Nama: Isep Lutpi Nur
NPM: 2113191079
Tugas: Komunikasi Data Minggu 14 Implementasi Djikstra

Implementasi Alogritma Djikstra

Program untuk menentukan jalur tercepat menggunakan algoritma djikstra, Tugas besar mata kuliah Komunikasi data semester 3 Menggunakan Bahasa pemrograman Javascript.

Kode Sumber: https://github.com/iseplutpinur/algoritma_dijkstra
Lihat Aplikasi: https://iseplutpinur.github.io/algoritma_dijkstra/

Daftar Isi

Implementasi Alogritma Djikstra
- Daftar Isi
- Script
- Contoh Kasus
  - Penyelesaian Dengan Penulisan langsung / Coding
  - Dengan GUI/ Halaman Web
- Cara kerja
- Script Untuk Web

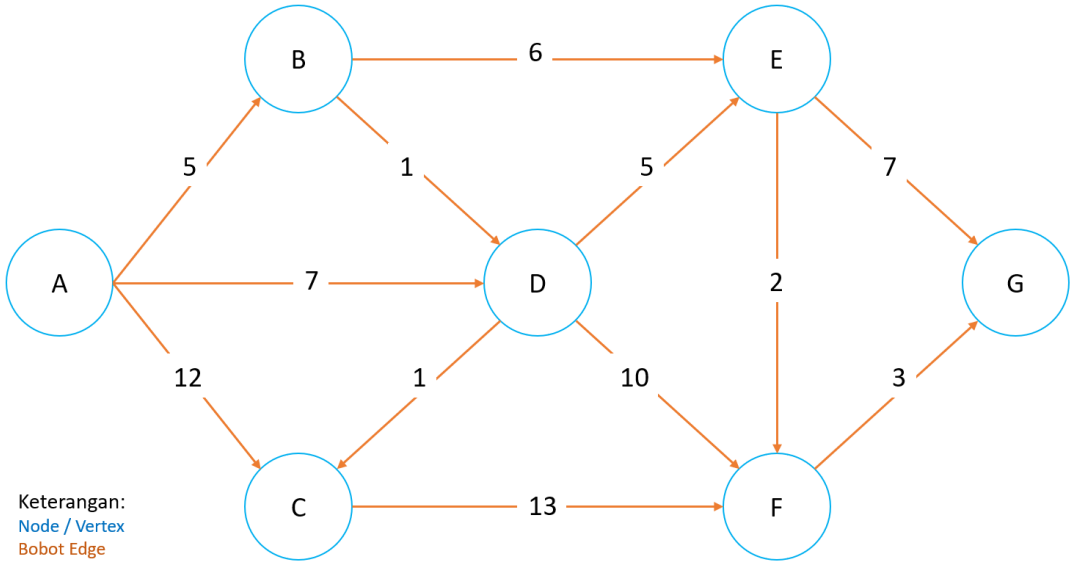
Script

```
// Nama   : Isep Lutpi Nur
// NPM    : 2113191079
// Matkul : Komunikasi Data
// Dosen  : Nanang Hunaifi, ST, MM


// membuat queue untuk keluar masuknya vertex / node
class PriorityQueue {
    constructor() {
        this.values = [];
    }
    //   menambah queue
    enqueue(val, priority) {
        this.values.push({ val, priority });
        this.sort();
    }
    //  menghapus atau mengeluarkan queue
    dequeue() {
        return this.values.shift();
    }
    //   mensorting queue yg lebih pendek
    sort() {
```

```
        this.values.sort((a, b) => a.priority - b.priority);
    }
}

//  membuat graph untuk short path djikstra
class WeightedGraph {
    constructor() {
        this.adjacencyList = {};
    }
    // function membuat vertex tempat  destinasi
    addVertex(vertex) {
        if (!this.adjacencyList[vertex]) this.adjacencyList[vertex] = [];
    }
    //    function membuat edge dan panjang jarak penghubung ke vertex lain
    addEdge(vertex1, vertex2, weight) {
        this.adjacencyList[vertex1].push({ node: vertex2, weight });
        this.adjacencyList[vertex2].push({ node: vertex1, weight });
    }
    //   Short-Path
    Dijkstra(start, finish) {
        const nodes = new PriorityQueue();
        const distances = {};
        const previous = {};
        let path = []; // tempat menembalikan nodes terakhir
        let smallest;
        // membangun initial state
        for (let vertex in this.adjacencyList) {
            if (vertex === start) {
                distances[vertex] = 0;
                nodes.enqueue(vertex, 0);
            } else {
                distances[vertex] = Infinity;
                nodes.enqueue(vertex, Infinity);
            }
            previous[vertex] = null;
        }

        // menentukan panjang path yg dikunjungi
        while (nodes.values.length) {
            smallest = nodes.dequeue().val;
            if (smallest === finish) {
                // selesai sampai tujuan mengembalikan nilai terekhir
                while (previous[smallest]) {
                    path.push(smallest);
                    smallest = previous[smallest];
                }
                break;
            }
            if (smallest || distances[smallest] !== Infinity) {
                for (let neighbor in this.adjacencyList[smallest]) {
                    //  mencari tetangga dari node
                    let nextNode = this.adjacencyList[smallest][neighbor];
                    //menjumlah tetangga node
                    let candidate = distances[smallest] + nextNode.weight;
                    let nextNeighbor = nextNode.node;
                    if (candidate < distances[nextNeighbor]) {
                        //update jarak terkecil antara node  dan tetangga
                        distances[nextNeighbor] = candidate;
```

```
                        //update previous - mendapatkan jarak antar tetanggaa
sebelumnya
                        previous[nextNeighbor] = smallest;
                        //enqueue hasil queue dengan hasil yg baru
                        nodes.enqueue(nextNeighbor, candidate);
                    }
                }
            }
        }
        return path.concat(smallest).reverse();
    }
}
```

