

# 그래프 탐색 스터디

2026년 2월 27일

# 농장 관리(1245)

## 문제

농부 민식이가 관리하는 농장은  $N \times M$  격자로 이루어져 있다. 민식이는 농장을 관리하기 위해 산봉우리마다 경비원을 배치하려 한다. 이를 위해 농장에 산봉우리가 총 몇 개 있는지를 세는 것이 문제다.

산봉우리의 정의는 다음과 같다. 산봉우리는 같은 높이를 가지는 하나의 격자 혹은 인접한 격자들의 집합으로 이루어져 있다. 여기서 “인접하다”의 정의는 X좌표 차이와 Y좌표 차이가 모두 1 이하인 경우이다. 또한 산봉우리와 인접한 격자는 모두 산봉우리의 높이보다 작아야 한다.

问题是 격자 내에 산봉우리의 개수가 총 몇 개인지 구하는 것이다.

## 입력

첫째 줄에 정수  $N(1 < N \leq 100)$ ,  $M(1 < M \leq 70)$ 이 주어진다. 둘째 줄부터  $N + 1$  번째 줄까지 각 줄마다 격자의 높이를 의미하는  $M$  개의 정수가 입력된다. 격자의 높이는 500보다 작거나 같은 음이 아닌 정수이다.

## 출력

첫째 줄에 산봉우리의 개수를 출력한다.

## 예제 격자

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

# 김시온 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Sion {
5     static int n, m;
6     static int[][] a;
7     static boolean[][] b;
8     static final int[] dx = { -1, -1, -1, 0, 0, 1, 1, 1 };
9     static final int[] dy = { -1, 0, 1, -1, 1, -1, 0, 1 };
10
11     static boolean inRange(int x, int y) {
12         return 0 <= x && x < n && 0 <= y && y < m;
13     }
14
15     static boolean bfs(int sx, int sy) {
16         Queue<int[]> q = new LinkedList<int[]>();
17         q.add(new int[] { sx, sy });
18         b[sx][sy] = true;
19         boolean f = true;
20         while (!q.isEmpty()) {
21             int[] t = q.poll();
22             int x = t[0], y = t[1];
23
24             for (int i = 0; i < 8; i++) {
25                 int nx = x + dx[i], ny = y + dy[i];
26                 // 8방향으로 인접한 칸 중에서 높이가 같은 칸으로 간선이 이어져 있다.
27                 if (inRange(nx, ny) && a[x][y] == a[nx][ny] && !b[nx][ny]) {
28                     q.add(new int[] { nx, ny });
29                     b[nx][ny] = true;
30                 }
31             }
32         }
33     }
34 }
```

# 김시온 소스 코드 2

```
31         }
32
33     for (int i = 0; i < 8; i++) {
34         int nx = x + dx[i], ny = y + dy[i];
35         // 8방향으로 인접한 칸 중에서 높이가 더 높은 칸이 있는지 확인.
36         if (inRange(nx, ny) && a[x][y] < a[nx][ny]) {
37             f = false;
38         }
39     }
40 }
41 return f;
42 }
43
44 public static void main(String[] args) throws IOException {
45     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
46     StringTokenizer st = new StringTokenizer(br.readLine());
47     Sion.n = Integer.parseInt(st.nextToken());
48     Sion.m = Integer.parseInt(st.nextToken());
49     Sion.a = new int[n][m];
50     for (int x = 0; x < n; x++) {
51         st = new StringTokenizer(br.readLine());
52         for (int y = 0; y < m; y++) {
53             a[x][y] = Integer.parseInt(st.nextToken());
54         }
55     }
56     Sion.b = new boolean[n][m];
57
58     int c = 0;
59     for (int x = 0; x < n; x++) {
```

# 김시온 소스 코드 3

```
60         for (int y = 0; y < m; y++) {
61             if (!b[x][y]) {
62                 if (bfs(x, y)) {
63                     c++;
64                 }
65             }
66         }
67     }
68     System.out.println(c);
69 }
70 }
71 }
```

# 김준형 소스 코드 1

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Junhyeong {
5     static class Cord {
6         int y;
7         int x;
8         int h;
9
10        public Cord(int y, int x, int h) {
11            super();
12            this.y = y;
13            this.x = x;
14            this.h = h;
15        }
16    }
17
18    static int n, m, ans;
19    static int[][] mountains;
20    static int[] dx = { 0, 1, 1, 1, 0, -1, -1, -1 };
21    static int[] dy = { -1, -1, 0, 1, 1, 1, 0, -1 };
22    static Queue<Cord> que = new ArrayDeque<>();
23    static boolean[][] visited;
24
25    public static void main(String[] args) throws IOException {
26        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
27        StringTokenizer st = new StringTokenizer(br.readLine());
28        n = Integer.parseInt(st.nextToken());
29        m = Integer.parseInt(st.nextToken());
30        mountains = new int[n][m];
```

# 김준형 소스 코드 2

```
31     visited = new boolean[n][m];
32
33     for (int i = 0; i < n; i++) {
34         st = new StringTokenizer(br.readLine());
35         for (int j = 0; j < m; j++) {
36             mountains[i][j] = Integer.parseInt(st.nextToken());
37         }
38     }
39
40     for (int i = 0; i < n; i++) {
41         for (int j = 0; j < m; j++) {
42             if (visited[i][j])
43                 continue;
44
45             boolean success = true;
46             que.offer(new Cord(i, j, mountains[i][j]));
47             visited[i][j] = true;
48
49             while (!que.isEmpty()) {
50                 Cord cur = que.poll();
51                 int curH = cur.h;
52
53                 for (int r = 0; r < 8; r++) {
54                     int nextY = cur.y + dy[r], nextX = cur.x + dx[r];
55                     if (nextY < 0 || nextX < 0 || nextY >= n || nextX >= m ||
56                         (visited[nextY][nextX] && mountains[nextY][nextX]
57                          == curH)) {
58                         continue;
59                     }
60                 }
61             }
62         }
63     }
64 }
```

# 김준형 소스 코드 3

```
59             int nextH = mountains[nextY][nextX];
60             if (curH < nextH) {
61                 success = false;
62             } else if (curH == nextH) {
63                 que.offer(new Cord(nextY, nextX, nextH));
64                 visited[nextY][nextX] = true;
65             }
66         }
67     }
68 }
69
70     if (success) {
71         ans++;
72     }
73 }
74 }
75 System.out.print(ans);
76 }
77 }
```

# 정의찬 소스 코드 1

```
1 import java.io.*;
2 import java.util.*;
3
4 class Uichan {
5     static int[] dy = { -1, -1, 0, 1, 1, 1, 0, -1 };
6     static int[] dx = { 0, 1, 1, 1, 0, -1, -1, -1 };
7     static int[][] map;
8
9     public static void main(String[] args) throws NumberFormatException,
10        IOException {
11         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12         StringBuilder sb = new StringBuilder();
13         StringTokenizer st = new StringTokenizer(br.readLine());
14         int n = Integer.parseInt(st.nextToken());
15         int m = Integer.parseInt(st.nextToken());
16         map = new int[n][m];
17
18         for (int i = 0; i < n; i++) {
19             st = new StringTokenizer(br.readLine());
20             for (int j = 0; j < m; j++) {
21                 map[i][j] = Integer.parseInt(st.nextToken());
22             }
23
24             int answer = 0;
25             Set<Integer> checked = new HashSet<>();
26             Set<Integer> temp;
27
28             for (int i = 0; i < n; i++) {
29                 Loop: for (int j = 0; j < m; j++) {
```

## 정의찬 소스 코드 2

```
30         if (checked.contains(i * 100 + j))
31             continue;
32
33         temp = new HashSet<>();
34         Queue<int[]> q = new LinkedList<>();
35         q.add(new int[] { i, j });
36         temp.add(i * 100 + j);
37
38         while (!q.isEmpty()) {
39             int[] current = q.poll();
40             for (int k = 0; k < 8; k++) {
41                 int nY = current[0] + dy[k];
42                 int nX = current[1] + dx[k];
43                 if (nY >= 0 && nY < m)
44                     && nX >= 0 && nX < m) {
45                     // 본인보다 낮은지검사
46                     if (map[current[0]][current[1]] < map[nY][nX])
47                         continue Loop;
48                     // 같은높이라면 큐 추가
49                     if (map[current[0]][current[1]] == map[nY][nX] &&
50                         !temp.contains(nY * 100 + nX)) {
51                         q.add(new int[] { nY, nX });
52                         temp.add(nY * 100 + nX);
53                     }
54                 }
55             }
56         }
57
58         checked.addAll(temp);
```

## 정의찬 소스 코드 3

```
58             answer++;
59         }
60     }
61
62     System.out.println(answer);
63 }
64 }
```

# 서민종 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Minjong {
5     static int N, M;
6     static int[][] farm;
7     static int[][] around = { { 1, 0 }, { 1, 1 }, { 0, 1 }, { -1, 1 }, { -1, 0 },
8     ↪ { -1, -1 }, { 0, -1 }, { 1, -1 } };
9     static boolean[][] checked;
10
11    public static void main(String[] args) throws IOException {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13        BufferedWriter bw = new BufferedWriter(new
14            OutputStreamWriter(System.out));
15        StringTokenizer st = new StringTokenizer(br.readLine());
16        N = Integer.parseInt(st.nextToken());
17        M = Integer.parseInt(st.nextToken());
18        farm = new int[N][M];
19        checked = new boolean[N][M];
20        for (int r = 0; r < N; r++) {
21            st = new StringTokenizer(br.readLine());
22            for (int c = 0; c < M; c++) {
23                farm[r][c] = Integer.parseInt(st.nextToken());
24            }
25        }
26        int answer = 0;
27        for (int r = 0; r < N; r++) {
28            for (int c = 0; c < M; c++) {
29                if (!checked[r][c]) {
30                    if (BFS(r, c)) {
```

# 서민종 소스 코드 2

```
29                     answer++;
30                 }
31             }
32         }
33     }
34     bw.write(answer + "\n");
35     bw.flush();
36 }
37
38 public static boolean BFS(int startR, int startC) {
39     boolean[][] visited = new boolean[N][M];
40     int location = startR * 100 + startC;
41     Queue<Integer> queue = new LinkedList<>();
42     queue.offer(location);
43     visited[startR][startC] = true;
44     checked[startR][startC] = true;
45     while (!queue.isEmpty()) {
46         location = queue.poll();
47         int r = location / 100;
48         int c = location % 100;
49         for (int i = 0; i < 8; i++) {
50             int nr = r + around[i][0];
51             int nc = c + around[i][1];
52             int nloc = nr * 100 + nc;
53             if (checkRange(nr, nc) && !visited[nr][nc]) {
54                 if (farm[nr][nc] == farm[startR][startC]) {
55                     queue.offer(nloc);
56                     visited[nr][nc] = true;
57                     checked[nr][nc] = true;
```

## 서민종 소스 코드 3

```
58             } else if (farm[nr][nc] < farm[startR][startC]) {
59                 visited[nr][nc] = true;
60             } else {
61                 return false;
62             }
63         }
64     }
65 }
66 return true;
67 }
68
69 public static boolean checkRange(int r, int c) {
70     return r >= 0 && r < N && c >= 0 && c < M;
71 }
72 }
```

# 손현준 소스 코드 1

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import java.util.ArrayDeque;
5 import java.util.Queue;
6 import java.util.StringTokenizer;
7
8 public class Hyeonjun {
9     // 1 2 3
10    // 8 x 4
11    // 7 6 5
12    static int[] dr = { -1, -1, -1, 0, 1, 1, 1, 0 };
13    static int[] dc = { -1, 0, 1, 1, 1, 0, -1, -1 };
14    static int N, M;
15    static int[][] map;
16    static boolean[][] isPeek;
17    static boolean[][] visited;
18    static int peekCnt;
19
20    public static void main(String[] args) throws IOException {
21        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
22        StringTokenizer st;
23
24        // 입력 받기
25        st = new StringTokenizer(br.readLine(), " ");
26        N = Integer.parseInt(st.nextToken());
27        M = Integer.parseInt(st.nextToken());
28        map = new int[N][M];
29        for (int r = 0; r < N; r++) {
30            st = new StringTokenizer(br.readLine(), " ");
```

