TRƯỜNG ĐẠI HỌC KĨ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TOÁN RỜI RẠC

Sinh viên thực hiện	**:**	**Nguyễn Văn Thứ**
MSSV	**:**	**K225480106062**
Lớp	**:**	**K58KMT-K01**
Giáo viên hướng dẫn	**:**	**Ths. Đỗ Duy Cốp**

Thái Nguyên - 2024.

TRƯỜNG ĐẠI HỌC KĨ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TOÁN RỜI RẠC

Sinh viên thực hiện : Nguyễn Văn Thứ

MSSV : K225480106062

Lớp : K58KMT-K01

Giáo viên hướng dẫn : Ths. Đỗ Duy Cốp

Thái Nguyên - 2024.

TRƯỜNG ĐHKTCN
KHOA ĐIỆN TỬ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập-Tự do-Hạnh phúc

.....🕮.....

BÀI TẬP LỚN

MÔN HỌC: TOÁN RỜI RẠC
BỘ MÔN CÔNG NGHỆ THÔNG TIN

Sinh viên thực hiện : Nguyễn Văn Thứ
MSSV : K225480106062
Ngành : Kỹ thuật máy tính
Lớp : K58KMT-K01
Giáo viên hướng dẫn : Ths. Đỗ Duy Cốp
Ngày giao đề: 07/03/2024 Ngày hoàn thành: 14/03/2024
Tên đề tài:
Yêu cầu: Mỗi sinh viên làm bài riêng cụ thể bao gồm có 7 phần, mỗi phần làm một bài toán theo danh sách đã được phân công. Mỗi sinh viên làm riêng và sau khi hoàn thiện bài làm in thành quyển báo cáo để nộp lại.

Thái Nguyên, ngày.... tháng.... năm 2024
GIÁO VIÊN HƯỚNG DẪN
(Kí và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Thái Nguyên, ngày.... tháng.... năm 2024

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN CHẤM

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Thái Nguyên, ngày.... tháng.... năm 2024

GIÁO VIÊN CHẤM

(Ký và ghi rõ họ tên)

Mục lục

I. PHẦN ĐẦU

1. Giới thiệu thông tin cá nhân

Họ và tên: Nguyễn Văn Thứ.
Ngày sinh: 16/06/2004
Giới tính: Nam
Sinh viên năm thứ 2, hiện tại đang theo học ngành Kĩ thuật máy tính, tại Trường Đại học kĩ thuật công nghiệp.
Email: mn9103541@gmail.com
Địa chỉ: Hương Lâm – Hiệp Hòa - Bắc Giang.

2. Hướng dẫn làm bài

Với mỗi bài toán được phân công, cần thực hiện các bước sau (lặp lại cho 7 bài):

1. Trình bày tên bài toán
2. Phân tích bài toán
3. Lập trình giải quyết bài toán bằng JavaScript
4. Chụp lại kết quả
5. Đánh giá kết quả

3. Dưới đây là yêu cầu đề bài được giao:

STT	Mã SV	Họ lót	Tên	Tên lớp	Phần I	Phần II	Phần III	Phần IV	Phần V	Phần VI	Phần VII
65	K225480106062	Nguyễn Văn	Thứ	K58KMT.K01	5	2	16	5	1	2	1

II. PHẦN NỘI DUNG

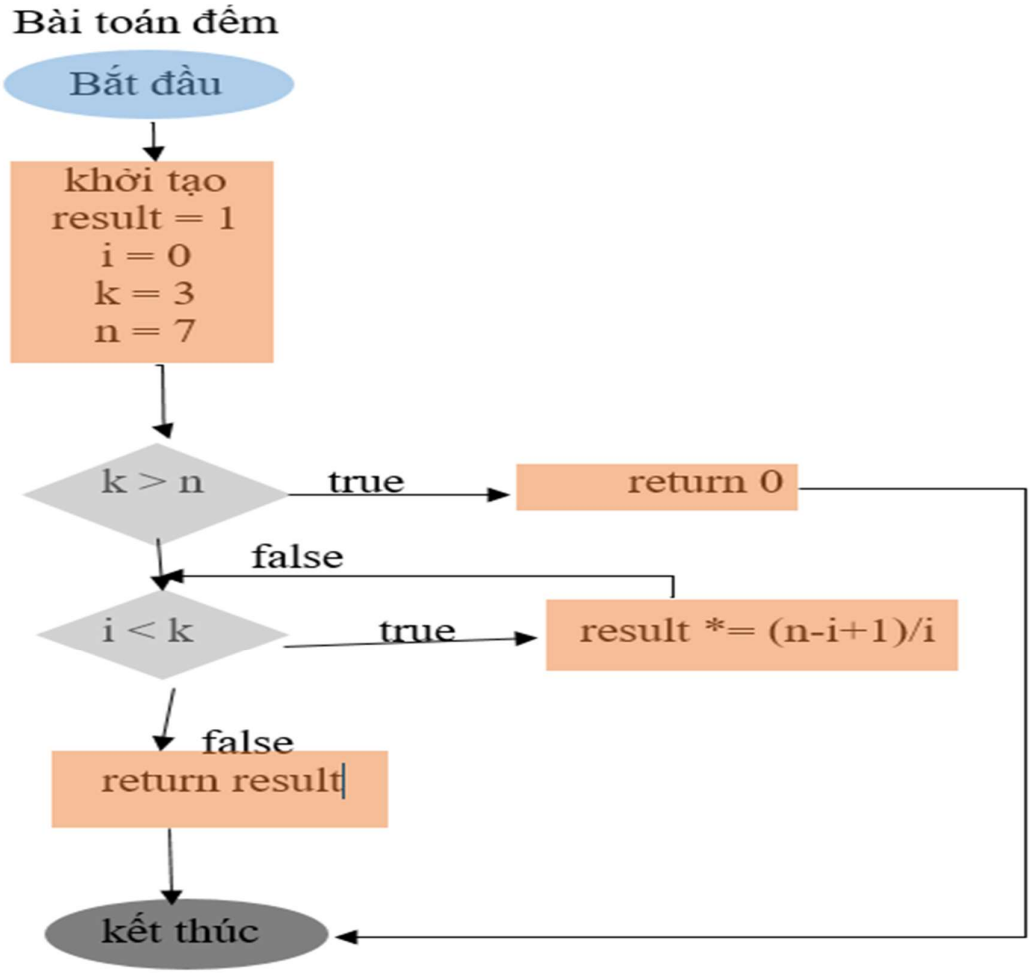
1. Phần I. Bài toán đếm

1.1 Đề bài

(5) Bạn muốn chọn 3 đám hoa từ 7 loại hoa khác nhau. Có bao nhiêu cách để chọn?

