

Tugas Eksplorasi Mandiri
The Maximum-Flow Problem
Perancangan dan Analisis Algoritma (INF1.62.4001)

Dosen Pengampu :
Randi Proska Sandra, S.Pd, M.Sc



Disusun Oleh:
Rendi Aigo Brandon
23343082

INFORMATIKA
DEPARTEMEN ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2025

A. RINGKASAN

Masalah aliran maksimum (Maximum-Flow Problem) adalah masalah optimisasi yang bertujuan untuk menentukan jumlah aliran maksimum yang dapat melewati sebuah jaringan dari sumber (source) ke tujuan (sink) tanpa melebihi kapasitas yang ditentukan pada setiap sisi (edge). Masalah ini banyak digunakan dalam berbagai bidang seperti perencanaan jaringan transportasi, pengelolaan sumber daya, dan komunikasi data. Salah satu algoritma yang umum digunakan untuk menyelesaikan masalah ini adalah algoritma Ford-Fulkerson. Algoritma ini bekerja dengan mencari jalur augmentasi, yaitu jalur dari sumber ke tujuan yang masih memiliki kapasitas tersisa untuk dilewati aliran tambahan. Setiap kali ditemukan jalur augmentasi, aliran pada jalur tersebut ditingkatkan hingga mencapai kapasitas maksimum jalur tersebut. Proses ini diulangi hingga tidak ada lagi jalur augmentasi yang dapat ditemukan dalam jaringan residual. Jaringan residual adalah representasi dari jaringan asli yang menunjukkan kapasitas tersisa setelah mempertimbangkan aliran yang sudah ada. Dengan kata lain, jaringan residual membantu dalam menentukan apakah masih ada ruang untuk menambah aliran dari sumber ke tujuan.

Dalam implementasinya, algoritma Ford-Fulkerson dapat menggunakan dua metode pencarian, yaitu pencarian lebar-pertama (BFS) atau pencarian dalam-pertama (DFS). BFS umumnya lebih disukai karena lebih mudah memastikan bahwa jalur augmentasi ditemukan secara optimal. Implementasi Ford-Fulkerson yang menggunakan BFS dikenal sebagai algoritma Edmonds-Karp, yang memiliki kompleksitas waktu $O(VE^2)$, di mana V adalah jumlah simpul dan E adalah jumlah sisi dalam jaringan. Kompleksitas ini menjadikan algoritma Edmonds-Karp lebih stabil dan dapat diprediksi dibandingkan dengan versi Ford-Fulkerson yang menggunakan DFS.

Penting untuk dicatat bahwa jika kapasitas sisi dinyatakan dalam bilangan real, algoritma Ford-Fulkerson mungkin tidak konvergen. Oleh karena itu, untuk memastikan konvergensi, kapasitas sisi biasanya dinyatakan dalam bilangan bulat. Selain itu, ada beberapa kondisi yang dapat mempengaruhi efisiensi algoritma ini, seperti struktur jaringan dan bagaimana aliran diatur dalam setiap iterasi. Dalam beberapa kasus, algoritma ini bisa berjalan dengan lebih cepat jika jalur augmentasi dipilih dengan strategi tertentu untuk menghindari terlalu banyak iterasi yang tidak efisien.

Algoritma Ford-Fulkerson memiliki aplikasi luas dalam berbagai bidang. Dalam optimisasi jaringan, algoritma ini sering digunakan untuk menemukan kapasitas maksimum dari suatu jaringan transportasi, misalnya dalam pengaturan lalu lintas jalan atau jaringan pipa distribusi air. Dalam bidang komunikasi data, algoritma ini dapat membantu dalam

menentukan jalur komunikasi yang paling efisien dalam suatu jaringan komputer. Selain itu, dalam masalah perencanaan sumber daya, algoritma ini dapat digunakan untuk mengalokasikan sumber daya secara optimal di antara berbagai titik permintaan dan suplai. Secara keseluruhan, algoritma Ford-Fulkerson merupakan salah satu solusi yang paling umum dan efektif untuk menyelesaikan masalah aliran maksimum dalam jaringan. Dengan prinsip dasar yang sederhana tetapi efektif, algoritma ini dapat diimplementasikan dalam berbagai skenario dunia nyata untuk meningkatkan efisiensi dalam distribusi sumber daya, transportasi, dan komunikasi.