## 손현준 소스 코드 2

```
31         for (int c = 0; c < M; c++) {
32             map[r][c] = Integer.parseInt(st.nextToken());
33         }
34     }
35
36     // 산 봉우리 확인
37     isPeek = new boolean[N][M];
38     peekCnt = 0;
39     for (int r = 0; r < N; r++) {
40         for (int c = 0; c < M; c++) {
41             if (isPeek[r][c])
42                 continue;
43
44             int now = map[r][c];
45             int state = -1; // -1: 봉우리, 0: 높이 같은거 있음, 1: 높이 큰거 있음
46             // 8방향 확인
47             for (int d = 0; d < 8; d++) {
48                 int nr = r + dr[d];
49                 int nc = c + dc[d];
50                 if (nr < 0 || nr >= N || nc < 0 || nc >= M)
51                     continue;
52
53                 if (now < map[nr][nc]) {
54                     state = 1;
55                     break;
56                 } else if (now == map[nr][nc])
57                     state = 0;
58             }
59         }
```

## 손현준 소스 코드 3

```
60         if (state == 1)
61             continue; // 더 높은 칸 있는 경우
62         else if (state == -1) { // 내가 봉우리면
63             isPeek[r][c] = true;
64             peekCnt++;
65         } else { // 높이 같은 칸이 존재하면 bfs 탐색
66             if (check.Peek(r, c, now)) {
67                 peekCnt++;
68
69                 // 기록된 봉우리 복사
70                 for (int ir = 0; ir < N; ir++) {
71                     for (int ic = 0; ic < M; ic++) {
72                         if (visited[ir][ic])
73                             isPeek[ir][ic] = true;
74                     }
75                 }
76             }
77         }
78     }
79 }
80 System.out.println(peekCnt);
81 }
82 }
83
84 static boolean check.Peek(int sr, int sc, int h) {
85     Queue<int[]> q = new ArrayDeque<>();
86     q.offer(new int[] { sr, sc });
87
88     visited = new boolean[N][M];
```

## 손현준 소스 코드 4

```
89
90     while (!q.isEmpty()) {
91         int[] now = q.poll();
92
93         for (int d = 0; d < 8; d++) {
94             int nr = now[0] + dr[d];
95             int nc = now[1] + dc[d];
96             if (nr < 0 || nr >= N || nc < 0 || nc >= M)
97                 continue;
98             if (map[nr][nc] > h)
99                 return false;
100            if (map[nr][nc] != h)
101                continue;
102            if (visited[nr][nc])
103                continue;
104
105            q.offer(new int[] { nr, nc });
106            visited[nr][nc] = true;
107        }
108    }
109    return true;
110 }
111 /*
112 * 산봉우리 : 주변 8칸이 현재 높이와 같거나 작다
113 *
114 * [배열 모든 칸에 대해 반복]
115 * - 주변 8칸 확인
116 * if 자신보다 높은 칸 있으면 continue
117 * else if 자신과 높이가 같은 칸 있으면
```

# 손현준 소스 코드 5

```
118     * else 그 칸은 봉우리
119     *
120     * 1) 자신과 높이가 같은 칸이 있을 때 탐색 방법
121     * (탐색 된 봉우리가 체크된 boolean 배열 확인하고 이미 방문한 봉우리면 continue)
122     * 방문 체크 하며 주변에 높이가 같은 칸으로 번지면서 확인
123     * └ 주변에 더 높은 칸이 있으면 봉우리가 아님
124     * 봉우리 확인에 성공했다면 방문 체크된 칸을 '탐색 된 봉우리'를 저장하는 배열에 복사
125     */
126 }
```

# 원찬혁 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 class Chanhyeok {
5     static int n, m;
6     static int ary[][];
7     static int dir[][] = { { 1, 0 }, { 1, 1 }, { 1, -1 }, { 0, 1 }, { 0, -1 }, {
8         -1, 1 }, { -1, 0 }, { -1, -1 } };
9     static boolean visited[][];
10
11     public static void main(String[] args) throws Exception {
12         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13         StringBuilder sb = new StringBuilder();
14         StringTokenizer st = new StringTokenizer(br.readLine());
15         n = Integer.parseInt(st.nextToken());
16         m = Integer.parseInt(st.nextToken());
17         ary = new int[n][m];
18         visited = new boolean[n][m];
19         int cnt = 0;
20         for (int i = 0; i < n; i++) {
21             st = new StringTokenizer(br.readLine());
22             for (int j = 0; j < m; j++)
23                 ary[i][j] = Integer.parseInt(st.nextToken());
24         }
25         for (int i = 0; i < n; i++) {
26             for (int j = 0; j < m; j++) {
27                 if (!visited[i][j]) {
28                     visited[i][j] = true;
29                     if (dfs(i, j))
                                cnt++;
30             }
31         }
32     }
33 }
```

## 원찬혁 소스 코드 2

```
30             }
31         }
32     }
33     System.out.println(cnt);
34 }
35
36     static boolean dfs(int i, int j) {
37         int ii, jj;
38         boolean ret = true;
39         for (int d = 0; d < 8; d++) {
40             ii = i + dir[d][0];
41             jj = j + dir[d][1];
42             if (ii >= 0 && ii < n && jj >= 0 && jj < m) {
43                 if (ary[ii][jj] > ary[i][j])
44                     ret = false;
45                 if (!visited[ii][jj]) {
46                     if (ary[ii][jj] == ary[i][j]) {
47                         visited[ii][jj] = true;
48                         ret = dfs(ii, jj) && ret;
49                     }
50                 }
51             }
52         }
53     }
54 }
55 }
```

# 김지훈 소스 코드 1

```
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3 import java.util.ArrayDeque;
4 import java.util.Queue;
5 import java.util.StringTokenizer;
6
7 /* 농장 관리 */
8 public class Jihoon {
9
10    static int N, M, res;
11    static int[] dr = { -1, 1, 0, 0, -1, -1, 1, 1 };
12    static int[] dc = { 0, 0, -1, 1, -1, 1, -1, 1 };
13    static int[][] map;
14    static boolean[][] visited;
15
16    public static void main(String[] args) throws Exception {
17
18        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
19        StringTokenizer st = new StringTokenizer(br.readLine());
20
21        N = Integer.parseInt(st.nextToken());
22        M = Integer.parseInt(st.nextToken());
23
24        map = new int[N][M];
25        for (int i = 0; i < N; i++) {
26            st = new StringTokenizer(br.readLine());
27            for (int j = 0; j < M; j++) {
28                map[i][j] = Integer.parseInt(st.nextToken());
29            }
30        }
31    }
32}
```

# 김지훈 소스 코드 2

```
31
32     res = 0;
33     visited = new boolean[N][M];
34     for (int r = 0; r < N; r++) {
35         for (int c = 0; c < M; c++) {
36             if (!visited[r][c])
37                 checkPeak(r, c);
38         }
39     }
40
41     System.out.println(res);
42 }
43
44 private static void checkPeak(int r, int c) {
45     Queue<Node> q = new ArrayDeque<>();
46     q.offer(new Node(r, c));
47
48     int height = map[r][c];
49     boolean isPeak = true;
50     visited[r][c] = true;
51
52     while (!q.isEmpty()) {
53         Node cur = q.poll();
54
55         for (int d = 0; d < 8; d++) {
56             int nr = cur.r + dr[d];
57             int nc = cur.c + dc[d];
58
59             if (nr < 0 || nr >= N || nc < 0 || nc >= M)
```

# 김지훈 소스 코드 3

```
60                     continue;
61
62             if (map[nr][nc] > height) {
63                 isPeak = false;
64             } else if (map[nr][nc] == height && !visited[nr][nc]) {
65                 visited[nr][nc] = true;
66                 q.offer(new Node(nr, nc));
67             }
68         }
69     }
70
71     if (isPeak)
72         res++;
73 }
74
75 private static class Node {
76     int r, c;
77
78     Node(int r, int c) {
79         this.r = r;
80         this.c = c;
81     }
82 }
83 }
```

# 임건애 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Keonae {
5
6     static int N, M;
7     static int[][] map;
8     static boolean[][] visited;
9     static int[] dx = { -1, -1, -1, 0, 0, 1, 1, 1 };
10    static int[] dy = { 1, 0, -1, 1, -1, 1, 0, -1 };
11
12    public static void main(String[] args) throws IOException {
13        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
14        StringTokenizer st = new StringTokenizer(br.readLine());
15
16        N = Integer.parseInt(st.nextToken());
17        M = Integer.parseInt(st.nextToken());
18
19        map = new int[N][M];
20
21        for (int i = 0; i < N; i++) {
22            st = new StringTokenizer(br.readLine());
23            for (int j = 0; j < M; j++) {
24                map[i][j] = Integer.parseInt(st.nextToken());
25            }
26        }
27
28        int count = 0;
29        visited = new boolean[N][M];
30        for (int i = 0; i < N; i++) {
```

## 임건애 소스 코드 2

## 임건애 소스 코드 3

```
60             flag = false;
61         }
62     }
63 }
64
65
66     return flag;
67 }
68
69 }
```

## DAG LCA(33851)

무가중치 방향 비순환 그래프  $G$ 가 주어진다. 다음 쿼리를 수행하는 프로그램을 작성하시오.

$u v$ : 두 정점  $u$ 와  $v$ 로 가는 경로가 모두 존재하는 모든 정점에 대해  $u$ 와  $v$ 로 가는 두 최단 경로의 길이 중 작지 않은 값의 최솟값을 출력한다. 두 정점  $u$ 와  $v$ 로 가는 경로가 존재하는 정점이 없다면 대신 -1을 출력한다.

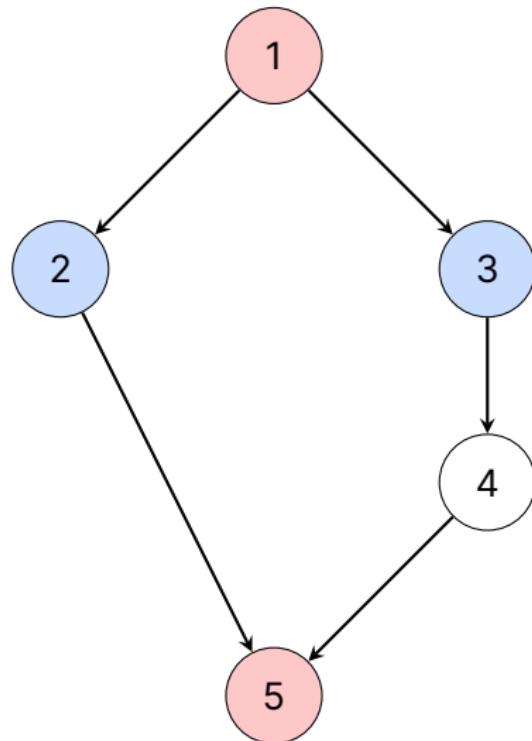
첫 번째 줄에 그래프  $G$ 의 정점의 개수  $n$ , 간선의 개수  $m$ , 질의의 개수  $q$ 가 공백으로 구분되어 주어진다.

