TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

TOÁN RỜI RẠC

Sinh viên: Nguyễn Tiến Thắng

Lớp: K58KMT.K01

Giáo viên giảng dạy: **Ths.Đỗ Duy Cốp**

Thái Nguyên – 2024

TRƯỜNG ĐHKTCN **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

KHOA ĐIỆN TỬ ***Độc lập - Tự do - Hạnh phúc***

BÀI TẬP LỚN

MÔN HỌC: TOÁN RỜI RẠC

Sinh viên ....Nguyễn Tiến Thắng

Lớp: ..K58KMT.K01.. Ngành: Kỹ thuật máy tính

Giáo viên hướng dẫn: Ths. Đỗ Duy Cốp

Ngày giao đề 07/03/2024 Ngày hoàn thành 14/03/2024

Yêu cầu : Mỗi sinh viên làm bài riêng, gồm có 7 phần, mỗi phần làm 1 bài toán. Cụ thể mỗi sinh viên làm 7 bài, theo danh sách đã phân công. Mỗi sinh viên làm riêng, in quyển báo cáo

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Thái Nguyên, ngày….tháng…..năm 20....

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

I. PHẦN MỞ ĐẦU

1. Giới thiệu thông tin cá nhân

Em tên là: Nguyễn Tiến Thắng
Ngày sinh: 09/02/2003
Quê quán: Tiền Châu ,Phúc Yên, Vĩnh Phúc
Chức vụ:
Lớp: K58KMT.K01
Sinh viên năm thứ 2
Ngành học: Kỹ thuật máy tính

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

Dưới đây là sô thứ tự đề bài được giao:

64	K225480106058	Nguyễn Tiến	Thắng	K58KMT.K01	4	1	17	4	2	1	2

II. PHẦN NỘI DUNG

1. Bài toán đếm

1.1. Tên bài toán

4. Một công ty gồm 8 nam và 5 nữ. Họ muốn chọn một nhóm làm việc gồm 3 nam và 2 nữ. Có bao nhiêu cách để chọn nhóm?

1.2. Phân tích bài toán

Bài toán này yêu cầu chọn một nhóm làm việc gồm 3 nam và 2 nữ từ trong một công ty có 8 nam và 5 nữ.
Để giải quyết bài toán, ta có thể sử dụng công thức tính tổ hợp để tính số cách chọn 3 năm từ 8 nam rồi nhân với số cách chọn 2 nữ từ 5 nữ.
Kết quả tính toán: $C_8^3 * C_5^2 = 560$

1.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html lang="vi">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Đếm Cách Chọn Nhóm</title>
```

```
</head>
<body>
    <h1>Đếm Cách Chọn Nhóm</h1>
    <p id="result"></p>

    <script>
        // Định nghĩa hàm combination và countWaysToChooseGroup
        function combination(n, k) {
            let numerator = 1;
            for (let i = n; i > n - k; i--) {
                numerator *= i;
            }
            let denominator = 1;
            for (let i = 1; i <= k; i++) {
                denominator *= i;
            }
            return numerator / denominator;
        }

        function countWaysToChooseGroup() {
            const soNam = 8;
            const soNu = 5;
            const soNamCan = 3;
            const soNuCan = 2;

            const cachChonNam = combination(soNam, soNamCan);
            const cachChonNu = combination(soNu, soNuCan);

