1. "최소 비행료" 정답코드

```
class Solution {
        public int solution(int n, int[][] flights, int s, int e, int k){
                ArrayList<ArrayList<int[]>> graph = new ArrayList<>();
                for(int i = 0; i < n; i++){
                        graph.add(new ArrayList<int[]>());
                }
                int[] costs = new int[n];
                Arrays.fill(costs, 1000000000);
                for(int[] x : flights){
                        graph.get(x[0]).add(new int[]{x[1], x[2]});
                }
                Queue<int[]> Q = new LinkedList<>();
                Q.offer(new int[]{s, 0});
                costs[s] = 0;
                int L = 0;
                while(!Q.isEmpty()){
                        int len = Q.size();
                        for(int i = 0; i < len; i++){
                                int[] p = Q.poll();
                                int now = p[0];
                                int nowcost = p[1];
                                for(int[] x : graph.get(now)){
                                        int next = x[0];
                                        int cost = x[1];
                                        if(nowcost + cost < costs[next]){
                                                costs[next] = nowcost + cost;
                                                Q.offer(new int[]{next, costs[next]});
                                        }
                                }
                        }
                        L++;
                        if(L > k) break;
                }
                if(costs[e] == 1000000000) return -1;
                else return costs[e];
        }
}
```

▶▶ Comment :

레벨 변수인 L 값을 비행기를 갈아탄 횟수로 대응해서 코드를 구현합니다.

2. "최소 환승 경로" 정답코드

```
class Solution {
        public int solution(int[][] routes, int s, int e){
                int answer = 0;
                HashMap<Integer, HashSet<Integer>> graph = new HashMap<>();
                int n = routes.length;
                for(int i = 0; i < n; i++){
                        for(int x : routes[i]){
                                graph.putIfAbsent(x, new HashSet<Integer>());
                                graph.get(x).add(i);
                        }
                }
                Queue<Integer> Q = new LinkedList<>();
                int[] ch = new int[n];
                Q.offer(s);
                int L = 0;
                while(!Q.isEmpty()){
                        int len = Q.size();
                        for(int i = 0; i < len; i++){
                                int curStop = Q.poll();
                                for(int line : graph.get(curStop)){
                                        if(ch[line] == 1) continue;
                                        ch[line] = 1;
                                        for(int stop : routes[line]){
                                                if(stop == e) return L;
                                                Q.offer(stop);
                                        }
                                }
                        }
                        L++;
                }
                return -1;
        }
}
```

▶▶ Comment :

레벨 변수인 L 값을 지하철을 환승한 횟수로 대응해서 코드를 구현합니다.

3. "벽 허물기" 정답코드

```
class Solution {
    public int solution(int[][] board) {
        int[] dx = {-1, 0, 1, 0};
        int[] dy = {0, 1, 0, -1};
        int n = board.length;
        int m = board[0].length;
        int[][] cost = new int[n][m];
        for(int i = 0; i < n; i++) Arrays.fill(cost[i], Integer.MAX_VALUE);
        cost[0][0] = 0;
        PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> a[2] - b[2]);
        pq.add(new int[]{0, 0, 0});
        while(!pq.isEmpty()) {
            int[] cur = pq.poll();
            if(cur[2] > cost[cur[0]][cur[1]]) continue;
            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) continue;
                if(board[nx][ny] == 0 && cost[nx][ny] > cur[2]) {
                    cost[nx][ny] = cur[2];
                    pq.offer(new int[]{nx, ny, cur[2]});
                }
                else if(board[nx][ny] == 1 && cost[nx][ny] > cur[2] + 1){
                    cost[nx][ny] = cur[2] + 1;
                    pq.offer(new int[]{nx, ny, cur[2] + 1});
                }
            }
        }
        return cost[n - 1][m - 1];
    }
}
```