1.2 Phân tích bài toán

Để giải quyết bài toán này, chúng ta có thể sử dụng công thức tổ hợp, vì chúng ta đang xem xét việc chọn từ một dãy gồm 3 đám hoa từ 10 loại hoa khác nhau. k phần tử từ một tập hợp gồm n phần tử mà không quan tâm tới thứ tự của các phần tử đó. Công thức tổ hợp được biểu diễn như sau: Công thức tổ hợp được thể hiện để tính số cách chọn C(n,k)=n!/(k!.(n-k)!). Áp dụng vào bài toán của chúng ta: C(7,3)=7!/(3!.(7-3)!). Vậy có 35 cách chọn dãy gồm 3 đám hoa từ 7 loại hoa khác nhau.



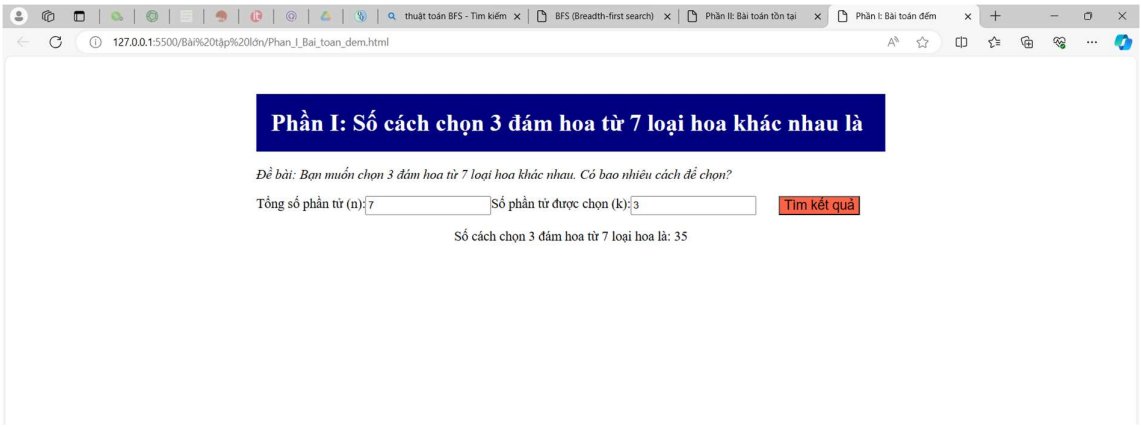
Hình 1.2 – sơ đồ khối phân tích bài toán đếm

1.3 Lập trình giải quyết bài toán bằng JavaScript

```
<body>
    <div id="wrapper">
    <h1>Phần I: Số cách chọn 3 đám hoa từ 7 loại hoa khác nhau là</h1>
    <p><i>Đề bài: Bạn muốn chọn 3 đám hoa từ 7 loại hoa khác nhau. Có bao
nhiêu cách để chọn?
    </i>
    </p>
    <div id="nhap">
    <label for="totalElements">Tổng số phần tử (n): </label>
    <input type="number" id="totalElements" min="0" value="7">

    <label for="selectedElements">Số phần tử được chọn (k): </label>
    <input type="number" id="selectedElements" min="0" value="3">

    <button onclick="tinhSoCachChon()">Tìm kết quả</button>
</div>
    <div id="result"></div>

    <script>
        function tinhGiaiThua(n) {
```

```
            if (n === 0 || n === 1) {
                return 1;
            }
            return n * tinhGiaiThua(n - 1);
        }

        function tinhToHop(n, k) {
            return tinhGiaiThua(n) / (tinhGiaiThua(k) * tinhGiaiThua(n - k));
        }

        function tinhSoCachChon() {
            var tongSoLoaiHoa =
parseInt(document.getElementById("totalElements").value);
            var soLoaiHoaDuocChon =
parseInt(document.getElementById("selectedElements").value);

            var resultElement = document.getElementById("result");

            if (isNaN(tongSoLoaiHoa) || isNaN(soLoaiHoaDuocChon) ||
tongSoLoaiHoa < 0 || soLoaiHoaDuocChon < 0) {
                resultElement.innerHTML = "Vui lòng nhập số nguyên không âm.";
            } else {
                var soCachChon = tinhToHop(tongSoLoaiHoa, soLoaiHoaDuocChon);
                resultElement.innerHTML = "Số cách chọn " + soLoaiHoaDuocChon
+ " đám hoa từ " + tongSoLoaiHoa + " loại hoa là: " + soCachChon;
            }
        }
    </script>
    </div>
</body>
</html>
```

1.4 Chụp lại kết quả



Hình 1.4 - kết quả cho bài toán đếm

1.5 đánh giá kết quả

Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

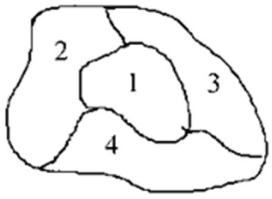
2. Phần II. Bài toán tồn tại

2.1 Đề bài

(2) Bài toán 4 mầu: Cho bản đồ gồn N quốc gia (mô tả bằng ma trận kề: 2 nước I và j là hàng xóm với nhà nhau thì Cij=1, ngược lại Cij=0). Hãy tô bằng 4 màu bản đồ sau cho 2 nước là hàng xóm với nhau thì khác màu nhau?

2.2 Phân tích bài toán

Có n quốc gia trên bản đồ, tìm cách dùng 4 màu để tô cho mỗi quốc gia, sao cho hàng xóm thì khác màu. Ma trận kề Cij=1 nếu nước i là hàng xóm với nước j: Chúng ta sử dụng định lý Fermat thì bài toán 4 màu là một bài toán như vậy. Bài toán có thể phát biểu trực quan như sau: Chứng minh rằng mọi bản đồ trên mặt phẳng đều có thể tô bằng 4 màu sao cho không có hai nước láng giềng nào bị tô bởi cùng một màu. Chú ý rằng, ta xem như mỗi nước là một vùng liên thông và hai nước được gọi là láng giềng nếu chúng có chung biên giới là một đường liên tục.



Hình 2.2 - Bản đồ không tô được bởi ít nhất hơn 4 mầu

Con số 4 không phải là ngẫu nhiên. Người ta đã chứng minh được rằng mọi bản đồ đều được tô với số mà hơn 4, còn với số màu ít hơn 4 thì không tô được, chẳng hạn bản đồ gồm 4 nước trên hình 1 không thể tô được với số màu ít hơn 4.

