```
function solution1_2_sol() {
    let input = gets().split(' ');
    let node = gets().split(' ');
    let nodes = node.slice();
    let ans = 0;
    //입력
    let edge = Array.from(Array(Number(input[1])), () => Array(3));
    for (let i = 0; i < Number(input[1]); i++) {
        let line = gets().split(' ');
        edge[i][0] = Number(line[2]);
        edge[i][1] = line[0];
        edge[i][2] = line[1];
    }

    edge = edge.sort((a, b) => {
        return a[0] >= b[0] ? 1 : -1;
    });

    function UnionNode(parent, child) {
        for (let i = 0; i < nodes.length; i++) {
            if (nodes[i] == child) {
                nodes[i] = parent;
            }
        }
    }
    for (let i of edge) {
        if (nodes[node.indexOf(i[1])] == nodes[node.indexOf(i[2])]) continue;
        if (nodes[node.indexOf(i[1])] != i[1]) {
            UnionNode(nodes[node.indexOf(i[1])], i[2]);
        } else if (nodes[node.indexOf(i[2])] != i[2]) {
            UnionNode(nodes[node.indexOf(i[2])], i[1]);
        } else {
            UnionNode(i[1], i[2]);
        }
        ans += i[0];
    }
    print(ans);
}

function solution2_1_sol() {
    let input = gets().split(' ');
    let node = gets().split(' ');
    //입력
    let edge = Array.from(Array(Number(input[1])), () => Array(3));
    for (let i = 0; i < Number(input[1]); i++) {
        let line = gets().split(' ');
        edge[i][0] = Number(line[2]);
        if (line[0] > line[1]) {
            edge[i][1] = line[0];
            edge[i][2] = line[1];
        } else {
            edge[i][1] = line[1];
            edge[i][2] = line[0];
        }
    }
```

```
    edge = edge.sort((a, b) => {
        if (a[0] != b[0]) {
            return a[0] - b[0];
        } else {
            if (a[1] == b[1]) {
                return a[2] > b[2];
            } else {
                return a[1] > b[1] ? 1 : -1;
            }
        }
    });

    function union(li) {
        let ans = 0;
        let nodes = node.slice();
        for (let i of li) {
            if (nodes[node.indexOf(i[1])] == nodes[node.indexOf(i[2])]) continue;
            if (nodes[node.indexOf(i[1])] != i[1]) {
                UnionNode(nodes[node.indexOf(i[1])], i[2]);
            } else if (nodes[node.indexOf(i[2])] != i[2]) {
                UnionNode(nodes[node.indexOf(i[2])], i[1]);
            } else {
                UnionNode(i[1], i[2]);
            }
            ans += i[0];
        }

        function UnionNode(parent, child) {
            for (let i = 0; i < nodes.length; i++) {
                if (nodes[i] == child) {
                    nodes[i] = parent;
                }
            }
        }
        return ans;
    }
    let m = union(edge);
    let ansarr = [];
    for (let j = 0; j < edge.length; j++) {
        let cop = edge.slice();
        cop.splice(j, 1);
        ansarr.push(union(cop));
    }
    ansarr = ansarr.filter((v, i) => {
        return ansarr.indexOf(v) === i
    });
    ansarr.sort((a, b) => a - b);
    if (ansarr[0] == m) {
        print(ansarr[1]);
    } else {
        print(ansarr[0]);
    }
}

function solution3_1() {
    let input = gets().split(' ');
```

```
    let self = gets().split(' ');
    //간선의 최소 길이
    let min = Array(self.length + 1).fill(101);
    //최소로 가는 간선 => index => value
    let select = Array(self.length + 1).fill(0);
    let graph = Array.from(Array(self.length + 1), () => Array(self.length + 1).fill(101));
    for (let i = 0; i < self.length; i++) {
        graph[i + 1][i + 1] = Number(self[i]);
    }
    for (let i = 0; i < Number(input[1]); i++) {
        let temp = gets().split(' ');
        graph[Number(temp[0])][Number(temp[1])] = Number(temp[2]);
        graph[Number(temp[1])][Number(temp[0])] = Number(temp[2]);
    }
    //graph.length == self.length + 1;
    //graph[0][0] == 101
    //graph[0][...] == 101
    //graph[...][0] == 101
    //graph의 최대값 == 101
    let ans = 0;
    //행에서의 최소값과 인덱스를 리턴
    function get_line_min(line) {
        let arr = [101, 0];
        for (let i = 1; i <= line.length; i++) {
            //비용이 같으면 선택이 안되었던것을 고름
            if (arr[0] > line[i] ) {
                arr[0] = line[i];
                arr[1] = i;
            }
        }
        return arr;
    }
    function prim(start, end, graph) {
        //행으로
        for (let i = start; i <= end; i++) {
            let min_line = get_line_min(graph[i]);
            min[i] = min_line[0];
            select[i] = min_line[1];
        }
    }
    prim(1, self.length, graph);
    //위성 기지국 설치
    //앞이 좋을경우
    //뒤가 좋을경우
    for(let i = 1; i <= self.length; i++){
        if(select[i] == i) ans += graph[i][i];
        else ans += graph[i][select[i]];
    }
    print(ans);

}

function solution3_2() {
```

```
let input = gets().split(' ');
let town = Number(input[0]);
let amount = Number(input[1]);
let nodes = gets().split(' ');
let group = Array(town + 1);
let select = Array(town + 1).fill(false);
let min = Array(town + 1);
let graph = Array.from(Array(town + 1), () => Array(town + 1).fill(101));
for (let i = 1; i <= town; i++) {
    graph[i][i] = Number(nodes[i - 1]);
    group[i] = i;
    min[i] = graph[i][i];
}
for (let i = 0; i < amount; i++) {
    let temp = gets().split(' ');
    graph[Number(temp[0])][Number(temp[1])] = Number(temp[2]);
    graph[Number(temp[1])][Number(temp[0])] = Number(temp[2]);
}

function get_min(n) {
    let v;
    for (let i = 1; i <= n; i++) {
        if (select[i] == false) {
            v = i;
            break;
        }
    }
    for (let i = 1; i <= n; i++) {
        if (select[i] == false && (min[i] < min[v])) {
            v = i;
        }
    }
    return v;
}
let ansr = [];
function prim(n) {
    for (let i = 1; i <= n; i++) {
        let u = get_min(n);
        select[u] = true;

        if (min[u] == 101) return;
        ansr.push(u);
        for (let v = 1; v <= n; v++) {
            if (graph[u][v] != 101) {
                if (select[v] == false && graph[u][v] < min[v]) {
                    min[v] = graph[u][v];
                    group[v] = u;
                }
            }

        }
    }
}
min[0] = 0;
prim(amount);
let ans = min.reduce((a, b) => a + b, 0);
print(ans, ' ', ...ansr, ' ', ...group);
```

```
}
```