Nama : febi rahmadia putri

Nim : 20220021

Algoritma Jaringan

Latihan 2

a. menggunakan fungsi minimum_spanning_tree dengan memberikan graf dan simpul awal sebagai argumen. Graf harus berbentuk representasi dengan menggunakan dictionary, di mana kunci adalah simpul dan nilai adalah dictionary lain yang berisi simpul tujuan dan bobotnya.

```python
import heapq

def minimum_spanning_tree(graph, start):
    # Inisialisasi variabel yang diperlukan
    visited = set([start])
    edges = [(cost, start, end) for end, cost in graph[start].items()]
    heapq.heapify(edges)
    mst_cost = 0
    mst_edges = []

    # Looping sampai semua simpul dikunjungi
    while edges:
        # Ambil simpul dengan bobot terkecil
        cost, start, end = heapq.heappop(edges)
        # Jika simpul tujuan belum dikunjungi
        if end not in visited:
            # Tambahkan simpul ke daftar simpul yang sudah dikunjungi
            visited.add(end)
            # Tambahkan jarak ke total biaya minimum
            mst_cost += cost
            # Tambahkan edge ke daftar MST
            mst_edges.append((start, end, cost))
            # Tambahkan edge dari simpul tujuan ke daftar edge
            for end_next, cost_next in graph[end].items():
                if end_next not in visited:
                    heapq.heappush(edges, (cost_next, end, end_next))

    # Kembalikan daftar edge dan total biaya minimum
    return mst_edges, mst_cost
```

Contoh penggunaan:
Output akan menampilkan daftar edge pada Minimum Spanning Tree (MST) beserta total biaya minimum.

```python
graph = {
    'A': {'B': 2, 'C': 3},
    'B': {'A': 2, 'C': 1},
    'C': {'A': 3, 'B': 1}
}

start_node = 'A'

# Panggil fungsi minimum_spanning_tree
mst_edges, mst_cost = minimum_spanning_tree(graph, start_node)

# Tampilkan hasil
print("Minimum Spanning Tree Edges:")
for edge in mst_edges:
    print(edge)
print("Total Cost:", mst_cost)
```

Dengan Hasil

```
Minimum Spanning Tree Edges:
('A', 'B', 2)
('B', 'C', 1)
Total Cost: 3
```

b. Sebuah perusahaan ingin membangun jaringan kabel fiber optik untuk menghubungkan beberapa kota besar di Indonesia. Perusahaan ingin meminimalkan biaya pembangunan infrastruktur dan jarak kabel yang harus ditempuh untuk menghubungkan semua kota. Perusahaan telah membuat peta wilayah dan menentukan lokasi kota-kota besar serta jarak antara setiap kota. Dalam hal ini, algoritma jaringan minimum dapat digunakan untuk menemukan rute kabel terpendek yang meminimalkan biaya pembangunan infrastruktur. Berikut ini adalah kode Python untuk membangun grafik dan mencari rute terpendek menggunakan algoritma jaringan minimum:

```python
import heapq

# Definisikan grafik dengan jarak antara setiap kota
graph = {
    'Jakarta': {'Bandung': 140, 'Semarang': 400, 'Surabaya': 800},
    'Bandung': {'Jakarta': 140, 'Semarang': 350, 'Surabaya': 900, 'Medan': 1250},
    'Semarang': {'Jakarta': 400, 'Bandung': 350, 'Surabaya': 650},
    'Surabaya': {'Jakarta': 800, 'Bandung': 900, 'Semarang': 650, 'Medan': 1100},
    'Medan': {'Bandung': 1250, 'Surabaya': 1100}
}

# Definisikan fungsi untuk mencari rute terpendek menggunakan algoritma jaringan minimum
def minimum_spanning_tree(graph, start):
    visited = set([start])
    edges = [(cost, start, end) for end, cost in graph[start].items()]
    heapq.heapify(edges)
    mst_cost = 0
    mst_edges = []

    while edges:
        cost, start, end = heapq.heappop(edges)
        if end not in visited:
            visited.add(end)
            mst_cost += cost
            mst_edges.append((start, end, cost))
            for end_next, cost_next in graph[end].items():
        cost, start, end = heapq.heappop(edges)
        if end not in visited:
            visited.add(end)
            mst_cost += cost
            mst_edges.append((start, end, cost))
            for end_next, cost_next in graph[end].items():
                if end_next not in visited:
                    heapq.heappush(edges, (cost_next, end, end_next))

    return mst_edges, mst_cost

# Cetak rute terpendek dan biaya minimum menggunakan algoritma jaringan minimum
rute, biaya = minimum_spanning_tree(graph, 'Jakarta')
print(f"Rute Terpendek: {rute}")
print(f"Biaya Minimum: {biaya}")
```

Dengan Hasil

```
Rute Terpendek: [('Jakarta', 'Bandung', 140), ('Bandung', 'Semarang', 350), ('Semarang', 'Surabaya', 650), ('Surabaya', 'M
edan', 1100)]
Biaya Minimum: 2240
```