B. PSEUDOCODE

Fungsi FordFulkerson(graf, sumber, tujuan):

 aliran_maksimum = 0

 graf_residual = salinGraf(graf)

 selama adaJalurAugmentasi(graf_residual, sumber, tujuan, jalur):

 aliran_jalur = kapasitasMinimum(jalur, graf_residual)

 perbaruiKapasitasResidual(jalur, graf_residual, aliran_jalur)

 aliran_maksimum = aliran_maksimum + aliran_jalur

 kembalikan aliran_maksimum

Fungsi adaJalurAugmentasi(graf, sumber, tujuan, jalur):

 inisialisasi struktur data untuk pencarian (misalnya, BFS atau DFS)

 tambahkan sumber ke struktur data pencarian

 tandai sumber sebagai dikunjungi

 selama struktur data pencarian tidak kosong:

 simpul = ambil elemen berikutnya dari struktur data pencarian

 jika simpul adalah tujuan:

 bangun jalur dari sumber ke tujuan

```

        kembalikan benar
    untuk setiap tetangga dari simpul:
        jika tetangga belum dikunjungi dan kapasitas
residual > 0:
            tambahkan tetangga ke struktur data pencarian
            tandai tetangga sebagai dikunjungi
            simpan informasi jalur

    kembalikan salah

```

```

Fungsi kapasitasMinimum(jalur, graf):
    kapasitas_min = tak hingga
    untuk setiap (u, v) dalam jalur:
        jika graf[u][v] < kapasitas_min:
            kapasitas_min = graf[u][v]
    kembalikan kapasitas_min

```

```

Fungsi perbaruiKapasitasResidual(jalur, graf, aliran):
    untuk setiap (u, v) dalam jalur:
        graf[u][v] = graf[u][v] - aliran
        graf[v][u] = graf[v][u] + aliran

```

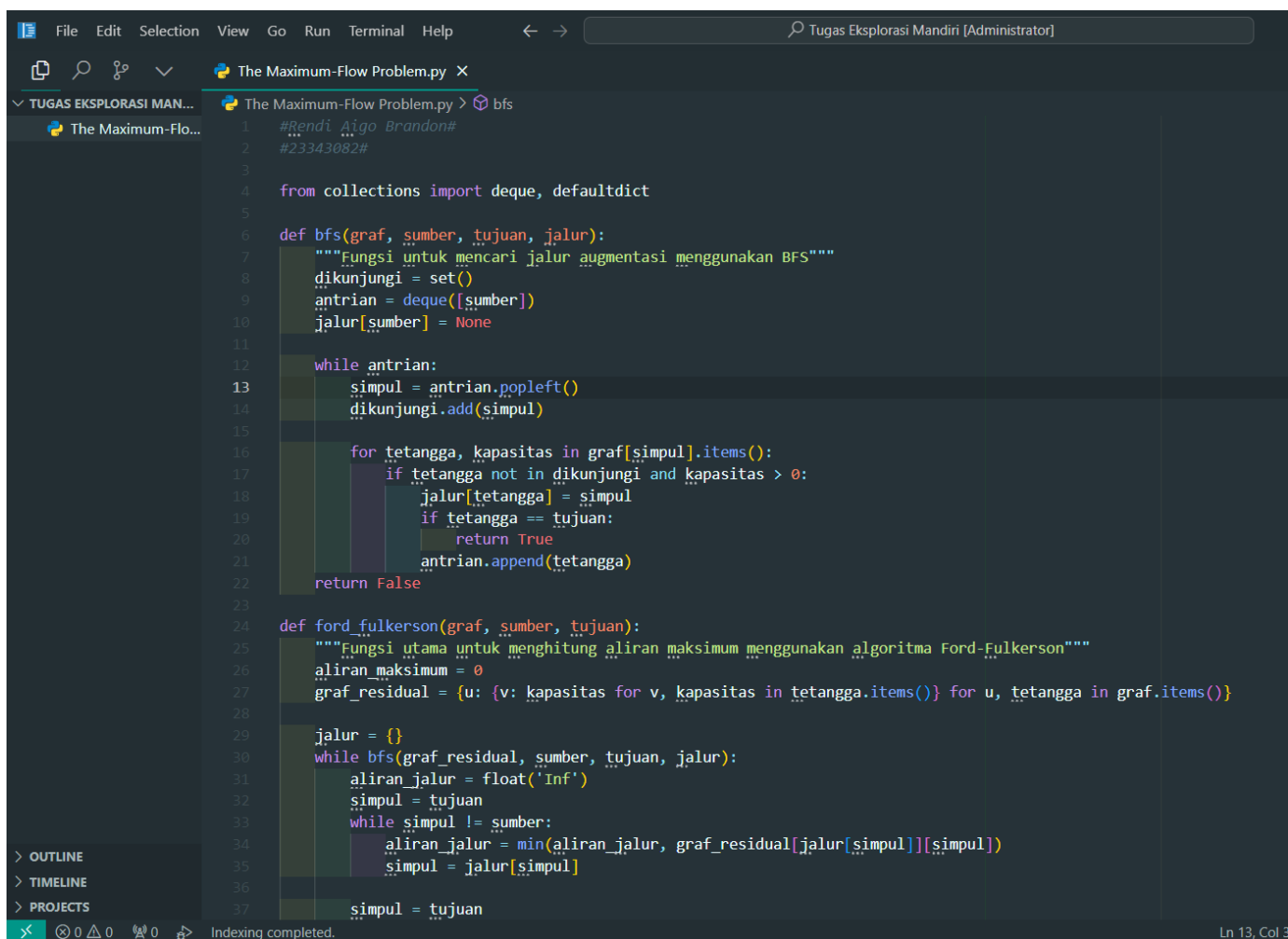
1. **Fungsi FordFulkerson:** Fungsi utama yang menerima parameter graf, sumber, dan tujuan. Fungsi ini menginisialisasi aliran maksimum menjadi 0 dan membuat salinan graf untuk graf residual. Selama masih ada jalur augmentasi yang ditemukan oleh fungsi `adaJalurAugmentasi`, fungsi akan menghitung kapasitas minimum pada jalur tersebut menggunakan fungsi `kapasitasMinimum`, memperbarui kapasitas residual dengan fungsi `perbaruiKapasitasResidual`, dan menambahkan aliran jalur ke aliran maksimum.

