BFS (연습 2)

최백준 choi@startlink.io

늑대와 양

https://www.acmicpc.net/problem/16956

- 크기가 R×C인 목장이 있고, 목장의 각 칸은 비어있거나, 양(S) 또는 늑대(W)가 있다.
- 늑대는 인접한 칸을 자유롭게 이동할 수 있다.
- 목장에 울타리(D)를 설치해 늑대가 양이 있는 칸으로 갈 수 없게 하려고 한다.
- 울타리를 설치해보자. (최소를 구하는 문제 아님)
- $1 \le R, C \le 500$

.	.	S	.	.	.
.	.	S	.	W	.
.	S	.	.	.	.
.	.	W	.	.	.
.	.	.	W	.	.
.	.	.	.	.	.

.	.	S	D	.	.
.	.	S	D	W	.
.	S	D	.	.	.
.	D	W	.	.	.
D	D	.	W	.	.
.	.	.	.	.	.

늑대와 양

https://www.acmicpc.net/problem/16956

- 늑대와 양이 인접한 경우만 없다면, 항상 분리시킬 수 있다.

늑대와 양

https://www.acmicpc.net/problem/16956

- 최소를 구하는 문제가 아니기 때문에, 가능하면 그냥 모든 빈 칸을 울타리로 바꾸자

늑대와 양

https://www.acmicpc.net/problem/16956

- 소스: http://codeplus.codes/0c677ed4c5a142c5b0f90b0665fb2230

스타트링크

https://www.acmicpc.net/problem/5014

- 스타트링크는 총 F층으로 이루어진 건물에 사무실이 있다. 사무실은 G층
- 지금 S층에 있는 사람이 G층으로 가려면 버튼을 최소 몇 번 눌러야 하는지 구하는 문제
- 엘리베이터의 버튼은 U (U층 위로 올라가는 버튼), D (D층 아래로 내려가는 버튼)
- $1 \le S, G \le F \le 1000000, 0 \le U, D \le 1000000$

스타트링크

https://www.acmicpc.net/problem/5014

- 총 백만개의 층으로 이루어져 있고, 여기서 BFS탐색을 이용하면 최소로 버튼을 누르는 횟수를 구할 수있다.

스타트링크

https://www.acmicpc.net/problem/5014

- 소스: http://codeplus.codes/780283a0d11741b8a97cda9a48a9635e

탈옥

https://www.acmicpc.net/problem/9376

- 빈 칸, 벽, 문으로 이루어진 지도가 주어진다.
- 두 죄수가 탈옥하기 위해서 열어야 하는 문의 최소 개수를 구하는 문제

탈옥

https://www.acmicpc.net/problem/9376

- 두 지도를 상하좌우로 한 칸씩 확장하면
- 두 죄수의 탈옥 경로는
- 어딘가에서 만나서 함께 이동하는 꼴이 된다
- 따라서, 지도의 밖에서 BFS 1번, 각 죄수별로 1번씩 BFS를 수행한다.
- 그 다음, 정답을 합친다
- 이 때, 문에서 만나는 경우는 조심해야 한다

탈옥

https://www.acmicpc.net/problem/9376

*	*	*	*	#	*	*	*	*
*	.	.	#	.	#	.	.	*
*	*	*	*	.	*	*	*	*
*	$	#	.	#	.	#	$	*
*	*	*	*	*	*	*	*	*

탈옥

https://www.acmicpc.net/problem/9376

.	.	.	.	.	.	.	.	.	.	.
.	*	*	*	*	#	*	*	*	*	.
.	*	.	.	#	.	#	.	.	*	.
.	*	*	*	*	.	*	*	*	*	.
.	*	$	#	.	#	.	#	$	*	.
.	*	*	*	*	*	*	*	*	*	.
.	.	.	.	.	.	.	.	.	.	.

