TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA ĐIỆN TỬ**

**BỘ MÔN**. **CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**TOÁN RỜI RẠC**

Sinh viên: Lường Văn Hạnh

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

BỘ MÔN CÔNG NGHỆ THÔNG TIN

*Sinh viên:* *Lường Văn Hạnh*

*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*

*Tên đề tài*

*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*.

|  |
| --- |
| **GIÁO VIÊN HƯỚNG DẪN** |
| *(Ký và ghi rõ họ tên)* |

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

### Thái Nguyên, ngày….tháng…..năm 20....

## GIÁO VIÊN HƯỚNG DẪN

*(Ký ghi rõ họ tên)*

**MỤC LỤC**

**Phần nội dung Trang**

1. PHẦN MỞ ĐẦU 6
2. Giới thiệu thông tin cá nhân 6
3. Hướng dẫn làm bài 6
4. PHẦN NỘI DUNG 6
5. Bài toán đếm 6
6. Bài toán tồn tại 8
7. Bài toán liệt kê 11
8. Bài toán tối ưu 13
9. Thuật toán tìm kiếm: hãy tìm hiểu thuật toán và cài đặt bằng JS 16
10. Đồ thị: Tìm cây khung nhỏ nhất bằng thuật toán 18
11. Đồ thị: Tìm đường đi ngắn nhất từ 1 đỉnh đến tất cả các đỉnh còn lại 21
12. PHẦN TỔNG KẾT 23
13. Sau khi học xong nhận được kiến thức gì? 23
14. Upload mã nguồn lên GitHub 23
15. **PHẦN MỞ ĐẦU**
16. **Giới thiệu thông tin cá nhân**

Em tên là: Lường Văn Hạnh

Ngày sinh: 20/04/2002

Quê quán: Thái nguyên

Chức vụ: Lớp Trưởng

Lớp: K58KMT.K01

Sinh viên năm thứ 2

Ngành học: Kỹ thuật máy tính

Chuyên ngành đam mê: Trí tuệ nhân tạo

1. **Hướng dẫn làm bài**

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

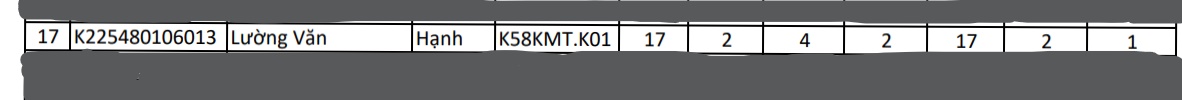
1. Trình bày tên bài toán

2. Phân tích bài toán

3. Lập trình giải quyết bài toán bằng JavaScript

4. Chụp lại kết quả

5. Đánh giá kết quảDưới đây là sô thứ tự đề bài được giao:



1. **PHẦN NỘI DUNG**
2. **Bài toán đếm**
   1. Tên bài toán

17. Một dãy gồm 10 quả bóng. Bạn muốn chọn một dãy gồm 3 quả bóng. Có bao nhiêu cách chọn nếu thứ tự không quan trọng?

* 1. 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 một dãy gồm 3 quả bóng từ 10 quả mà thứ tự không quan trọng. k phần tử từ một tập hợp gồm n phần tử mà không quan tâm đến thứ tự của các phần tử đó. Công thức tổ hợp được biểu diễn như sau: Công thức tổ hợp được sử dụng để tính số cách chọn C(n,k)= n!/( k!⋅(n−k)!). Áp dụng vào bài toán của chúng ta: C(10,3)= 10!/( 3!⋅(10−3)!). Vậy có 120 cách chọn dãy gồm 3 quả bóng từ 10 quả mà thứ tự không quan trọng.

A diagram of a function

Description automatically generated

*Hình 1: Sơ đồ khối bài toán đếm*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN ĐẾM \*/

function Giai\_Bai\_Toan\_Dem() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[0].stt\_bai\_toan

    let tag\_ket\_qua = document.getElementById(id\_tagnames[stt\_bt-1].ket\_qua)

    tag\_ket\_qua.innerHTML = ''

    // Số lượng quả bóng và số lượng quả bóng bạn muốn chọn

    value\_inputs = []

    Lay\_Du\_Lieu\_Input(stt\_bt,value\_inputs)

    function To\_Hop(n,k) {

        if (k > n) return 0

        let result = 1;

        for (let i = 1; i <= k; i++) result \*= (n - i + 1) / i;

        return result;

