Penggunaan Breadth First Search (BFS) pada Algoritma Edmonds-Karp untuk Penyelesaian Maximum Flow Problem

Jova Andres Riski Sirait - 13520072
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520072@std.stei.itb.ac.id

***Abstract*—*Maximum flow problem* merupakan masalah yang terlihat cukup rumit untuk diselesaikan. Suatu *flow network* dapat direpresentasikan dalam bentuk graf berarah dan berbobot, dengan bobot menunjukkan kapasitas *flow* pada sisi tertentu. Dengan menggunakan algoritma Edmonds-Karp, kita dapat menyelesaikan permasalahan ini dengan cukup mudah. Algoritma ini mengimplementasikan metode yang ditemukan oleh Ford dan Fulkerson, yang menggunakan Breadth First Search pada pencarian setiap jalurnya karena algoritma Breadth First Search akan mencari jalur terpendek terlebih dahulu. *Maximum flow problem* memiliki banyak pemanfaatan contohnya pada pemodelan jalur lalu lintas yang optimal, pemodelan pipa air rumah tangga, aliran arus listrik pada sirkuit elektrik, dan masih banyak lagi.**

Keywords—maximum flow problem, ford-fulkerson, bfs, edmons-karp

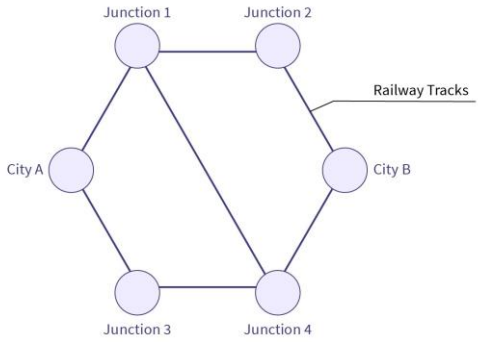
I. PENDAHULUAN

Pencarian suatu jalur pada struktur data pohon atau graf merupakan salah satu pencarian yang cukup banyak diimplementasikan. Beberapa contoh permasalahan yang memanfaatkan algoritma pencarian dalam graf ini adalah pencarian jalur terpendek, web crawler, folder crawler, penyelesaian puzzle, dan masih banyak lagi.

Salah satu permasalahan yang cukup terkenal adalah *Maximum Flow Problem* yang bertujuan untuk memaksimalkan aliran dalam suatu jaringan yang direpresentasikan dalam bentuk graf berarah. Aplikasi atau pemanfaatan dari *maximum flow problem* ini cukup banyak, diantaranya adalah untuk menghitung distribusi air maksimum pada pipa yang menghubungkan saluran air antar rumah, persoalan M *job* dan N *applicants* (*bipartite matching*), pemodelan lalu lintas, aliran listrik pada sirkuit elektrik, dan menghitung permintaan-penawaran pada sebuah sirkulasi pasar.

Permasalah tersebut dapat diselesaikan menggunakan suatu algoritma bernama Edmonds-Karp yang menggunakan metode Ford-Fulkerson. Metode Ford-Fulkerson ditemukan oleh Ford dan Fulkerson pada tahun 1956. Secara sederhana, algoritma ini bekerja dengan cara mencari semua jalur yang masih memiliki kapasitas kosong, kemudian menambahkan sejumlah aliran ke dalamnya sampai aliran-aliran dalam graf tersebut menjadi maksimum.

Pendekatan Edmonds-Karp ini menggunakan algoritma pencarian Breadth First Search (BFS) dalam mencari semua jalur dari simpul awal ke simpul tujuan.



Gambar 1.1 Maximum flow problem pada pemodelan lalu lintas
Sumber: https://www.scaler.com/topics/data-structures/ford-fulkerson-algorithm-for-maximum-flow-problem/

II. TEORI DASAR

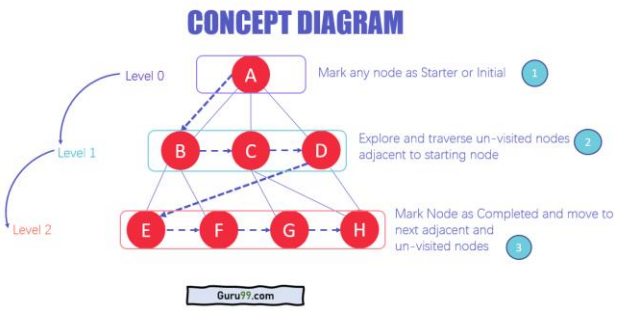
A. *Breadth First Search*

Breadth First Search (BFS) adalah algoritma pencarian node dalam suatu struktur yang berbentuk pohon atau graf. Adapun graf merupakan struktur yang terdiri atas simpul atau vertex, yang dihubungkan oleh sisi atau edge. Sutau graf dapat memiliki arah maupun tidak. Pada suatu graf, simpul 1 dikatakan bertetangga dengan simpul 2 apabila terdapat suatu sisi yang menghubungkan kedua simpul tersebut.

Pencarian BFS dilakukan secara traversal mulai dari simpul awal. Adapun algoritma dari BFS secara umum adalah sebagai berikut:

1. Tandai suatu simpul sebagai simpul awal.
2. Kunjungi semua simpul yang bertetangga dengan simpul awal dan dimasukkan dalam queue.

3. Dari queue tersebut, ambil suatu simpul dengan aturan FIFO, kemudian kunjungi semua simpul yang bertetangga dengan simpul tersebut dan belum pernah dikunjungi sebelumnya. Begitu seterusnya sampai queue kosong.