( $5 \leq n \leq 2\,000$ ;  $0 \leq m \leq 4\,000$ ;  $1 \leq q \leq 1\,000$ )

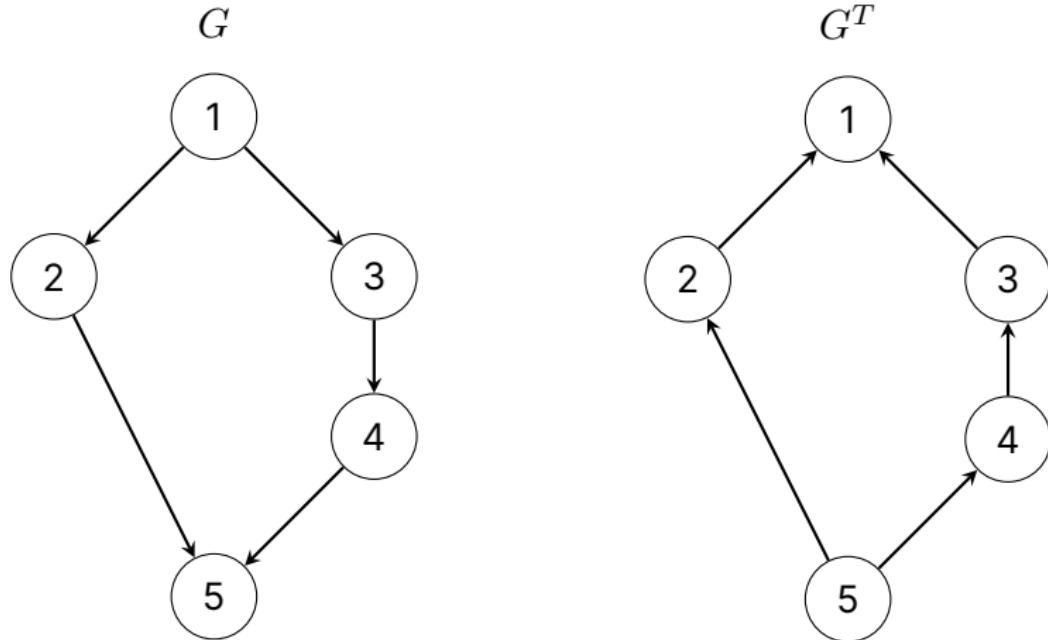
두 번째 줄부터  $m$ 개의 줄에 걸쳐 간선으로 연결된 두 정점의 번호  $u, v$ 가 한 줄에 하나씩 주어진다. 이는  $u$ 에서  $v$ 로 향하는 방향의 간선을 의미한다. ( $1 \leq u, v \leq n$ ;  $u \neq v$ )

$m + 2$ 번째 줄부터  $q$ 개의 줄에 걸쳐 쿼리가 한 줄에 하나씩 주어진다.

## 예제 방향 그래프와 쿼리



## 간선의 방향을 뒤집은 그래프 $G^T$



# 정의찬 소스 코드 1

```
1 import java.io.*;
2 import java.util.*;
3
4 class Uichan {
5     static int[][] distance;
6     static List<Integer>[] parent;
7     static int n;
8
9     public static void main(String[] args) throws NumberFormatException,
10    ↵ IOException {
11         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12         StringBuilder sb = new StringBuilder();
13         StringTokenizer st = new StringTokenizer(br.readLine());
14
15         // 가장가까운 공통 부모찾아서 간선합 긴쪽 출력
16
17         n = Integer.parseInt(st.nextToken());
18         int m = Integer.parseInt(st.nextToken());
19         int q = Integer.parseInt(st.nextToken());
20
21         distance = new int[n + 1][n + 1]; // idx1(자식)->idx2(부모) 거리
22         parent = new List[n + 1];
23
24         for (int i = 1; i <= n; i++) {
25             Arrays.fill(distance[i], Integer.MAX_VALUE);
26             parent[i] = new ArrayList<>();
27         }
28
29         for (int i = 0; i < m; i++) {
30             st = new StringTokenizer(br.readLine());
```

## 정의찬 소스 코드 2

```
30         int from = Integer.parseInt(st.nextToken());
31         int to = Integer.parseInt(st.nextToken());
32         parent[to].add(from);
33     }
34
35     for (int i = 1; i <= n; i++) {
36         getDistance(i, i, 0);
37     }
38
39     for (int i = 0; i < q; i++) {
40         st = new StringTokenizer(br.readLine());
41         int u = Integer.parseInt(st.nextToken());
42         int v = Integer.parseInt(st.nextToken());
43         int answer = Integer.MAX_VALUE;
44         for (int p = 1; p <= n; p++) {
45             // 한쪽이라도 p가 부모가 아닌 경우 continue
46             if (distance[u][p] == Integer.MAX_VALUE || distance[v][p] ==
47                 Integer.MAX_VALUE)
48                 continue;
49             int maxDist = Math.max(distance[u][p], distance[v][p]);
50             answer = Math.min(answer, maxDist);
51         }
52         sb.append(answer == Integer.MAX_VALUE ? -1 : answer).append("\n");
53     }
54     System.out.println(sb);
55 }
56
57 static void getDistance(int start, int current, int depth) {
```

## 정의찬 소스 코드 3

```
58         distance[start][current] = Math.min(distance[start][current], depth);
59
60     for (int parent : parent[current]) {
61         getDistance(start, parent, depth + 1);
62     }
63 }
64 }
```

# 서민종(구) 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class MinjongOld {
5     static int n, m, q;
6     static List<Integer>[] connectedTo, connectedFrom;
7     static Integer[][] memo;
8
9     public static void main(String[] args) throws IOException {
10         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11         BufferedWriter bw = new BufferedWriter(new
12             ↳ OutputStreamWriter(System.out));
13         StringTokenizer st = new StringTokenizer(br.readLine());
14         n = Integer.parseInt(st.nextToken());
15         m = Integer.parseInt(st.nextToken());
16         q = Integer.parseInt(st.nextToken());
17         connectedFrom = new List[n + 1];
18         connectedTo = new List[n + 1];
19         for (int i = 1; i <= n; i++) {
20             connectedFrom[i] = new ArrayList<>();
21             connectedTo[i] = new ArrayList<>();
22         }
23         for (int i = 0; i < m; i++) {
24             st = new StringTokenizer(br.readLine());
25             int u = Integer.parseInt(st.nextToken());
26             int v = Integer.parseInt(st.nextToken());
27             connectedFrom[v].add(u);
28             connectedTo[u].add(v);
29         }
30     }
31 }
```

## 서민종(구) 소스 코드 2

```
29         memo = new Integer[n + 1][n + 1];
30         for (int i = 0; i < q; i++) {
31             int answer = 1000000;
32             st = new StringTokenizer(br.readLine());
33             int num1 = Integer.parseInt(st.nextToken());
34             int num2 = Integer.parseInt(st.nextToken());
35             List<Integer> list = getCommonAncestor(num1, num2);
36             if (list.size() == 0) {
37                 answer = -1;
38             } else {
39                 for (int ancestor : list) {
40                     int dist1 = getDistance(ancestor, num1);
41                     int dist2 = getDistance(ancestor, num2);
42                     answer = Math.min(answer, Math.max(dist1, dist2));
43                 }
44             }
45             bw.write(answer + "\n");
46         }
47         bw.flush();
48     }
49
50     public static int getDistance(int ancestor, int num) {
51         if (memo[num][ancestor] != null) {
52             return memo[num][ancestor];
53         } else {
54             Queue<Integer> numQueue = new LinkedList<>();
55             Queue<Integer> distQueue = new LinkedList<>();
56             numQueue.offer(num);
57             distQueue.offer(0);
```

# 서민종(구) 소스 코드 3

```
58         while (!numQueue.isEmpty()) {
59             int current = numQueue.poll();
60             int distance = distQueue.poll();
61             if (memo[num][current] == null) {
62                 memo[num][current] = distance;
63             } else {
64                 memo[num][current] = Math.min(memo[num][current], distance);
65             }
66             for (int next : connectedFrom[current]) {
67                 numQueue.offer(next);
68                 distQueue.offer(distance + 1);
69             }
70         }
71     }
72     return memo[num][ancestor];
73 }
74
75 public static List<Integer> getCommonAncestor(int num1, int num2) {
76     boolean[] visited1 = new boolean[n + 1];
77     List<Integer> list = new ArrayList<>();
78     Queue<Integer> queue = new LinkedList<>();
79     queue.offer(num1);
80     visited1[num1] = true;
81     while (!queue.isEmpty()) {
82         int current = queue.poll();
83         for (int next : connectedFrom[current]) {
84             if (!visited1[next]) {
85                 queue.offer(next);
86                 visited1[next] = true;
```

## 서민종(구) 소스 코드 4

```
87             }
88         }
89     }
90     boolean[] visited2 = new boolean[n + 1];
91     if (visited1[num2]) {
92         list.add(num2);
93     }
94     queue.offer(num2);
95     visited2[num2] = true;
96     while (!queue.isEmpty()) {
97         int current = queue.poll();
98         for (int next : connectedFrom[current]) {
99             if (visited1[next]) {
100                 list.add(next);
101             }
102             if (!visited2[next]) {
103                 queue.offer(next);
104                 visited2[next] = true;
105             }
106         }
107     }
108     return list;
109 }
110 }
```

# 서민종(신) 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class MinjongNew {
5     static int n, m, q;
6     static List<Integer>[] connectedFrom;
7     static Integer[][] memo;
8
9     public static void main(String[] args) throws IOException {
10         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11         BufferedWriter bw = new BufferedWriter(new
12             ↳ OutputStreamWriter(System.out));
13         StringTokenizer st = new StringTokenizer(br.readLine());
14         n = Integer.parseInt(st.nextToken());
15         m = Integer.parseInt(st.nextToken());
16         q = Integer.parseInt(st.nextToken());
17         connectedFrom = new List[n + 1];
18         for (int i = 1; i <= n; i++) {
19             connectedFrom[i] = new ArrayList<>();
20         }
21         for (int i = 0; i < m; i++) {
22             st = new StringTokenizer(br.readLine());
23             int u = Integer.parseInt(st.nextToken());
24             int v = Integer.parseInt(st.nextToken());
25             connectedFrom[v].add(u);
26         }
27         memo = new Integer[n + 1][n + 1];
28         for (int i = 1; i <= n; i++) {
29             BFS(i);
30         }
31     }
32 }
```

## 서민종(신) 소스 코드 2

```
29      }
30      for (int i = 0; i < q; i++) {
31          int answer = 1000000;
32          st = new StringTokenizer(br.readLine());
33          int num1 = Integer.parseInt(st.nextToken());
34          int num2 = Integer.parseInt(st.nextToken());
35          for (int j = 1; j <= n; j++) {
36              if (memo[num1][j] != null && memo[num2][j] != null) {
37                  answer = Math.min(answer, Math.max(memo[num1][j],
38                                              memo[num2][j]));
39              }
40          }
41          if (answer == 1000000) {
42              answer = -1;
43          }
44          bw.write(answer + "\n");
45      }
46      bw.flush();
47  }
48  public static void BFS(int startNum) {
49      Queue<Integer> numQueue = new LinkedList<>();
50      Queue<Integer> distQueue = new LinkedList<>();
51      numQueue.offer(startNum);
52      distQueue.offer(0);
53      while (!numQueue.isEmpty()) {
54          int current = numQueue.poll();
55          int distance = distQueue.poll();
56          if (memo[startNum][current] == null) {
```

## 서민종(신) 소스 코드 3

```
57         memo[startNum][current] = distance;
58     } else {
59         memo[startNum][current] = Math.min(memo[startNum][current],
60             ↪ distance);
61     }
62     for (int next : connectedFrom[current]) {
63         numQueue.offer(next);
64         distQueue.offer(distance + 1);
65     }
66 }
67 }
```

# 원찬혁 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 class Chanhyeok {
5     static int n, m, q, res, a, b;
6     static List<Integer>[] e;
7     static int[][] dp;
8
9     public static void main(String[] args) throws Exception {
10         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11         StringBuilder sb = new StringBuilder();
12         StringTokenizer st = new StringTokenizer(br.readLine());
13         n = Integer.parseInt(st.nextToken());
14         m = Integer.parseInt(st.nextToken());
15         q = Integer.parseInt(st.nextToken());
16         int u, v;
17         e = new List[n];
18         dp = new int[n][n];
19         Queue<Integer> qq = new ArrayDeque<>();
20         for (int i = 0; i < n; i++)
21             e[i] = new ArrayList<>();
22
23         for (int i = 0; i < m; i++) {
24             st = new StringTokenizer(br.readLine());
25             u = Integer.parseInt(st.nextToken()) - 1;
26             v = Integer.parseInt(st.nextToken()) - 1;
27             e[v].add(u);
28         }
29         for (int i = 0; i < n; i++)
30             Arrays.fill(dp[i], -1);
```

## 원찬혁 소스 코드 2