4. "방향 바꾸기" 정답코드

```
class Solution {
    public int solution(int[][] board) {
        int[] dx = {0, 0, 1, -1};
        int[] dy = {1, -1, 0, 0};
        int n = board.length;
        int m = board[0].length;
        int[][] cost = new int[n][m];
        for(int i = 0; i < n; i++) Arrays.fill(cost[i], Integer.MAX_VALUE);
        cost[0][0] = 0;
        PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> a[2] - b[2]);
        pq.add(new int[]{0, 0, 0});
        while(!pq.isEmpty()) {
            int[] cur = pq.poll();
            int dir = board[cur[0]][cur[1]] - 1;
            if(cur[2] > cost[cur[0]][cur[1]]) continue;
            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) continue;
                if(k == dir && cost[nx][ny] > cur[2]) {
                    cost[nx][ny] = cur[2];
                    pq.offer(new int[]{nx, ny, cur[2]});
                }
                else {
                        if (cost[nx][ny] > cur[2] + 1) {
                                cost[nx][ny] = cur[2] + 1;
                                pq.offer(new int[]{nx, ny, cur[2] + 1});
                        }
                }
            }
        }
        return cost[n - 1][m - 1];
    }
}
```

▶▶ Comment :

if(k == dir && cost[nx][ny] > cur[2])

board에 있는 표시된 방향 dir과 현재 이동하려고 하는 방향 k가 같다면 (nx, ny)지점으로 cur[2]값으로 이동 가능합니다.

5. "공 굴리기" 정답코드

```
class Solution {
        public int solution(int[][] board, int[] s, int[] e){
                int n = board.length;
                int m = board[0].length;
                int[][] cost = new int[n][m];
                for(int i = 0; i < n; i++) Arrays.fill(cost[i], Integer.MAX_VALUE);
                PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> a[2] - b[2]);
                pq.add(new int[]{s[0], s[1], 0});
                cost[s[0]][s[1]] = 0;
                while(!pq.isEmpty()) {
                    int[] cur = pq.poll();
                    if(cur[2] > cost[cur[0]][cur[1]]) continue;
                    for(int[] dir : new int[][]{{-1, 0}, {0, 1}, {1, 0}, {0, -1}}){
                        int nx = cur[0];
                        int ny = cur[1];
                        int len = cur[2];
                        while(nx>= 0 && nx < n&& ny >= 0 && ny < m && board[nx][ny] == 0){
                                nx += dir[0];
                                ny += dir[1];
                                len ++;
                        }
                        nx -= dir[0];
                        ny -= dir[1];
                        len --;
                        if(cost[nx][ny] > len){
                                cost[nx][ny] = len;
                                pq.add(new int[]{nx, ny, len});
                        }
                    }
                }
                if(cost[e[0]][e[1]] == Integer.MAX_VALUE) return -1;
                else return cost[e[0]][e[1]];
        }
}
```

▶▶ Comment :

```
x -= dir[0];
y -= dir[1];
len --;
```

벽은 만나 멈춘 x, y지점은 벽이 있는 지점이기 때문에 공이 굴러간 방향의 반대방향으로 한 칸 뒤로 이동합니다.

6. "교육 과정" 정답코드

```
class Solution {
        public String[] solution(String[] subjects, String[] course){
                int n = subjects.length;
                HashMap<String, Integer> node = new HashMap<>();
                for(int i = 0; i < n; i++) node.put(subjects[i], i);
                ArrayList<ArrayList<Integer>> graph = new ArrayList<>();
                for(int i = 0; i < n; i++){
                        graph.add(new ArrayList<Integer>());
                }
                int[] indegree = new int[n];
                for(String x : course){
                        int a = node.get(x.split(" ")[0]);
                        int b = node.get(x.split(" ")[1]);
                        graph.get(b).add(a);
                        indegree[a]++;
                }
                ArrayList<Integer> order = new ArrayList<>();
                Queue<Integer> queue = new LinkedList<>();
                for(int i = 0; i < n; i++) {
                        if(indegree[i] == 0) queue.offer(i);
                }
                while(!queue.isEmpty()){
                        int pre = queue.poll();
                        order.add(pre);
                        for(int x : graph.get(pre)){
                                indegree[x]--;
                                if(indegree[x] == 0){
                                        queue.offer(x);
                                }
                        }
                }
                String[] answer = new String[n];
                System.out.println(order);
                for(int i = 0; i < n; i++){
                        answer[i] = subjects[order.get(i)];
                }
                return answer;
        }
}
```

▶▶ Comment :

indegree[a]++; 방향그래프에서 a정점의 진입차수를 계산하는 코드입니다.