```
function solution1() {
    let get = gets().trim();
    //기본 그래프 오브젝트형식으로 입력
    let obj = {
        'A': ['B'],
        'B': ['A', 'C', 'H'],
        'C': ['B', 'D'],
        'D': ['C', 'E', 'G'],
        'E': ['D', 'F'],
        'F': ['E'],
        'G': ['D'],
        'H': ['B', 'I', 'J', 'M'],
        'I': ['H'],
        'J': ['H', 'K'],
        'K': ['J', 'L'],
        'L': ['K'],
        'M': ['H']
    };
    //기본적인 bfs
    function BFS(g, start) {
        let visit = [];
        let need = [];

        need.push(start);

        while (need.length != 0) {
            let node = need.shift();
            if (!visit.includes(node)) {
                visit.push(node);
                need = [...need, ...g[node]];
            }
        }
        return visit;
    };
    //기본적인 dfs
    function DFS(g, start) {
        let need = [];
        let visit = [];
```

```
        need.push(start);

        while (need.length != 0) {
            let node = need.pop();
            if (!visit.includes(node)) {
                visit.push(node);
                need = [...need, ...g[node]];
            }
        }
        return visit;
    }
    //출력
    print(BFS(obj, get).join(' '));
    print(DFS(obj, get).join(' '));
}

function solution2() {
    let n = Number(gets().trim());
    let obj = [];
    //입력
    for (let i = 0; i < n; i++) {
        let k = gets().split(' ');
        for (let _ of k) {
            k.push(Number(k.shift()));
        }
        obj.push(k);
    }
    let ans = "False";

    function DFS(g, start, visit) {
        visit[start] = true;
        //결괏값에 도달하면 ans에 true를 넣고 리턴
        if (start == (n - 1)) {
            ans = "True";
            return;
        }
        g[start].forEach(i => {
            if (!visit[i]) {
                DFS(g, i, visit);
```

```
                }
            })
        }
        let visit = Array.from({
            length: n
        }).fill(false);
        DFS(obj, 0, visit);
        print(ans);
}

function solution3() {
        let f = gets().split(' ');
        let obj = [];
        //입력
        for (let k = 0; k < Number(f[1]); k++) {
            let j = gets().split(' ');
            obj.push([Number(j[0]), Number(j[1])]);
        }

        //DFS
        function DFS(g, v, visit) {
            visit[v] = true;

            for (let i = 0; i < g.length; i++) {
                if (g[i][0] == v) {
                    if (!visit[g[i][1]]) {
                        DFS(g, g[i][1], visit);
                    }
                }
                if (g[i][1] == v) {
                    if (!visit[g[i][0]]) {
                        DFS(g, g[i][0], visit);
                    }
                }
            }
        }
        let ans = [];
        for (let i = 0; i < Number(f[1]); i++) {
            let visit = Array(Number(f[0])).fill(false);
```

```
        //간선 복사
        let ob = obj.slice();
        //특정 간선을 제거했을때 그래프 전체 순회가 제대로 되는지
        ob.splice(i, 1);
        DFS(ob, 0, visit);
        //되지 않으면 정답배열에 추가
        if (visit.includes(false)) {
            ans.push(obj[i]);
        }
    }
    //출력을 위한 정렬
    ans = ans.sort((a, b) => {
        if (a[0] == b[0]) {
            return a[1] - b[1];
        } else {
            return a[0] - b[0];
        }
    })
    for (let i of ans) {
        print(i.join(' '));
    }

}
```