Contoh Kasus



Penyelesaian Dengan Penulisan langsung / Coding

1. Inisialisasi Variable dengan class WeightedGraph()

```
const graph = new WeightedGraph();
```

2. Menambah label vertex dengan menggunakan method addVertex(labelvertex).

```
graph.addVertex("A");
graph.addVertex("B");
graph.addVertex("C");
graph.addVertex("D");
graph.addVertex("E");
graph.addVertex("F");
graph.addVertex("G");
```

3. Menambah bobotEdge denan method addEdge(vertex1, vertex2, bobot).

```
graph.addEdge("A", "B", 5);
graph.addEdge("A", "C", 12);
graph.addEdge("A", "D", 7);

graph.addEdge("B", "D", 1);
graph.addEdge("B", "E", 6);

graph.addEdge("C", "F", 13);

graph.addEdge("D", "C", 1);
graph.addEdge("D", "E", 5);
graph.addEdge("D", "F", 10);
```

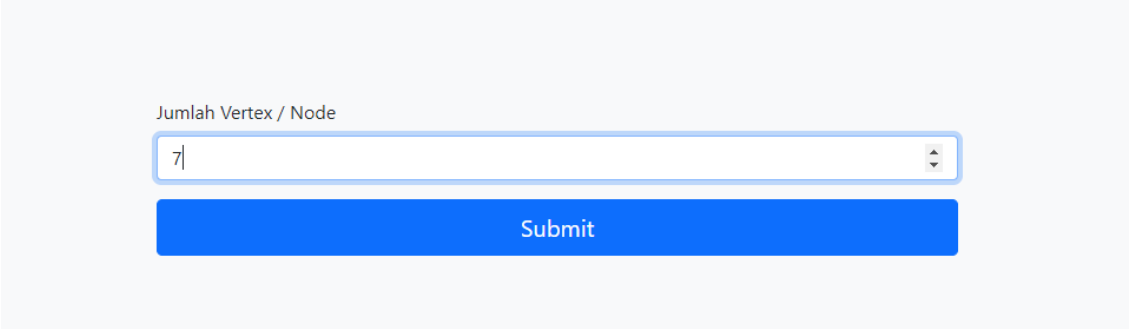
```
graph.addEdge("E", "F", 2);
graph.addEdge("E", "G", 7);

graph.addEdge("F", "G", 3);
```

4. Memanggil methood Djikstra(node awal, node tujuan).

```
console.log(graph.Dijkstra("A", "D"));
// hasil [ 'A', 'B', 'E', 'F', 'G' ]
```

Dengan GUI/ Halaman Web

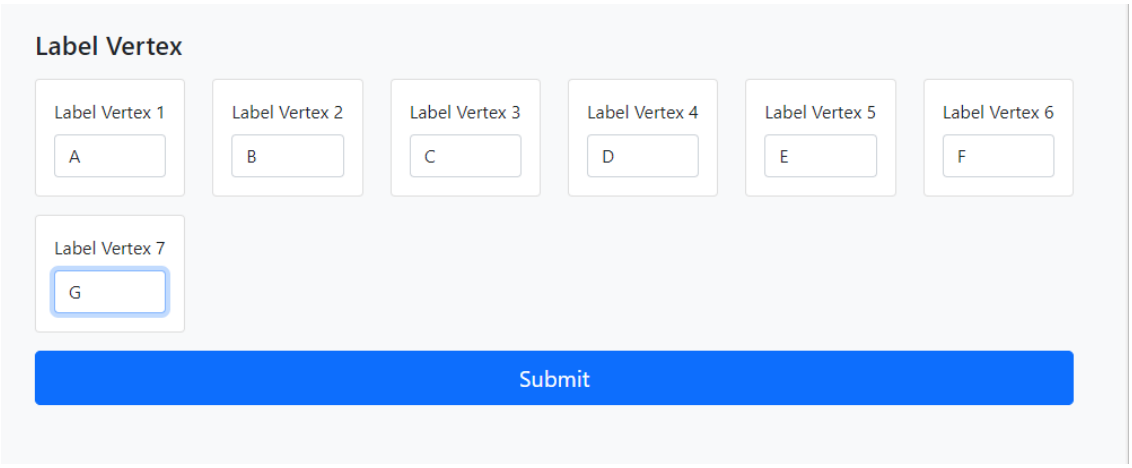
1. Kunjungi halaman: https://iseplutpinur.github.io/algoritma_dijkstra/
2. Masukan Jumlah Vertex sesuai contoh kasus, Lalu Klik Submit.