2.3 Lập trình giải quyết bài toán bằng JavaScrip

```
<title>Phần II: Bài toán tồn tại</title>
    <script>
        var n, C, x = [],d
        function show_kq() {
            var kq = document.getElementById('ketqua')
            kq.innerHTML += 'Nghiệm '+(++d)+': Cách tô màu: ' + x + '<br>'
        }

        function check_hx_khac_mau_nhau(i) {
            //đã tô màu cho các nước từ x[0]..x[i]:
            //cần check cặp ij mà Cij=1 xem màu khác nhau ko?
            for (var j= 0; j < i; j++)
                if (C[i][j] == 1 && x[i] == x[j])
                    return false //ko được
            return true //được
        }

        var Color=['ko biết','Xanh','Đỏ','Tím','Vàng']
        //tô 1 màu cho x[i]
        function to_mau(i) {
```

```
            for (var mau = 1; mau <= 4; mau++) {
                x[i] = Color[mau]
                if (check_hx_khac_mau_nhau(i)) { //nhánh cận
                    if (i == n - 1) {
                        //đã đến thằng cuối cùng
                        show_kq() //suy biến: ko đệ quy
                    } else {
                        to_mau(i + 1) //đệ quy
                    }
                }
                x[i] = 0 // quay lui
            }
        }
        function giai_toan() {
            //chuyển dữ liệu từ ma-tran-ke vào mảng 2 chiều C
            var s = document.getElementById('ma-tran-ke').innerText
            C = s.split('\n'); // C là ma trận kề ==mảng 2 chiều
            n = C.length; //số quốc gia
            for (var i = 0; i < n; i++) {
                x[i] = 0 // khởi tạo chưa tô mầu cho qgia nào
            }
            d = 0
            to_mau(0) //tô 1 màu cho quốc gia đầu tiên
        }
    </script>
</head>
<body>
    <div id="wrapper">
    <h1>
        Phần II: Bài toán tô màu bản đồ
        </h1>
    <p><i>
    Đề bài: Cho bản đồ gồn N quốc gia (mô tả bằng ma trận kề: 2 nước I và j là hàng xóm với nhà nhau thì Cij=1, ngược lại Cij=0).
    Hãy tô bằng 4 màu bản đồ sau cho 2 nước là hàng xóm với nhau thì khác màu nhau?
</i>
</p>
    <pre id="ma-tran-ke" contenteditable="true">
11111
11101
11111
10111
11111
</pre>
    <button onclick="giai_toan()">Tìm cách tô màu</button>
    <div id="ketqua"></div>
</div>
</body>
</html>
```

2.4 Chụp lại kết quả

Hình 2.4 - kết quả cho bài toán tồn tại

2.5 đánh giá kết quả

Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

3. Phần III. Bài toán liệt kê

3.1 Đề bài

(16) Liệt kê tất cả các cách chia 5 viên bi vào 3 hộp, mỗi hộp có thể chứa từ 0 đến 5 viên bi?

3.2 Phân tích bài toán

Số biến viên bi trong mỗi hộp. Vì mỗi hộp có thể chứa từ 0 đến 5 viên bi, nên chúng ta cần 3 biến để đại diện cho số bi viên bi trong mỗi hộp. Tổng số viên bi phải là 5, vì chúng ta đang chia 5 viên bi vào 3 hộp. Chúng ta sẽ duyệt qua tất cả các trường hợp có thể của 3 biến, với mỗi biến có giá trị từ 0 đến 5, và kiểm tra rằng tổng số viên bi là 5.

Mỗi khi tìm thấy một trường hợp hợp lệ, chúng ta sẽ lưu trữ nó và sau đó hiển thị kết quả cuối cùng.

Dưới đây là một ví dụ về một số cách chia 5 viên bi vào 3 hộp:

Hộp 1: 0 viên, Hộp 2: 0 viên, Hộp 3: 5 viên

Hộp 1: 0 viên, Hộp 2: 0 viên, Hộp 3: 5 viên

Hộp 1: 2 viên, Hộp 2: 0 viên, Hộp 3: 3 viên

...

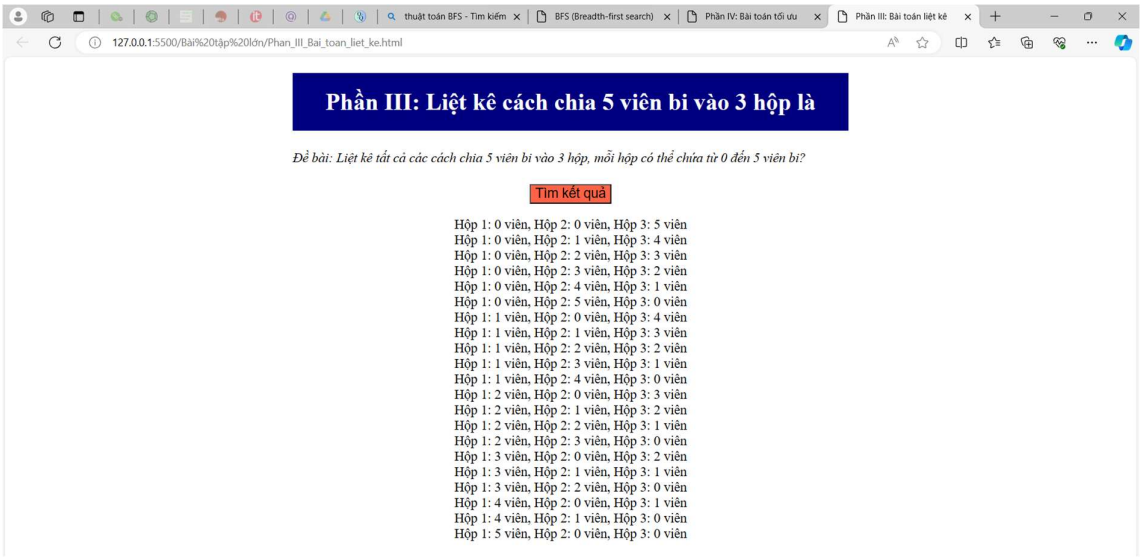
Hộp 1: 5 viên, Hộp 2: 0 viên, Hộp 3: 0 viên

Chúng ta cần duyệt qua tất cả các trường hợp có thể và lưu trữ những trường hợp hợp lệ.

3.3 Lập trình giải quyết bài toán bằng JavaScrip

```
<body>
    <div id="wrapper">
    <h1>
        Phần III: Liệt kê cách chia 5 viên bi vào 3 hộp là
        </h1>
        <p>
            <i>Đề bài: Liệt kê tất cả các cách chia 5 viên bi vào 3 hộp, mỗi
hộp có thể chứa từ 0 đến 5 viên bi?
            </i>
            </p>

    <button onclick="lietKeCachChia()">Tìm kết quả</button>

    <div id="results"></div>

    <script>
        function lietKeCachChia() {
            var resultsElement = document.getElementById("results");
            resultsElement.innerHTML = ""; // Xóa kết quả trước đó (nếu có)

            for (var hop1 = 0; hop1 <= 5; hop1++) {
                for (var hop2 = 0; hop2 <= 5; hop2++) {
                    for (var hop3 = 0; hop3 <= 5; hop3++) {
                        if (hop1 + hop2 + hop3 === 5) {
                            resultsElement.innerHTML += "Hộp 1: " + hop1 + "
viên, Hộp 2: " + hop2 + " viên, Hộp 3: " + hop3 + " viên<br>";
                        }
                    }
                }
            }
        }
    </script>
    </div>
</body>
</html>
```

3.4 Chụp lại kết quả



Hình 3.4 – kết quả cho bài toán liệt kê

3.5 đánh giá kết quả

Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

4. Phần IV. Bào toán tối ưu

4.1 Đề bài

(5) Bài toán lập lịch: Mỗi một chi tiết trong số n chi tiết D_1, D2,..., Dn, cần phải được lần lượt gia công trên m máy M1, M2,..., Mm. Thời gian gia công chi tiết D, trên máy Mj là tij. Hãy tìm lịch (trình tự gia công) các chi tiết trên các máy sao cho việc hoàn thành gia công tất cả các chi tiết là sớm nhất có thể được?