탈옥

https://www.acmicpc.net/problem/9376

밖에서 부터

0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	1	-	-	-	-	0
0	-	2	2	2	1	2	2	2	-	0
0	-	-	-	-	1	-	-	-	-	0
0	-	3	3	2	2	2	3	3	-	0
0	-	-	-	-	-	-	-	-	-	0
0	0	0	0	0	0	0	0	0	0	0

죄수 1부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	0	1	1	2	2	3	3	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

죄수 2부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	3	3	2	2	1	1	0	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

탈옥

https://www.acmicpc.net/problem/9376

밖에서 부터

0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	1	-	-	-	-	0
0	-	2	2	2	1	2	2	2	-	0
0	-	-	-	-	1	-	-	-	-	0
0	-	3	3	2	2	2	3	3	-	0
0	-	-	-	-	-	-	-	-	-	0
0	0	0	0	0	0	0	0	0	0	0

죄수 1부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	0	1	1	2	2	3	3	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

죄수 2부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	3	3	2	2	1	1	0	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

탈옥

https://www.acmicpc.net/problem/9376

- 소스: http://codeplus.codes/6372ed074277455c9406e6daf9776c35

물통

https://www.acmicpc.net/problem/2251

- 세 물통 A, B, C가 있을 때
- C만 가득차있다
- 어떤 물통에 들어있는 물을 다른 물통으로 쏟아 부을 수 있는데, 이 때에는 앞의 물통이 빌 때까지 붓거나, 뒤의 물통이 가득 찰 때까지 붓게 된다
- 이 과정에서 손실되는 물은 없다
- 이 때, A가 비어있을 때, C에 들어있을 수 있는 양을 모두 구하는 문제

물통

https://www.acmicpc.net/problem/2251

- 3차원 배열을 만들 필요는 없다
- 중간에 물이 손실되지 않기 때문에
- 첫 번째 물통, 두 번째 물통에 들어있는 물의 양만 알면 세 번째 물통에 들어있는 물이 양을 알 수 있다

물통

https://www.acmicpc.net/problem/2251

```
queue<pair<int,int>> q;
q.push(make_pair(0, 0)); check[0][0] = true; ans[c] = true;
while (!q.empty()) {
    int x = q.front().first, y = q.front().second;
    int z = sum - x - y;
    q.pop();
    // x -> y
    // x -> z
    // y -> x
    // y -> z
    // z -> x
    // z -> y
}
```

물통

https://www.acmicpc.net/problem/2251

```
// x -> y
ny += nx; nx = 0;
if (ny >= b) {
    nx = ny-b;
    ny = b;
}
if (!check[nx][ny]) {
    check[nx][ny] = true;
    q.push(make_pair(nx,ny));
    if (nx == 0) {
        ans[nz] = true;
    }
}
```

물통

https://www.acmicpc.net/problem/2251

- 소스: http://codeplus.codes/a400ad5d5dfd48cfb67d7556d1906c6f

모양 만들기

https://www.acmicpc.net/problem/16932

- 0과 1이 들어있는 N×M 크기의 배열에서 모양을 찾으려고 한다. ($2 \le N, M \le 1,000$)
- 모양은 연결되어 있는 1의 최대 개수
- 칸 하나의 수를 변경할 수 있다. 변경해서 만들 수 있는 모양의 최대 크기를 구하는 문제

모양 만들기

https://www.acmicpc.net/problem/16932

- 칸을 바꾸지 않는다면, DFS/BFS를 이용해서 1의 최대 개수를 구할 수 있다.
- 시간 복잡도: O(NM)
- 모든 칸을 바꿔보고 BFS를 수행하는 것은 시간이 매우 오래 걸린다.
- 바꿀 수 있는 칸의 수 = NM, BFS = O(NM)
- 시간 복잡도: O(NMNM)

모양 만들기

https://www.acmicpc.net/problem/16932

- BFS를 한 번 이용해서 모든 칸을 그룹짓고, 그룹의 크기를 미리 구해 놓는다.
- 그 다음, 0을 1로 바꾸는 경우만 고려해서, 인접한 네 칸의 그룹 정보를 이용해서 BFS 없이 계산해볼 수 있다.