            return cachChonNam * cachChonNu;
        }

        // Gọi hàm và hiển thị kết quả
        const ketQua = countWaysToChooseGroup();
        document.getElementById('result').innerText = "Số cách chọn nhóm: " +
ketQua;
    </script>
</body>
```

</html>

1.4. Kiểm tra kết quả

Kết quả chạy chương trình

1.5. Đánh giá kết quả

Kết quả được tính toán chính xác và tối ưu dựa vào thuật toán tính tổ hợp.

Kết quả tính toán: $C_8^3 * C_5^2 = 560$

2. Bài toán tồn tại

2.1. Tên bài toán

1. Hình lục giác thần bí: Hãy xếp các số nguyên từ 1..19 vào các ô của lục giác thần bí sao cho tổng 6 hướng của lục giác đều bằng nhau

2.2. Phân tích bài toán

Bài toán yêu cầu xếp các số nguyên từ 1 đến 19 vào các ô của hình lục giác thần bí sao cho tổng của 6 hướng của hình lục giác đều bằng nhau.

Để giải quyết bài toán, ta sử dụng thuật toán tìm kiếm.

2.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html>
<head>
<title>Giải bài tập lục giác thần bí </title>
<script>
const n = 19;
var x = []; // khai báo mảng X
var m = []; // mảng đánh dấu những số đã dùng
var stt;

function init() {
```

```
  for (var i = 1; i <= n; i++) {
    m[i] = 0; //chưa dùng ô thứ i
    x[i] = 0; //ô thứ i: chưa điền số nào
  }
  stt = 0;
}

function f(i) {
  for (var so = 1; so <= n; so++) {
    if (m[so] == 0) { // chưa dùng
      x[i] = so; //điền vào ô i giá trị so
      m[so] = 1; //đánh dấu đã dùng nó
      if (check_ok(i)) {
        if (i == n)
          show_kq();
        else
          f(i + 1);
      }
      x[i] = 0;
      m[so] = 0;
    }
  }
}
const v = [
  [9, 10, 11],
  [8, 12, 17, 18],
  [1, 7, 13, 16, 19],
  [2, 6, 14, 15],
  [3, 4, 5],
  [1, 2, 3],
  [4, 12, 13, 14],
  [5, 11, 15, 18, 19],
  [6, 10, 16, 17],
  [7, 8, 9],
  [1, 11, 12],
  [2, 10, 13, 18],
  [3, 9, 14, 17, 19],
```

```
  [4, 8, 15, 16],
  [5, 6, 7],
];

function check_ok(k) {
  for (var mang of v) {
    var sum = 0,
      isFull = 1;
    for (var i of mang) {
      if (i <= k) {
        sum += x[i];
        if (x[i] == 0) isFull = 0;
      } else {
        isFull = 0;
      }
    }
    //console.log([k,mang,isFull,sum])
    if (sum > 38) return false;
    if (isFull) {
      if (sum != 38) return false;
    } else if (sum >= 38) return false;
  }
  return true;
}

function show_kq() {
  var loigiai = 'Nghiệm thứ ' + ++stt + ': ';
  for (var i = 1; i <= n; i++) loigiai += x[i] + ', ';
  document.getElementById('ketqua').innerHTML += loigiai + '<br>';
  if (stt == 7) {
    for (var i = 1; i <= n; i++) document.getElementById('v' + i).innerHTML =
x[i];
  }
}

function giai() {
  init();
```

```
  f(1); //bắt đầu tìm xem x[1] điền cái gì
}
</script>
<style>
.cell {
  width: 30px;
  height: 20px;
  text-align: center;
  font-size: 40px;
}
</style>
</head>
<body style="cursor: auto">
<h3>Giải bài tập lục giác thần bí:</h3>
<button onclick="giai()">Nhấn để giải bài tập</button>
<div id="ketqua"></div>
</body>
</html>;
```

2.4. Kiểm tra kết quả

Giải bài tập lục giác thần bí:

