍는 것이 최댓값이 된다.

13회 9번(2019-C)

9. (10점) 이화는 길을 걷고 있는데, 어느 날 산신령이 나타나서 길에 돈을 일렬로 놓으며 "돈을 마음껏 주워가라. 단, 연속해서 3개의 돈을 가질 수 없다."라고 말하였다. 이화가 최대 주울 수 있는 돈의 액수를 구하는 프로그램을 작성하시오.

다음과 같이 길에 돈이 있을 경우, 최대 주울 수 있는 돈은 37원이 된다.

5	7	10	1	2	10	10	8

[입력 형식]
- 첫 번째 줄에 입력받을 돈의 개수 n을 입력한다. $(1 \leq n \leq 30000)$
- 두 번째 줄에 n개의 돈의 액수 M_i를 공백으로 구분하여 입력한다. $(1 \leq M_i \leq 10)$

[출력 형식]
- 주울 수 있는 돈의 최대 금액을 출력한다.

[입력 예1]

```
8
5 7 10 1 2 10 10 8
```

[출력 예1]

```
37
```

[입력 예2]

```
8
1 2 3 4 5 6 7 8
```

[출력 예2]

```
27
```

C 소스코드

```c
#include<stdio.h>

int MAX(int a, int b) {
    return a > b ? a : b;
}//a와 b 중 더 큰 것을 반환

int solution(int n, int *arr) {
    int dp[30001] = { 0, };//dp 배열 선언 및 배열 전체를 0으로 초기화
    dp[1] = arr[1];//1인 경우 초기화
    dp[2] = arr[1] + arr[2];//2인 경우 초기화
    for (int i = 3; i <= n; i++) {
        dp[i] = MAX(dp[i - 2] + arr[i], MAX(dp[i - 3] + arr[i - 1] + arr[i], dp[i - 1]));
        /*
        dp[i] : i번째까지 주웠을 때 최대로 주울 수 있는 돈
        1. dp[i - 2] + money[i]
        2. dp[i - 3] + money[i-1]+ money[i]
        3. dp[i - 1]
        중 최댓값을 dp[i]에 저장함.
        */
    }//3부터 n 까지 dp값을 계산
    return dp[n];
}

int main() {
    int n;
    scanf("%d", &n);
    int money[30001] = { 0, };//money 배열 선언 및 배열 전체를 0으로 초기화
    for (int i = 1; i <= n; i++) {
        scanf("%d", &money[i]);
    }//money 배열에 입력 받음

    printf("%d₩n", solution(n, money));//최댓값 출력
}
```

(참고) int dp[30001] = { 0, }; 를 하면 배열의 모든 값이 자동으로 0 으로 초기화된다.
C언어의 경우 배열이 자동 초기화 되지 않으므로 유의해야 한다.
n이 3 미만이더라도 배열의 값이 0으로 저장되어 있기 때문에 null pointer 에러가 발생하지 않는다.