/\*\*

\* @param {string[][]} equations

\* @param {number[]} values

\* @param {string[][]} queries

\* @return {number[]}

\*/

var calcEquation = function(equations, values, queries) {

// 1. 构建图的邻接表

const graph = {};

// 填充邻接表

for (let i = 0; i < equations.length; i++) {

const [A, B] = equations[i];

const value = values[i];

if (!graph[A]) graph[A] = [];

if (!graph[B]) graph[B] = [];

graph[A].push([B, value]);

graph[B].push([A, 1 / value]);

}

// 2. 处理每个查询

const result = [];

for (const [C, D] of queries) {

// 如果变量不存在于图中

if (!graph[C] || !graph[D]) {

result.push(-1.0);

continue;

}

// 如果查询的是同一个变量

if (C === D) {

result.push(1.0);

continue;

}

// BFS查找路径

const visited = new Set();

const queue = [[C, 1.0]];

let found = false;

while (queue.length > 0 && !found) {

const [current, product] = queue.shift();

visited.add(current);

for (const [neighbor, value] of graph[current]) {

if (neighbor === D) {

result.push(product \* value);

found = true;

break;

}

if (!visited.has(neighbor)) {

queue.push([neighbor, product \* value]);

}

}

}

if (!found) {

result.push(-1.0);

}

}

return result;

};