Nhấn để giải bài tập

Nghiệm thứ 1: 3, 17, 18, 11, 9, 14, 15, 13, 10, 12, 16, 19, 7, 1, 6, 8, 4, 2, 5,
Nghiệm thứ 2: 3, 19, 16, 12, 10, 13, 15, 14, 9, 11, 18, 17, 7, 2, 4, 8, 6, 1, 5,
Nghiệm thứ 3: 9, 11, 18, 17, 3, 19, 16, 12, 10, 13, 15, 14, 6, 1, 7, 2, 4, 8, 5,
Nghiệm thứ 4: 9, 14, 15, 13, 10, 12, 16, 19, 3, 17, 18, 11, 6, 8, 4, 2, 7, 1, 5,
Nghiệm thứ 5: 10, 12, 16, 19, 3, 17, 18, 11, 9, 14, 15, 13, 4, 2, 7, 1, 6, 8, 5,
Nghiệm thứ 6: 10, 13, 15, 14, 9, 11, 18, 17, 3, 19, 16, 12, 4, 8, 6, 1, 7, 2, 5,
Nghiệm thứ 7: 15, 13, 10, 12, 16, 19, 3, 17, 18, 11, 9, 14, 8, 4, 2, 7, 1, 6, 5,

2.5. Đánh giá kết quả

Kết quả hiển thị hình lục giác với tổng 6 hướng bằng nhau.

3. Bài toán liệt kê

3.1. Tên bài toán

17. Liệt kê tất cả các tập hợp con của tập {1, 2, 3} có kích thước từ 0 đến 3.

3.2. Phân tích bài toán

Bài toán yêu cầu liệt kê tất cả các tập hợp con của tập {1, 2, 3} từ tập rỗng đến tập gồm 3 phần tử.

Để giải quyết bài toán, ta có thể sử dụng phương pháp đệ quy để duyệt qua mọi kích thước của tập hợp con và tạo ra các tập hợp con tương ứng.

3.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Trình tạo tập hợp con</title>
    <script>
        function subsetsOfSizeK(arr, k) {
            const result = [];

            function generateSubsets(curr, start, k) {
                if (curr.length === k) {
                    result.push(curr.slice());
                    return;
                }
                for (let i = start; i < arr.length; i++) {
                    curr.push(arr[i]);
                    generateSubsets(curr, i + 1, k);
                    curr.pop();
                }
            }

            generateSubsets([], 0, k);
            return result;
        }

        function hienThiTapHopCon() {
            const originalSet = [1, 2, 3];
            const outputDiv = document.getElementById("output");
            outputDiv.innerHTML = "";

            for (let k = 0; k <= originalSet.length; k++) {
                const subsets = subsetsOfSizeK(originalSet, k);
```