모양 만들기

https://www.acmicpc.net/problem/16932

- 소스: http://codeplus.codes/cb5160064e014c7cbd0638abd23871fe

말이 되고픈 원숭이

https://www.acmicpc.net/problem/1600

- 격자판 위에 원숭이가 있다.
- 원숭이의 시작 칸에서 도착 칸으로 가는 최소 이동 동작의 횟수를 구하는 문제
- 원숭이는 인접한 4방향만 이동할 수 있고, 최대 K번 체스판의 나이트처럼 이동할 수 있다.

말이 되고픈 원숭이

https://www.acmicpc.net/problem/1600

- 벽 부수고 이동하기 2 문제와 같이 나이트처럼 이동을 한 횟수에 따라서, 정점을 나누어서 BFS를 구현할 수 있다.

말이 되고픈 원숭이

https://www.acmicpc.net/problem/1600

- 격자판 위에 원숭이가 있다.
- 원숭이의 시작 칸에서 도착 칸으로 가는 최소 이동 동작의 횟수를 구하는 문제
- 원숭이는 인접한 4방향만 이동할 수 있고, 최대 K번 체스판의 나이트처럼 이동할 수 있다.

말이 되고픈 원숭이

https://www.acmicpc.net/problem/1600

- 소스: http://codeplus.codes/63f37523a47c46d4904529a3f4c67283

아기 상어 2

https://www.acmicpc.net/problem/17086

- N×M 크기의 공간에 아기 상어 여러 마리가 있다. 한 칸에는 최대 1마리의 아기 상어가 있다.
- 어떤 칸의 안전 거리는 그 칸과 거리가 가장 가까운 아기 상어와의 거리이다.
- 거리는 지나야 하는 칸의 수이고, 이동은 인접한 8방향 가능 (대각선 포함)
- 안전 거리가 가장 큰 칸을 구하는 문제

아기 상어 2

https://www.acmicpc.net/problem/17086

- 각각의 칸에서 아기 상어까지 거리를 BFS로 계산할 수 있다.

아기 상어 2

https://www.acmicpc.net/problem/17086

- 소스: http://codeplus.codes/1282529da3784753be8d7dfa5dd02824

로봇 청소기

https://www.acmicpc.net/problem/4991

- 직사각형 모양의 방이 주어졌을 때
- 모든 더러운 칸을 깨끗한 칸으로 바꾸기 위해 필요한 최소 이동 횟수를 구하는 문제
- 너비, 높이 ≤ 20, 더러운 칸의 개수 ≤ 10

로봇 청소기

https://www.acmicpc.net/problem/4991

- 최소로 이동해야 한다.
- 직사각형 모양의 지도에서 임의의 두 칸 사이의 최소 이동 거리는 BFS로 구할 수 있다

로봇 청소기

https://www.acmicpc.net/problem/4991

- 더러운 칸의 개수가 10개 이하이기 때문에
- 모든 순열을 구하고 거리를 계산해서 해결한다.

로봇 청소기

https://www.acmicpc.net/problem/4991

- 소스: http://codeplus.codes/a7eaacbf4a064ec18ba587cf608e0fe0

거울 설치

https://www.acmicpc.net/problem/2151

- 문이 두 개 있을 때, 한 문에서 다른 문을 보기 위해 필요한 거울 개수의 최소값을 구하는 문제

*	*	*	#	*
*	.	!	.	*
*	!	.	!	*
*	.	!	.	*
*	#	*	*	*

거울 설치

https://www.acmicpc.net/problem/2151

- 문이 두 개 있을 때, 한 문에서 다른 문을 보기 위해 필요한 거울 개수의 최소값을 구하는 문제

*	*	*	#	*
*	!	!	!	*
*	.	!	!	*
*	!	.	!	*
*	#	*	*	*

거울 설치

https://www.acmicpc.net/problem/2151

- 각각의 거울마다 네 방향에 대해서 다음 거울을 미리 찾아야 한다.