```
31     for (int i = 0; i < n; i++) {
32         dp[i][i] = 0;
33         qq.offer(i);
34         while (!qq.isEmpty()) {
35             int t = qq.poll();
36             for (int it : e[t]) {
37                 if (dp[i][it] < 0) {
38                     dp[i][it] = dp[i][t] + 1;
39                     qq.offer(it);
40                 }
41             }
42         }
43     }
44     for (int k = 0; k < q; k++) {
45         st = new StringTokenizer(br.readLine());
46         a = Integer.parseInt(st.nextToken()) - 1;
47         b = Integer.parseInt(st.nextToken()) - 1;
48         res = -1;
49         for (int i = 0; i < n; i++) {
50             if (dp[a][i] >= 0 && dp[b][i] >= 0) {
51                 if (res < 0)
52                     res = Math.max(dp[a][i], dp[b][i]);
53                 res = Math.min(res, Math.max(dp[a][i], dp[b][i]));
54             }
55         }
56         sb.append(res).append("\n");
57     }
58     System.out.println(sb);
```

## 원찬혁 소스 코드 3

```
59     }
60 }
```

# 손현준 소스 코드 1

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import java.util.ArrayDeque;
5 import java.util.ArrayList;
6 import java.util.Arrays;
7 import java.util.List;
8 import java.util.Queue;
9 import java.util.StringTokenizer;
10
11 public class Hyeonjun {
12     static int N, M;
13     static List<Integer>[] connected;
14
15     public static void main(String[] args) throws IOException {
16         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
17         StringBuilder sb = new StringBuilder();
18         StringTokenizer st;
19
20         // 입력 받기
21         st = new StringTokenizer(br.readLine(), " ");
22         N = Integer.parseInt(st.nextToken()); // 정점의 수
23         M = Integer.parseInt(st.nextToken()); // 간선의 수
24         int Q = Integer.parseInt(st.nextToken()); // 입력 받을 쿼리의 수
25
26         // 간선 정보를 가진 리스트 초기화
27         connected = new ArrayList[N];
28         for (int i = 0; i < N; i++)
29             connected[i] = new ArrayList<>();
30         // 간선 입력 받기
```

## 손현준 소스 코드 2

```
31     for (int m = 0; m < M; m++) {
32         st = new StringTokenizer(br.readLine(), " ");
33         int nodeFrom = Integer.parseInt(st.nextToken());
34         int nodeTo = Integer.parseInt(st.nextToken());
35         connected[nodeTo - 1].add(nodeFrom - 1);
36     }
37
38     // 쿼리 입력 받고 처리
39     for (int q = 0; q < Q; q++) {
40         st = new StringTokenizer(br.readLine(), " ");
41         int u = Integer.parseInt(st.nextToken());
42         int v = Integer.parseInt(st.nextToken());
43
44         sb.append(getResult(u - 1, v - 1)).append("\n");
45     }
46     System.out.println(sb.toString());
47 }
48
49 static int getResult(int u, int v) {
50     int result = Integer.MAX_VALUE;
51
52     int[] uNodes = getAccessibleNodes(u);
53     int[] vNodes = getAccessibleNodes(v);
54
55     for (int i = 0; i < N; i++) {
56         if (uNodes[i] <= 0 || vNodes[i] <= 0)
57             continue;
58         int len = Math.max(uNodes[i], vNodes[i]) - 1;
59         result = Math.min(result, len);
60     }
61 }
```

# 손현준 소스 코드 3

```
60         }
61
62     return (result != Integer.MAX_VALUE ? result : -1);
63 }
64
65 static int[] getAccessibleNodes(int from) {
66     Queue<Integer> q = new ArrayDeque<>();
67     int[] dist = new int[N];
68
69     q.offer(from);
70     dist[from] = 1;
71
72     while (!q.isEmpty()) {
73         int now = q.poll();
74         int len = dist[now] + 1;
75
76         for (int c : connected[now]) {
77             // BFS는 최단거리! > 처음 방문만 체크하도록 변경
78             if (dist[c] != 0)
79                 continue;
80             dist[c] = len;
81             q.offer(c);
82         }
83     }
84
85     return dist;
86 }
87
88 /*
```

# 손현준 소스 코드 4

```
89     * 각 정점이 어디로 연결되었는지 정보 입력 받음
90     * u나 v 둘다 도달할 수 있는 정점들의 거리의 최댓값들을 구한다
91     * ┌ 이 최대값중에 최솟값을 구한다
92     *
93     * 두 정점으로부터 방문 가능한 정점을 하나씩 체크하면서 이동.
94     * int 배열에 거리 저장 (초기값 -1. 0보다 같거나 크면 연결됨)
95     * ┌ u, v 배열 둘다 0보다 같거나 큰 값 가지고 있는 정점 = 둘다 연결된 정점
96     * ┌ 둘중 큰값 가져옴
97     * ┌ 큰 값 중에 최솟값 찾음
98     */
99 }
```

# 김지훈 소스 코드 1

```
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3 import java.util.ArrayDeque;
4 import java.util.ArrayList;
5 import java.util.Arrays;
6 import java.util.Queue;
7 import java.util.StringTokenizer;
8
9 public class Jihoon {
10
11     static int N, M, Q, res;
12     static ArrayList<Integer>[] adjList;
13     static int[][] dist;
14
15     public static void main(String[] args) throws Exception {
16
17         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
18         StringTokenizer st = new StringTokenizer(br.readLine());
19         StringBuilder sb = new StringBuilder();
20
21         N = Integer.parseInt(st.nextToken()); // 정점의 개수 [5, 2,000]
22         M = Integer.parseInt(st.nextToken()); // 간선의 개수 [0, 4,000]
23         Q = Integer.parseInt(st.nextToken()); // 질의의 개수 [1, 1,000]
24
25         // adjList init
26         adjList = new ArrayList[N + 1];
27         for (int i = 1; i <= N; i++) {
28             adjList[i] = new ArrayList<>();
29         }
30     }
31 }
```

# 김지훈 소스 코드 2

```
31         // input
32         for (int i = 0; i < M; i++) {
33             st = new StringTokenizer(br.readLine());
34             int a = Integer.parseInt(st.nextToken());
35             int b = Integer.parseInt(st.nextToken());
36             adjList[a].add(b);
37         }
38
39         // dist init
40         dist = new int[N + 1][N + 1];
41         for (int i = 1; i <= N; i++) {
42             Arrays.fill(dist[i], -1);
43         }
44
45         // dist bfs, O(N(V+E))
46         for (int j = 1; j <= N; j++) {
47             bfs(j);
48         }
49
50         // query, O(Q*N)
51         for (int i = 0; i < Q; i++) {
52             st = new StringTokenizer(br.readLine());
53             int a = Integer.parseInt(st.nextToken());
54             int b = Integer.parseInt(st.nextToken());
55
56             res = Integer.MAX_VALUE;
57             boolean found = false;
58             for (int j = 1; j <= N; j++) {
59                 if (dist[j][a] == -1 || dist[j][b] == -1)
```

# 김지훈 소스 코드 3

```
60             continue;
61             int temp = Math.max(dist[j][a], dist[j][b]);
62             res = Math.min(res, temp);
63             found = true;
64         }
65
66         if (!found)
67             sb.append("-1\n");
68         else
69             sb.append(res).append("\n");
70     }
71
72     System.out.println(sb);
73 }
74
75 private static void bfs(int start) {
76     Queue<Integer> q = new ArrayDeque<>();
77
78     q.offer(start);
79     dist[start][start] = 0;
80
81     while (!q.isEmpty()) {
82         int cur = q.poll();
83
84         for (int next : adjList[cur]) {
85             if (dist[start][next] == -1) {
86                 dist[start][next] = dist[start][cur] + 1;
87                 q.offer(next);
88             }
89         }
90     }
91 }
```

## 김지훈 소스 코드 4

```
89         }
90     }
91 }
92 }
```

# 김시온 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Sion {
5     static int n;
6     static List<List<Integer>> g = new ArrayList<>();
7
8     static int[] bfs(int x) {
9         Queue<Integer> q = new LinkedList<>();
10        q.add(x);
11        int[] d = new int[n + 1];
12        Arrays.fill(d, -1);
13        d[x] = 0;
14
15        while (!q.isEmpty()) {
16            int u = q.poll();
17
18            for (int v : g.get(u)) {
19                if (d[v] == -1) {
20                    q.add(v);
21                    d[v] = d[u] + 1;
22                }
23            }
24        }
25
26        return d;
27    }
28
29    public static void main(String[] args) throws IOException {
30        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

## 김시온 소스 코드 2

```
31     StringBuilder bw = new StringBuilder();
32     StringTokenizer st = new StringTokenizer(br.readLine());
33     Sion.n = Integer.parseInt(st.nextToken());
34     int m = Integer.parseInt(st.nextToken());
35     int q = Integer.parseInt(st.nextToken());
36     for (int i = 0; i <= n; i++) {
37         g.add(new ArrayList<>());
38     }
39
40     for (int i = 0; i < m; i++) {
41         st = new StringTokenizer(br.readLine());
42         int u = Integer.parseInt(st.nextToken());
43         int v = Integer.parseInt(st.nextToken());
44         g.get(v).add(u);
45     }
46
47     for (int i = 0; i < q; i++) {
48         st = new StringTokenizer(br.readLine());
49         int u = Integer.parseInt(st.nextToken());
50         int v = Integer.parseInt(st.nextToken());
51
52         int[] d1 = bfs(u), d2 = bfs(v);
53
54         int min = (int) 1e9;
55         for (int x = 1; x <= n; x++) {
56             if (d1[x] == -1 || d2[x] == -1) {
57                 continue;
58             }
59             min = Math.min(min, Math.max(d1[x], d2[x]));
}
```

## 김시온 소스 코드 3

```
60         }
61         bw.append(min == (int) 1e9 ? -1 : min).append('\n');
62     }
63     System.out.print(bw);
64 }
65 }
```

# 임건애 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Keonae {
5
6     static int N, M, q; // 정점, 간선, 쿼리
7     static List<ArrayList<Integer>> graph;
8     static boolean[] visited;
9     static int[][] dist;
10
11    public static void main(String[] args) throws IOException {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13        StringTokenizer st = new StringTokenizer(br.readLine());
14
15        N = Integer.parseInt(st.nextToken());
16        M = Integer.parseInt(st.nextToken());
17        q = Integer.parseInt(st.nextToken());
18
19        graph = new ArrayList<>();
20        for (int i = 0; i <= N; i++) {
21            graph.add(new ArrayList<>());
22        }
23
24        dist = new int[N + 1][N + 1];
25        for (int i = 0; i < M; i++) {
26            st = new StringTokenizer(br.readLine());
27            int u = Integer.parseInt(st.nextToken());
28            int v = Integer.parseInt(st.nextToken());
29
30            graph.get(u).add(v);
31        }
32
33        for (int i = 0; i < N; i++) {
34            Arrays.fill(dist[i], -1);
35        }
36
37        for (int i = 0; i < q; i++) {
38            st = new StringTokenizer(br.readLine());
39            int u = Integer.parseInt(st.nextToken());
40            int v = Integer.parseInt(st.nextToken());
41
42            if (dist[u][v] == -1) {
43                dist[u][v] = 1;
44            } else {
45                dist[u][v] += 1;
46            }
47        }
48
49        for (int i = 0; i < N; i++) {
50            for (int j = 0; j < N; j++) {
51                if (dist[i][j] == -1) {
52                    System.out.print("0 ");
53                } else {
54                    System.out.print(dist[i][j] + " ");
55                }
56            }
57            System.out.println();
58        }
59    }
60}
```

## 임건애 소스 코드 2