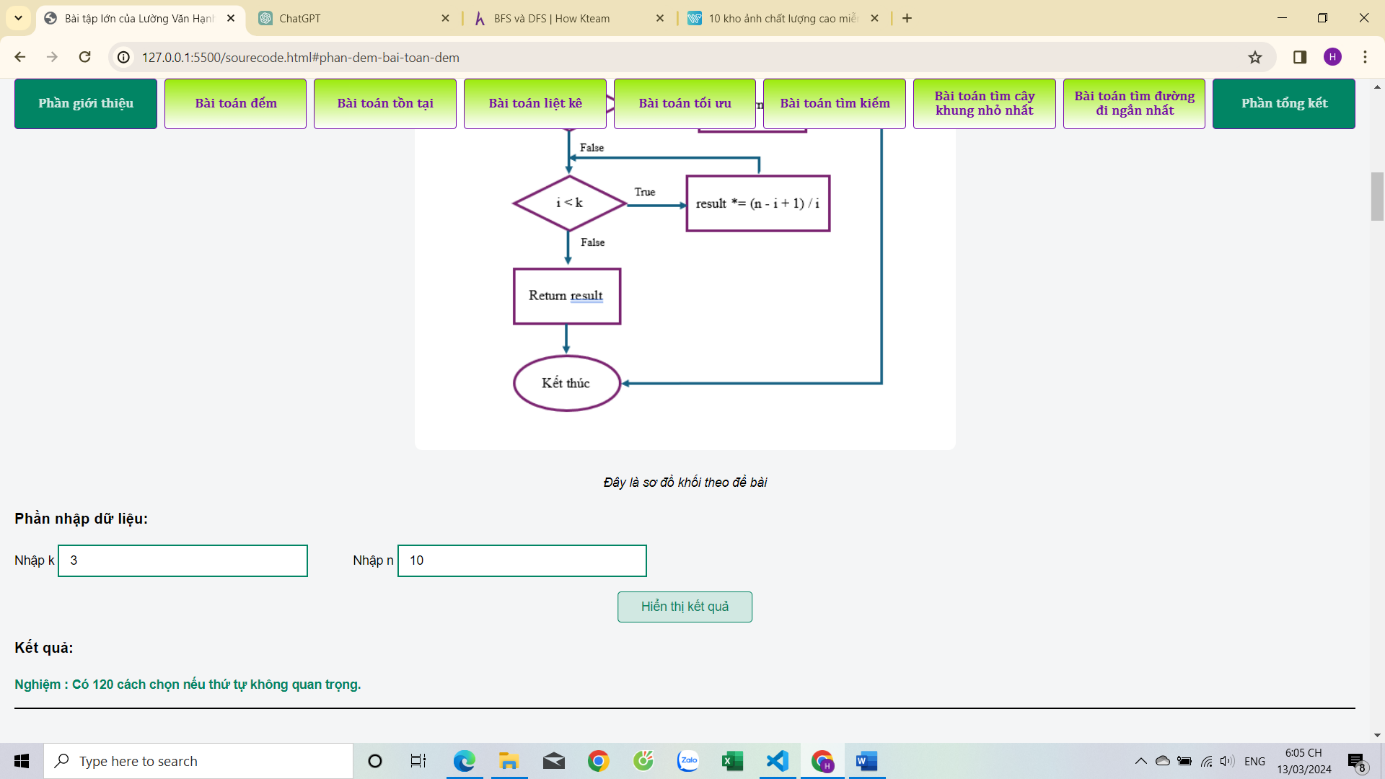
    }

    const nghiem = To\_Hop(value\_inputs[1],value\_inputs[0])

    Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,1,nghiem)

}

* 1. Kiểm tra kết quả



*Hình 2.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quảChương trình chạy ra đúng kết quả.