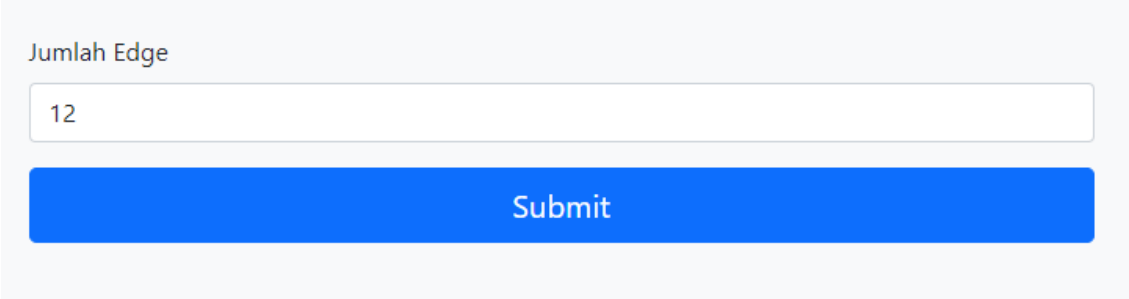
Jumlah vertex makasimal dan minimal serta jumlah karakter untuk label vertex diatur di file assets/script/index.js, Baris 103.

```
const vtrxrule = {
    min: 3,
    max: 10,
    labelcharmax: 1
};
```