```
31         }
32
33     // 거리 계산
34     for (int i = 1; i <= N; i++) {
35         visited = new boolean[N + 1];
36         Arrays.fill(dist[i], -1);
37         cal(i);
38     }
39
40     StringBuilder sb = new StringBuilder();
41     for (int i = 0; i < q; i++) {
42         st = new StringTokenizer(br.readLine());
43         int u = Integer.parseInt(st.nextToken());
44         int v = Integer.parseInt(st.nextToken());
45
46         int min = Integer.MAX_VALUE;
47         for (int j = 1; j <= N; j++) {
48             if (dist[j][u] != -1 && dist[j][v] != -1) {
49                 min = Math.min(min, Math.max(dist[j][u], dist[j][v]));
50             }
51         }
52
53         sb.append(min == Integer.MAX_VALUE ? -1 : min).append("\n");
54     }
55
56     System.out.println(sb);
57 }
58
59     static void cal(int start) {
```

# 임건애 소스 코드 3

```
60     visited[start] = true;
61     Queue<Integer> q = new LinkedList<>();
62     q.add(start);
63     dist[start][start] = 0;
64
65     while (!q.isEmpty()) {
66         int cur = q.poll();
67         for (int next : graph.get(cur)) {
68             if (!visited[next]) {
69                 dist[start][next] = dist[start][cur] + 1;
70                 visited[next] = true;
71                 q.add(next);
72             }
73         }
74     }
75 }
76 }
77 }
78 }
```

# 비밀번호 제작(20304)

## 문제

서강대학교 전산실에서 보안직원으로 일하는 향빈이는 한 통의 이메일을 받게 되었다. 이메일에는 서버 관리자 계정에 대한 비정상적인 로그인 시도가 감지되었다는 내용이 적혀 있었고, 첨부된 파일에는 지금까지 로그인 시도에 사용된 비밀번호 목록이 있었다. 참고로, 서버 관리자 계정의 비밀번호로는 0 이상  $N$  이하의 정수 중 하나를 사용할 수 있다.

두 비밀번호의 안전 거리는 이진법으로 표현한 두 비밀번호의 서로 다른 자리의 개수로 정의한다. 예를 들어 3을 이진법으로 표현하면 0011, 8을 이진법으로 표현하면 1000이 되고, 이때 서로 다른 자리의 개수는 3개이므로 3과 8의 안전 거리는 3이 된다.

어떤 비밀번호의 안전도는 지금까지 로그인 시도에 사용된 모든 비밀번호와의 안전 거리 중 최솟값으로 정의한다. 예를 들어 지금까지 로그인 시도에 사용된 비밀번호가 3과 4이라고 가정하면, 새로운 비밀번호 8에 대해 3과 8의 안전 거리는 3, 4와 8의 안전 거리는 2이므로 비밀번호 8의 안전도는 2가 된다.

향빈이는 해커가 비밀번호를 알아내기까지의 시간을 최대한 늦추기 위해 현재 사용 중인 관리자 계정 비밀번호의 안전도가 가장 높게끔 바꾸고 싶다. 이때, 안전도가 제일 높은 비밀번호의 안전도를 구하여라.

## 입력

첫째 줄에 관리자 계정 비밀번호의 최댓값을 나타내는 정수  $N$ 이 주어진다. ( $0 \leq N \leq 1\,000\,000$ )

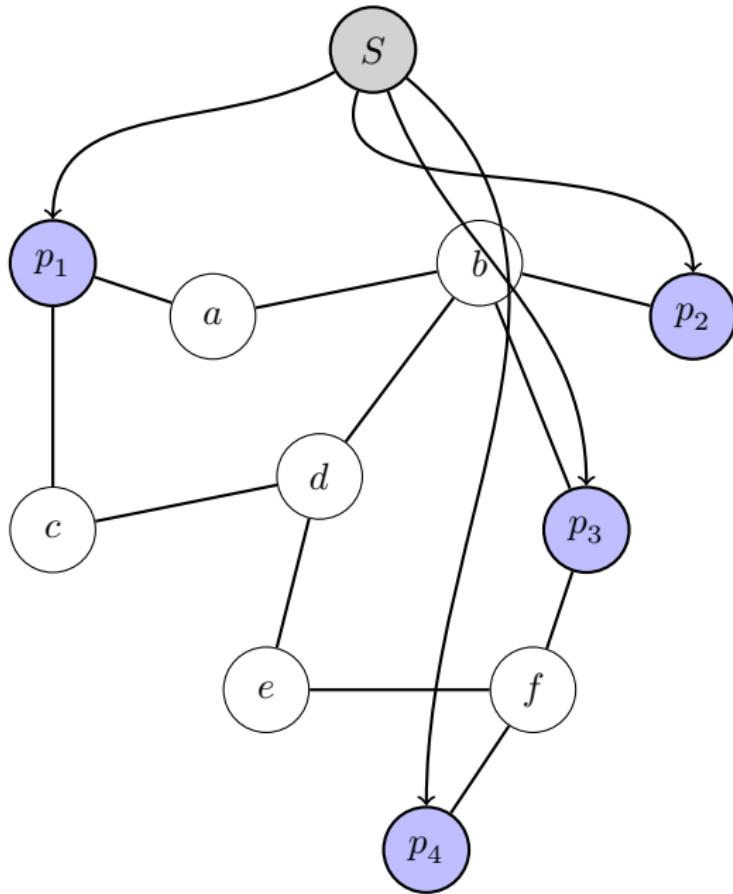
둘째 줄에는 로그인 시도에 사용된 비밀번호의 개수를 나타내는 정수  $M$ 이 주어진다. ( $1 \leq M \leq 100\,000$ )

셋째 줄에는 로그인 시도에 사용된 비밀번호 값인 정수  $p_1, p_2, \dots, p_M$ 이 주어진다. ( $0 \leq p_i \leq N$ )

## 출력

안전도가 제일 높은 비밀번호의 안전도를 출력한다.

## 가상의 정점 $S$ 를 추가한 Multi-source BFS



# 원찬혁1 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Chanhyeok1 {
5     static boolean visited[];
6
7     public static void main(String[] args) throws Exception {
8         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
9         StringTokenizer st;
10        int res = -1, qsize, t, i, temp;
11        int n = Integer.parseInt(br.readLine());
12        int m = Integer.parseInt(br.readLine());
13        Queue<Integer> q = new ArrayDeque<>();
14        visited = new boolean[n + 1];
15        st = new StringTokenizer(br.readLine());
16        for (i = 0; i < m; i++) {
17            t = Integer.parseInt(st.nextToken());
18            visited[t] = true;
19            q.offer(t);
20        }
21
22        while (!q.isEmpty()) {
23            qsize = q.size();
24            while (qsize-- > 0) {
25                t = q.poll();
26                for (i = 0; i < 32; i++) {
27                    temp = t ^ (1 << i);
28                    if (temp > n || temp < 0)
29                        continue;
30                    if (!visited[temp]) {
```

# 원찬혁1 소스 코드 2

```
31             visited[temp] = true;
32             q.offer(temp);
33         }
34     }
35 }
36 res++;
37 }
38 System.out.println(res);
39 }
40 }
```

## 원찬혁2 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Chanyeok2 {
5     static boolean visited[];
6
7     public static void main(String[] args) throws Exception {
8         BufferedReader br = new BufferedReader(new
9             InputStreamReader(System.in));
10        StringTokenizer st;
11        int res = -1, qsize, t, i, temp;
12        int n = Integer.parseInt(br.readLine());
13        int m = Integer.parseInt(br.readLine());
14        Queue<Integer> q = new ArrayDeque<>();
15        visited = new boolean[n + 1];
16        st = new StringTokenizer(br.readLine());
17        for (i = 0; i < m; i++) {
18            t = Integer.parseInt(st.nextToken());
19            visited[t] = true;
20            q.offer(t);
21        }
22
23        while (!q.isEmpty()) {
24            qsize = q.size();
25            while (qsize-- > 0) {
26                t = q.poll();
27                for (i = 0; i < 32; i++) {
28                    temp = t ^ (1 << i);
29                    if ((1 << i) > n)
                        break;
                    }
                }
            }
        }
    }
```

## 원찬혁2 소스 코드 2

```
30             if (temp > n || temp < 0)
31                 continue;
32             if (!visited[temp]) {
33                 visited[temp] = true;
34                 q.offer(temp);
35             }
36         }
37     }
38     res++;
39 }
40 System.out.println(res);
41 }
42 }
```

# 김시온 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Sion {
5
6     public static void main(String[] args) throws IOException {
7         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
8         int n = Integer.parseInt(br.readLine());
9         int m = Integer.parseInt(br.readLine());
10
11         Queue<Integer> q = new LinkedList<>();
12         int[] d = new int[n + 1];
13         Arrays.fill(d, -1);
14
15         StringTokenizer st = new StringTokenizer(br.readLine(), " ");
16         for (int i = 0; i < m; i++) {
17             int u = Integer.parseInt(st.nextToken());
18             q.add(u);
19             d[u] = 0;
20         }
21
22         while (!q.isEmpty()) {
23             int u = q.poll();
24             for (int i = 0; i <= 20; i++) {
25                 int v = u ^ (1 << i);
26                 if (0 <= v && v <= n && d[v] == -1) {
27                     q.add(v);
28                     d[v] = d[u] + 1;
29                 }
30             }
31         }
32     }
33 }
```

## 김시온 소스 코드 2

```
31      }
32
33      int max = 0;
34      for (int u = 0; u <= n; u++) {
35          max = Math.max(max, d[u]);
36      }
37      System.out.println(max);
38  }
39
40 }
```

# 대전 도시철도 2호선 (32408)

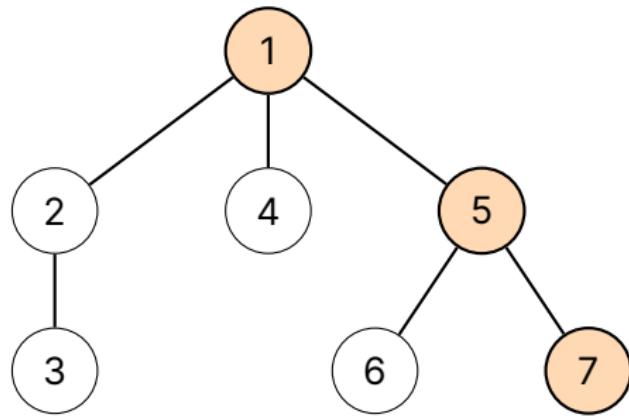
## 문제

대전시는  $N$  개의 교차로와  $N - 1$  개의 도로가 교차로를 정점, 도로를 간선으로 하는 트리 구조를 이루고 있다. 교차로는 1번부터  $N$  번까지 번호가 매겨져 있다. 대전시는 1번 교차로와  $N$  번 교차로를 잇는 경로를 따라 대전 도시철도 1호선을 운행하고 있다. 이 경로에 포함되는 교차로와 도로에는 각각 역과 철도가 설치되어 있다.

대전시는 도시철도 2호선을 추가로 운행하려고 한다. 1호선과 비슷하게 2호선도  $S$  번 교차로와  $E$  번 교차로를 잇는 경로를 따라 역과 철도를 설치하려고 한다.  $S$  번과  $E$  번 교차로는 1호선 역이 설치되지 않은 서로 다른 교차로여야 하고, 이용객이 1호선으로 환승할 수 있도록 2호선은 1호선 역을 적어도 하나 포함해야 한다.

대전 도시철도 2호선을 운행하기 위해 조건을 만족하는 두 교차로를 고르는 경우의 수를 구해보자. 단, 두 교차로의 순서를 바꾼 것은 같은 경우로 본다.