4.2 Phân tích bài toán

Để giải quyết bài toán này, có nhiều thuật toán khác nhau để giải quyết bài toán lập lịch nhiệm vụ, nhưng đối với bài này em sử dụng thuật toán lập lịch LPT (Longest Processing Time).

Thuật toán này hoạt động như sau:

- Sắp xếp các công việc theo thời gian gia công giảm dần: Sắp xếp danh sách các công việc theo thời gian gia công từ lớn đến nhỏ.
- Gia công từng công việc theo trình tự đã sắp xếp: Bắt đầu từ công việc có thời gian gia công lớn nhất, lập lịch gia công cho từng công việc lần lượt trên các máy.

Lập lịch gia công từng công việc theo thứ tự đã sắp xếp:

- Đặt mỗi công việc vào máy có thời gian hoàn thành gần nhất.
- Cập nhật thời gian hoàn thành của mỗi máy sau khi gia công một công việc.

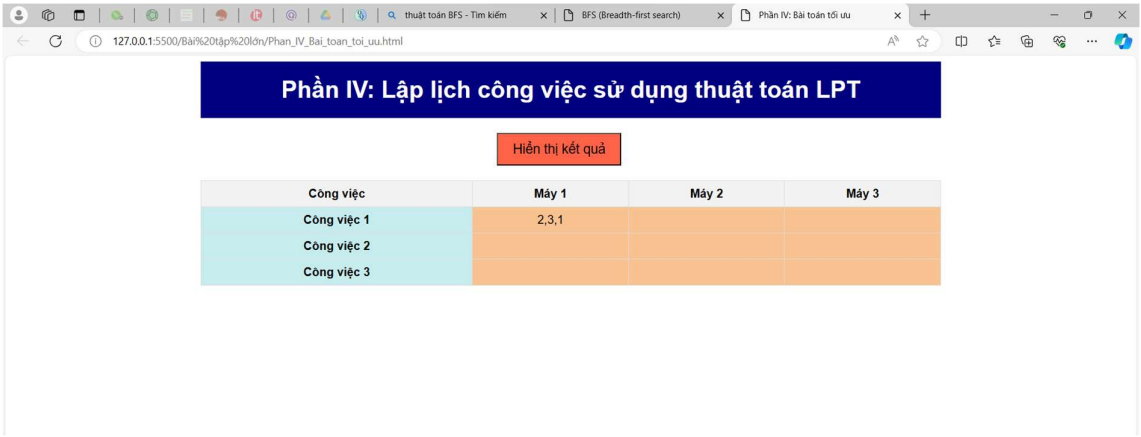
Trả về lịch trình sau khi tất cả công việc đã được gia công.

4.3 Lập trình giải quyết bài toán bằng JavaScript

```
<body>
    <div id="wrapper">
<h1>Phần IV: Lập lịch công việc sử dụng thuật toán LPT</h1>

<button onclick="runLPT()">Hiển thị kết quả</button>

<table id="scheduleTable">
    <!-- Bảng sẽ được tạo động bằng JavaScript -->
</table>

<script>
function lptAlgorithm(processingTimes) {
    // Sắp xếp các công việc theo thời gian gia công giảm dần
    let sortedJobs = processingTimes.slice().sort((a, b) => b - a);

    // Khởi tạo mảng lưu trữ thời gian hoàn thành của mỗi máy
    let machines = new Array(processingTimes[0].length).fill(0);

    // Lập lịch gia công từng công việc
    let schedule = [];
    sortedJobs.forEach(job => {
        let minTime = Math.min(...machines);
        let machineIndex = machines.indexOf(minTime);
        machines[machineIndex] += job;
        schedule.push({ job: job, machine: machineIndex + 1 });
    });

    return schedule;
}

function runLPT() {
    const processingTimes = [
        [2, 3, 1],
        [5, 2, 4],
        [1, 2, 3]
    ];

    const schedule = lptAlgorithm(processingTimes);
    displaySchedule(schedule);
}

function displaySchedule(schedule) {
    let table = document.getElementById("scheduleTable");
    table.innerHTML = "";

    // Header
    let headerRow = table.insertRow();
    let th = document.createElement("th");
    th.textContent = "Công việc";
    headerRow.appendChild(th);
```

```
    for (let i = 1; i <= schedule.length; i++) {
        th = document.createElement("th");
        th.textContent = `Máy ${i}`;
        headerRow.appendChild(th);
    }

    // Body
    schedule.forEach((item, index) => {
        let row = table.insertRow();
        let cell = row.insertCell();
        cell.textContent = `Công việc ${index + 1}`;
        cell.className = "job-cell";
        for (let i = 0; i < schedule.length; i++) {
            cell = row.insertCell();
            cell.textContent = (item.machine === i + 1) ? item.job : "";
            cell.className = "machine-cell";
        }
    });
}
</script>
</div>
</body>
</html>
```

4.4 Chụp lại kết quả



Hình 4.4 – kết quả cho bài toán tối ưu

4.5 đánh giá kết quả

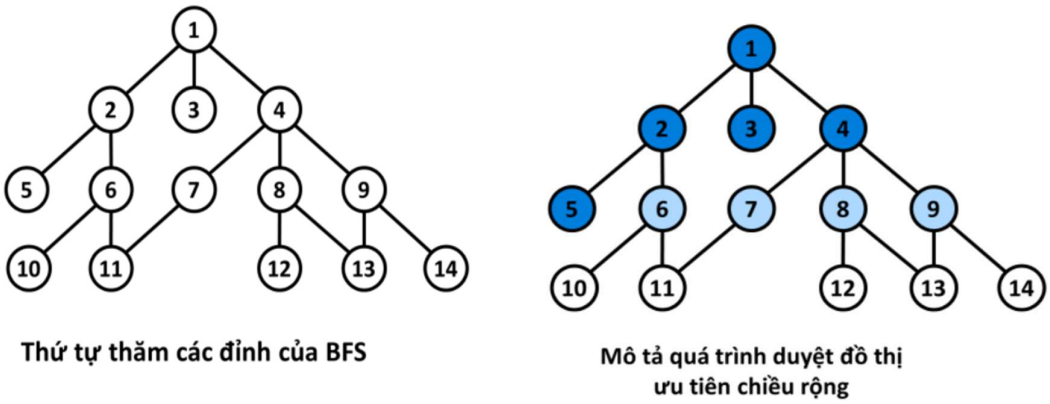
Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

5. Phần V. Thuật toán tìm kiếm:Hãy tìm hiểu thuật toán và cài đặt bằng JS

5.1 Đề bài

(1) Thuật toán BFS?