3. Masukan Label Vertex sesuai contoh kasus, Lalu Klik Submit.

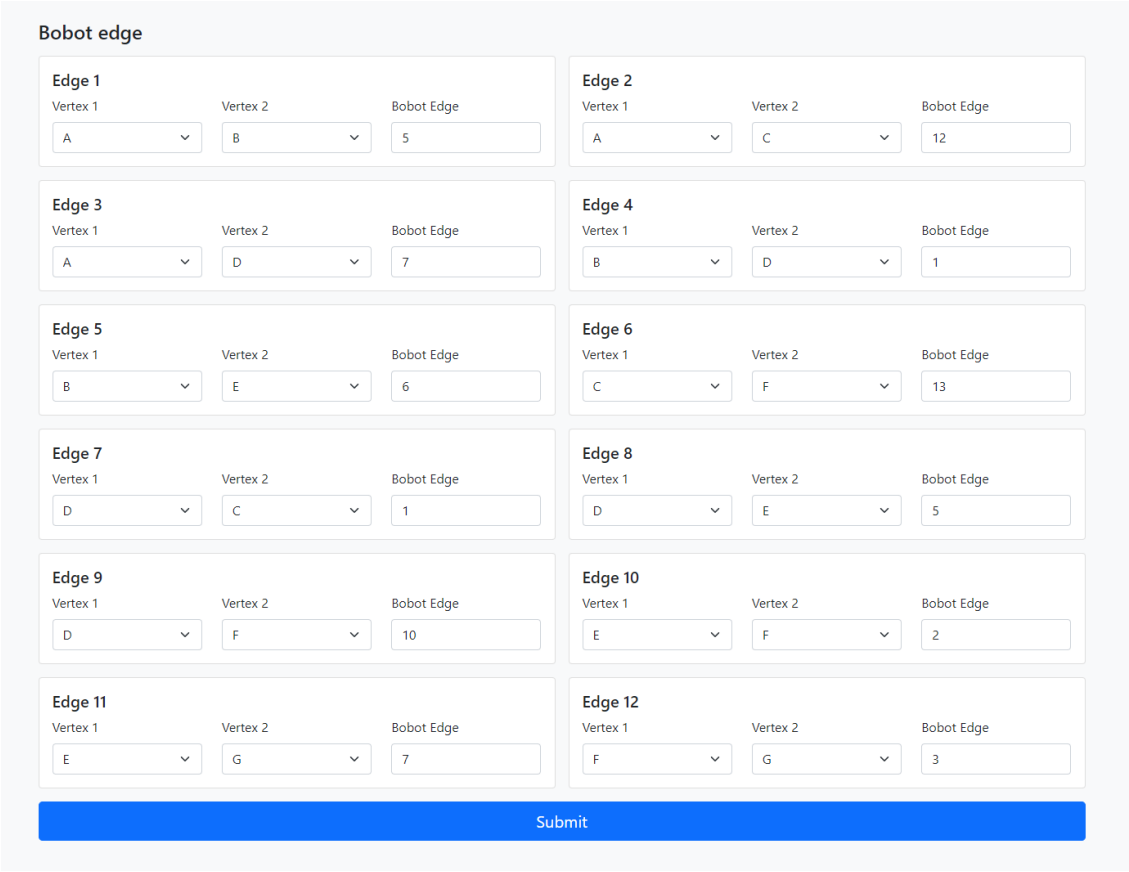


4. Masuka Jumlah edge atau sambungan antara node/vertex sesuai contoh kasus, Lalu Klk Submit.

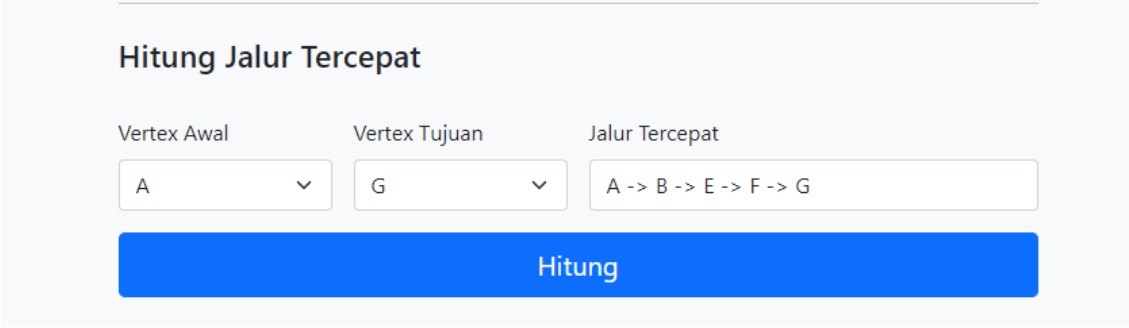


Jumlah Edge minimal adalah sama dengan jumlah vertex dan jumlah makasimal adalah jumlah vertex dikali jumlah vertex min satu (jv*(jv-1)) jv: jumlah vertex/node.

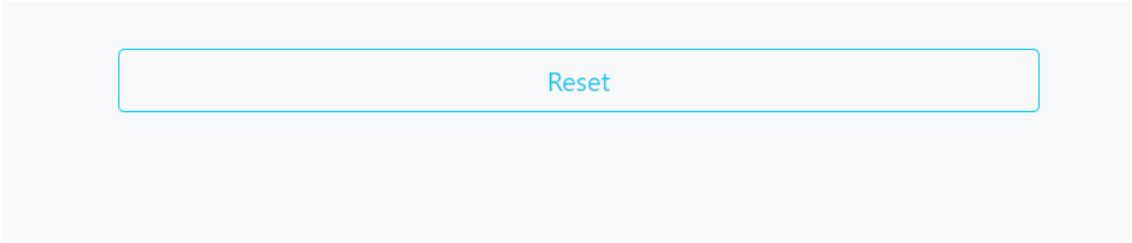
5. Masukan Bobot tiap tiap ede sesuai dengan contoh kasus, lalu klik Submit. *Pada pilihan vertex 2 dan bobot edge akan terbuka jika Vertex 1 Telah di pilih.



6. Pilih Vertex/ Node awal dan Vertex/ Node tujuan, Lalu klik Hitung, Maka kolom jalur tercepat akan menampilkan hasil.



7. Untuk Menggunakan kembali, Klik Tombol reset.



Cara kerja

1. Ketika graph di inisialisasi maka constructor akan mendeklarasikan adjacencyList atau daftar node yang terhubung untuk perhitungan, dideklarasikan dengan tipe data objek.

```
class WeightedGraph {
    constructor() {
        this.adjacencyList = {};
    }
}


......


const graph = new WeightedGraph();
```

2. Ketika method addVertex dijalankan dengan parameter label vertex atau node akan ditambahkan ke properti adjacencyList dengan tipe data array yang nantinya array tersebut akan memuat detal data bobot edge vertex.

```
// function membuat vertex tempat  destinasi
addVertex(vertex) {
    if (!this.adjacencyList[vertex]) this.adjacencyList[vertex] = [];
}

graph.addVertex("A");
graph.addVertex("B");
graph.addVertex("C");
graph.addVertex("D");
graph.addVertex("E");
graph.addVertex("F");
graph.addVertex("G");
```

3. Setelah selesai menambahkan label dari vertex, bobot edge vertex akan ditambahkan sesuai dengan labelnya masing masing kedalam properti adjacencyList[vertex] masing masng.

```
//    function membuat edge dan panjang jarak penghubung ke vertex lain
addEdge(vertex1, vertex2, weight) {
    this.adjacencyList[vertex1].push({ node: vertex2, weight });
    this.adjacencyList[vertex2].push({ node: vertex1, weight });
}

graph.addEdge("A", "B", 5);
graph.addEdge("A", "C", 12);
graph.addEdge("A", "D", 7);

graph.addEdge("B", "D", 1);
graph.addEdge("B", "E", 6);

graph.addEdge("C", "F", 13);

graph.addEdge("D", "C", 1);
graph.addEdge("D", "E", 5);
```

```
graph.addEdge("D", "F", 10);

graph.addEdge("E", "F", 2);
graph.addEdge("E", "G", 7);

graph.addEdge("F", "G", 3);
```

