미로 탐색

https://www.acmicpc.net/problem/2178

- 가중치가 없는 최단거리를 구하는 문제로 BFS로 구현할 수 있다.
- 입력 받은 행렬 외, visited 정보를 저장하는 행렬, 이동 거리를 저장한 행렬을 이용하여 최단 거리를 구한다.

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1					

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3	4	5			
4	5				

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2					

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3	4	5	6		
4	5	6			

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3					

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3		7		
3	4	5	6	7	
4	5	6	7		

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3	4				
4					

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

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

Gaussian

미로 탐색

https://www.acmicpc.net/problem/2178

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

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

- 탐색 시 배열의 범위를 벗어나지 않기 위해 0행과 0열은 참조하지 않는다.

```
queue<ii> q;
q.push(ii(1, 1));
dist[1][1] = 1;
visited[1][1] = true;

while (!q.empty()) {
    ii here = q.front();
    q.pop();

    int y = here.first;
    int x = here.second;

    for (int i = 0; i < 4; ++i) {
        int ny = y + dy[i];
        int nx = x + dx[i];

        if (!visited[ny][nx] && A[ny][nx] == 1) {
            visited[ny][nx] = true;
            q.push(ii(ny, nx));
            dist[ny][nx] = dist[y][x] + 1;
        }
    }
}
printf("%d\n", dist[N][M]);
```