```
                outputDiv.innerHTML += `<p>Tất cả các tập hợp con có kích thước
${k}: ${JSON.stringify(subsets)}</p>`;
            }
        }
    </script>
</head>
<body>
    <h1>Trình tạo tập hợp con</h1>
    <button onclick="hienThiTapHopCon()">Tạo Tập Hợp Con</button>
    <div id="output"></div>
</body>
</html>
```

3.4. Kiểm tra kết quả

3.5. Đánh giá kết quả

Kết quả in ra được tập con của tập {1, 2, 3} một cách chính xác.

4. Bài toán tối ưu

4.1. Tên bài toán

4. Bài toán đóng thùng: Có n đồ vật với trọng lượng là w_1, w_2, ..., wn. Cần tìm cách xếp các đồ vật này vào các cái thùng có cùng dung lượng là b sao cho số thùng cần sử dụng là nhỏ nhất có thể được.

4.2. Phân tích bài toán

Đề bài: Có n đồ vật với trọng lượng là w1, w2, w3 . Cần tìm cách xếp các đồ vật này vào các thùng có cùng dung lượng là b sao cho số lượng thùng cần sử dụng là ít nhất.

Ràng buộc: Mỗi thùng chỉ chứa được một lượng trọng lượng nhất định, và mỗi đồ vật phải được đặt vào một thùng.

Mục tiêu: Tối ưu hóa số lượng thùng cần sử dụng.

4.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html lang="vi">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Bài toán đóng thùng</title>
</head>
<body>
<h1>Bài toán đóng thùng</h1>
<div>
   <label for="n">Nhập số lượng đồ vật :</label>
   <input type="number" id="n" min="1">
</div>
<div>
   <label for="b">Nhập dung tích của thùng :</label>
   <input type="number" id="b" min="1">
</div>
<div>
   <label for="weights">Nhập trọng lượng của từng đồ vật (cách nhau bằng dấu phẩy):</label>
   <input type="text" id="weights">
</div>
<button onclick="solve()">Giải</button>
<div id="result"></div>

<script>
function solve() {
   var n = parseInt(document.getElementById("n").value);
   var b = parseInt(document.getElementById("b").value);
   var weightsInput = document.getElementById("weights").value;
   var weights = weightsInput.split(",").map(function(item) {
      return parseInt(item.trim());
   });
```

```
    // Lọc ra các đồ vật có trọng lượng nhỏ hơn hoặc bằng dung tích của thùng
    weights = weights.filter(weight => weight <= b);
    weights.sort((a, b) => b - a); // Sắp xếp các đồ vật theo trọng lượng giảm dần
    var bins = [b]; // Mảng lưu trữ dung lượng còn lại của các thùng

    for (var i = 0; i < weights.length; i++) {
        var j;
        for (j = 0; j < bins.length; j++) {
            if (weights[i] <= bins[j]) {
                bins[j] -= weights[i];
                break;
            }
        }
        if (j === bins.length) { // Nếu không thể đặt vào các thùng hiện có, tạo thêm một thùng mới
            bins.push(b - weights[i]);
        }
    }
    document.getElementById("result").innerText = "Số lượng thùng cần sử dụng là: " + bins.length;
}
</script>
</body>
</html>
```

4.4. Kiểm tra kết quả

4.5. Đánh giá kết quả

Kết quả in ra được số thùng sử dụng ít nhất theo yêu cầu.

5. Thuật toán tìm kiếm: hãy tìm hiểu thuật toán và cài đặt bằng JS

5.1. Tên bài toán

2. Thuật toán DFS

5.2. Phân tích bài toán

DFS là một thuật toán dùng để duyệt một đồ thị theo chiều sâu. Thuật toán này bắt đầu từ một đỉnh bất kỳ của đồ thị, sau đó duyệt đỉnh kề của nó một bằng phương pháp đệ quy. Quá trình này tiếp tục cho đến khi không còn đỉnh nào để duyệt hoặc tất cả các đỉnh đã được duyệt.

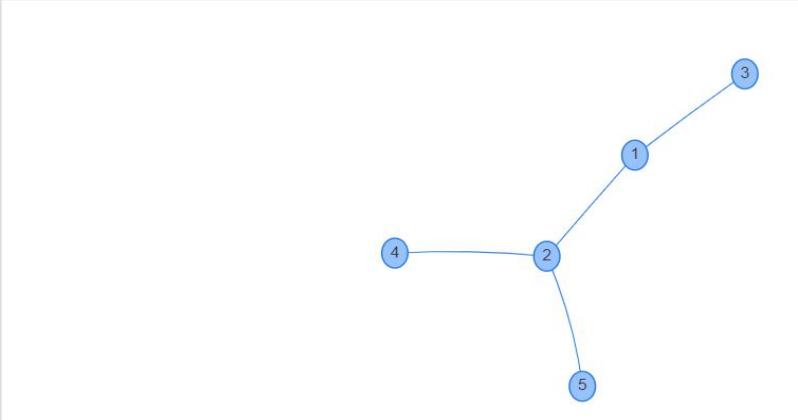
5.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Biểu diễn DFS</title>
    <style>
        #container {
            width: 600px;
            height: 400px;
            border: 1px solid #ccc;
        }
    </style>
    <link href="https://cdnjs.cloudflare.com/ajax/libs/vis/4.21.0/vis.min.css"
rel="stylesheet" type="text/css" />
    <script src="https://cdnjs.cloudflare.com/ajax/libs/vis/4.21.0/vis.min.js"></script>
</head>
<body>
    <h2>Biểu diễn DFS</h2>
    <div id="container"></div>
    <script>
        const graph = {
            1: [2, 3],
            2: [4, 5],
            3: [],
            4: [],
```