Soal :

1. Jelaskan algoritma aliran maksimum dan bagaimana ia dapat diterapkan dalam pemecahan masalah jaringan pada kehidupan nyata.
   Algoritma aliran maksimum adalah metode yang digunakan untuk mencari aliran maksimum di dalam jaringan yang terdiri dari simpul-simpul yang terhubung oleh jalur-jalur dengan kapasitas tertentu. Algoritma ini berguna dalam pemecahan masalah jaringan di kehidupan nyata karena dapat membantu mengoptimalkan penggunaan sumber daya dan meningkatkan efisiensi aliran di dalam jaringan.
   Dalam pemecahan masalah jaringan pada kehidupan nyata, algoritma aliran maksimum dapat diterapkan sebagai berikut:
   a. Distribusi Barang: Dalam rantai pasokan, algoritma aliran maksimum dapat membantu menentukan jalur pengiriman yang efisien dan mengoptimalkan aliran barang dari pabrik ke gudang-gudang atau toko-toko. Dengan memodelkan jaringan distribusi sebagai grafik dengan kapasitas pada setiap jalur, algoritma ini dapat membantu mengatur aliran barang dengan memaksimalkan penggunaan sumber daya dan meminimalkan biaya transportasi.
   b. Jaringan Komunikasi: Dalam jaringan komunikasi seperti internet atau jaringan telekomunikasi, algoritma aliran maksimum dapat digunakan untuk mengatur aliran data di antara node-node jaringan. Misalnya, dengan memodelkan kapasitas saluran komunikasi antar node, algoritma ini dapat membantu mengalokasikan kapasitas dengan efisien, meminimalkan tumpukan data (congestion), dan memastikan pengiriman data yang optimal.
   c. Transportasi: Dalam sistem transportasi seperti jalan raya atau jaringan rel, algoritma aliran maksimum dapat digunakan untuk mengoptimalkan aliran lalu lintas. Dengan memodelkan jaringan transportasi sebagai grafik dengan kapasitas pada setiap jalur, algoritma ini dapat membantu dalam perencanaan rute, penjadwalan transportasi, dan pengaturan lalu lintas yang efisien.
   d. Manajemen Proyek: Dalam manajemen proyek, algoritma aliran maksimum dapat digunakan untuk mengoptimalkan penggunaan sumber daya dan mengatur aliran pekerja atau peralatan di dalam proyek. Dengan memodelkan jaringan proyek sebagai grafik dengan kapasitas pada setiap jalur, algoritma ini dapat membantu dalam mengoptimalkan jadwal, mengurangi konflik sumber daya, dan meningkatkan efisiensi pelaksanaan proyek.
      Dengan menggunakan algoritma aliran maksimum, kita dapat mencapai solusi yang optimal dalam pemecahan masalah jaringan di kehidupan nyata, menghasilkan penggunaan sumber daya yang efisien, mengurangi biaya, dan meningkatkan kinerja sistem secara keseluruhan.
2. Apa yang dimaksud dengan algoritma jaringan minimum? Jelaskan secara detail bagaimana algoritma ini dapat digunakan untuk menentukan rute terpendek pada jaringan.
   Algoritma jaringan minimum, juga dikenal sebagai algoritma rute terpendek, adalah metode yang digunakan untuk mencari rute terpendek antara dua simpul di dalam jaringan yang terdiri dari simpul-simpul yang terhubung oleh jalur-jalur dengan bobot

atau jarak tertentu. Algoritma ini bertujuan untuk menemukan rute dengan total bobot atau jarak yang paling kecil.

Salah satu contoh algoritma jaringan minimum yang umum digunakan adalah Algoritma Dijkstra. Berikut adalah langkah-langkah yang dilakukan oleh algoritma ini untuk menentukan rute terpendek pada jaringan:

a. Inisialisasi: Setiap simpul dalam jaringan diberi label jarak tak terhingga kecuali simpul awal, yang diberi label jarak 0. Semua simpul diberi status "belum dikunjungi".
b. Pilih simpul awal: Simpul awal dipilih sebagai simpul saat ini.
c. Perbarui jarak: Dari simpul saat ini, perbarui label jarak ke semua tetangganya. Jarak baru dihitung sebagai jumlah jarak saat ini dengan bobot jalur antara simpul saat ini dan tetangga, kemudian dibandingkan dengan label jarak tetangga saat ini. Jika jarak baru lebih kecil, maka label jarak tetangga diperbarui.
d. Tandai simpul saat ini sebagai "sudah dikunjungi".
e. Pilih simpul dengan jarak terkecil: Dari simpul yang belum dikunjungi, pilih simpul dengan label jarak terkecil sebagai simpul saat ini. Jika tidak ada simpul yang tersedia, berarti proses selesai.
f. Ulangi langkah 3 hingga langkah 5 sampai semua simpul dikunjungi atau jika simpul tujuan telah dikunjungi.
g. Rekonstruksi rute terpendek: Setelah semua simpul dikunjungi atau simpul tujuan dikunjungi, rute terpendek dapat direkonstruksi dengan mengikuti jalur terpendek dari simpul tujuan ke simpul awal menggunakan label jarak yang telah diperbarui selama proses.