5.2 Phân tích bài toán



Hình 2.5.2 – ví dụ mô tả thuật toán BFS trước và sau khi tự thăm các đỉnh

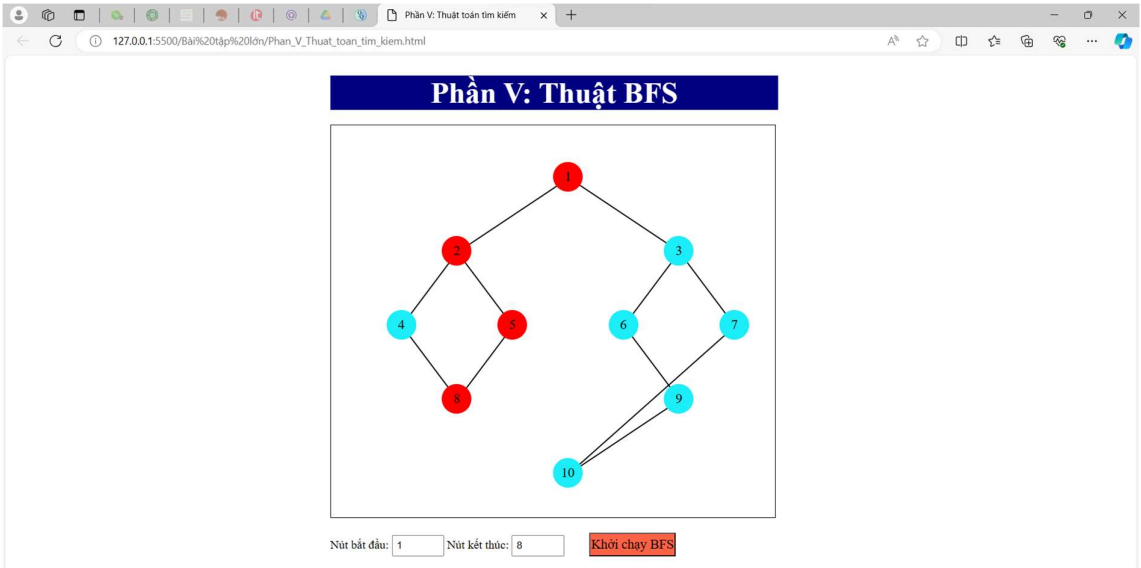
5.3 Lập trình giải quyết bài toán bằng JavaScript

```
<body>
<div id="wrapper">
<h1>Phần V: Thuật BFS</h1>
<div id="graph"></div>
<div id="nut">
<label for="startNode">Nút bắt đầu:</label>
<input type="number" id="startNode" min="1" max="10">
<label for="endNode">Nút kết thúc:</label>
<input type="number" id="endNode" min="1" max="10">
<button id="runBtn">Khởi chạy BFS</button>
</div>
<script>
const graph = [
[0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 1, 1, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 1, 1, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
[0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
[0, 0, 1, 0, 0, 0, 0, 0, 0, 1],
[0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
[0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
];

const nodes = [
{x: 300, y: 50},
{x: 150, y: 150},
{x: 450, y: 150},
{x: 75, y: 250},
{x: 225, y: 250},
```

```
{x: 375, y: 250},
{x: 525, y: 250},
{x: 150, y: 350},
{x: 450, y: 350},
{x: 300, y: 450}
];

const graphContainer = document.getElementById('graph');
const nodeElements = [];
for (let i = 0; i < nodes.length; i++) {
const node = document.createElement('div');
node.classList.add('node');
node.textContent = i + 1;
node.style.left = `${nodes[i].x}px`;
node.style.top = `${nodes[i].y}px`;
graphContainer.appendChild(node);
nodeElements.push(node);
}

const edgeElements = [];
for (let i = 0; i < graph.length; i++) {
for (let j = i + 1; j < graph[i].length; j++) {
if (graph[i][j] === 1) {
const edge = document.createElement('div');
edge.classList.add('edge');
edge.style.left = `${nodes[i].x + 20}px`;
edge.style.top = `${nodes[i].y + 20}px`;
const dx = nodes[j].x - nodes[i].x;
const dy = nodes[j].y - nodes[i].y;
const distance = Math.sqrt(dx * dx + dy * dy);
edge.style.width = `${distance}px`;
edge.style.transform = `rotate(${Math.atan2(dy, dx)}rad)`;
graphContainer.appendChild(edge);
edgeElements.push(edge);
}
}
}

function BFS(start, end) {
const queue = [start];
const visited = new Set();
const parent = {};

while (queue.length > 0) {
const current = queue.shift();
if (current === end) {
return reconstructPath(parent, start, end);
}

visited.add(current);
```

```
for (let neighbor = 0; neighbor < graph[current].length; neighbor++) {
if (graph[current][neighbor] === 1 && !visited.has(neighbor)) {
queue.push(neighbor);
parent[neighbor] = current;
}
}
}

return null;
}

function reconstructPath(parent, start, end) {
const path = [end];
let current = end;
while (current !== start) {
current = parent[current];
path.unshift(current);
}
return path;
}

function clearPath() {
nodeElements.forEach(node => {
node.classList.remove('path');
});
}

document.getElementById('runBtn').addEventListener('click', function() {
const startNode = parseInt(document.getElementById('startNode').value);
const endNode = parseInt(document.getElementById('endNode').value);
if (isNaN(startNode) || isNaN(endNode) || startNode < 1 || startNode >
nodes.length || endNode < 1 || endNode > nodes.length) {
alert('Invalid input!');
return;
}

clearPath();
const path = BFS(startNode - 1, endNode - 1);
if (path) {
path.forEach(nodeIndex => {
nodeElements[nodeIndex].classList.add('path');
});
} else {
alert('No path found!');
}
});
</script>
</div>
</body>
</html>
```

