

Laporan praktikum daa

Nama : Yohanes Yeningga

Nim : 20220047

Matkul : prak.daa

1. Berikut adalah contoh kode Python untuk menyelesaikan masalah aliran maksimum dengan algoritma Ford-Fulkerson:

Kode yang Anda berikan adalah implementasi dari algoritma Breadth-First Search (BFS). Itu melakukan pencarian pada grafik mulai dari simpul awal yang diberikan dan mencoba mencapai simpul tujuan. Itu juga melacak node induk dari setiap node yang dikunjungi.

Berikut adalah kode Python lengkap untuk program yang mengimplementasikan Breadth-First Search (BFS) seperti yang dijelaskan sebelumnya:

```
def bfs(graph, start, goal, parent):
    visited = [False] * len(graph)
    queue = []
    queue.append(start)
    visited[start] = True

    while queue:
        node = queue.pop(0)
        for index, val in enumerate(graph[node]):
            if not visited[index] and val > 0:
                queue.append(index)
                visited[index] = True
                parent[index] = node
                if index == goal:
                    return True

    return False

# Contoh penggunaan
adjacency_matrix = [
    [0, 1, 1, 0, 0],
    [1, 0, 0, 1, 0],
    [1, 0, 0, 1, 0],
    [1, 0, 0, 1, 1],
    [1, 0, 0, 1, 0]
]

start_node = 0
goal_node = 4
parent = [None] * len(adjacency_matrix)

if bfs(adjacency_matrix, start_node, goal_node, parent):
    # Path dari start_node ke goal_node ditemukan
    path = []
    current_node = goal_node
    while current_node is not None:
        path.append(current_node)
        current_node = parent[current_node]
    path.reverse()
    print("Path:", path)
else:
    print("Tidak ada path dari start_node ke goal_node")

Path: [0, 2, 4]
```

Dalam contoh di atas, kami menggunakan matriks ketetanggaan (adjacency_matrix) untuk merepresentasikan grafik. Anda dapat mengganti adjacency_matrix dengan representasi grafik yang sesuai, seperti daftar ketetanggaan.

Kode tersebut akan mencetak jalur (path) dari start_node ke goal_node jika jalur tersebut ada, atau mencetak pesan bahwa tidak ada jalur yang tersedia.

2. Berikut adalah kode Python lengkap yang menggabungkan fungsi ford_fulkerson dengan implementasi fungsi bfs untuk mencari jalur penambahan dalam algoritma Ford-Fulkerson:

```
from collections import deque

def bfs(graph, source, sink, parent):
    visited = [False] * len(graph)
    queue = deque()
    queue.append(source)
    visited[source] = True

    while queue:
        node = queue.popleft()
        for index, val in enumerate(graph[node]):
            if not visited[index] and val > 0:
                queue.append(index)
                visited[index] = True
                parent[index] = node
                if index == sink:
                    return True

    return False

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

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

```
0 d ▶ return max_flow

# Contoh penggunaan
adjacency_matrix = [
    [0, 16, 13, 0, 0, 0],
    [0, 0, 10, 12, 0, 0],
    [0, 4, 0, 0, 14, 0],
    [0, 0, 9, 0, 0, 20],
    [0, 0, 0, 7, 0, 4],
    [0, 0, 0, 0, 0, 0]
]

source_node = 0
sink_node = 5

max_flow = ford_fulkerson(adjacency_matrix, source_node, sink_node)
print("Maximum Flow:", max_flow)
```

Maximum Flow: 23

Dalam contoh di atas, kami menggunakan matriks ketetanggaan (`adjacency_matrix`) untuk merepresentasikan jaringan aliran. Anda dapat mengganti `adjacency_matrix` dengan representasi jaringan aliran yang sesuai.


Kode tersebut akan mencetak `max_flow`, yang merupakan aliran maksimum dari `source_node` ke `sink_node` dalam jaringan aliran yang diberikan.

3. Algoritma Ford-Fulkerson adalah algoritma yang digunakan untuk mencari aliran maksimum dalam jaringan aliran. Berikut ini adalah contoh penggunaan algoritma Ford-Fulkerson dengan grafik yang diberikan:

```
0 d ▶ def bfs(graph, s, t, parent):
    visited = [False] * len(graph)
    queue = []
    queue.append(s)
    visited[s] = True

    while queue:
        u = queue.pop(0)
        for v in range(len(graph)):
            if visited[v] is False and graph[u][v] > 0:
                queue.append(v)
                visited[v] = True
                parent[v] = u
                if v == t:
                    return True

    return False
```

```
0 d 
def ford_fulkerson(graph, source, sink):
    parent = [-1] * len(graph)
    max_flow = 0

    while bfs(graph, source, sink, parent):
        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
        while v != source:
            u = parent[v]
            graph[u][v] -= path_flow
            graph[v][u] += path_flow
            v = parent[v]

    return max_flow

graph = [[0, 16, 13, 0, 0, 0],
         [0, 0, 10, 12, 0, 0],
         [0, 4, 0, 0, 14, 0],
         [0, 0, 9, 0, 0, 20],
         [0, 0, 0, 7, 0, 4],
         [0, 0, 0, 0, 0, 0]]

source = 0
sink = 5

max_flow = ford_fulkerson(graph, source, sink)
print("Aliran maksimum adalah: %d" % max_flow)

Aliran maksimum adalah: 23
```

Hasil output dari kode di atas akan menampilkan aliran maksimum yang ditemukan dalam grafik yang diberikan.