## 예제 입력



# 정의찬 소스 코드 1

```
1 import java.io.*;
2 import java.util.*;
3
4 class Uichan {
5     static int[] dy = { -1, -1, 0, 1, 1, 1, 0, -1 };
6     static int[] dx = { 0, 1, 1, 1, 0, -1, -1, -1 };
7     static int[][] map;
8
9     public static void main(String[] args) throws NumberFormatException,
10        ↵ IOException {
11         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12         StringBuilder sb = new StringBuilder();
13         StringTokenizer st = new StringTokenizer(br.readLine());
14         int n = Integer.parseInt(st.nextToken());
15         int m = Integer.parseInt(st.nextToken());
16         map = new int[n][m];
17
18         for (int i = 0; i < n; i++) {
19             st = new StringTokenizer(br.readLine());
20             for (int j = 0; j < m; j++) {
21                 map[i][j] = Integer.parseInt(st.nextToken());
22             }
23
24             int answer = 0;
25             Set<Integer> checked = new HashSet<>();
26             Set<Integer> temp;
27
28             for (int i = 0; i < n; i++) {
29                 Loop: for (int j = 0; j < m; j++) {
```

## 정의찬 소스 코드 2

```
30         if (checked.contains(i * 100 + j))
31             continue;
32
33         temp = new HashSet<>();
34         Queue<int[]> q = new LinkedList<>();
35         q.add(new int[] { i, j });
36         temp.add(i * 100 + j);
37
38         while (!q.isEmpty()) {
39             int[] current = q.poll();
40             for (int k = 0; k < 8; k++) {
41                 int nY = current[0] + dy[k];
42                 int nX = current[1] + dx[k];
43                 if (nY >= 0 && nY < m)
44                     && nX >= 0 && nX < m) {
45                     // 본인보다 낮은지검사
46                     if (map[current[0]][current[1]] < map[nY][nX])
47                         continue Loop;
48                     // 같은높이라면 큐 추가
49                     if (map[current[0]][current[1]] == map[nY][nX] &&
50                         !temp.contains(nY * 100 + nX)) {
51                         q.add(new int[] { nY, nX });
52                         temp.add(nY * 100 + nX);
53                     }
54                 }
55             }
56         }
57
58         checked.addAll(temp);
```

## 정의찬 소스 코드 3

```
58             answer++;
59         }
60     }
61
62     System.out.println(answer);
63 }
64 }
```

# 원찬혁 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Chanyeok {
5     static boolean visited[], fir[];
6     static List<Integer> e[];
7     static int n;
8     static long res;
9
10    public static void main(String[] args) throws Exception {
11        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12        StringTokenizer st;
13        int u, v;
14        n = Integer.parseInt(br.readLine());
15        visited = new boolean[n];
16        fir = new boolean[n];
17        e = new List[n];
18        for (int i = 0; i < n; i++)
19            e[i] = new ArrayList<>();
20        for (int i = 0; i < n - 1; i++) {
21            st = new StringTokenizer(br.readLine());
22            u = Integer.parseInt(st.nextToken()) - 1;
23            v = Integer.parseInt(st.nextToken()) - 1;
24            e[u].add(v);
25            e[v].add(u);
26        }
27        visited[0] = true;
28        fir[0] = true;
29        dfs(0);
30        count(0);
```

## 원찬혁 소스 코드 2

```
31         System.out.println(res);
32     }
33
34     static boolean dfs(int idx) {
35         if (idx == n - 1) {
36             fir[idx] = true;
37             return true;
38         }
39         boolean ret = false;
40         for (int it : e[idx]) {
41             if (!visited[it]) {
42                 visited[it] = true;
43                 if (dfs(it)) {
44                     ret = true;
45                     fir[idx] = true;
46                 }
47                 visited[it] = false;
48             }
49         }
50         return ret;
51     }
52
53     static long count(int idx) {
54         long ret = 0, val = 0, temp;
55         if (!fir[idx])
56             ret++;
57         for (int it : e[idx]) {
58             if (!visited[it]) {
59                 visited[it] = true;
```

## 원찬혁 소스 코드 3

```
60         temp = count(it);
61         ret += temp;
62         visited[it] = false;
63         if (fir[idx]) {
64             res += val * temp;
65             val += temp;
66         }
67     }
68 }
69 return ret;
70 }
71 }
```

# 손현준 소스 코드 1

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import java.util.ArrayDeque;
5 import java.util.ArrayList;
6 import java.util.List;
7 import java.util.StringTokenizer;
8
9 public class Hyeonjun {
10     static int N, llCnt;
11     static List<Integer>[] connected;
12     static boolean[] llVisited;
13
14     public static void main(String[] args) throws IOException {
15         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
16         StringTokenizer st;
17
18         N = Integer.parseInt(br.readLine());
19         connected = new ArrayList[N];
20         for (int i = 0; i < N; i++)
21             connected[i] = new ArrayList<>();
22
23         for (int i = 0; i < N - 1; i++) {
24             st = new StringTokenizer(br.readLine(), " ");
25             int from = Integer.parseInt(st.nextToken());
26             int to = Integer.parseInt(st.nextToken());
27             connected[from - 1].add(to - 1);
28             connected[to - 1].add(from - 1);
29         }
30     }
}
```

## 손현준 소스 코드 2

```
31         // 1호선 역 받아오기
32         l1Cnt = 0;
33         boolean[] isL1Stations = getL1Stations();
34         l1Visited = new boolean[N];
35         System.arraycopy(isL1Stations, 0, l1Visited, 0, N);
36
37         long l2Cnt = N - l1Cnt;
38         long result = (l2Cnt * (l2Cnt - 1)) / 2;
39         for (int i = 0; i < N; i++) {
40             if (!isL1Stations[i])
41                 continue;
42
43             for (int c : connected[i]) { // 연결된 정점으로 부터 정점 개수 탐색
44                 if (isL1Stations[c])
45                     continue; // 1호선은 계산 x
46
47                 // 정점 개수 계산
48                 long groupCnt = getStationCnt(c);
49                 result -= (groupCnt * (groupCnt - 1)) / 2;
50             }
51         }
52
53         System.out.println(result);
54     }
55
56     static boolean[] getL1Stations() {
57         ArrayDeque<Integer> q = new ArrayDeque<>();
58         int[] parent = new int[N];
59         boolean[] visited = new boolean[N];
```

## 손현준 소스 코드 3

```
60
61     q.offer(0); // 0 ~ N - 1까지 경로 찾기
62     parent[0] = -1;
63     visited[0] = true;
64
65     while (!q.isEmpty()) {
66         int now = q.poll();
67         if (now == N - 1)
68             break; // N 까지의 경로만 찾으면 됨
69
70         for (int c : connected[now]) {
71             if (visited[c])
72                 continue;
73             visited[c] = true;
74             parent[c] = now;
75             q.offer(c);
76         }
77     }
78
79     boolean[] isL1Stations = new boolean[N];
80     int cur = N - 1; // N 부터 1까지 경로 체크
81     while (cur != -1) {
82         isL1Stations[cur] = true;
83         cur = parent[cur];
84         l1Cnt++;
85     }
86
87     return isL1Stations;
88 }
```

## 손현준 소스 코드 4

```
89
90     static int getStationCnt(int strNode) {
91         ArrayDeque<Integer> q = new ArrayDeque<>();
92         l1Visited[strNode] = true;
93         q.offer(strNode);
94
95         int stationCnt = 1;
96         while (!q.isEmpty()) {
97             int now = q.poll();
98
99             for (int c : connected[now]) {
100                 if (l1Visited[c])
101                     continue;
102                 l1Visited[c] = true;
103                 q.offer(c);
104                 stationCnt++;
105             }
106         }
107
108         return stationCnt;
109     }
110
111     /*
112     * N개 교차로(정점), N-1개 도로(간선)
113     *
114     * 1호선은 1부터 N까지를 연결하는 간선 (지나는 모든 정점에는 역이 있음)
115     *
116     * S, E 정점(2호선)에 역 설치. 단, 1호선 역이 설치되어 있으면 안됨
117     * └ 1호선 역이 있는 곳을 한번 이상 지나가야함
```

# 손현준 소스 코드 5

```
118      * > 조건을 만족하는 S, E가 나오는 경우의 수 구하기(S, E 순서를 바꿔도 동일한 경우로 취급)
119      *
120      * 1부터 N까지 연결하면서 각각 역에서부터 연결된 모든 정점을 탐색?
121      * or 1부터 N위치까지의 역을 담아두고 각 역에서 출발하는 노드들을 모두 하나의 집합으로 본다.
122      * > 특정역으로부터 연결된 그래프를 하나의 집합으로 생각한다.
123      * 서로다른 집합에서 2개의 정점을 뽑을 경우의 수를 계산한다.
124      * └ 집합 서로서로 다 비교하기 보단, 모든 경우의 수(1호선 정점 제외)에서 각 집합에서의 경우의 수
125          ↳ 를 빼주자
126      */
126 }
```

# 서민종 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Minjong {
5     static int N;
6     static int[] tree;
7     static List<Integer>[] connection;
8     static HashSet<Integer> line1;
9     static Integer[] subTreeSizes;
10
11    public static void main(String[] args) throws IOException {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13        BufferedWriter bw = new BufferedWriter(new
14            OutputStreamWriter(System.out));
15        N = Integer.parseInt(br.readLine());
16        tree = new int[N + 1];
17        connection = new List[N + 1];
18        subTreeSizes = new Integer[N + 1];
19        for (int i = 1; i <= N; i++) {
20            connection[i] = new ArrayList<>();
21        }
22        for (int i = 0; i < N - 1; i++) {
23            StringTokenizer st = new StringTokenizer(br.readLine());
24            int u = Integer.parseInt(st.nextToken());
25            int v = Integer.parseInt(st.nextToken());
26            connection[u].add(v);
27            connection[v].add(u);
28        }
29        getTree();
30        line1 = new HashSet<>();
```

## 서민종 소스 코드 2

```
30     List<Integer> list = new ArrayList<>();
31     int current = N;
32     int previous = N;
33     while (current != 0) {
34         line1.add(current);
35         for (int child : connection[current]) {
36             if (child == tree[current] || child == previous) {
37                 continue;
38             }
39             list.add(child);
40         }
41         previous = current;
42         current = tree[current];
43     }
44     long answer = calculateC((long) N - line1.size());
45     for (int i : list) {
46         int n = getSubTreeSize(i);
47         answer -= calculateC(n);
48     }
49     bw.write(answer + "\n");
50     bw.flush();
51 }
52
53 public static long calculateC(long n) {
54     return n * (n - 1) / 2;
55 }
56
57 public static void getTree() {
58     boolean[] visited = new boolean[N + 1];
```

# 서민종 소스 코드 3

```
59         Queue<Integer> queue = new LinkedList<>();
60         queue.offer(1);
61         visited[1] = true;
62         while (!queue.isEmpty()) {
63             int current = queue.poll();
64             for (int next : connection[current]) {
65                 if (!visited[next]) {
66                     tree[next] = current;
67                     queue.offer(next);
68                     visited[next] = true;
69                 }
70             }
71         }
72     }
73
74     public static int getSubTreeSize(int parent) {
75         if (subTreeSizes[parent] == null) {
76             int result = 1;
77             for (int child : connection[parent]) {
78                 if (child == tree[parent]) {
79                     continue;
80                 }
81                 result += getSubTreeSize(child);
82             }
83             subTreeSizes[parent] = result;
84         }
85         return subTreeSizes[parent];
86     }
87 }
```

# 김준형 소스 코드 1

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Junhyeong {
5     static class Cord {
6         int n;
7         boolean visitOne;
8
9         public Cord(int n, boolean visitOne) {
10            this.n = n;
11            this.visitOne = visitOne;
12        }
13    }
14
15    static int n;
16    static long ans;
17    static List<List<Integer>> station = new ArrayList<List<Integer>>();
18    static boolean[] oneStation;
19    static List<Integer> twoStationSize = new ArrayList<Integer>();
20
21    public static void main(String[] args) throws IOException {
22        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
23        n = Integer.parseInt(br.readLine());
24        StringTokenizer st;
25
26        init();
27        for (int i = 0; i < n - 1; i++) {
28            st = new StringTokenizer(br.readLine());
29            int start = Integer.parseInt(st.nextToken());
30            int end = Integer.parseInt(st.nextToken());
```