5.4 Chụp lại kết quả



Hình 5.4 – kết quả cho thuật toán tìm kiếm BFS

5.5 đánh giá kết quả

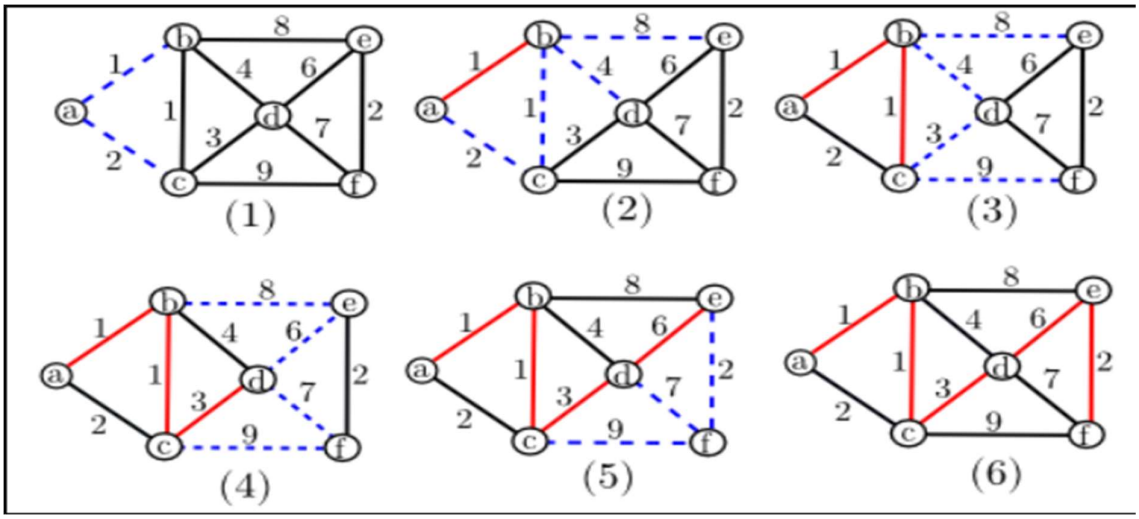
Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

6. Phần VI. Đồ thị: Tìm cây khung nhỏ nhất bằng thuật toán

6.1 Đề bài

(2) Thuật toán Prim?