1. **Bài toán tồn tại**
   1. Tên bài toán

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

* 1. Phân tích bài toán

A diagram of a algorithm

Description automatically generated

*Hình 3: Sơ đồ khối bài toán tồn tại*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN TỒN TẠI \*/

function Giai\_Bai\_Toan\_Ton\_Tai() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[1].stt\_bai\_toan

    let tag\_ket\_qua = document.getElementById(id\_tagnames[stt\_bt-1].ket\_qua)

    tag\_ket\_qua.innerHTML = ''

    let n, C, x = [],dem = 0

    const Color=['ko biết','Xanh','Đỏ','Vàng']

    let pre\_tag = document.getElementById(id\_tagnames[1].pre\_id\_1).innerText

    C = pre\_tag.split('\n'); // C là ma trận kề ==mảng 2 chiều

    n = C.length - 1;

    x = new Array(n).fill(0)// khởi tạo chưa tô mầu cho qgia nào

    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

    }

    //tô 1 màu cho x[i]

    function To\_Mau\_Quoc\_Gia(i) {

        for (var mau = 1; mau < n; mau++) {

            x[i] = Color[mau]

            if (Check\_Hx\_Khac\_Mau\_Nhau(i)) { //nhánh cận

                if (i == n-1) {

                    //đã đến thằng cuối cùng

                    ++dem

                    Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,dem,x) //suy biến: ko đệ quy

                } else {

                    To\_Mau\_Quoc\_Gia(i + 1) //đệ quy

                }

            }

            x[i] = 0 // quay lui

        }

    }

    To\_Mau\_Quoc\_Gia(0) //tô 1 màu cho quốc gia đầu tiên

}

* 1. Kiểm tra kết quả

A screenshot of a computer

Description automatically generated

*Hình 4.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quả

Chương trình chạy ra đúng kết quả.

1. **Bài toán liệt kê**
   1. Tên bài toán

Liệt kê tất cả các số tự nhiên có 3 chữ số và không chứa chữ số 5.

* 1. Phân tích bài toán

A diagram of a flowchart

Description automatically generated

*Hình 5: Sơ đồ khối bài toán liệt kê*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN LIỆT KÊ \*/

function Giai\_Bai\_Toan\_Liet\_Ke() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[2].stt\_bai\_toan

    let tag\_ket\_qua = document.getElementById(id\_tagnames[stt\_bt-1].ket\_qua)

    tag\_ket\_qua.innerHTML = ''

    let dem = 0

    value\_inputs = []

    Lay\_Du\_Lieu\_Input(stt\_bt,value\_inputs)

    const cau\_hinh\_khong\_mong\_muon = value\_inputs[1]

    const so\_bat\_dau = Math.pow(10, (value\_inputs[0] - 1))

    const so\_ket\_thuc = Math.pow(10, (value\_inputs[0]))

    // Liệt kê tất cả các số tự nhiên có 3 chữ số và không chứa chữ số 5

    for (let num = so\_bat\_dau; num < so\_ket\_thuc; num++) {

        if (!num.toString().includes(cau\_hinh\_khong\_mong\_muon)) {

            ++dem

            Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,dem,num)

        }

    }

}

* 1. Kiểm tra kết quả

A screenshot of a computer

Description automatically generated

*Hình 6.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quảChương trình chạy ra đúng kết quả.

1. **Bài toán tối ưu**
   1. Tên bài toán

2. Bài toán cái túi: Một nhà thám hiểm cần đem theo một cái túi có trọng lượng không quá b. Có n đồ vật có thể đem theo. Đồ vật thứ j có trọng lượng là ai và giá trị sử dụng là cj (j = 1, 2,..., n). Hỏi rằng nhà thám hiểm cần đem theo các đồ vật nào để cho tổng giá trị sử dụng của các đồ vật đem theo là lớn nhất?

* 1. Phân tích bài toán

A diagram of a computer program

Description automatically generated with medium confidence

*Hình 7: Sơ đồ khối bài toán tối ưu*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN TỐI ƯU \*/

function Giai\_Bai\_Toan\_Toi\_Uu() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[3].stt\_bai\_toan

    let tag\_ket\_qua = document.getElementById(id\_tagnames[stt\_bt-1].ket\_qua)

    tag\_ket\_qua.innerHTML = ''

    let ten\_do\_vat\_tag = document.getElementById(id\_tagnames[3].pre\_id\_1).innerText

    let gia\_tri\_tag = document.getElementById(id\_tagnames[3].pre\_id\_2).innerText

    let trong\_luong\_tag = document.getElementById(id\_tagnames[3].pre\_id\_3).innerText

    let max\_trong\_luong\_tag = document.getElementById(id\_tagnames[3].pre\_id\_4).innerText

    let cac\_do\_vat = ten\_do\_vat\_tag.split(/,\s\*/).map(String)

    let cac\_gia\_tri = gia\_tri\_tag.split(/,\s\*/).map(Number)

    let trong\_luong\_cac\_vat = trong\_luong\_tag.split(/,\s\*/).map(Number)

    let max\_trong\_luong = max\_trong\_luong\_tag.split(/,\s\*/).map(Number)

    let n = cac\_gia\_tri.length

    let nghiem = []

    function Ba\_Lo(max\_trong\_luong, trong\_luong\_cac\_vat, cac\_gia\_tri, n) {

        let K = new Array(n + 1);

