Nama : febi rahmadia putri

Nim : 20220021

Algoritma jaringan

Latihan 1

a. Berikut adalah contoh kasus penggunaan algoritma aliran maksimum pada jaringan dengan bobot yang tidak terbatas (infinite capacity). Kasus ini dapat diaplikasikan pada jaringan dengan banyak jalur yang dapat dilalui antara simpul sumber dan simpul tujuan.
Berikut adalah contoh kode Python untuk menyelesaikan kasus tersebut menggunakan algoritma Ford-Fulkerson:

```python
def dfs(graph, u, visited, parent, sink):
    # Implementasi algoritma Depth-First Search
    visited[u] = True
    for v, capacity in enumerate(graph[u]):
        if not visited[v] and capacity > 0:
            parent[v] = u
            if v == sink:
                return True
            if dfs(graph, v, visited, parent, sink):
                return True
    return False


def ford_fulkerson(graph, source, sink):
    # Inisialisasi variabel
    max_flow = 0
    parent = [-1] * len(graph)

    # Jalankan algoritma Ford-Fulkerson
    while dfs(graph, source, [False] * len(graph), parent, sink):
        path_flow = float('inf')
        s = sink
        while s != source:
            path_flow = min(path_flow, graph[parent[s]][s])
            s = parent[s]

        max_flow += path_flow
        v = sink
```

```
        v = sink
        while v != source:
            u = parent[v]
            graph[u][v] -= path_flow
            graph[v][u] += path_flow
            v = parent[v]

    return max_flow


# Contoh penggunaan algoritma Ford-Fulkerson
graph = [
    [0, 10, 10, 0, 0, 0, 0, 0],
    [0, 0, 2, 4, 8, 0, 0, 0],
    [0, 0, 0, 0, 9, 0, 0, 0],
    [0, 0, 0, 0, 0, 10, 0, 0],
    [0, 0, 0, 0, 0, 0, 10, 0],
    [0, 0, 0, 0, 0, 0, 10, 0],
    [0, 0, 0, 0, 0, 0, 0, 10],
    [0, 0, 0, 0, 0, 0, 0, 0],
]
source = 0
sink = 7
max_flow = ford_fulkerson(graph, source, sink)
print(f'Aliran maksimum adalah {max_flow}')
```

Dengan Hasil

```
Aliran maksimum adalah 10
```

b. Jaringan telekomunikasi: Dalam jaringan telekomunikasi, algoritma aliran maksimum dapat digunakan untuk menentukan aliran data maksimum yang dapat dikirimkan melalui kabel atau jalur nirkabel. Hal ini dapat membantu perusahaan telekomunikasi untuk mengoptimalkan penggunaan infrastrukturnya.

```
class Edge:
    def __init__(self, v, capacity, reverse_edge):
        self.v = v
        self.capacity = capacity
        self.flow = 0
        self.reverse_edge = reverse_edge


def add_edge(graph, u, v, capacity):
    forward_edge = Edge(v, capacity, None)
    reverse_edge = Edge(u, 0, forward_edge)
    forward_edge.reverse_edge = reverse_edge

    graph[u].append(forward_edge)
    graph[v].append(reverse_edge)


def dfs(graph, u, sink, visited, path_flow):
    visited[u] = True

    if u == sink:
        return True

    for edge in graph[u]:
        v = edge.v
        residual_capacity = edge.capacity - edge.flow

        if not visited[v] and residual_capacity > 0:
            min_flow = min(path_flow, residual_capacity)
            if dfs(graph, v, sink, visited, min_flow):
                edge.flow += min_flow
                edge.reverse_edge.flow -= min_flow
                return True

    return False


def ford_fulkerson(graph, source, sink):
    max_flow = 0

    while True:
        visited = [False] * len(graph)
        path_flow = float('inf')
        if not dfs(graph, source, sink, visited, path_flow):
            break

        max_flow += path_flow

    return max_flow


# Contoh penggunaan algoritma Ford-Fulkerson pada jaringan telekomunikasi
graph = [[] for _ in range(6)]  # Inisialisasi graf dengan 6 node
```

```
# Menambahkan edge dan kapasitas antar node
add_edge(graph, 0, 1, 10)
add_edge(graph, 0, 2, 10)
add_edge(graph, 1, 2, 2)
add_edge(graph, 1, 3, 4)
add_edge(graph, 1, 4, 8)
add_edge(graph, 2, 4, 9)
add_edge(graph, 3, 5, 10)
add_edge(graph, 4, 5, 10)

source = 0
sink = 5

max_flow = ford_fulkerson(graph, source, sink)
print(f"Aliran maksimum dalam jaringan telekomunikasi adalah {max_flow}")
```

Dengan Hasil

```
Aliran maksimum dalam jaringan telekomunikasi adalah inf
```

c. Algoritma jaringan minimum

fungsi dijkstra(graph, start) untuk mencari jarak terpendek dari start ke setiap simpul dalam graph. graph adalah representasi graf menggunakan dictionary, di mana setiap kunci adalah simpul dan nilai adalah dictionary yang berisi tetangga simpul beserta bobotnya.

Contoh kode Python untuk algoritma Dijkstra:

```
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
```

```
    return distances

# Contoh penggunaan algoritma Dijkstra
graph = {
    'A': {'B': 5, 'C': 2},
    'B': {'D': 4, 'E': 2},
    'C': {'B': 8, 'E': 7},
    'D': {'F': 2},
    'E': {'D': 6, 'F': 1},
    'F': {}
}

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

for node, distance in distances.items():
    print(f"Jarak terpendek dari {start_node} ke {node} adalah {distance}")
```

Dengan Hasil

```
Jarak terpendek dari A ke A adalah 0
Jarak terpendek dari A ke B adalah 5
Jarak terpendek dari A ke C adalah 2
Jarak terpendek dari A ke D adalah 9
Jarak terpendek dari A ke E adalah 7
Jarak terpendek dari A ke F adalah 8
```