2. **Fungsi `adaJalurAugmentasi`:** Fungsi ini mencari jalur dari sumber ke tujuan yang masih memiliki kapasitas tersisa menggunakan metode pencarian seperti BFS atau DFS. Jika jalur tersebut ditemukan, fungsi akan membangun jalur dari sumber ke tujuan dan mengembalikan nilai benar. Jika tidak ada jalur yang ditemukan, fungsi mengembalikan nilai salah.

3. **Fungsi kapasitasMinimum:** Fungsi ini menelusuri jalur yang ditemukan dan menentukan kapasitas minimum (bottleneck) di sepanjang jalur tersebut. Kapasitas minimum ini menentukan berapa banyak aliran tambahan yang dapat ditambahkan melalui jalur tersebut.

4. **Fungsi perbaruiKapasitasResidual:** Fungsi ini memperbarui kapasitas residual di sepanjang jalur augmentasi dengan mengurangi kapasitas sisi sesuai dengan aliran dan menambahkan aliran ke sisi balik (reverse edge) di graf residual. Hal ini memungkinkan algoritma untuk mempertimbangkan aliran balik dalam iterasi berikutnya.

C. PROGRAM



```
File Edit Selection View Go Run Terminal Help
Tugas Eksplorasi Mandiri [Administrator]

The Maximum-Flow Problem.py x
TUGAS EKSPLORASI MAN...
The Maximum-Flo...
1 #Rendi Aigo Brandon#
2 #23343082#
3
4 from collections import deque, defaultdict
5
6 def bfs(graf, sumber, tujuan, jalur):
7     """Fungsi untuk mencari jalur augmentasi menggunakan BFS"""
8     dikunjungi = set()
9     antrian = deque([sumber])
10    jalur[sumber] = None
11
12    while antrian:
13        simpul = antrian.popleft()
14        dikunjungi.add(simpul)
15
16        for tetangga, kapasitas in graf[simpul].items():
17            if tetangga not in dikunjungi and kapasitas > 0:
18                jalur[tetangga] = simpul
19                if tetangga == tujuan:
20                    return True
21                antrian.append(tetangga)
22    return False
23
24 def ford_fulkerson(graf, sumber, tujuan):
25     """Fungsi utama untuk menghitung aliran maksimum menggunakan algoritma Ford-Fulkerson"""
26     aliran_maksimum = 0
27     graf_residual = {u: {v: kapasitas for v, kapasitas in tetangga.items()} for u, tetangga in graf.items()}
28
29     jalur = {}
30     while bfs(graf_residual, sumber, tujuan, jalur):
31         aliran_jalur = float('Inf')
32         simpul = tujuan
33         while simpul != sumber:
34             aliran_jalur = min(aliran_jalur, graf_residual[jalur[simpul]][simpul])
35             simpul = jalur[simpul]
36
37     simpul = tujuan
```

```
File Edit Selection View Go Run Terminal Help
Tugas Eksplorasi Mandiri [Administrator]

The Maximum-Flow Problem.py x
TUGAS EKSPLORASI MAN... The Maximum-Flow Problem.py > bfs
def ford_fulkerson(graf, sumber, tujuan):
    while simpul != sumber:
        u = jalur[simpul]
        graf_residual[u][simpul] -= aliran_jalur
        if simpul not in graf_residual:
            graf_residual[simpul] = {}
        if u not in graf_residual[simpul]:
            graf_residual[simpul][u] = 0
        graf_residual[simpul][u] += aliran_jalur
        simpul = jalur[simpul]
    aliran_maksimum += aliran_jalur
    return aliran_maksimum

def input_graf():
    """Fungsi untuk menerima input graf dari pengguna"""
    graf = defaultdict(dict)
