Algoritma dan penjelasan Dijkstra shortest patch 1

1.Inisialisasi:

Algoritma dimulai dengan menginisialisasi beberapa struktur data,seperti PriorityQueue untuk mengelola simpul yang akan dieksplorasi berikutnya,distances unutk menyimpan jarak terpendek dari titik awal ke setial titik lain,patchs untuk menyimpan jalur terpendek dari titik awal ke setiap titik lain.

Scriptnya:

```
queue = PriorityQueue()

queue.put((0, start))

distances = {node: float('infinity') for node in graph}

distances[start] = 0

paths = {start: []}
```

2.Penelusuran Graf:

Algoritma menggunakan perulangan while untuk melakukan penelusuran graf.Selama masih ada simpul yang perlu dieksplorasi,program akan terus melakukan literasi.

Scriptnya:

```
while not queue.empty():

  (dist, current_node) = queue.get()
```

3.Pengecekan Titik Akhir:

Selama penelusuran,program akan memeriksa apakah simpul yang sedang diesplorasi adalah titik akhir yang diinginkan.jika iya,maka algoritma selesai dan mengembalikan jarak terpendek dan jalur terpendek dari titik awal ke titik akhir

Scriptnya:

```
if current_node == end:

  return distances[end], paths[end]
```

4.Relaksasi:

Program akan mengekplorasi tetangga-tetangga dari simpul saat ini,dan melakukan relaksasi jika ditemukan jarak lebih pendek.Relaksasi dilakukan dengan membandingkan jarak saat ini dengan jarak yang baru dihitung melalui simpul saat ini.

Scriptnya:

```
for neighbor, weight in graph[current_node]:
    old_dist = distances[neighbor]
    new_dist = dist + weight


    if new_dist < old_dist:
        distances[neighbor] = new_dist
        paths[neighbor] = paths[current_node] + [neighbor]
        queue.put((new_dist, neighbor))
```

5.Pembentukan Graf:

Sebelum menjalankan algoritma Dijkstra,graf harus dibentuk terlebih dahulu.Fungsi 'build_graph' digunakan untuk mengonversi matriks bobot menjadi representasi graf dalam bentuk dictionary

Scriptnya:

```
def build_graph(matrix):
    graph = {i: [] for i in range(len(matrix))}
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if i != j:
                graph[i].append((j, matrix[i][j]))
    return graph
```

6.Tampilan Graf:

Fungsi 'display_graph' digunakan untuk menampilkan representasi graf yang telah dibentuk.

Scriptnya:

```
def display_graph(graph):
    for node in graph:
        print(f"Node {node} is connected to {graph[node]}")
```

7.Pemanggilan Fungsi dan Output:

Terakhir,fungsi-fungsi ini dipanggil matriks bobot sebagai input,dan hasilnya ditampilkan.

Scriptnya:

```
matrix = [
    [0, 4, 8, 1],
    [4, 0, 8, 11],
    [8, 8, 0, 7],
    [1, 11, 7, 0]
]

graph = build_graph(matrix)
display_graph(graph)

shortest_path = dijkstra(graph, 0, 3)
print(f"Shortest path: {shortest_path[0]}, Path: {shortest_path[1]}")
```