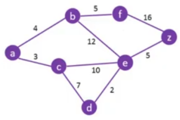
DIJKSTRA ALGORITHM



Set distance to startNode to zero.

Set all other distances to an infinite value.

We add the startNode to the unsettled nodes set.

While the unsettled nodes set is not empty we:

- Choose an evaluation node from the unsettled nodes set, the evaluation node should be the one with the lowest distance from the source.
- Calculate new distances to direct neighbors by keeping the lowest distance at each evaluation.
- Add neighbors that are not yet settled to the unsettled nodes set.

aturan!

- jika bobotnya sama. maka yang vertex selanjutnya yang akan dieksekusi adalah vertex yang pertama kali dimasukan

- bobot edge harus positif

unsettledVertices

settledVertices

	a	b	c	d	e	f	z
a	0a	4a	3a	-	-	-	-
c	0a	4a	3a	10c	13c	-	-
b	0a	4a	3a	10c	13c	9b	-
f	0a	4a	3a	10c	13c	9b	25f
d	0a	4a	3a	10c	12d	9b	25f
e	0a	4a	3a	10c	12d	9b	17e
z	0a	4a	3a	10c	12d	9b	17e

16b > 13c.

jadi disimpennya adalah 13c

12d < 13c.

jadi disimpennya adalah 12d

while = 7
looping vertex = 49
looping edge= 13

a → edge ke b (4a), ke c (3a) → 2 edge
c → edge ke d (10c), ke e (13c) → 2 edge
b → edge ke c (10c), ke e (13c), ke f (9b) → 3 edge
f → edge ke d (10c), ke e (13c), ke z (25f) → 3 edge
d → edge ke e (12d), ke z (25f) → 2 edge
e → edge ke z (17e) → 1 edge
z → 0 edge

karna sudah selesai, selanjutnya
1. tentukan nilai z yang terkecil
2. tentukan vertex yang didahului oleh z, maka ke e, maka ke d, maka ke c, maka ke a
3. dibalik jadi a - c - d - e - z

ada 3 looping

looping untuk memasukan vertex kedalam settledVertices.

```
while (!unsettledVertices.isEmpty()) {
```

maksimal berjalan sesuai jumlah vertex yang diinput = V

Looping mencari vertex dengan jarak terkecil

```
for (Vertex vertex : unsettledVertices) {
```

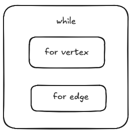
dipanggil setiap kali while berjalan = V x V

Looping memproses semua edge dari currentVertex

```
for (Edge edge : currentVertex.getEdges()) {
```

dipanggil setiap kali currentVertex ditemukan = E

step in baeldung algorithm
1. set jarak vertex source = 0
2. looping vertex (unsettledVertex) untuk mendapatkan jarak sementara yang paling kecil.
    1. ambil vertex yang jarak terkecil (akan ditetapkan menjadi currentVertex)
    1. kemudian proses lagi edge dari currentVertex untuk mendapatkan vertex tujuan dan bobotnya
    2. update jarak untuk vertex (unsettledVertex)
    3. update jarak minimum jika lebih pendek
        1. Hitung jarak baru: jarak_source + bobot_edge
        2. Bandingkan dengan jarak lama di evaluationVertex
        3. Jika lebih kecil → update jarak
        4. Update juga shortestPath (jalur menuju vertex)
    4. Tambahkan currentVertex ke settledVertices
    5. lalu looping kembali sampai semua vertex selesai


while

for vertex

for edge

input data graph

```
{
    "vertices": ["a","b","c","d","e","f","z"],
    "edges": [
        { "source": "a", "destination": "b", "weight": 4 },
        { "source": "a", "destination": "c", "weight": 3 },
        { "source": "b", "destination": "f", "weight": 5 },
        { "source": "b", "destination": "e", "weight": 12 },
        { "source": "f", "destination": "z", "weight": 16 },
        { "source": "z", "destination": "e", "weight": 5 },
        { "source": "c", "destination": "e", "weight": 10 },
        { "source": "c", "destination": "d", "weight": 7 },
        { "source": "d", "destination": "e", "weight": 2 }
    ]
}
```

output Algorithm

```
{
"vertices": [
    {
        "name": "a",
        "distance": 0,
        "shortestPath": []
    },
    {
         "name": "c",
         "distance": 3,
         "shortestPath": ["a"]
    },
    {
          "name": "b",
          "distance": 4,
          "shortestPath": ["a"]
    },
    {
          "name": "f",
          "distance": 9,
          "shortestPath": ["a", "b"]
    },
    {
          "name": "d",
          "distance": 10,
          "shortestPath": ["a", "c"]
    },
    {
         "name": "e",
         "distance": 12,
         "shortestPath": ["a", "c", "d"]
    },
    {
         "name": "z",
         "distance": 17,
         "shortestPath": ["a", "c", "d", "e"]
    }
  ]
}
```

input source and destination

```
start = "a"
end = "z"
```

output DijkstraResult

```
"path": ["a", "c", "d", "e", "z"],
"distance": 17
```