*	*	*	#	*
*	!	!	!	*
*	.	!	!	*
*	!	.	!	*
*	#	*	*	*

거울 설치

https://www.acmicpc.net/problem/2151

- 각각의 거울마다 네 방향에 대해서 다음 거울을 미리 찾아야 한다.
- 그 다음, 각각의 거울만 가지고 BFS 탐색을 수행해야 한다.

*	*	*	#	*
*	!	!	!	*
*	.	!	!	*
*	!	.	!	*
*	#	*	*	*

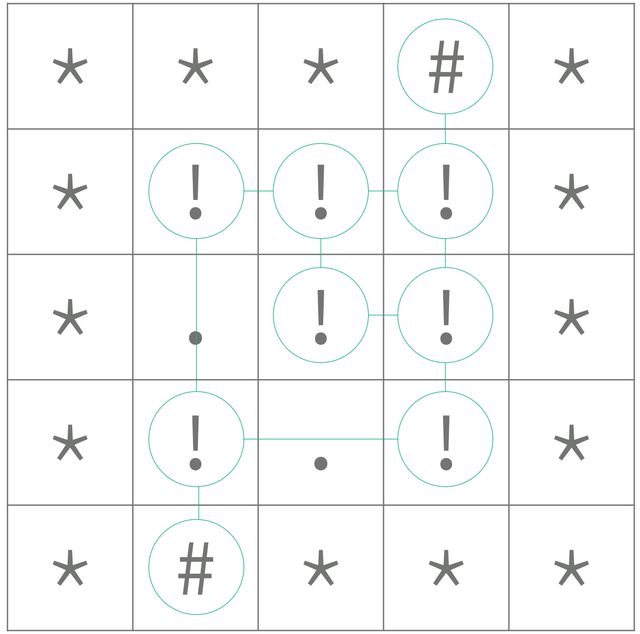
*	*	*	#	*
*	!	!	!	*
*	.	!	!	*
*	!	.	!	*
*	#	*	*	*

거울 설치

https://www.acmicpc.net/problem/2151

- 각각의 거울마다 네 방향에 대해서 다음 거울을 미리 모두 찾아야 한다.
- 그 다음, 각각의 거울만 가지고 BFS 탐색을 수행해야 한다.

*	*	*	#	*
*	!	!	!	*
*	.	!	!	*
*	!	.	!	*
*	#	*	*	*



거울 설치

https://www.acmicpc.net/problem/2151

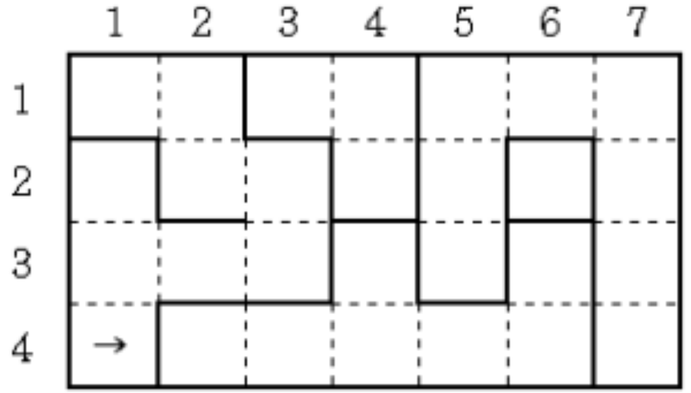
- 소스: http://codeplus.codes/204bc904b07e44a1ae2b9c5e89fd77da