4. setelah label dan bobot tiap tiap node/ vertex didapat maka pencarian jalur tecepat dapat dilakukan dengan cara memanggil method Djikstra(awal, tujuan), dengan dua parameter yaitu parameter awal dan parameter tujuan, method ini mengembalikan/return hasil dari pencarian jalur tercepat dengan tipe data array.

```
class PriorityQueue {
    constructor() {
        this.values = [];
    }
    //   menambah queue
    enqueue(val, priority) {
        this.values.push({ val, priority });
        this.sort();
    }
    //  menghapus atau mengeluarkan queue
    dequeue() {
        return this.values.shift();
    }
    //   mensorting queue yg lebih pendek
    sort() {
        this.values.sort((a, b) => a.priority - b.priority);
    }
}

Dijkstra(start, finish) {
    const nodes = new PriorityQueue();
    const distances = {};
    const previous = {};
    let path = []; // tempat menembalikan nodes terakhir
    let smallest;
    // membangun initial state
    for (let vertex in this.adjacencyList) {
        if (vertex === start) {
            distances[vertex] = 0;
            nodes.enqueue(vertex, 0);
        } else {
            distances[vertex] = Infinity;
            nodes.enqueue(vertex, Infinity);
        }
        previous[vertex] = null;
    }

    // menentukan panjang path yg dikunjungi
    while (nodes.values.length) {
        smallest = nodes.dequeue().val;
        if (smallest === finish) {
            // selesai sampai tujuan mengembalikan nilai terekhir
            while (previous[smallest]) {
                path.push(smallest);
                smallest = previous[smallest];
```

```
            }
            break;
        }
        if (smallest || distances[smallest] !== Infinity) {
            for (let neighbor in this.adjacencyList[smallest]) {
                //  mencari tetangga dari node
                let nextNode = this.adjacencyList[smallest][neighbor];
                //menjumlah tetangga node
                let candidate = distances[smallest] + nextNode.weight;
                let nextNeighbor = nextNode.node;
                if (candidate < distances[nextNeighbor]) {
                    //update jarak terkecil antara node  dan tetangga
                    distances[nextNeighbor] = candidate;
                    //update previous - mendapatkan jarak antar tetanggaa
sebelumnya
                    previous[nextNeighbor] = smallest;
                    //enqueue hasil queue dengan hasil yg baru
                    nodes.enqueue(nextNeighbor, candidate);
                }
            }
        }
    }
    return path.concat(smallest).reverse();
}

console.log(graph.Dijkstra("A", "G"));
// hasil [ 'A', 'B', 'E', 'F', 'G' ]
```

Script Untuk Web

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"
crossorigin="anonymous">

    <meta name="description" content="Implementasi Algoritma Djikstra">
    <meta name="keywords" content="HTML, CSS, JavaScript, Djikstra, Algoritma
Djikstra">
    <meta name="author" content="Isep Lutpi Nur">
    <title>Algoritma Djikstra</title>
</head>

<body class="bg-light">

    <div class="container">
        <main>
```

```
            <div class="py-5 text-center">
                <h2>Implementasi Alogritma Djikstra</h2>
                <p class="lead">Program untuk menentukan jalur tercepat
menggunakan algoritma djikstra,
                    Tugas besar mata kuliah Komunikasi data semester 3
                    <br>
                    <br>Dosen Mata Kuliah:
                    <br>Nanang Hunaifi, ST., MM.
                    <br>
                    <br>Dibuat Oleh:
                    <br>Nama: Isep Lutpi Nur
                    <br>NPM: 2113191079
                </p>
            </div>

            <div class="row g-3">
                <div class="row g-3">
                    <!-- Input jumlah vertex/node -->
                    <div class="col-sm-12">
                        <label for="totalvertex" class="form-label">Jumlah
Vertex / Node</label>
                        <input type="number" class="form-control"
id="totalvertex" placeholder="" value="" required="">
                    </div>
                    <div class="d-grid gap-2">
                        <button class="btn btn-primary btn-lg" type="button"
onclick="totalVertexBtn()">Submit</button>
                    </div>

                    <!-- input vertex -->
                    <div id="inputVertex"></div>

                    <!-- input edge -->
                    <div id="inputEdge"></div>

                    <!-- input jarak vertex -->
                    <div id="countDistance"></div>


                    <hr class="my-4">

                    <button class="btn btn-outline-info btn-lg" type="submit"
onclick="reset()">Reset</button>
                </div>
            </div>
        </main>

        <footer class="my-5 pt-5 text-muted text-center text-small">
            <p class="mb-1">© 2020 Isep Lutpi Nur</p>
            <ul class="list-inline">
                <li class="list-inline-item"><a
href="https://api.whatsapp.com/send?phone=+6285798132505"
                        target="blank">Whatsapp</a></li>
                <li class="list-inline-item"><a
href="https://t.me/+6285798132505" target="blank">Telegram</a></li>
                <li class="list-inline-item"><a
href="https://facebook.com/iseplutpinur7" target="blank">Facebook</a>
                </li>
```