```
        5: []
    };

    function DFS(node, visited, nodes, edges) {
        if (!visited[node]) {
            console.log("Đã thăm đỉnh:", node);
            visited[node] = true;
            nodes.add({ id: node, label: String(node) });
            graph[node].forEach(neighbor => {
                edges.add({ from: node, to: neighbor });
                DFS(neighbor, visited, nodes, edges);
            });
        }
    }

    const nodes = new vis.DataSet();
    const edges = new vis.DataSet();

    const container = document.getElementById("container");
    const data = {
        nodes: nodes,
        edges: edges
    };
    const options = {};

    const network = new vis.Network(container, data, options);

    const visited = {};
    DFS(1, visited, nodes, edges);
  </script>
</body>
</html>
```

5.4. Kiểm tra kết quả

Biểu diễn DFS



Kết quả duyệt đồ thị bằng thuật toán DFS chính xác với thứ tự duyệt các đỉnh: 1 -> 2 -> 4 -> 5 -> 3

5.5. Đánh giá kết quả

Kết quả của thuật toán DFS chính xác dựa trên đồ thị đã cung cấp.

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

6.1. Tên bài toán

1. Thuật toán Kruskal

6.2. Phân tích bài toán

Thuật toán Kruskal là một thuật toán được sử dụng để tìm cây khung nhỏ nhất trong một đồ thị có trọng số không âm. Cây khung nhỏ nhất là một cây con của đồ thị ban đầu, bao gồm tất cả các đỉnh và một số cạnh sao cho tổng trọng số của các cạnh là nhỏ nhất có thể.

Chọn đồ thị dạng ;

0 4 2 0 0
4 0 5 0 0
2 5 0 8 0
0 0 8 0 3
0 0 0 3 0

6.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html lang="vi">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Thuật toán Kruskal</title>
```