    print("Masukkan sisi-sisi graf dalam format 'simpul1 simpul2 kapasitas'. Ketik 'selesai' untuk mengakhiri input.")
    while True:
        data = input("Masukkan sisi (atau 'selesai' untuk selesai): ")
        if data.lower() == 'selesai':
            break
        try:
            simpul1, simpul2, kapasitas = data.split()
            kapasitas = int(kapasitas)
            graf[simpul1][simpul2] = kapasitas
        except ValueError:
            print("Input tidak valid. Pastikan formatnya 'simpul1 simpul2 kapasitas'.")
    return graf

if __name__ == "__main__":
    print("Algoritma Ford-Fulkerson untuk Mencari Aliran Maksimum")
    graf = input_graf()
    sumber = input("Masukkan simpul sumber: ")
    tujuan = input("Masukkan simpul tujuan: ")
    bfs
```

```
File Edit Selection View Go Run Terminal Help
Tugas Eksplorasi Mandiri [Administrator]

The Maximum-Flow Problem.py x
TUGAS EKSPLORASI MAN... The Maximum-Flow Problem.py > bfs
71 graf = input_graf()
72 sumber = input("Masukkan simpul sumber: ")
73 tujuan = input("Masukkan simpul tujuan: ")
74
75 aliran_maksimum = ford_fulkerson(graf, sumber, tujuan)
76 print(f"Aliran maksimum dari '{sumber}' ke '{tujuan}' adalah {aliran_maksimum}")
77
```

OUTPUT

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 100 Python Debug Console
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Users\muall\Documents\tugas Eksplorasi Mandiri> cmd /c "c:\Users\muall\AppData\Local\Programs\Python\Python313\python.exe c:\Users\muall\cursor\extensions\ms-python
.debugpy-2024.6.0-win32-x64\bundle\libs\debugpy\adapter\..\..\debugpy\launcher 50857 -- "C:\Users\muall\Documents\tugas Eksplorasi Mandiri\The Maximum-Flow Problem.py"

Algoritma Ford-Fulkerson untuk Mencari Aliran Maksimum
Masukkan sisi-sisi graf dalam format 'simpul1 simpul2 kapasitas'. Ketik 'selesai' untuk mengakhiri input.
Masukkan sisi (atau 'selesai' untuk selesai): A B 10
Masukkan sisi (atau 'selesai' untuk selesai): A C 5
Masukkan sisi (atau 'selesai' untuk selesai): B C 15
Masukkan sisi (atau 'selesai' untuk selesai): B D 10
Masukkan sisi (atau 'selesai' untuk selesai): C D 10
Masukkan sisi (atau 'selesai' untuk selesai): selesai
Masukkan simpul sumber: A
Masukkan simpul tujuan: D
Aliran maksimum dari 'A' ke 'D' adalah 15

C:\Users\muall\Documents\tugas Eksplorasi Mandiri>
```

E. ANALISIS KEBUTUHAN WAKTU

1) Analisis Menyeluruh Berdasarkan Operasi/Instruksi yang Dieksekusi

Algoritma Ford-Fulkerson dalam program ini terdiri dari dua bagian utama:

- Mencari jalur augmentasi menggunakan BFS (*bfs()* function)
- Memperbarui kapasitas jalur dalam *ford_fulkerson()* function

Kita menganalisis kebutuhan waktu berdasarkan operator **penugasan** (`=`, `+=`, `-=`, dll.) dan **aritmatika** (`+`, `-`, `*`, `/`, `%`).

a. Operasi pada Fungsi BFS

```
dikunjungi = set() # O(1)
antrian = deque([sumber]) # O(1)
jalur[sumber] = None # O(1)
```

- Penugasan dan inisialisasi set/list: $O(1)$

Loop utama BFS:

```
while antrian: # Looping  $O(V)$ 
    simpul = antrian.popleft() #  $O(1)$ 
    dikunjungi.add(simpul) #  $O(1)$ 
```

- while antrian dijalankan sebanyak V kali (V = jumlah simpul).
- `popleft()` dari deque berjalan dalam $O(1)$.
- `add()` ke dalam set berjalan dalam $O(1)$.

Iterasi melalui tetangga:

