1. "타일점프" 정답코드

```
class Solution {
        public int solution(int[] nums){
                int n = nums.length;
                int[] ch = new int[n];
                Queue<Integer> Q = new LinkedList<>();
                Q.offer(0);
                ch[0] = 1;
                int L = 0;
                while(!Q.isEmpty()){
                        int len = Q.size();
                        for(int i = 0; i < len; i++){
                                int x = Q.poll();
                                for(int j = 1; j <= nums[x]; j++){
                                        int nx = x+j;
                                        if(nx == n-1) return L + 1;
                                        if(nx < n && ch[nx] == 0){
                                                ch[nx] = 1;
                                                Q.offer(nx);
                                        }
                                }
                        }
                        L++;
                }
                return -1;
        }
}
```

2. "집으로 이동" 정답코드

```
class Solution {
	public int solution(int[] pool, int a, int b, int home){
		int[][] ch = new int[2][10001];
		for(int x : pool){
			ch[0][x] = 1;
			ch[1][x] = 1;
		}
		Queue<int[]> Q = new LinkedList<>();
		ch[0][0] = 1;
		ch[1][0] = 1;
		Q.offer(new int[]{0, 0});
		int L = 0;
		while(!Q.isEmpty()){
			int len = Q.size();
			for(int i = 0; i < len; i++){
				int[] cur = Q.poll();
				if(cur[0] == home) return L;
				int nx = cur[0] + a;
				if(nx <= 10001 && ch[0][nx] == 0){
					ch[0][nx] = 1;
					Q.offer(new int[]{nx, 0});
				}
				nx = cur[0] - b;
				if(nx >= 0 && ch[1][nx] == 0 && cur[1] == 0){
					ch[1][nx] = 1;
					Q.offer(new int[]{nx, 1});
				}
			}
			L++;
		}
		return -1;
	}
}
```

▶▶ Comment :

```
nx = cur[0] - b;
if(nx >= 0 && ch[1][nx] == 0 && cur[1] == 0)
```

뒤로 점프는 현지 지점을 오는데 앞으로 점프해서 온 경우 cur[1] == 0 일 때 뒤로 점프를 합니다.

3. "송아지를 잡자" 정답코드

```
class Solution {
        public int solution(int s, int e){
                int[][] ch = new int[2][200001];
                Queue<Integer> Q = new LinkedList<>();
                ch[0][s] = 1;
                Q.offer(s);
                int L = 0;
                while(!Q.isEmpty()){
                        int len = Q.size();
                        L++;
                        for(int i = 0; i < len; i++){
                                int x = Q.poll();
                                for(int nx : new int[]{x-1, x+1, x*2}){
                                        if(nx >= 0 && nx <= 200000 && ch[L%2][nx] == 0){
                                                ch[L%2][nx] = 1;
                                                Q.offer(nx);
                                        }
                                }
                        }
                        e = e + L;
                        if(e > 200000) return -1;
                        if(ch[L%2][e] == 1) return L;
                }
                return -1;
        }
}
```

▶▶ Comment :

if(ch[L%2][e] == 1) return L;

홀수 레벨 ch[1][nx]에 체크된 지점은 현수가 현재의 홀수 레벨에서 모두 방문할 수 있는 지점이고,

짝수 레벨 ch[0][nx]에 체크된 지점은 현수가 현재의 짝수 레벨에서 모두 방문할 수 있는 지점입니다.

4. "미로의 최단거리 통로(L 탐색)" 정답코드

```
class Solution {
    public int solution(int[][] board){
        int[] dx = {-1, 0, 1, 0};
        int[] dy = {0, 1, 0, -1};
        Queue<int[]> Q = new LinkedList<>();
        int[][] dist = new int[7][7];
        Q.offer(new int[]{0, 0});
        int L = 0;
        while(!Q.isEmpty()){
            L++;
            int len = Q.size();
            for(int i = 0; i < len; i++){
                int[] p = Q.poll();
                for(int k = 0; k < 4; k++){
                        int nx=p[0]+dx[k];
                        int ny=p[1]+dy[k];
                        if(nx >= 0 && nx < 7 && ny >= 0 && ny < 7 && board[nx][ny] == 0){
                                board[nx][ny] = 1;
                                Q.offer(new int[]{nx, ny});
                                dist[nx][ny] = L;
                        }
                }
            }
        }
        if(dist[6][6]==0) return -1;
        else return dist[6][6];
    }
}
```

5. "집을 짓자" 정답코드

```
class Solution {
    public int solution(int[][] board){
        int answer = 0;
        int[] dx = {-1, 0, 1, 0};
        int[] dy = {0, 1, 0, -1};
        int n = board.length;
        int[][] dist = new int[n][n];
        Queue<int[]> Q = new LinkedList<>();
        int emptyLand = 0;
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                if(board[i][j] == 1){
                    answer = Integer.MAX_VALUE;
                    Q.offer(new int[]{i, j});
                    int L = 0;
                    while(!Q.isEmpty()){
                        L++;
                        int len = Q.size();
                        for(int r = 0; r < len; r++){
                            int[] cur = Q.poll();
                            for(int k = 0; k < 4; k++){
                                int nx = cur[0] + dx[k];
                                int ny = cur[1] + dy[k];
                                if(nx >= 0 && nx < n && ny >= 0 && ny < n && board[nx][ny] == emptyLand){
                                    board[nx][ny]--;
                                    dist[nx][ny] += L;
                                    Q.offer(new int[]{nx, ny});
                                    answer = Math.min(answer, dist[nx][ny]);
                                }
                            }
                        }
                    }
                    emptyLand--;
                }
            }
        }
        return answer == Integer.MAX_VALUE ? -1 : answer;
    }
}
```

6. "숲속의 기사" 정답코드

```
class Solution {
    public int solution(int[][] board){
        int answer = Integer.MAX_VALUE;
        int[] dx = {-1, 0, 1, 0};
        int[] dy = {0, 1, 0, -1};
        int n = board.length;
        int m = board[0].length;
        int[][] dist = new int[n][m];
        Queue<int[]> Q = new LinkedList<>();
        for(int i = 0; i < n; i++){
            for(int j = 0; j < m; j++){
                if(board[i][j] == 2 || board[i][j] == 3){
                    Q.offer(new int[]{i, j});
                    int[][] ch = new int[n][m];
                    ch[i][j] = 1;
                    int L = 0;
                    while(!Q.isEmpty()){
                        L++;
                        int len = Q.size();
                        for(int r = 0; r < len; r++){
                            int[] cur = Q.poll();
                            for(int k = 0; k < 4; k++){
                                int nx = cur[0] + dx[k];
                                int ny = cur[1] + dy[k];
                                if(nx >= 0 && nx < n && ny >= 0 && ny < m && board[nx][ny] != 1){
                                    if(ch[nx][ny] == 0){
                                        ch[nx][ny] = 1;
                                        dist[nx][ny] += L;
                                        Q.offer(new int[]{nx, ny});
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        for(int i = 0; i < n; i++){
            for(int j = 0; j < m; j++){
                if(board[i][j] == 4 && dist[i][j] > 0){
                    answer = Math.min(answer, dist[i][j]);
                }
            }
        }
        return answer;
    }
}
```