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

            K[i] = new Array(max\_trong\_luong + 1).fill(0);

        }

        // Xây dựng bảng K[][] theo phương pháp từ dưới lên

        for (let i = 0; i <= n; i++) {

            for (let w = 0; w <= max\_trong\_luong; w++) {

                if (i === 0 || w === 0) {

                    K[i][w] = 0;

                } else if (trong\_luong\_cac\_vat[i - 1] <= w) {

                    K[i][w] = Math.max(cac\_gia\_tri[i - 1] + K[i - 1][w - trong\_luong\_cac\_vat[i - 1]], K[i - 1][w]);

                } else {

                    K[i][w] = K[i - 1][w];

                }

            }

        }

        // Lưu kết quả của cái túi

        let res = K[n][max\_trong\_luong];

        let w = max\_trong\_luong;

        for (let i = n; i > 0 && res > 0; i--) {

            if (res === K[i - 1][w]) {

                continue;

            } else {

                // Đưa đồ vật này vào túi.

                nghiem.push(cac\_do\_vat[i - 1] + ' -----> có giá trị sử dụng là: '+ cac\_gia\_tri[i-1]+' và trọng lượng là: '+trong\_luong\_cac\_vat[i-1])

                // Giảm giá trị của đồ vật này khỏi tổng giá trị

                res -= cac\_gia\_tri[i - 1];

                w -= trong\_luong\_cac\_vat[i - 1];

            }

        }

        Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,K[n][max\_trong\_luong],nghiem)

    }

    Ba\_Lo(max\_trong\_luong, trong\_luong\_cac\_vat, cac\_gia\_tri, n);

}

* 1. Kiểm tra kết quả

Chương trình chạy ra đúng kết quả.

A screenshot of a computer

Description automatically generated

*Hình 8.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quảChương trình chạy ra đúng kết quả.

1. **Thuật toán tìm kiếm: hãy tìm hiểu thuật toán và cài đặt bằng JS**
   1. Tên bài toán

Hãy tìm hiểu thuật toán BFS và cài đặt bằng JS

* 1. Phân tích bài toán

A screenshot of a computer

Description automatically generated

*Hình 9.Đây là hình ảnh ví dụ về thuật toán BFS*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN TÌM KIẾM \*/

function Giai\_Bai\_Toan\_Tim\_Kiem() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[4].stt\_bai\_toan

    let tag\_ket\_qua =  document.getElementById(id\_tagnames[4].ket\_qua)

    let pre\_tag = document.getElementById(id\_tagnames[4].pre\_id\_1).innerText

    costMatrix = pre\_tag.split('\n'); // costMatrix là ma trận kề ==mảng 2 chiều

    n\_dinh = costMatrix.length - 1

    let ma\_tran\_ke = []

    for(let i=0; i< n\_dinh; i++){   //// định nghĩa ma trận kề full 0

        let x = new Array(n\_dinh).fill(0)

        ma\_tran\_ke.push(x)

    }

    for(let i=0; i< n\_dinh; i++){   //// khởi tạo ma trận kề từ ma trận chi phí

        for(let j=0; j< n\_dinh; j++){

            if(parseInt(costMatrix[i][j]) === 0) ma\_tran\_ke[i][j] = 0

            else ma\_tran\_ke[i][j] = 1

        }

    }

    value\_inputs = []

    Lay\_Du\_Lieu\_Input(stt\_bt,value\_inputs)

    let sumcost = 0

  // Hàm thực hiện BFS và trả về lộ trình từ start đến end

  function BFS(ma\_tran\_ke, start, end) {

    const visited = new Array(ma\_tran\_ke.length).fill(false); // Mảng đánh dấu các đỉnh đã được thăm

    const queue = []; // Hàng đợi để lưu các đỉnh cần duyệt

    const parent = new Array(ma\_tran\_ke.length).fill(-1); // Mảng để lưu các đỉnh cha

    visited[start] = true;

    queue.push(start);

    while (queue.length > 0) {

      const currentVertex = queue.shift();

