Nama : Muhammad Endriansyah Rahul Sudarsono
NIM : 20220029

Laporan praktikum Diivde and Conquer

contoh kode python untuk mengetahui jarak terpendek dari node A ke Node E, jika diketahui : jarak node A ke B =30 jarak node A ke C = 50 jarak node B ke C =10 jarak node C ke D = 40 jarak node dari B ke D = 10 jarak dari node C ke D = 20 jarak dari node C ke E = 30 jarak dari node D ke E = 50

```python
import heapq

def dijkstra(graph, start, end):
    # Inisialisasi jarak ke setiap node dengan nilai infinity
    distances = {node: float('inf') for node in graph}
    # Jarak ke node awal diatur ke 0
    distances[start] = 0
    # Setiap node yang sudah dievaluasi disimpan di set visited
    visited = set()
    # Queue untuk menyimpan node yang akan dievaluasi berikutnya
    heap = [(0, start)]

    while heap:
        # Ambil node dengan jarak terdekat dari heap
        (distance, current) = heapq.heappop(heap)
        # Jika node tersebut belum dievaluasi sebelumnya
        if current not in visited:
            # Tandai node tersebut sebagai dievaluasi
            visited.add(current)
            # Loop melalui semua node yang terhubung dengan node tersebut
            for neighbor, weight in graph[current].items():
                # Hitung jarak baru ke node tersebut
                new_distance = distance + weight
                # Jika jarak baru lebih pendek dari jarak sebelumnya ke node tersebut
                if new_distance < distances[neighbor]:
                    # Update jarak ke node tersebut
                    distances[neighbor] = new_distance
                    # Tambahkan node tersebut ke heap
                    heapq.heappush(heap, (new_distance, neighbor))
```

```
    # Kembalikan jarak terpendek ke node tujuan
    return distances[end]

# Definisikan grafik
graph = {
    'A': {'B': 30, 'C': 50},
    'B': {'C': 10, 'D': 10},
    'C': {'D': 20, 'E': 30},
    'D': {'E': 50},
    'E': {}
}

# Hitung jarak terpendek dari A ke E
shortest_distance = dijkstra(graph, 'A', 'E')

# Cetak jarak terpendek
print(f"Jarak terpendek dari A ke E: {shortest_distance}")
```

hasil dasi source code diatas ketika dijalankan adalah

```
Jarak terpendek dari A ke E: 70
```

Penjelasan:

Script Python di atas mengimplementasikan algoritma Dijkstra untuk mencari jarak terpendek antara dua node dalam sebuah grafik yang diberikan.

Pertama-tama, grafik didefinisikan sebagai kamus Python, di mana setiap kunci dalam kamus adalah simpul grafik dan nilai-nilainya adalah kamus lain yang berisi simpul lain yang terhubung ke simpul itu beserta bobotnya.

Fungsi dijkstra kemudian didefinisikan dengan tiga parameter masukan, yaitu grafik yang diberikan, simpul awal dari mana pencarian dimulai, dan simpul tujuan di mana pencarian berakhir.

Di dalam fungsi, jarak ke setiap simpul dalam grafik diinisialisasi dengan nilai tak terbatas, kecuali simpul awal yang diatur ke 0. Selanjutnya, setiap simpul yang dievaluasi akan ditandai sebagai "visited" dan dimasukkan ke dalam set "visited".

Heap queue digunakan untuk menyimpan simpul yang akan dievaluasi berikutnya, dengan jarak terdekat diprioritaskan. Kemudian, simpul dengan jarak terdekat diambil dari heap queue dan jika simpul tersebut belum dievaluasi sebelumnya, maka loop dilakukan untuk semua simpul yang terhubung dengan simpul tersebut.

Jarak baru ke simpul tetangga dihitung dan jika jarak baru lebih pendek dari jarak sebelumnya, maka jarak ke simpul tersebut akan diupdate. Kemudian, simpul akan dimasukkan ke dalam heap queue untuk dievaluasi di iterasi selanjutnya. Proses ini akan terus berlanjut hingga simpul tujuan mencapai dan jarak terpendek dari simpul awal ke simpul tujuan akan dikembalikan sebagai output dari fungsi.

Terakhir, jarak terpendek dari simpul A ke simpul E dicetak menggunakan fungsi print.