   Algoritma jaringan minimum seperti Algoritma Dijkstra memiliki kegunaan yang luas dalam pemecahan masalah rute terpendek pada jaringan nyata. Contoh penggunaannya meliputi penentuan rute terpendek dalam jaringan transportasi, perencanaan rute dalam sistem logistik dan distribusi, perutean paket data dalam jaringan komunikasi, dan penjadwalan transportasi dalam sistem transportasi.

3. Bagaimana cara menentukan rute terpendek pada sebuah grafik yang memiliki bobot atau jarak antara setiap node yang berbeda menggunakan algoritma Dijkstra?
   Berikut ini adalah contoh implementasi algoritma Dijkstra dalam bahasa pemrograman Python:

```python
import heapq

def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    queue = [(0, start)]
    while queue:
        current_distance, current_node = heapq.heappop(queue)
        if current_distance > distances[current_node]:
            continue
        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(queue, (distance, neighbor))
    return distances

# Contoh penggunaan algoritma Dijkstra untuk menentukan rute terpendek
graph = {
    'A': {'B': 10, 'C': 15},
    'B': {'D': 12, 'E': 15},
    'C': {'E': 10, 'F': 12},
    'D': {'G': 10},
    'E': {'G': 5, 'H': 8},
    'F': {'H': 5},
    'G': {'I': 15},
    'H': {'I': 10},
    'I': {}
}

start = 'A'
distances = dijkstra(graph, start)

print("Rute Terpendek dan Total Jarak:")
for node, distance in distances.items():
    if distance != float('inf'):
        print(f"{start} -> {node}: {distance}")
```

Dengan Hasil

```
Rute Terpendek dan Total Jarak:
A -> A: 0
A -> B: 10
A -> C: 15
A -> D: 22
A -> E: 25
A -> F: 27
A -> G: 30
A -> H: 32
A -> I: 42
```

Output:

```
Rute
```

4. Jelaskan bagaimana algoritma Ford-Fulkerson bekerja dalam mencari aliran maksimum pada grafik terarah dengan kapasitas edge yang ditentukan.
   - Algoritma Ford-Fulkerson adalah salah satu algoritma yang digunakan untuk mencari aliran maksimum dalam grafik terarah dengan kapasitas edge yang ditentukan. Algoritma ini bekerja dengan mengulangi proses peningkatan aliran melalui jalur-jalur yang memungkinkan di dalam grafik hingga mencapai aliran maksimum yang dapat mengalir dari sumber ke tujuan.
   - Algoritma Ford-Fulkerson berulang kali mencari jalur peningkatan dan memperbarui aliran hingga tidak ada jalur peningkatan yang tersedia lagi. Algoritma ini berhenti saat mencapai kondisi di mana tidak ada lagi jalur peningkatan dari sumber ke tujuan.
   - Hasil akhir dari algoritma Ford-Fulkerson adalah aliran maksimum yang dapat mengalir dari sumber ke tujuan melalui grafik dengan memperhatikan kapasitas edge yang ditentukan. Algoritma ini dapat digunakan dalam berbagai aplikasi, seperti perencanaan jaringan, perutean paket dalam jaringan komunikasi, pengalokasian sumber daya, dan perencanaan produksi.
5. Sebuah perusahaan ingin membangun jaringan pipa minyak untuk menghubungkan beberapa lokasi di Indonesia. Jelaskan bagaimana algoritma jaringan dapat digunakan untuk menentukan rute pipa terpendek yang meminimalkan biaya pembangunan infrastruktur. Berikan contoh penerapan algoritma ini dalam bahasa pemrograman Python.
   - Algoritma jaringan dapat digunakan untuk menentukan rute pipa terpendek yang meminimalkan biaya pembangunan infrastruktur dengan memodelkan jaringan sebagai grafik dan menerapkan algoritma jaringan minimum seperti Algoritma Dijkstra atau Algoritma Bellman-Ford.

     Berikut ini adalah contoh penerapan algoritma jaringan menggunakan Algoritma Dijkstra dalam bahasa pemrograman Python untuk menentukan rute pipa terpendek yang meminimalkan biaya pembangunan infrastruktur:

python Copy code

```python
import heapq

def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    queue = [(0, start)]
    while queue:
        current_distance, current_node = heapq.heappop(queue)
        if current_distance > distances[current_node]:
            continue
        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(queue, (distance, neighbor))
    return distances
```

```python
# Contoh penggunaan algoritma jaringan untuk menentukan rute pipa terpendek
graph = {
    'Sumber Minyak': {'Lokasi A': 10, 'Lokasi B': 15, 'Lokasi C': 20},
    'Lokasi A': {'Lokasi D': 12, 'Lokasi E': 8},
    '
```