# 김준형 소스 코드 2

```
31         station.get(start).add(end);
32         station.get(end).add(start);
33     }
34     oneStation[1] = true;
35     findOneStation(1, oneStation);
36
37     boolean[] visited = new boolean[n + 1];
38     visited[1] = true;
39     findTwoStation(1, false, visited);
40
41     long sum = 0;
42     long powSum = 0;
43     for (int i = 0; i < twoStationSize.size(); i++) {
44         int twoStation = twoStationSize.get(i);
45         sum += twoStation;
46         powSum += Math.pow(twoStation, 2);
47     }
48     ans = (long) (Math.pow(sum, 2) - powSum) / 2;
49     System.out.print(ans);
50 }
51
52     static int findTwoStation(int cur, boolean counting, boolean[] visited) {
53         int cnt = 1;
54         for (int i = 0; i < station.get(cur).size(); i++) {
55             int next = station.get(cur).get(i);
56             if (!visited[next]) {
57                 visited[next] = true;
58                 if (!oneStation[next] && !counting) {
59                     twoStationSize.add(findTwoStation(next, true, visited));
60                 }
61             }
62         }
63     }
64 }
```

# 김준형 소스 코드 3

```
60             } else {
61                 cnt += findTwoStation(next, counting, visited);
62             }
63             visited[next] = false;
64         }
65     }
66     return cnt;
67 }
68
69 static boolean findOneStation(int cur, boolean[] oneStation) {
70     if (cur == n) {
71         oneStation[cur] = true;
72         return true;
73     }
74     boolean find = false;
75     for (int i = 0; i < station.get(cur).size(); i++) {
76         int next = station.get(cur).get(i);
77         if (!oneStation[next]) {
78             oneStation[next] = true;
79             find = findOneStation(next, oneStation);
80             if (find) {
81                 oneStation[next] = true;
82                 break;
83             }
84             oneStation[next] = false;
85         }
86     }
87     return find;
88 }
```

## 김준형 소스 코드 4

```
89
90     static void init() {
91         oneStation = new boolean[n + 1];
92         for (int i = 0; i <= n; i++) {
93             station.add(new ArrayList<>());
94         }
95     }
96 }
```

# 김시온 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Sion {
5     static int b[];
6     static List<List<Integer>> g = new ArrayList<>();
7
8     static int dfs(int u, int p) {
9         int s = 1;
10        for (int v : g.get(u)) {
11            if (v != p) {
12                s += dfs(v, u);
13                b[v] = u;
14            }
15        }
16        return s;
17    }
18
19    public static void main(String[] args) throws IOException {
20        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
21        int n = Integer.parseInt(br.readLine());
22        for (int i = 0; i <= n; i++) {
23            g.add(new ArrayList<>());
24        }
25
26        for (int i = 0; i < n - 1; i++) {
27            StringTokenizer st = new StringTokenizer(br.readLine());
28            int u = Integer.parseInt(st.nextToken());
29            int v = Integer.parseInt(st.nextToken());
```

## 김시온 소스 코드 2

```
31         g.get(u).add(v);
32         g.get(v).add(u);
33     }
34
35     b = new int[n + 1];
36     dfs(1, 1);
37
38     Set<Integer> s = new HashSet<>();
39     for (int u = n; u != 0; u = b[u]) {
40         s.add(u);
41     }
42
43     List<Integer> a = new ArrayList<>();
44     for (int u : s) {
45         for (int v : g.get(u)) {
46             if (!s.contains(v)) {
47                 a.add(dfs(v, u));
48             }
49         }
50     }
51
52     int t = a.stream().mapToInt(Integer::intValue).sum();
53     long r = 0;
54     for (int v : a) {
55         r += (long) v * (t - v);
56     }
57
58     System.out.println(r / 2);
59 }
```

## 김시온 소스 코드 3

```
60  
61 }
```

# 임건애 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Keonae {
5
6     static int N;
7     static List<ArrayList<Integer>> list;
8     static boolean[] visited;
9     static boolean[] isLine1;
10
11    public static void main(String[] args) throws IOException {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13
14        N = Integer.parseInt(br.readLine());
15
16        list = new ArrayList<>();
17        for (int i = 0; i <= N; i++) {
18            list.add(new ArrayList<Integer>());
19        }
20
21        for (int i = 0; i < N - 1; i++) {
22            StringTokenizer st = new StringTokenizer(br.readLine());
23            int a = Integer.parseInt(st.nextToken());
24            int b = Integer.parseInt(st.nextToken());
25
26            list.get(a).add(b);
27            list.get(b).add(a);
28        }
29
30        isLine1 = new boolean[N + 1];
```

## 임건애 소스 코드 2

```
31         check();
32
33     visited = new boolean[N + 1];
34     for (int i = 1; i <= N; i++) {
35         if (isLine1[i]) {
36             visited[i] = true;
37         }
38     }
39
40     List<Integer> size = new ArrayList<>();
41     for (int i = 1; i <= N; i++) {
42         if (!visited[i]) {
43             size.add(BFS(i));
44         }
45     }
46
47     long result = 0, sum = 0;
48     for (long s : size) {
49         result += sum * s;
50         sum += s;
51     }
52
53     System.out.println(result);
54 }
55
56 static void check() { // 1호선 체크
57     Queue<Integer> q = new LinkedList<>();
58     q.add(1);
59     boolean[] visited = new boolean[N + 1];
```

# 임건애 소스 코드 3

```
60         visited[1] = true;
61
62         int[] parent = new int[N + 1];
63         while (!q.isEmpty()) {
64             int cur = q.poll();
65
66             if (cur == N)
67                 break;
68
69             for (int next : list.get(cur)) {
70                 if (!visited[next]) {
71                     visited[next] = true;
72                     parent[next] = cur;
73                     q.add(next);
74                 }
75             }
76         }
77
78         int track = N;
79         while (track != 0) {
80             isLine1[track] = true;
81             track = parent[track];
82         }
83     }
84
85     static int BFS(int start) {
86         visited[start] = true;
87         Queue<Integer> q = new LinkedList<>();
88         q.add(start);
```

## 임건애 소스 코드 4

```
89         int count = 1;
90         while (!q.isEmpty()) {
91             int cur = q.poll();
92
93             for (int next : list.get(cur)) {
94                 if (!visited[next]) {
95                     visited[next] = true;
96                     q.add(next);
97                     count++;
98                 }
99             }
100         }
101     }
102
103     return count;
104 }
105
106 }
```

# 전단지 돌리기(19542)

## 문제

현민이는 트리 모양의 길 위에서 오토바이를 타고 전단지를 돌리려고 한다. 현민이의 목표는 케니소프트에서 출발하여 모든 노드에 전단지를 돌리고, 다시 케니소프트로 돌아오는 것이다. 현민이는 힘이 좋기 때문에 현재 노드에서 거리가  $D$  이하인 모든 노드에 전단지를 돌릴 수 있다. 날씨가 매우 덥기 때문에, 현민이는 최소한만 이동해서 목표를 달성하고 싶다! 현민이를 위해 현민이가 이동해야 하는 총 거리를 구해주자.

## 입력

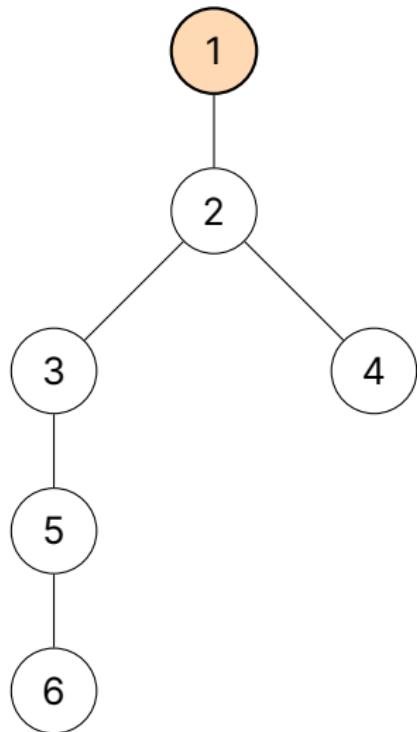
첫번째 줄에는 노드의 개수  $N(1 \leq N \leq 100\,000)$ 과 케니소프트의 위치  $S(1 \leq S \leq N)$ , 힘  $D(0 \leq D \leq N)$ 이 주어진다.

두 번째 줄부터  $N$  번째 줄까지, 트리의 간선 정보를 의미하는 두 자연수  $x, y$ 가 공백으로 구분되어 주어진다. 이는  $x$  번 노드와  $y$  번 노드가 연결되어 있음을 의미한다. ( $1 \leq x, y \leq N, x \neq y$ ) 주어지는 연결관계는 트리를 구성하며, 모든 간선의 길이는 1이다.

## 출력

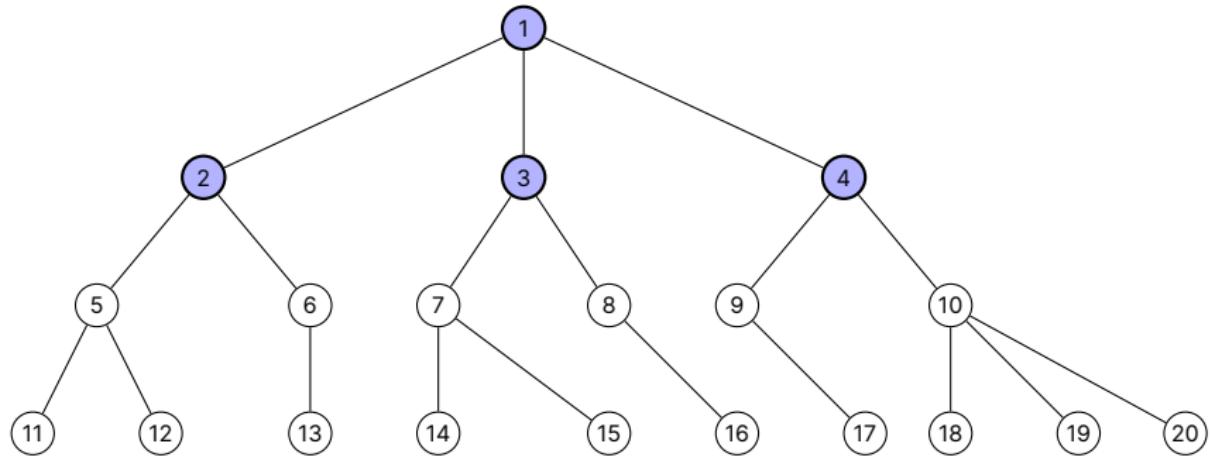
현민이가 목표를 완수하기 위해 이동해야 하는 최소 거리를 출력하여라.