      // Kiểm tra nếu đến được đỉnh kết thúc

      if (currentVertex === end) {

        const path = [];

        let current = end;

        // Tạo lộ trình bằng cách lùi lại từ đỉnh kết thúc đến đỉnh bắt đầu

        while (current !== -1) {

          path.unshift(current);

          current = parent[current];

        }

        return path;

      }

      // Duyệt qua các đỉnh kề của đỉnh hiện tại

      for (let i = 0; i < ma\_tran\_ke[currentVertex].length; i++) {

        if (ma\_tran\_ke[currentVertex][i] === 1 && !visited[i]) {

          visited[i] = true;

          parent[i] = currentVertex;

          queue.push(i);

        }

      }

    }

    // Trả về null nếu không tìm thấy đường đi

    return null;

  }

  // Thực hiện BFS từ đỉnh 1 (đỉnh bắt đầu) đến đỉnh 3 (đỉnh kết thúc)

  const path = BFS(ma\_tran\_ke, value\_inputs[0], value\_inputs[1]);

  if(path !== null){

    path.push(path[0])

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

      if(path[i+1] !== undefined) sumcost += parseInt(costMatrix[path[i]][path[i+1]])

      else if(parseInt(costMatrix[value\_inputs[1]][value\_inputs[0]]) === 0) sumcost \*=2

    }

  }

  let nghiem

  // In lộ trình nếu tìm thấy

  if (path) nghiem = "Lộ trình từ thành phố "+value\_inputs[0]+" đến thành phố "+value\_inputs[1]+" rồi quay lại là  : "+ path.join(" =>> ")

  else nghiem = "Không tồn tại đường đi từ thành phố "+value\_inputs[0]+" đến thành phố "+value\_inputs[1]

  Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,sumcost,nghiem)

}

* 1. Kiểm tra kết quả

A screenshot of a computer

Description automatically generated

*Hình 10.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quả

Chương trình chạy ra đúng kết quả.

1. **Đồ thị: Tìm cây khung nhỏ nhất bằng thuật toán**
   1. Tên bài toán

Đồ thị: Tìm cây khung nhỏ nhất bằng thuật toán

* 1. Phân tích bài toán

A group of lines and dots

Description automatically generated

*Hình 11..Đây là hình ảnh ví dụ về thuật toán Prim*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN TÌM CÂY KHUNG NHỎ NHẤT \*/

function Giai\_Bai\_Toan\_Tim\_Cay\_Khung\_Nho\_Nhat() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[5].stt\_bai\_toan

    let tag\_ket\_qua =  document.getElementById(id\_tagnames[5].ket\_qua)

    tag\_ket\_qua.innerHTML = '' //Reset hiển thị kết quả mới

    let ma\_tran\_duong\_di = []

    let tong\_trong\_so = 0

    value\_inputs = []

    Lay\_Du\_Lieu\_Input(stt\_bt,value\_inputs)

    Khoi\_Tao\_Ma\_Tran\_Ke(stt\_bt,ma\_tran\_duong\_di)

    function Check\_Lap\_Dinh\_Dich(MST, n) {

        for(let i=0; i < n - 1; i++){

            if(MST[i][1] === MST[i+1][1]) MST.splice(i+1)

        }

        return MST

    }

    function PRIM(ma\_tran\_duong\_di) {

        const numVertices = ma\_tran\_duong\_di.length;

        const visited = new Array(numVertices).fill(false);

        const MST = [];

        const queue = [];

        // Bắt đầu từ đỉnh 0

        visited[value\_inputs[0]] = true;

        for (let i = 0; i < numVertices - 1; i++) {

            for (let j = 0; j < numVertices; j++) {

                if (visited[j]) {

                    for (let k = 0; k < numVertices; k++) {

                        if (!visited[k] && ma\_tran\_duong\_di[j][k] !== 0) {

                            queue.push([j, k, ma\_tran\_duong\_di[j][k]]);