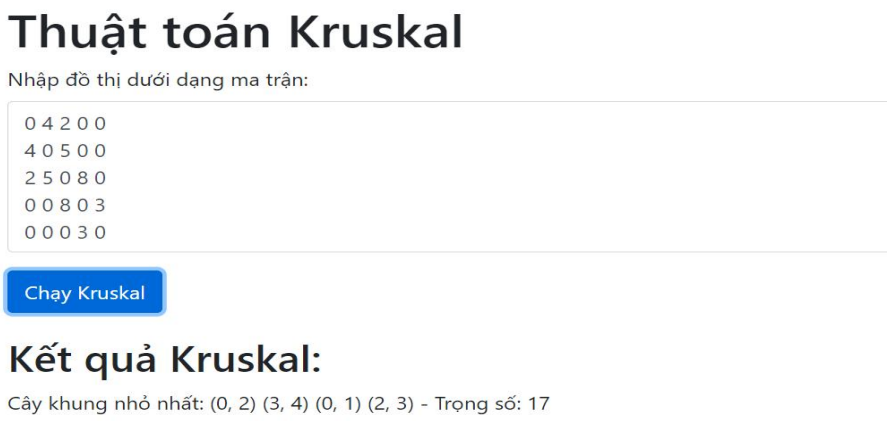
```
<link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/vis-network@7.12.0/dist/vis-
network.min.js"></script>
<style>
   /* Định dạng cho phần kết quả */
   #result {
      margin-top: 20px;
   }
   #result h2 {
      margin-top: 20px;
   }
   #result p {
      margin-bottom: 5px;
   }
</style>
</head>
<body>
<div class="container">
   <h1 class="mt-5">Thuật toán Kruskal</h1>
   <div class="form-group">
      <label for="graph">Nhập đồ thị dưới dạng ma trận:</label>
      <textarea id="graph" class="form-control" rows="5" cols="50">
0 4 2 0 0
4 0 5 0 0
2 5 0 8 0
0 0 8 0 3
0 0 0 3 0</textarea>
   </div>
   <button onclick="runKruskal()" class="btn btn-primary">Chạy Kruskal</button>
   <div id="result"></div>
   <div id="network" class="mt-3"></div>
</div>

<script>
function kruskal(graph) {
```

```
const n = graph.length;
const edges = [];

for (let i = 0; i < n; i++) {
    for (let j = i + 1; j < n; j++) {
        if (graph[i][j] !== 0) {
            edges.push([i, j, graph[i][j]]);
        }
    }
}

edges.sort((a, b) => a[2] - b[2]);

const parent = new Array(n).fill(-1);
const result = [];

function find(u) {
    if (parent[u] === -1) return u;
    return find(parent[u]);
}

function union(u, v) {
    const rootU = find(u);
    const rootV = find(v);
    parent[rootU] = rootV;
}

let edgeCount = 0;
let i = 0;
while (edgeCount < n - 1) {
    const [u, v, weight] = edges[i++];
    const rootU = find(u);
    const rootV = find(v);
    if (rootU !== rootV) {
        result.push([u, v, weight]);
        union(u, v);
        edgeCount++;
```

```
        }
    }

    return result;
}

function runKruskal() {
    const input = document.getElementById("graph").value;
    const graph = input.split("\n").map(row => row.split(" ").map(Number));
    const result = kruskal(graph);

    let output = "<h2>Kết quả Kruskal:</h2>";
    const minSpanningTree = [];
    let totalWeight = 0;

    for (let i = 0; i < result.length; i++) {
        const [u, v, weight] = result[i];
        minSpanningTree.push([u, v]);
        totalWeight += weight;
    }

    output += "<p>Cây khung nhỏ nhất: ";
    for (let j = 0; j < minSpanningTree.length; j++) {
        output += "(" + minSpanningTree[j][0] + ", " + minSpanningTree[j][1] + ") ";
    }
    output += "- Trọng số: " + totalWeight + "</p>";

    document.getElementById("result").innerHTML = output;

    const container = document.getElementById("network");
    const nodes = new vis.DataSet(
        new Array(graph.length).fill().map((_, idx) => ({id: idx, label: String(idx)}))
    );
    const edges = new vis.DataSet(minSpanningTree.map(edge => ({from: edge[0], to:
edge[1]})));
    const data = {nodes, edges};
    const options = {};
```

```
    new vis.Network(container, data, options);
}
</script>
</body>
</html>
```

6.4. Kiểm tra kết quả



6.5. Đánh giá kết quả

Kết quả của thuật toán Kruskal cho ra các cây khung nhỏ nhất chính xác.

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

7.1. Tên bài toán

2. Thuật toán Dijkstra

7.2. Phân tích bài toán

Thuật toán Dijkstra được sử dụng để tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại trong một đồ thị có trọng số không âm. Thuật toán này hoạt động bằng cách duyệt qua các đỉnh một cách tuần tự và cập nhật khoảng cách ngắn nhất từ đỉnh ban đầu đến các đỉnh khác.

Chọn đồ thị dạng ;

0 4 2 0 0
4 0 5 0 0
2 5 0 8 0
0 0 8 0 3
0 0 0 3 0

7.3. Lập trình giải quyết bài toán bằng JavaScript

```
<!DOCTYPE html>
<html lang="vi">
<head>
```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Thuật toán Dijkstra</title>
<link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
    <h1 class="mt-5">Thuật toán Dijkstra</h1>
    <div>
        <label for="graph">Nhập đồ thị dưới dạng ma trận:</label>
        <textarea id="graph" class="form-control" rows="5" cols="50">
0 4 2 0 0
4 0 5 0 0
2 5 0 8 0
0 0 8 0 3
0 0 0 3 0</textarea>
    </div>
    <button onclick="runDijkstra()" class="btn btn-primary mt-3">Chạy
Dijkstra</button>
    <div id="result" class="mt-3"></div>
</div>