6.2 Phân tích bài toán



Hình 6.2 – đây là hình ảnh ví dụ minh họa về thuật toán Prim

6.3 Lập trình giải quyết bài toán bằng JavaScrip

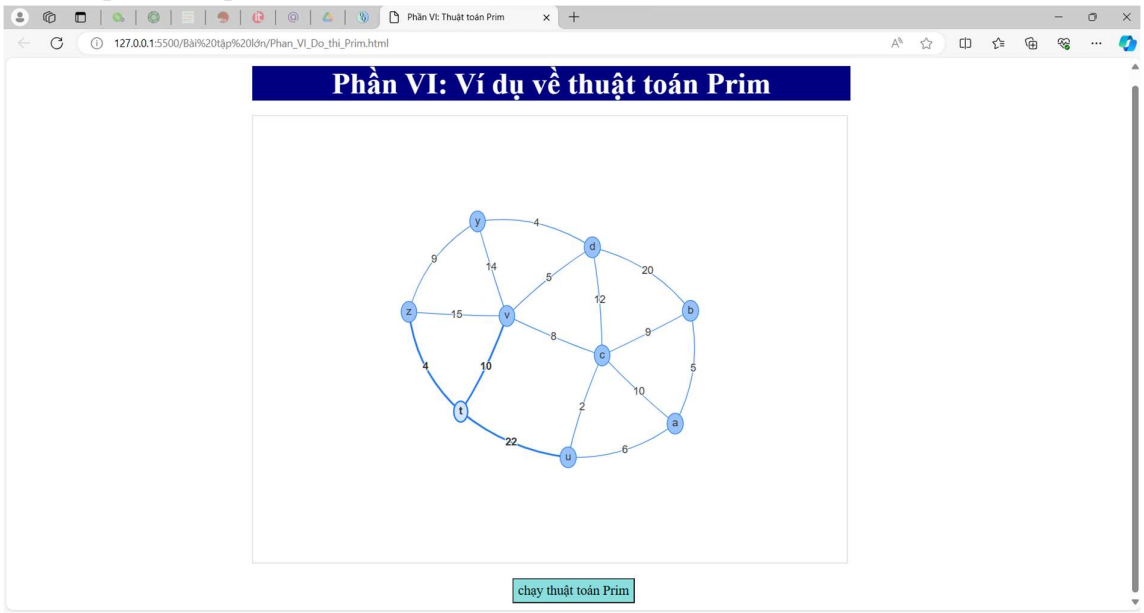
```
<body>
```

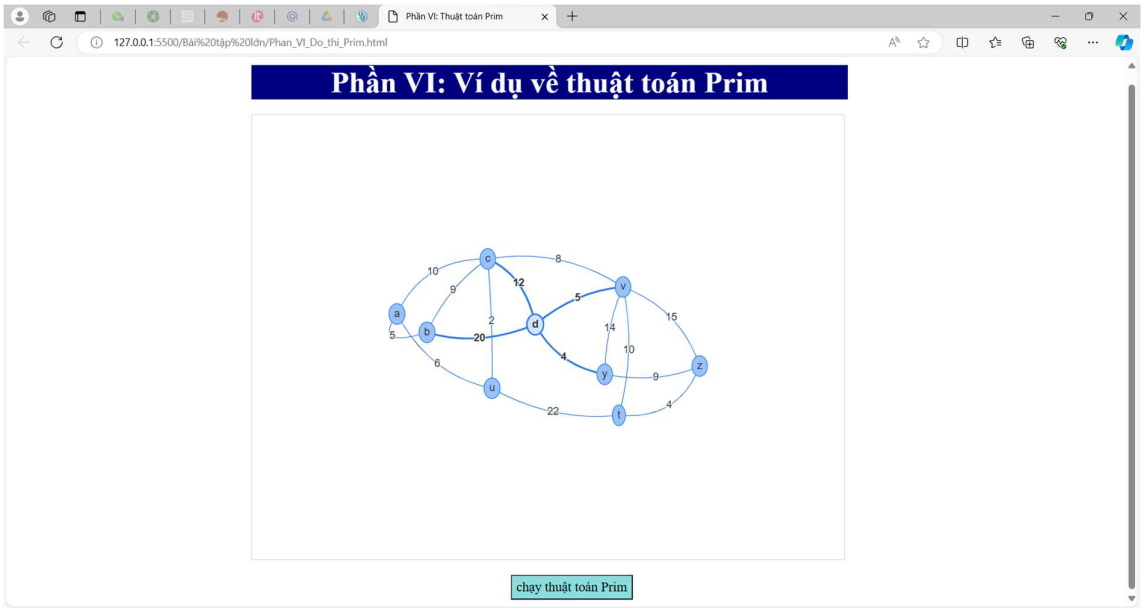
```
<div id="wrapper">
<h1>Phần VI: Ví dụ về thuật toán Prim</h1> <!-- Đổi tiêu đề -->

<div id="mynetwork"></div>
<button id="button">chạy thuật toán Prim</button>
<div id="totalLength"></div>

<script type="text/javascript">
    // Danh sách các đỉnh
    const nodes = new vis.DataSet([
        {id: 'a', label: 'a'},
        {id: 'b', label: 'b'},
        {id: 'c', label: 'c'},
        {id: 'u', label: 'u'},
        {id: 'd', label: 'd'},
        {id: 'v', label: 'v'},
        {id: 'y', label: 'y'},
        {id: 'z', label: 'z'},
        {id: 't', label: 't'}
    ]);

    // Danh sách các cạnh và độ dài
    const edges = new vis.DataSet([
        {from: 'a', to: 'b', label: '5'},
        {from: 'a', to: 'c', label: '10'},
        {from: 'a', to: 'u', label: '6'},
        {from: 'b', to: 'd', label: '20'},
        {from: 'b', to: 'c', label: '9'},
        {from: 'c', to: 'u', label: '2'},
        {from: 'c', to: 'd', label: '12'},
        {from: 'c', to: 'v', label: '8'},
        {from: 'v', to: 'd', label: '5'},
        {from: 'd', to: 'y', label: '4'},
        {from: 'v', to: 'y', label: '14'},
        {from: 'y', to: 'z', label: '9'},
        {from: 'v', to: 'z', label: '15'},
        {from: 'z', to: 't', label: '4'},
        {from: 'v', to: 't', label: '10'},
        {from: 'u', to: 't', label: '22'}
    ]);

    // Tạo một mạng
    const container = document.getElementById('mynetwork');
    const data = {
        nodes: nodes,
        edges: edges
    };
    const options = {};
    const network = new vis.Network(container, data, options);
```

```
    // Function to run Prim's Algorithm
    function runPrimAlgorithm() {
        let visited = {};
        let resultEdges = [];
        let minDist = {};
        let minEdge = {};
        let totalLength = 0;

        // Initialize visited and minDist
        nodes.forEach(node => {
            visited[node.id] = false;
            minDist[node.id] = Infinity;
        });

        // Choose start node arbitrarily (here we choose 'a')
        let currentNode = 'a';
        minDist[currentNode] = 0;

        // Loop until all nodes are visited
        while (Object.values(visited).includes(false)) {
            // Mark current node as visited
            visited[currentNode] = true;

            // Update minDist for neighbors of current node
edges.forEach(edge => {
                if (edge.from === currentNode && edge.label <
minDist[edge.to]) {
                    minDist[edge.to] = edge.label;
                    minEdge[edge.to] = edge;
                }
            });

            // Find unvisited node with minimum distance
            let min = Infinity;
            let minNode;
            Object.keys(minDist).forEach(node => {
                if (!visited[node] && minDist[node] < min) {
                    min = minDist[node];
                    minNode = node;
                }
            });

            // Add edge from minNode to current node to result
            if (minNode) {
                resultEdges.push(minEdge[minNode]);
                totalLength += parseInt(minDist[minNode]);
                currentNode = minNode;
            }
        }
```

```
        // Update the graph with the result edges
        network.setData({nodes: nodes, edges: resultEdges});

        // Display the total length
        document.getElementById('totalLength').innerText = 'Tổng trọng số nhỏ
nhất: ' + totalLength;
    }

    // Add click event listener to the button
    document.getElementById('runButton').addEventListener('click',
runPrimAlgorithm);
</script>
</div>
</body>
</html>
```

6.4 Chụp lại kết quả





Hình 6.4 – kết quả cho bài toán đồ thị sử dụng thuật toán Prim tìm cây khung nhỏ nhất

6.5 đánh giá kết quả

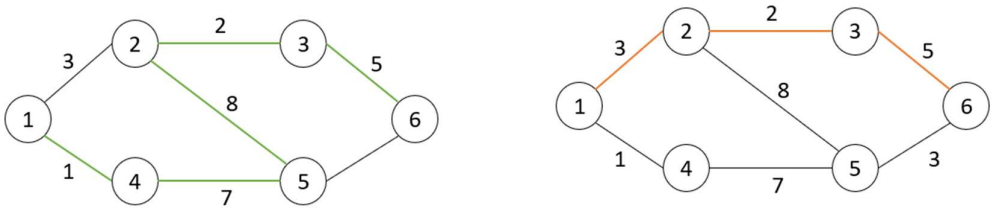
Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

7. Phần VII. Đồ thị: Tìm đường đi ngắn nhất từ 1 đỉnh đến tất cả các đỉnh còn lại

7.1 Đề bài

(1) Thuật toán Floyd?