                        }

                    }

                }

            }

            // Tìm cạnh nhỏ nhất từ các cạnh kết nối

            let minEdge = queue.reduce((min, curr) => curr[2] < min[2] ? curr : min, queue[0]);

            queue.splice(queue.indexOf(minEdge), 1);

            // Thêm cạnh nhỏ nhất vào cây bao trùm nhỏ nhất (MST)

            MST.push(minEdge);

            visited[minEdge[1]] = true;

        }

        Check\_Lap\_Dinh\_Dich(MST, MST.length)

        return MST;

    }

    const minimumSpanningTree = PRIM(ma\_tran\_duong\_di);

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

        tong\_trong\_so += minimumSpanningTree[i][minimumSpanningTree[i].length - 1]

    }

    Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,tong\_trong\_so,minimumSpanningTree)

}

* 1. Kiểm tra kết quả

A screenshot of a computer

Description automatically generated

*Hình 12.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quảChương trình chạy ra đúng kết quả.

1. **Đồ thị: Tìm đường đi ngắn nhất từ 1 đỉnh đến tất cả các đỉnh còn lại**
   1. Tên bài toán

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

* 1. Phân tích bài toán

A screenshot of a game

Description automatically generated

*Hình 13..Đây là hình ảnh ví dụ về thuật toán Floyd*

* 1. Lập trình giải quyết bài toán bằng JavaScript

/\*BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT \*/

function Giai\_Bai\_Toan\_Tim\_Duong\_Di\_Ngan\_Nhat() {

    const stt\_bt = thong\_tin\_cac\_bai\_toan[6].stt\_bai\_toan

    let tag\_ket\_qua =  document.getElementById(id\_tagnames[6].ket\_qua)

    tag\_ket\_qua.innerHTML = ''  //Reset hiển thị kết quả mới

    let ma\_tran\_duong\_di = []

    value\_inputs = []

    Lay\_Du\_Lieu\_Input(stt\_bt,value\_inputs)

    Khoi\_Tao\_Ma\_Tran\_Ke(stt\_bt,ma\_tran\_duong\_di)

    function FloydWarshall(ma\_tran\_duong\_di) {

        const n = ma\_tran\_duong\_di.length;

        const dist = []

        const next = [] // Ma trận lưu lộ trình

        // Khởi tạo ma trận khoảng cách và lộ trình

        for (let i = 0; i < n; i++) {

            dist[i] = []

            next[i] = []

            for (let j = 0; j < n; j++) {

                if (i === j) {

                    dist[i][j] = 0

                    next[i][j] = j

                } else if (ma\_tran\_duong\_di[i][j] === 0) {

                    dist[i][j] = Infinity

                    next[i][j] = null

                } else {

                    dist[i][j] = ma\_tran\_duong\_di[i][j]

                    next[i][j] = j

                }

            }

        }

        // 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] !== Infinity && dist[k][j] !== Infinity && dist[i][j] > dist[i][k] + dist[k][j]) {

                        dist[i][j] = dist[i][k] + dist[k][j];

                        next[i][j] = next[i][k] // Cập nhật lộ trình

                    }

                }

            }

        }

        return { distances: dist, next: next };

    }

    // Hiển thị đường đi

    function PrintAllPaths(start, next) {

        const n = next.length

        for (let end = 0; end < n; end++) {

            if (start !== end ) {

                const path = GetPath(start, end, next)

                let nghiem = "Đường đi ngắn nhất từ " + start + " đến " + end + ": <br>"+path

                let min\_distances = result.distances[start][end]

                Hien\_Thi\_Ket\_Qua(stt\_bt,tag\_ket\_qua,min\_distances,nghiem)

            }

        }

    }

    // Tạo path

    function GetPath(start, end, next) {

        let path = "Từ " + start

        let current = start

        while (current !== end) {

            current = next[current][end]

            path += " ==> " +current

        }

        return path

    }

    const result = FloydWarshall(ma\_tran\_duong\_di);

    const start = value\_inputs[0];

    PrintAllPaths(start, result.next);

}

* 1. Kiểm tra kết quả

A screenshot of a computer

Description automatically generated

*Hình 14.Kết quả sau khi chạy chương trình*

* 1. Đánh giá kết quảChương trình chạy ra đúng kết quả.

1. **PHẦN TỔNG KẾT**
2. **Sau khi học xong nhận được kiến thức gì?**

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.

1. **Upload mã nguồn lên GitHub**