```
            </ul>
        </footer>
    </div>
    <script>
        // Nama   : Isep Lutpi Nur
        // NPM    : 2113191079
        // Matkul : Komunikasi Data
        // Dosen  : Nanang Hunaifi, ST, MM


        // membuat queue untuk keluar masuknya vertex / node
        class PriorityQueue {
            constructor() {
                this.values = [];
            }
            //   menambah queue
            enqueue(val, priority) {
                this.values.push({ val, priority });
                this.sort();
            }
            //  menghapus atau mengeluarkan queue
            dequeue() {
                return this.values.shift();
            }
            //   mensorting queue yg lebih pendek
            sort() {
                this.values.sort((a, b) => a.priority - b.priority);
            }
        }

        //  membuat graph untuk short path djikstra
        class WeightedGraph {
            constructor() {
                this.adjacencyList = {};
            }
            // function membuat vertex tempat  destinasi
            addVertex(vertex) {
                if (!this.adjacencyList[vertex]) this.adjacencyList[vertex] =
[];
            }
            //    function membuat edge dan panjang jarak penghubung ke vertex
lain
            addEdge(vertex1, vertex2, weight) {
                this.adjacencyList[vertex1].push({ node: vertex2, weight });
                this.adjacencyList[vertex2].push({ node: vertex1, weight });
            }
            //   Short-Path
            Dijkstra(start, finish) {
                const nodes = new PriorityQueue();
                const distances = {};
                const previous = {};
                let path = []; // tempat menembalikan nodes terakhir
                let smallest;
                // membangun initial state
                for (let vertex in this.adjacencyList) {
                    if (vertex === start) {
                        distances[vertex] = 0;
                        nodes.enqueue(vertex, 0);
```

```
                    } else {
                        distances[vertex] = Infinity;
                        nodes.enqueue(vertex, Infinity);
                    }
                    previous[vertex] = null;
                }

                // menentukan panjang path yg dikunjungi
                while (nodes.values.length) {
                    smallest = nodes.dequeue().val;
                    if (smallest === finish) {
                        // selesai sampai tujuan mengembalikan nilai terekhir
                        while (previous[smallest]) {
                            path.push(smallest);
                            smallest = previous[smallest];
                        }
                        break;
                    }
                    if (smallest || distances[smallest] !== Infinity) {
                        for (let neighbor in this.adjacencyList[smallest]) {
                            //  mencari tetangga dari node
                            let nextNode = this.adjacencyList[smallest]
[neighbor];
                            //menjumlah tetangga node
                            let candidate = distances[smallest] +
nextNode.weight;
                            let nextNeighbor = nextNode.node;
                            if (candidate < distances[nextNeighbor]) {
                                //update jarak terkecil antara node  dan
tetangga
                                distances[nextNeighbor] = candidate;
                                //update previous - mendapatkan jarak antar
tetanggaa sebelumnya
                                previous[nextNeighbor] = smallest;
                                //enqueue hasil queue dengan hasil yg baru
                                nodes.enqueue(nextNeighbor, candidate);
                            }
                        }
                    }
                }
                return path.concat(smallest).reverse();
            }
        }

        // Label untuk vertex
        let vertexlabel = [];

        // bobot untuk edge yang terhubung
        let vertexweight = [];

        // dgunakan untuk perulangan saat input bobot edge
        let jmledge = null;

        // Atur jumlah maksmal dan minimal vertex yang akan dihitung
        const vtrxrule = {
            min: 3,
            max: 10,
            labelcharmax: 1
```

```
        };

        // element display
        const inputVertex = document.getElementById("inputVertex");
        const inputEdge = document.getElementById("inputEdge");
        const countDistance = document.getElementById("countDistance");

        // menangani inputan jumlah vertex
        function totalVertexBtn() {
            const totalvertex = document.getElementById("totalvertex");

            // validasi jumlah vertex minimal 3 dan maxsimal 10
            if (totalvertex.value >= vtrxrule.min && totalvertex.value <=
vtrxrule.max) {
                let strhtml = `<hr class="my-4">
        <h4 class="mb-3">Label Vertex</h4>
        <div class="row gy-3">
        `;
                for (let i = 0; i < totalvertex.value; i++) {
                    strhtml += `
            <div class="col-sm-3 col-md-2">
            <div class="card">
                <div class="card-body">
            <label for="labelvertex${i}" class="form-label">Label Vertex ${i +
1}</label>
            <input type="text" class="form-control labelvertex"
id="labelvertex${i}"
             placeholder="" required="" onkeyup="checkLabelVertex(this)">

            </div>
            </div>
            </div>
            `;