7.2 Phân tích bài toán



Hình 7.2 – đây là hình ảnh ví dụ minh họa về thuật toán Floyd

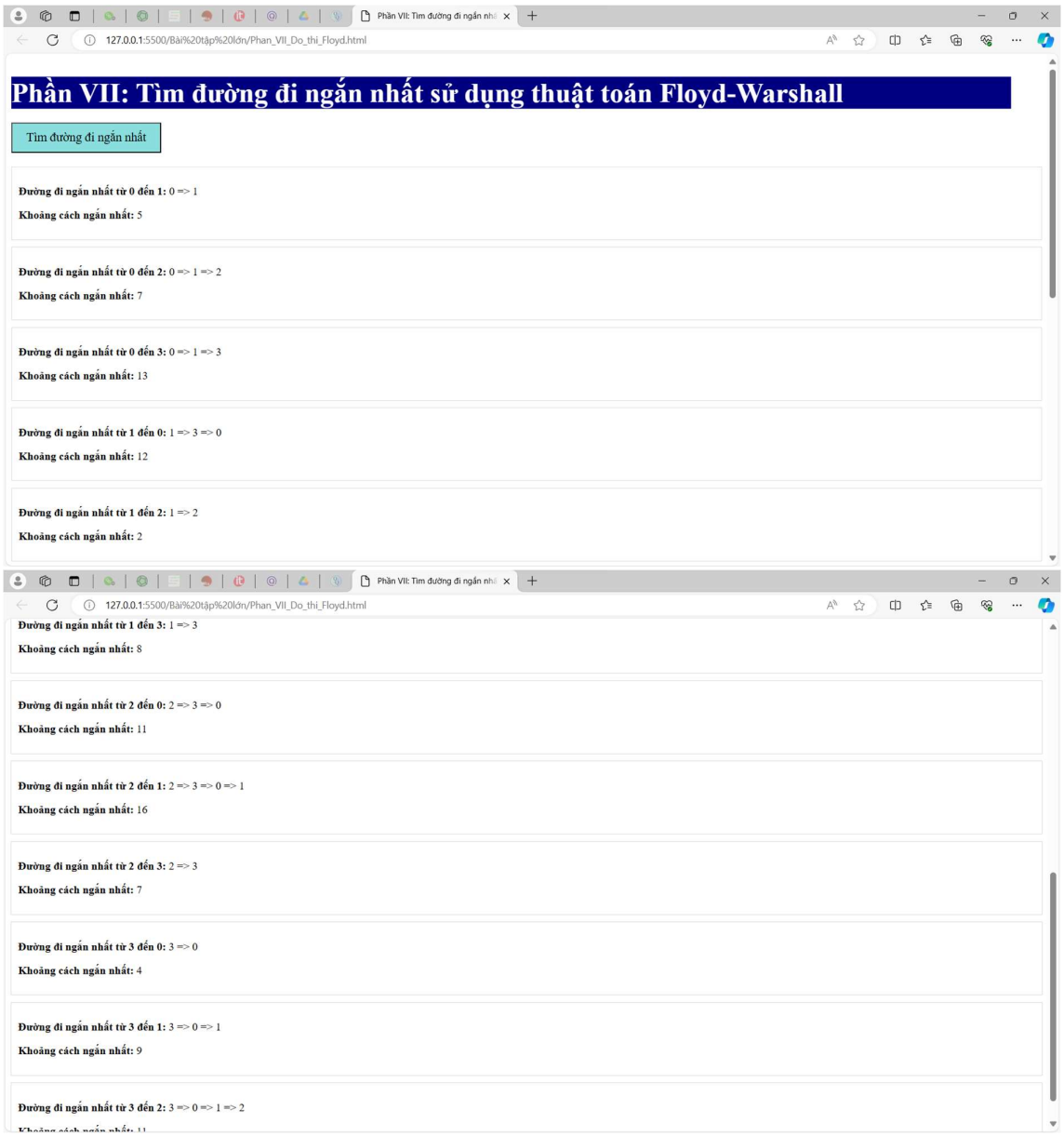
7.3 Lập trình giải quyết bài toán bằng JavaScrip

```
<body>
<h2>Phần VII: Tìm đường đi ngắn nhất sử dụng thuật toán Floyd-Warshall</h2>

<button onclick="findShortestPaths()">Tìm đường đi ngắn nhất</button>

<div id="result"></div>

<script>
function findShortestPaths() {
    const n = 4; // Số lượng đỉnh
    const distances = [
        [0, 5, 9, Infinity],
        [Infinity, 0, 2, 8],
        [Infinity, Infinity, 0, 7],
        [4, Infinity, Infinity, 0]
    ]; // Ma trận khoảng cách

    function floydWarshall() {
        const next = []; // Lưu lộ trình
        const dist = distances.slice(); // Khởi tạo ma trận khoảng cách

        // Khởi tạo ma trận lộ trình
        for (let i = 0; i < n; i++) {
            next[i] = [];
            for (let j = 0; j < n; j++) {
```

```
                if (i !== j && dist[i][j] !== Infinity) {
                    next[i][j] = j;
                } else {
                    next[i][j] = null;
                }
            }
        }

        // Áp dụng thuật toán Floyd-Warshall
        for (let k = 0; k < n; k++) {
            for (let i = 0; i < n; i++) {
                for (let j = 0; j < n; j++) {
                    if (dist[i][k] + dist[k][j] < dist[i][j]) {
                        dist[i][j] = dist[i][k] + dist[k][j];
                        next[i][j] = next[i][k];
                    }
                }
            }
        }

        return { distances: dist, next: next };
    }

    const result = floydWarshall();

    // Hiển thị đường đi ngắn nhất
    const resultContainer = document.getElementById('result');
    resultContainer.innerHTML = '';
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            if (i !== j) {
                const shortestPath = getShortestPath(i, j, result.next);
                const shortestDistance = result.distances[i][j];
                const resultItem = document.createElement('div');
                resultItem.classList.add('result-item');
                resultItem.innerHTML = `<p><b>Đường đi ngắn nhất từ ${i} đến
${j}:</b> ${shortestPath}</p>`;
                resultItem.innerHTML += `<p><b>Khoảng cách ngắn nhất:</b>
${shortestDistance}</p>`;
                resultContainer.appendChild(resultItem);
            }
        }
    }
}

function getShortestPath(start, end, next) {
    let path = `${start}`;
    let current = start;
    while (current !== end) {
        current = next[current][end];
```

```
            path += ` => ${current}`;
        }
        return path;
}
</script>
</body>
</html>
```

7.4 Chụp lại kết quả



Hình 7.4 – kết quả cho bài toán đồ thị sử dụng thuật toán Floyd tìm đường đi ngắn nhất

7.5 đánh giá kết quả

Kết quả hiển thị bằng javascript trùng khớp với kết quả đã phân tích bài toán trên, qua đó em đánh giá kết quả đã đúng như mong muốn.

III. PHẦN TỔNG KẾT

1. Sau khi học xong nhận được kiến thức gì?

- Hiểu rõ về web: HTML là ngôn ngữ đánh dấu sử dụng để tạo cấu trúc và nội dung của một trang web, trong khi JS là ngôn ngữ lập trình cho phép tạo ra tính năng tương tác trên trang web. Học HTML và JS giúp bạn hiểu rõ cách hoạt động của các trang web.
- Tạo ra sản phẩm thực tế: Với HTML và JS, bạn có thể tạo ra các trang web hoạt động đầy đủ với giao diện người dùng tương tác và hấp dẫn.
- Nâng cao kỹ năng giải quyết vấn đề: Lập trình, bao gồm HTML và JS, đòi hỏi kỹ năng giải quyết vấn đề. Bạn sẽ phải tìm ra cách để thực hiện các tác vụ cụ thể, giải quyết các lỗi và vấn đề mà bạn gặp phải.
- Cơ hội nghề nghiệp: Biết HTML và JS mở ra nhiều cơ hội nghề nghiệp trong lĩnh vực phát triển web, thiết kế giao diện người dùng, phân tích dữ liệu web, và nhiều lĩnh vực khác.
- Nền tảng cho việc học các ngôn ngữ lập trình khác: Một khi bạn đã nắm vững HTML và JS, bạn sẽ có nền tảng vững chắc để học thêm các ngôn ngữ lập trình khác như CSS, Python, Java. Giúp em ôn lại các kiến thức cũ về HTML,CSS,JS,GITHUB và học thêm nhiều kiến thức mới, trau dồi tư duy thiết kế, tư duy logic và kỹ năng code chuyên nghiệp hơn.

2. Upload mã nguồn lên GitHub, lấy link github chuyển thành mã QrCode, đưa mã Qrcode này vào trong báo cáo (+1điểm).

Hình 2.1 – Mã Qr Code cho bài làm đã upload trên Github

Link Github bài code:
https://github.com/NguyenVanThu24/Baitaplon.143/commit/2dbe915861df0d1783ca6782613d637c222a1100