```
for tetangga, kapasitas in graf[simpul].items(): #  $O(E)$  total
    if tetangga not in dikunjungi and kapasitas > 0: #  $O(1)$ 
        jalur[tetangga] = simpul #  $O(1)$ 
        antrian.append(tetangga) #  $O(1)$ 
```

- Loop berjalan sebanyak E kali dalam seluruh eksekusi BFS (karena memeriksa semua sisi).
- Operasi `if` dan `append` berjalan dalam $O(1)$.

Kompleksitas total BFS: $O(V + E)$

b. Operasi pada Fungsi Ford-Fulkerson

Loop utama:

```
while bfs(graf_residual, sumber, tujuan, jalur): # O(f)
    iterasi
```

- Ford-Fulkerson dapat mengulang hingga f kali, di mana f adalah aliran maksimum.

Mencari aliran jalur minimum:

```
aliran_jalur = float('Inf') # O(1)
simpul = tujuan
while simpul != sumber: # O(V)
    aliran_jalur = min(aliran_jalur,
        graf_residual[jalur[simpul]][simpul]) # O(1)
    simpul = jalur[simpul] # O(1)
```

- Loop berjalan dalam $O(V)$ karena menelusuri jalur augmentasi.

Memperbarui kapasitas:

```
simpul = tujuan
while simpul != sumber: # O(V)
    graf_residual[u][simpul] -= aliran_jalur # O(1)
    graf_residual[simpul][u] += aliran_jalur # O(1)
    simpul = jalur[simpul] # O(1)
```

- Operasi dalam $O(V)$ untuk setiap iterasi.

Kompleksitas total Ford-Fulkerson (tanpa optimasi) adalah: $O(E * f)$

Jika BFS digunakan (Edmonds-Karp), kompleksitas menjadi: $O(VE^2)$

2) Analisis Berdasarkan Jumlah Operasi Abstrak atau Operasi Khas

- **Operasi BFS** (pencarian jalur augmentasi) $\rightarrow O(V + E)$
- **Operasi pembaruan kapasitas jaringan residual** $\rightarrow O(V)$
- **Jumlah iterasi maksimum** $\rightarrow O(f)$

Total operasi dalam satu iterasi Ford-Fulkerson:

$$O(V+E)+O(V)=O(V+E)$$

Total operasi dalam semua iterasi:

$$O(f) \times O(V+E) = O(E \cdot f)$$

Jika menggunakan BFS (Edmonds-Karp), maka jumlah iterasi maksimal adalah $O(VE)$, sehingga kompleksitas menjadi $O(VE^2)$.

3) Analisis Menggunakan Pendekatan Best-Case, Worst-Case, dan Average-Case

a) Best-Case (Kasus Terbaik)

- Jika setiap jalur augmentasi langsung mencapai **aliran maksimum dalam sedikit iterasi**, maka jumlah iterasi menjadi kecil.
- Kompleksitas terbaik terjadi jika kita hanya butuh satu iterasi BFS.
- **Kompleksitas best-case: $O(V + E)$**

b) Worst-Case (Kasus Terburuk)

- Jika peningkatan aliran maksimum sangat kecil setiap iterasi (misalnya, hanya meningkat 1 unit setiap iterasi).
- Jumlah iterasi bisa mencapai $O(f)$.
- Kompleksitas **tanpa BFS: $O(E * f)$**
- Kompleksitas **dengan BFS (Edmonds-Karp): $O(VE^2)$**

c) Average-Case (Kasus Rata-Rata)

- Dalam kebanyakan aplikasi nyata, performa Ford-Fulkerson mendekati $O(VE^2)$ karena jalur augmentasi tidak selalu optimal tetapi juga tidak selalu buruk.

F, SUMBER REFERENSI

- Anany Levitin, *Introduction to the Design & Analysis of Algorithms (3rd Edition)*
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms (3rd Edition)*
- GeeksForGeeks - Ford-Fulkerson Algorithm for Maximum Flow Problem
<https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>
- Brilliant - Ford-Fulkerson Algorithm
<https://brilliant.org/wiki/ford-fulkerson-algorithm/>
- Stack Overflow - Complexity of Ford-Fulkerson Algorithm
<https://stackoverflow.com/questions/20036483/complexity-of-the-ford-fulkerson-algorithm>
- OpenAI. (2025). *ChatGPT (Model GPT-4)*.
<https://chat.openai.com>