                }
                strhtml += `
            <div class="d-grid gap-2">
            <button class="btn btn-primary btn-lg" type="button"
onclick="btninpvertex()">Submit</button>
            </div>
            </div>
        `;
                inputVertex.innerHTML = strhtml;

                inputEdge.innerHTML = "";
                countDistance.innerHTML = "";
            } else {
                if (totalvertex.value > vtrxrule.max) totalvertex.value =
vtrxrule.max;
                else if (totalvertex.value < vtrxrule.min) totalvertex.value =
vtrxrule.min;
                alert(`Jumlah vertex minimal ${vtrxrule.min} dan maksimal
${vtrxrule.max}`);
                totalvertex.focus();
            }
        }

        // validasi Karakter label
```

```
        function checkLabelVertex(th) {
            th.value = th.value.toUpperCase();
            document.querySelectorAll(".labelvertex").forEach(m => {
                if (m.value != "" && m != th) {
                    if (th.value == m.value) {
                        th.value = "";
                        alert(`Label Vertex ${th.value} sudah digunakan`);
                    }
                }
            });

            if (th.value.length > vtrxrule.labelcharmax) {
                th.value = th.value.slice(0, vtrxrule.labelcharmax);
            }
        }

        // validasi label vertex sekaligus membuat inputan jumlah edge
        function btninpvertex() {
            // reset vertex label
            vertexlabel = [];
            // digunakan untuk memvalidasi label vertex apakah sudah di isi atau
belum
            let cek = true;

            document.querySelectorAll(".labelvertex").forEach((m, i) => {
                vertexlabel.push(m.value);
                if (m.value == "" || m.value.length > 1) {
                    if (cek) {
                        m.focus()
                        alert(`Label Vertex ${i + 1} Belum Di Isi`);
                    }
                    cek = false;
                }
            });

            if (cek) {

                // membuat elemen untuk jumlah inputan bobot edge
                inputEdge.innerHTML = `
                    <hr class="my-4">
                    <div class="col-sm-12 g-3">
                        <label for="jmledge" class="form-label">Jumlah
Edge</label>
                        <input type="number" class="form-control" id="jmledge"
                        required="">
                    </div>
                    <div class="d-grid gap-2 mt-3">
                            <button class="btn btn-primary btn-lg "
type="button"
                            onclick="btnjmledge()">Submit</button>
                    </div>
                    <div id="edgevalueinput"></div>
                `;
            }
        }

        // validasi bobot edge
        function btnjmledge() {
```

```
            const totalvertex = document.getElementById("totalvertex");
            const edgevalueinput = document.getElementById("edgevalueinput");
            const jmledgerule = {
                min: Number(totalvertex.value),
                max: Number(totalvertex.value) * (Number(totalvertex.value) - 1)
            }
            let strhtml = `<hr class="my-4">
    <h4 class="mb-3">Bobot edge</h4>
    <div class="row gy-3 gx-3">`;
            let stropt = ``;
            jmledge = document.getElementById("jmledge");
            if (jmledge.value >= jmledgerule.min &&
                jmledge.value <= jmledgerule.max) {
                vertexlabel.forEach(n => {
                    stropt += `<option value="${n}">${n}</option>`;
                })
                for (let i = 1; i <= jmledge.value; i++) {
                    strhtml += `
            <div class="col-md-6">
            <div class="card">
                <div class="card-body">
                    <h5 class="card-title">Edge ${i}</h5>
                        <div class="row gy-3">
                            <div class="col-4">
                                <label for="edge_${i}_1" class="form-
label">Vertex 1</label>
                                <select class="form-select" id="edge_${i}_1"
                                    onclick="vertex1Click(this)"
onchange="vertex1Click(this)">
                                    ${stropt}
                                </select>
                            </div>

                            <div class="col-4">
                                <label for="edge_${i}_2" class="form-
label">Vertex 2</label>
                                <select class="form-select" id="edge_${i}_2"
disabled>
                                </select>
                            </div>

                            <div class="col-4">
                                <label for="bobotedge_${i}" class="form-
label">Bobot Edge</label>
                                <input type="number" class="form-control"
id="bobotedge_${i}"
                                required="" disabled>
                            </div>

                        </div>
                    </div>
                </div>
            </div>

            `;
                }
                strhtml += `
        <div class="d-grid gap-2 mt-3">
```

```
        <button class="btn btn-primary btn-lg" type="button"
onclick="countToVertexWeight()">Submit</button>
        </div>
        </div>
        `;

            } else {
                if (jmledge.value > jmledgerule.max) {
                    jmledge.value = jmledgerule.max;
                } else if (jmledge.value < jmledgerule.min) {
                    jmledge.value = jmledgerule.min;
                }

                alert(`Jumlah vertex minimal ${jmledgerule.min} dan maksimal
${jmledgerule.max}`);
                jmledge.focus();
            }


            jmledge = Number(jmledge.value);
            edgevalueinput.innerHTML = strhtml;
        }