성곽

https://www.acmicpc.net/problem/2234

- 성곽이 있다

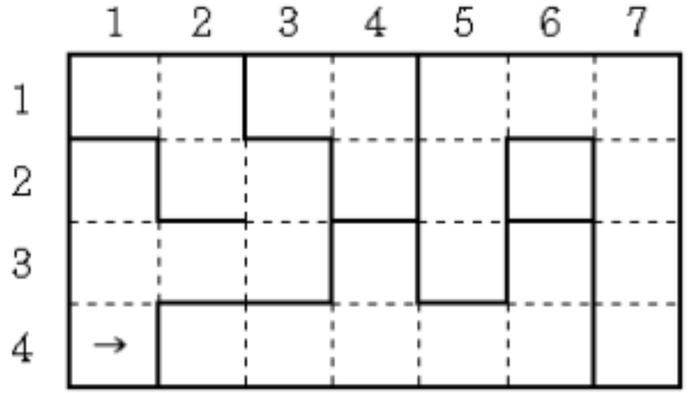
1. 방의 개수
2. 가장 넓은 방의 넓이
3. 하나의 벽을 제거해서 얻을 수 있는 가장 넓은 방의 크기



성곽

https://www.acmicpc.net/problem/2234

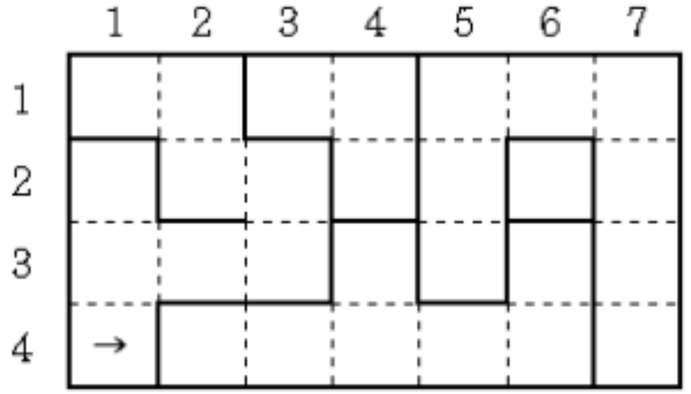
- 1, 2번 문제를 풀기 위해 BFS를 이용해서
- 각 칸의 방 번호와
- 각 방의 크기를 모두 구할 수 있다



성곽

https://www.acmicpc.net/problem/2234

- 3번 문제를 풀기 위해 각각의 벽을 하나씩 없애보고
- BFS를 돌리는 것은 너무 오래 시간이 걸린다
- 1, 2번 문제를 풀면서 얻은 정보를 이용해
- 모든 벽에 대해서, 각 벽을 제거했을 때 방이 합쳐지는지
- 그리고, 넓이는 몇 인지 구할 수 있다



성곽

https://www.acmicpc.net/problem/2234

- 소스: http://codeplus.codes/3008e95a555546ce9a03a99a86ffc300

새로운 하노이 탑

https://www.acmicpc.net/problem/12906

- 총 3가지 종류의 막대가 있다. 막대 A, B, C
- 각 막대에는 원판이 0개 이상 놓여져 있다.
- 모든 원판의 크기는 같고, 원판의 종류도 A, B, C이다.
- 한 번 이동: 한 막대의 가장 위에 있는 원판을 다른 막대의 가장 위로 옮기는 것
- 막대 A에는 원판 A만, 막대 B에는 원판 B만, 막대 C에는 원판 C만 있게 옮겨야 한다.
- 최소 이동 횟수를 구하는 문제
- $1 \le$ 모든 막대에 놓여져 있는 원판 개수의 합 ≤ 10

새로운 하노이 탑

https://www.acmicpc.net/problem/12906

```
1 B
1 C
1 A
```

- 원판 A를 막대 A로
- 원판 C를 막대 C로
- 원판 A를 막대 C로
- 원판 B를 막대 B로
- 원판 A를 막대 A로

새로운 하노이 탑

https://www.acmicpc.net/problem/12906

```
3 CBA
0

0
```

- 원판 A를 막대 C로
- 원판 B를 막대 B로
- 원판 A를 막대 B로
- 원판 C를 막대 C로
- 원판 A를 막대 A로

새로운 하노이 탑

https://www.acmicpc.net/problem/12906

- 막대의 개수가 3개, 원판의 총 개수가 10개밖에 안된다.
- 경우의 수가 많지 않아 BFS 탐색으로 모든 이동을 해본다.