<script>
function dijkstra(graph, start) {
    const distances = {};
    const visited = {};
    const queue = [];

    for (let vertex in graph) {
        distances[vertex] = Infinity;
    }
    distances[start] = 0;

    queue.push({ node: start, weight: 0 });
```

```
    while (queue.length) {
        const { node, weight } = queue.shift();

        visited[node] = true;

        for (let neighbor of graph[node]) {
            if (!visited[neighbor.node] && distances[neighbor.node] > distances[node] +
neighbor.weight) {
                distances[neighbor.node] = distances[node] + neighbor.weight;
                queue.push({ node: neighbor.node, weight: distances[neighbor.node] });
            }
        }
    }

    return distances;
}

function runDijkstra() {
    const input = document.getElementById("graph").value;
    const rows = input.trim().split("\n");
    const graph = {};
    for (let i = 0; i < rows.length; i++) {
        const cols = rows[i].trim().split(" ").map(Number);
        graph[i] = [];
        for (let j = 0; j < cols.length; j++) {
            if (cols[j] !== 0) {
                graph[i].push({ node: j, weight: cols[j] });
            } } }
    const startVertex = 0;
    const shortestDistances = dijkstra(graph, startVertex);

    let output = "<h2 class='mt-3'>Độ dài đường đi ngắn nhất từ đỉnh " + startVertex
+ ":</h2>";
    for (let vertex in shortestDistances) {
        output += "<p>Đỉnh " + vertex + ": " + shortestDistances[vertex] + "</p>";
    }
```

```
    document.getElementById("result").innerHTML = output;
}
</script>
</body>
</html>
```

7.4. Kiểm tra kết quả

Kết quả in ra chính xác khoảng cách ngắn nhất từ đỉnh 0 đến các đỉnh khác.

7.5. Đánh giá kết quả

Kết quả của thuật toán Dijkstra thể hiện chính xác đường đi ngắn nhất từ một đỉnh đến các đỉnh khác của đồ thị.

III. PHẦN TỔNG KẾT

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

Sau khi học xong học em nhận được các kiến thức rất có ích của môn Toán rời rạc về các chủ đề như bài toán đếm, bài toán tồn tại, bài toán liệt kê, bài toán tối ưu và các thuật toán tìm kiếm, tìm cây khung nhỏ nhát, tìm đường đi ngắn nhất. Bên cạnh đó em cũng rèn luyện được khả năng chuyển từ ý tưởng của thuộc toán thành cài đặt bằng chương trình.

2. Upload mã nguồn lên GitHub

3 Ý Kiến cá nhân

Lời cảm ơn

Lời đầu tiên, Em xin gửi lời cảm ơn chân thành nhất đến Thầy Dỗ Duy Cốp. Trong quá trình học tập và tìm hiểu về môn học Toán rời rạc : Em đã nhận được sự quan tâm, giúp đỡ và hướng dẫn tận tình, sâu sắc,tâm huyết của Thầy.Em đã tích lũy thêm kiến thức về ngôn ngữ "javascript". Thông qua Bài tập lớn này em xin trình bày lại kiến thức về ngôn ngữ javascript. Trong quá trình làm bài, chắc chắn không thể chánh khỏi những thiếu sót. Bản thân em rất mong nhận được sự góp ý đến từ thầy (email : k225480106058@tnut.edu.vn).

Em kính chúc thầy sức khỏe, hạnh phúc, thành công trong công việc và trong sự nghiệp giảng dậy.

Em xin trân trọng cảm ơn thầy

Thái Nguyên..13..tháng 03..năm 2024

Sinh viên

Nguyễn Tiến Thắng