        // validasi ketika vertex 1 dipilih
        function vertex1Click(th, ht = false) {
            let inpedge = th.id.split("_");
            let inpel = document.getElementById(`edge_${inpedge[1]}_2`);
            let stropt = ``;

            inpel.disabled = false;
            vertexlabel.forEach(n => {
                if (ht) {
                    if (n != th.value) stropt += `<option value="${n}">${n}
</option>`;
                } else {
                    if (n != th.value && vertex2Check(th, n)) stropt += `<option
value="${n}">${n}</option>`;
                }
            })
            inpel.innerHTML = stropt;
            document.getElementById(`bobotedge_${inpedge[1]}`).disabled = false;
        }

        // Validasi sambungan vertex label
        function vertex2Check(v1, v2) {
            let cek = true;
            for (let i = 1; i <= jmledge; i++) {
                let vrtx1 = document.getElementById(`edge_${i}_1`);
                let vrtx2 = document.getElementById(`edge_${i}_2`).value;
                if (vrtx1.value == v1.value && vrtx2 == v2 & v1 != vrtx1) cek =
false
            }

            return cek;
        }

        // validasi bobot edge sekaligus submit data ke variable utama
"vertexweight"
```

```
        function countToVertexWeight() {
            vertexweight = [];
            let cekbobot = true;

            // Pengecekan inputan bobot edge sudah di isi atau belum
            for (let i = 1; i <= jmledge; i++) {
                let vrtx1 = document.getElementById(`edge_${i}_1`).value;
                let vrtx2 = document.getElementById(`edge_${i}_2`).value;
                let bobot = document.getElementById(`bobotedge_${i}`);
                if (bobot.value != "" && vrtx2 != "") {
                    vertexweight[i] = {
                        v1: vrtx1,
                        v2: vrtx2,
                        bb: bobot.value
                    }
                } else {
                    if (cekbobot) {
                        if (bobot.value == "") {
                            alert(`Bobot Edge ${i} Belum di isi..`);
                            bobot.focus();
                        } else {
                            alert(`Vertex 2 Edge ${i} Belum di pilih..`);
                        }
                    }
                    cekbobot = false;
                }
            }

            // membuat elemen untuk memilih node awal dan node tujuan
            if (cekbobot) {

                let stropt = ``;
                vertexlabel.forEach(n => {
                    stropt += `<option value="${n}">${n}</option>`;
                })
                let strhtml = `
                    <hr class="my-4">
                    <div class="row gy-3 gx-3">
                    <h4 class="mb-3">Hitung Jalur Tercepat</h4>
                        <div class="col-sm-3 g-3">
                            <label for="edge_999_1" class="form-label">Vertex
Awal</label>
                            <select class="form-select"  id="edge_999_1"
                            onclick="vertex1Click(this,true)"
onchange="vertex1Click(this,true)">
                                ${stropt}
                            </select>
                        </div>
                        <div class="col-sm-3 g-3">
                            <label for="edge_999_2" class="form-label">Vertex
Tujuan</label>
                            <select id="edge_999_2" class="form-select"
disabled>

                            </select>
                        </div>
                        <div class="col-sm-6 g-3">
```

```
                            <label for="jmledge" class="form-label">Jalur
Tercepat</label>
                            <input class="form-control" type="text" id="result">
                        </div>

                        <div class="d-grid gap-2 mt-3">
                                <button class="btn btn-primary btn-lg "
type="button"
                                onclick="btnHitung()">Hitung</button>
                        </div>
                    </div>
                    <input type="number"
                    id="bobotedge_999"
                    disabled style="display:none;">
            `;

                document.getElementById("countDistance").innerHTML = strhtml;

            }
        }

        function btnHitung() {
            const vinp1 = document.getElementById(`edge_999_1`);
            const vinp2 = document.getElementById(`edge_999_2`);
            const result = document.getElementById(`result`);
            const v1 = vinp1.value;
            const v2 = vinp2.value;
            //  instanisasi  Graph
            const graph = new WeightedGraph();

            vertexlabel.forEach(n => {
                //  menambah vertex
                graph.addVertex(n);

            });

            for (let i = 1; i <= jmledge; i++) {
                //  menambah vertex konektor Edge
                graph.addEdge(vertexweight[i].v1,
                    vertexweight[i].v2,
                    Number(vertexweight[i].bb)
                );
            }
            //  menampilkan isi graph
            // console.log(graph.adjacencyList);

            // memanggil dijkstra dan mentukan jarak terdekat
            // console.log(graph.Dijkstra(v1, v2));

            let hasil = "";
            graph.Dijkstra(v1, v2).forEach(n => {
                if (hasil == "") hasil += n;
                else hasil += ` -> ${n}`;
            });

            result.value = hasil;
            result.removeAttribute("hidden");
        }
```

```
        function reset() {
            inputVertex.innerHTML = "";
            inputEdge.innerHTML = "";
            countDistance.innerHTML = "";
            document.getElementById("totalvertex").focus();
        }

    </script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW"
        crossorigin="anonymous"></script>

</body>

</html>
```