새로운 하노이 탑

https://www.acmicpc.net/problem/12906

- 소스: http://codeplus.codes/4432990d578d4a48b16bd9e98e2f8ce7

연구소 2

https://www.acmicpc.net/problem/17141

- 크기가 N×N인 연구소가 있고, 빈 칸, 벽으로 이루어져 있다.
- 일부 빈 칸에는 바이러스를 놓을 수 있다.
- M개의 빈 칸에 바이러스를 놓으려고 한다. 바이러스는 상하좌우로 인접한 모든 빈 칸으로 동시에 복제되고, 1초가 걸린다.
- 모든 빈 칸에 바이러스를 퍼뜨리는 최소 시간을 구하는 문제

연구소 2

https://www.acmicpc.net/problem/17141

```
2 0 0 0 1 1 0
0 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0
0 0 0 2 0 1 1
0 1 0 0 0 0 0
2 1 0 0 0 0 2
```

- 0: 빈 칸, 1: 벽, 2: 바이러스를 놓을 수 있는 칸

연구소 2

https://www.acmicpc.net/problem/17141

```
6 6 5 4 - - 2
5 6 - 3 - 0 1
4 - - 2 - 1 2
3 - 2 1 2 2 3
2 2 1 0 1 - -
1 - 2 1 2 3 4
0 - 3 2 3 4 5
```

- M = 3인 경우 방법 하나
- 바이러스를 놓은 위치: 0, 벽: -, 정수: 바이러스가 퍼지는 시간

연구소 2

https://www.acmicpc.net/problem/17141

```
0 1 2 3 - - 2
1 2 - 3 - 0 1
2 - - 2 - 1 2
3 - 2 1 2 2 3
3 2 1 0 1 - -
4 - 2 1 2 3 4
5 - 3 2 3 4 5
```

- M = 3인 경우 최소 시간이 걸리는 방법
- 바이러스를 놓은 위치: 0, 벽: -, 정수: 바이러스가 퍼지는 시간

연구소 2

https://www.acmicpc.net/problem/17141

- N ≤ 50
- M ≤ 10
- M ≤ 바이러스를 놓을 수 있는 칸의 수 ≤ 10

연구소 2

https://www.acmicpc.net/problem/17141

- $N \leq 50$
- $M \leq 10$
- $M \leq$ 바이러스를 놓을 수 있는 칸의 수 ≤ 10
- 바이러스를 놓고, 모든 칸으로 퍼뜨리는 시간을 계산해본다.
- 바이러스를 놓을 수 있는 방법의 수: 2^{10}
- 모든 칸으로 퍼뜨리는 시간의 계산: BFS를 이용해 $O(N^2)$

연구소 2

https://www.acmicpc.net/problem/17141

- 소스: http://codeplus.codes/1ef683b7d7a34dd69622d45afe4883e2

연구소 3

https://www.acmicpc.net/problem/17142

- 크기가 N×N인 연구소가 있고, 빈 칸, 벽, 바이러스이다.
- 바이러스는 활성/비활성 상태가 있다.
- 가장 처음에 모든 바이러스는 비활성 상태이고, 바이러스 M개를 활성 상태로 바꾸려고 한다.
- 활성 바이러스는 상하좌우로 인접한 모든 빈 칸으로 동시에 복제되고, 1초가 걸린다.
- 비활성 바이러스가 있는 곳으로 활성 바이러스가 이동하면 비활성이 활성으로 변한다.
- 모든 빈 칸에 바이러스를 퍼뜨리는 최소 시간을 구하는 문제