## 예제 입력



예제 출력: **6**

# 트리의 높이



높이가 2 이상인 정점

# 원찬혁 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Chanhyeok {
5     static int n, s, d;
6     static List<Integer> e[];
7     static boolean visited[];
8     static int dep[];
9     static int res;
10
11    public static void main(String[] args) throws Exception {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13        StringTokenizer st = new StringTokenizer(br.readLine());
14        int u, v;
15        n = Integer.parseInt(st.nextToken());
16        s = Integer.parseInt(st.nextToken()) - 1;
17        d = Integer.parseInt(st.nextToken());
18        visited = new boolean[n];
19        dep = new int[n];
20        e = new List[n];
21        for (int i = 0; i < n; i++)
22            e[i] = new ArrayList<>();
23        for (int i = 0; i < n - 1; i++) {
24            st = new StringTokenizer(br.readLine());
25            u = Integer.parseInt(st.nextToken()) - 1;
26            v = Integer.parseInt(st.nextToken()) - 1;
27            e[u].add(v);
28            e[v].add(u);
29        }
30        visited[s] = true;
```

## 원찬혁 소스 코드 2

```
31         dfs(s);
32         sol(s);
33         res *= 2;
34         System.out.println(res);
35     }
36
37     static int dfs(int v) {
38         dep[v] = 1;
39         for (int it : e[v]) {
40             if (!visited[it]) {
41                 visited[it] = true;
42                 dep[v] = Math.max(dep[v], dfs(it) + 1);
43                 visited[it] = false;
44             }
45         }
46         return dep[v];
47     }
48
49     static void sol(int v) {
50         for (int it : e[v]) {
51             if (!visited[it]) {
52                 if (dep[it] > d) {
53                     visited[it] = true;
54                     res++;
55                     sol(it);
56                     visited[it] = false;
57                 }
58             }
59         }
    }
```

## 원찬혁 소스 코드 3

```
60      }
61 }
```

# 정의찬 소스 코드 1

```
1 import java.io.*;
2 import java.util.*;
3
4 class Uichan {
5     static boolean[] visited;
6     static List<Integer>[] nexts;
7     static int dist;
8     static int d;
9
10    public static void main(String[] args) throws NumberFormatException,
11        ↵ IOException {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13        StringBuilder sb = new StringBuilder();
14        StringTokenizer st = new StringTokenizer(br.readLine());
15
16        int n = Integer.parseInt(st.nextToken()); // 노드개수
17        int s = Integer.parseInt(st.nextToken()); // 위치
18        d = Integer.parseInt(st.nextToken()); // 힘
19
20        visited = new boolean[n + 1];
21        nexts = new List[n + 1];
22        dist = 0;
23
24        for (int i = 1; i <= n; i++)
25            nexts[i] = new ArrayList<>();
26
27        for (int i = 0; i < n - 1; i++) {
28            st = new StringTokenizer(br.readLine());
29            int n1 = Integer.parseInt(st.nextToken());
30            int n2 = Integer.parseInt(st.nextToken());
```

## 정의찬 소스 코드 2

```
30         nexts[n1].add(n2);
31         nexts[n2].add(n1);
32     }
33
34     dfs(s);
35
36     System.out.println(dist * 2);
37 }
38
39 static int dfs(int current) {
40     visited[current] = true;
41     int ret = 0;
42     for (int next : nexts[current])) {
43         if (visited[next])
44             continue;
45         int nextNodeDepth = dfs(next);
46         if (nextNodeDepth > d)
47             dist++;
48         ret = Math.max(ret, nextNodeDepth);
49     }
50     return ret + 1;
51 }
52 }
```

# 손현준 소스 코드 1

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import java.util.ArrayDeque;
5 import java.util.ArrayList;
6 import java.util.List;
7 import java.util.Queue;
8 import java.util.StringTokenizer;
9
10 public class Hyeonjun {
11     static int N, S, D;
12     static List<Integer>[] graph;
13     static int[] distance;
14     static Queue<Integer> leafNodes;
15     static boolean[] bikeVisited;
16
17     public static void main(String[] args) throws IOException {
18         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
19         StringTokenizer st;
20
21         // 기본값 입력 받기
22         st = new StringTokenizer(br.readLine(), " ");
23         N = Integer.parseInt(st.nextToken());
24         S = Integer.parseInt(st.nextToken());
25         D = Integer.parseInt(st.nextToken());
26
27         // 간선 정보 입력 받기
28         graph = new ArrayList[N];
29         for (int i = 0; i < N; i++)
30             graph[i] = new ArrayList<>();
```

## 손현준 소스 코드 2

```
31     for (int i = 0; i < N - 1; i++) {
32         st = new StringTokenizer(br.readLine(), " ");
33         int from = Integer.parseInt(st.nextToken());
34         int to = Integer.parseInt(st.nextToken());
35         graph[from - 1].add(to - 1);
36         graph[to - 1].add(from - 1);
37     }
38
39     distance = new int[N]; // 회사의 위치로부터 떨어져 있는 거리 저장
40     leafNodes = new ArrayDeque<>();
41     markDistances(S - 1, 0, new boolean[N]);
42
43     int result = 0;
44     bikeVisited = new boolean[N];
45     while (!leafNodes.isEmpty()) {
46         int now = leafNodes.poll();
47         result += getDistance(now);
48     }
49
50     System.out.println(result * 2);
51 }
52
53 static void markDistances(int pos, int dist, boolean[] visited) {
54     visited[pos] = true;
55     distance[pos] = dist;
56
57     boolean hasChild = false;
58     for (int g : graph[pos]) {
59         if (visited[g])
```

# 손현준 소스 코드 3

```
60         continue;
61         hasChild = true;
62         markDistances(g, dist + 1, visited);
63     }
64
65     if (!hasChild)
66         leafNodes.offer(pos);
67 }
68
69 static int getDistance(int start) {
70     int target = -1;
71
72     int now = start;
73     int dist = 0; // start(리프)에서 now까지의 거리
74
75     while (distance[now] > 0) { // 회사까지 최단경로 이동
76         if (dist >= D) { // 현민이가 방문 할 필요 없는건 스킵
77             if (bikeVisited[now]) {
78                 if (target == -1)
79                     return 0;
80                 target -= distance[now];
81                 break;
82             }
83             if (target == -1)
84                 target = distance[now];
85             bikeVisited[now] = true;
86         }
87
88         for (int g : graph[now]) { // 부모로 1칸 이동
```

## 손현준 소스 코드 4

```
89             if (distance[g] == distance[now] - 1) {
90                 now = g;
91                 break;
92             }
93         }
94
95         dist++;
96     }
97
98     return (target == -1) ? 0 : target;
99 }
100
101 /*
102 * 입력 받을때 연결된 간선이 하나밖에 없다면 끝 노드 임으로 큐에 저장
103 *
104 * 1. 출발점으로 부터의 모든 노드의 깊이 기록
105 * 2. 끝노드를 큐에서 하나씩 꺼내면서 그래프 역탐색
106 * BFS로 역탐색 할 때, 간선개수 V^2면 시간초과 될 듯..
107 */
108 }
```

# 서민종 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Minjong {
5     static int N, S, D;
6     static List<Integer>[] connections;
7     static int[] tree;
8     static int[] depths;
9     static int[] leafDistance;
10    static int answer;
11
12    public static void main(String[] args) throws IOException {
13        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
14        BufferedWriter bw = new BufferedWriter(new
15            OutputStreamWriter(System.out));
16        StringTokenizer st = new StringTokenizer(br.readLine());
17        N = Integer.parseInt(st.nextToken());
18        S = Integer.parseInt(st.nextToken());
19        D = Integer.parseInt(st.nextToken());
20        connections = new List[N + 1];
21        for (int i = 1; i <= N; i++) {
22            connections[i] = new ArrayList<>();
23        }
24        for (int i = 0; i < N - 1; i++) {
25            st = new StringTokenizer(br.readLine());
26            int x = Integer.parseInt(st.nextToken());
27            int y = Integer.parseInt(st.nextToken());
28            connections[x].add(y);
29            connections[y].add(x);
        }
    }
}
```

## 서민종 소스 코드 2

```
30         answer = 0;
31         getTree();
32         getLeafDistance();
33         bw.write(answer + "\n");
34         bw.flush();
35     }
36
37     public static void getLeafDistance() {
38         leafDistance = new int[N + 1];
39         Arrays.fill(leafDistance, -1);
40         for (int i = 1; i <= N; i++) {
41             leafDistance[i] = getLeafDistance(i);
42             if (i != S && leafDistance[i] >= D) {
43                 answer += 2;
44             }
45         }
46     }
47
48     public static int getLeafDistance(int n) {
49         if (leafDistance[n] == -1) {
50             if (connections[n].size() == 1 && n != S) {
51                 leafDistance[n] = 0;
52             } else {
53                 int max = 0;
54                 for (int next : connections[n]) {
55                     if (next == tree[n]) {
56                         continue;
57                     }
58                     max = Math.max(max, getLeafDistance(next));
59                 }
60             }
61         }
62         return leafDistance[n];
63     }
64 }
```

## 서민종 소스 코드 3

```
59             }
60         leafDistance[n] = max + 1;
61     }
62 }
63 return leafDistance[n];
64 }
65
66 public static void getTree() {
67     tree = new int[N + 1];
68     depths = new int[N + 1];
69     Queue<Integer> queue = new LinkedList<>();
70     queue.offer(S);
71     while (!queue.isEmpty()) {
72         int parent = queue.poll();
73         for (int child : connections[parent]) {
74             if (child == tree[parent]) {
75                 continue;
76             }
77             tree[child] = parent;
78             depths[child] = depths[parent] + 1;
79             queue.offer(child);
80         }
81     }
82 }
83 }
```

# 임건애 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Keonae {
5
6     static int N, S, D;
7     static List<ArrayList<Integer>> graph;
8     static boolean[] visited;
9     static int count;
10
11    public static void main(String[] args) throws IOException {
12        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13        StringTokenizer st = new StringTokenizer(br.readLine());
14
15        N = Integer.parseInt(st.nextToken());
16        S = Integer.parseInt(st.nextToken());
17        D = Integer.parseInt(st.nextToken());
18
19        graph = new ArrayList<>();
20        for (int i = 0; i <= N; i++) {
21            graph.add(new ArrayList<>());
22        }
23
24        for (int i = 1; i < N; i++) {
25            st = new StringTokenizer(br.readLine());
26            int x = Integer.parseInt(st.nextToken());
27            int y = Integer.parseInt(st.nextToken());
28
29            graph.get(x).add(y);
30            graph.get(y).add(x);
31        }
32    }
33}
```

## 임건애 소스 코드 2

```
31         }
32
33         count = 0;
34         visited = new boolean[N + 1];
35         DFS(S);
36
37         System.out.println(count);
38     }
39
40     static int DFS(int start) {
41         visited[start] = true;
42         int max = 0;
43
44         for (int next : graph.get(start)) {
45             if (!visited[next]) {
46                 int depth = DFS(next);
47
48                 if (depth >= D)
49                     count += 2;
50
51                 max = Math.max(max, depth + 1);
52             }
53         }
54
55         return max;
56     }
57
58 }
```

# 김시온 소스 코드 1

```
1 import java.util.*;
2 import java.io.*;
3
4 public class Sion {
5     static List<List<Integer>> g = new ArrayList<>();
6     static int[] h;
7
8     static void dfs(int u, int p) {
9         for (int v : g.get(u)) {
10             if (v != p) {
11                 dfs(v, u);
12                 h[u] = Math.max(h[u], h[v] + 1);
13             }
14         }
15     }
16
17     public static void main(String[] args) throws IOException {
18         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
19         StringTokenizer st = new StringTokenizer(br.readLine());
20         int n = Integer.parseInt(st.nextToken());
21         int s = Integer.parseInt(st.nextToken());
22         int d = Integer.parseInt(st.nextToken());
23
24         for (int i = 0; i <= n; i++) {
25             g.add(new ArrayList<>());
26         }
27
28         for (int i = 0; i < n - 1; i++) {
29             st = new StringTokenizer(br.readLine());
30             int u = Integer.parseInt(st.nextToken());
```

## 김시온 소스 코드 2

```
31         int v = Integer.parseInt(st.nextToken());
32         g.get(u).add(v);
33         g.get(v).add(u);
34     }
35
36     h = new int[n + 1];
37     dfs(s, s);
38
39     int c = 0;
40     for (int u = 1; u <= n; u++) {
41         if (h[u] >= d) {
42             c++;
43         }
44     }
45     System.out.println(2 * Math.max(c - 1, 0));
46 }
47
48 }
49 }
```