연구소 3

https://www.acmicpc.net/problem/17142

```
2 0 0 0 1 1 0
0 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0
0 0 0 2 0 1 1
0 1 0 0 0 0 0
2 1 0 0 0 0 2
```

- 0: 빈 칸, 1: 벽, 2: 비활성 바이러스

연구소 3

https://www.acmicpc.net/problem/17142

```
* 6 5 4 - - 2
5 6 - 3 - 0 1
4 - - 2 - 1 2
3 - 2 1 2 2 3
2 2 1 0 1 - -
1 - 2 1 2 3 4
0 - 3 2 3 4 *
```

- M = 3인 경우 방법 하나
- 활성 바이러스: 0, 비활성 바이러스: *, 벽: -, 정수: 바이러스가 퍼지는 시간

연구소 3

https://www.acmicpc.net/problem/17142

```
0 1 2 3 - - 2
1 2 - 3 - 0 1
2 - - 2 - 1 2
3 - 2 1 2 2 3
3 2 1 0 1 - -
4 - 2 1 2 3 4
* - 3 2 3 4 *
```

- M = 3인 경우 최소 시간이 걸리는 방법
- 활성 바이러스: 0, 비활성 바이러스: *, 벽: -, 정수: 바이러스가 퍼지는 시간

연구소 3

https://www.acmicpc.net/problem/17142

- N ≤ 50
- M ≤ 10
- M ≤ 비활성 바이러스의 수 ≤ 10

연구소 3

https://www.acmicpc.net/problem/17142

- $N \leq 50$
- $M \leq 10$
- $M \leq$ 비활성 바이러스의 수 ≤ 10
- 바이러스를 선택하고, 모든 칸으로 퍼뜨리는 시간을 계산해본다.
- 바이러스를 선택할 수 있는 방법의 수: 2^{10}
- 모든 칸으로 퍼뜨리는 시간의 계산: BFS를 이용해 $O(N^2)$

연구소 3

https://www.acmicpc.net/problem/17142

- 연구소 2와 거의 같은 문제이기 때문에, 일부 구현만 변경하면 된다.
- 바이러스를 놓을 수 있는 칸 → 비활성 바이러스
- 바이러스를 놓는다 → 비활성 바이러스를 활성 상태로 변경한다.
- M개를 활성 바이러스로 변경한 이후 비활성 바이러스는 빈 칸과 같은 의미를 갖게 된다.

연구소 3

https://www.acmicpc.net/problem/17142

- BFS에서는 바이러스가 빈 칸과 같은 의미를 갖지만, 정답을 구할 때는 아니다.
- 정답을 구할 때는 빈 칸까지 가는 거리의 최댓값만 구해야 한다.

연구소 3

https://www.acmicpc.net/problem/17142

- 연구소 2의 소스와 세 부분이 다르다.
- 차이 1: 연구소 2는 바이러스를 놓을 수 있는 칸을 후보에 넣고 빈 칸으로 변경했다. 연구소 3은 변경하지 않는다.
- 차이 2: 차이 1 때문에, 바이러스를 놓을 칸을 결정할 때, 바이러스를 놓지 않는 경우 (활성 바이러스로 바꾸지 않는 경우)를 연구소 2는 0으로, 연구소 3은 2로 설정한다.
- 차이 3: 정답을 구할 때, 연구소 2는 벽이 아닌 경우 (`a[i][j] != 1`) 거리의 최댓값을 구하고, 연구소 3은 빈 칸인 경우 (`a[i][j] == 0`) 거리의 최댓값을 구한다.

연구소 3

https://www.acmicpc.net/problem/17142

- 소스: http://codeplus.codes/2f894cc24c7a443a97d13796cd2be563