Gambar 2.1 Penelusuran dengan BFS
Sumber: https://www.guru99.com/breadth-first-search-bfs-graph-example.html

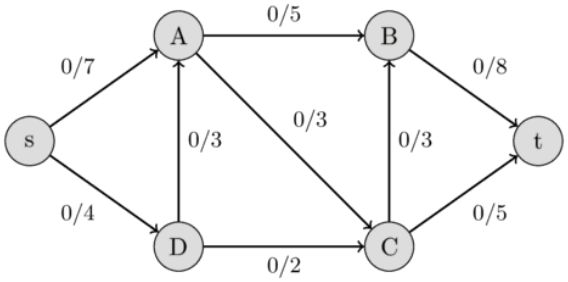
BFS menggunakan struktur data Queue karena sifatnya yang akan mengunjungi tetangga dari simpul-simpul secara berurutan, sedangkan pada metode Depth First Search (DFS) menggunakan struktur data Stack karena menggunakan aturan LIFO (Last In First Out).

B. *Maximum Flow Problem*

Maximum flow problem merupakan suatu permasalahan yang bertujuan untuk menemukan nilai aliran dari suatu jaringan (*network*) sehingga aliran dari sumber ke tujuan bernilai maksimum.

Jaringan (*network*) yang dimaksud di sini adalah suatu graf berarah dengan simpul V dan sisi E. Pada setiap sisi yang ada, terdapat *flow*, yaitu jumlah aliran pada sisi tersebut dan *capacity*, yaitu kapasitas aliran maksimum yang mungkin melewati sisi tersebut. Nilai *flow* selalu lebih kecil atau sama dengan *capacity*.

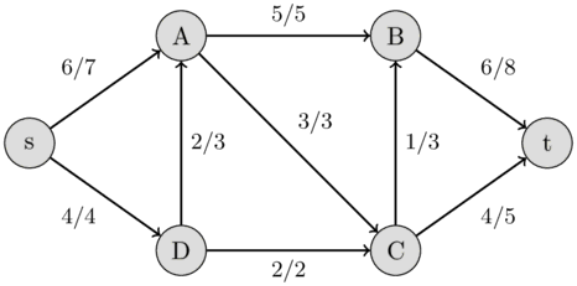
Berikut adalah contoh dari sebuah graf yang merepresentasikan *network flow*.



Gambar 2.2 *Flow Network*
Sumber: https://cp-algorithms.com/graph/edmonds_karp.html#flow-network

Pada gambar di atas, terdapat dua angka pada setiap sisi graf, angka yang pertama menunjukkan jumlah aliran (*flow)* yang diinisiasi dengan 0, dan angka yang kedua menunjukkan kapasitas (*capacity)* dari setiap sisi yang ada. Tujuan yang ingin dicapai pada *maximum flow problem* adalah untuk mencari jumlah aliran pada setiap sisi agar jumlah aliran dari simpul s (*source)* ke simpul t (*sink)* maksimum.

Berikut adalah penyelesaian dari *maximum flow* dari gambar di atas:



Gambar 2.3 *Maximum Flow in Flow Network*
Sumber: https://cp-algorithms.com/graph/edmonds_karp.html#flow-network

Jumlah aliran maksimum pada graf di atas adalah 10, dihitung dari jumlah semua aliran yang keluar dari simpul s, atau bisa juga dengan menghitung jumlah semua aliran yang masuk ke simpul t.

C. *Ford-Fulkerson Method*

Ford-Fulkerson merupakan suatu metode yang digunakan untuk menyelesaikan *maximum flow problem.* Input dari metode ini adalah sebuah graf berarah, simpul asal dan simpul tujuan. Pada metode ini, diperkenalkan sebuah konsep yaitu *residual capacity. Residual capacity* merupakan hasil pengurangan dari *capacity* dengan *flow* atau bisa juga dianggap sebagai jumlah *flow* yang tersedia dari sebuah sisi.

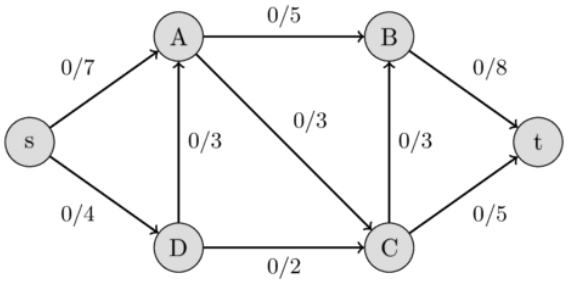
Secara umum, langkah-langkah dari *ford-fulkerson* untuk mencari *maximum flow* adalah sebagai berikut:

1. Atur nilai aliran dari setiap sisi menjadi 0.
2. Cari semua *augmenting path* dari simpul asal (*source)* ke simpul tujuan (*sink). Augmenting path* merupakan jalur pada graf dimana semua sisi sepanjang jalur tersebut memiliki *residual capacity* lebih besar daripada 0.
3. Untuk setiap jalur yang ditemukan, tambahkan *flow* untuk setiap sisi sepanjang jalur tersebut. Jumlah *flow* yang ditambahkan untuk setiap sisi tersebut adalah nilai minimum dari semua *residual capacity* sisi sepanjang jalur tersebut.

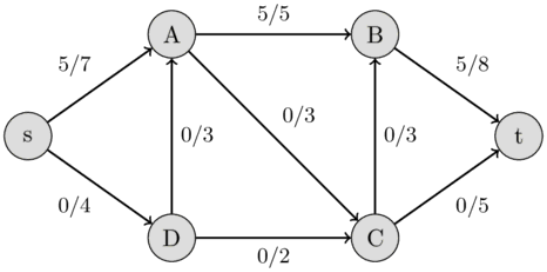
Sebagai tambahan, kita juga bisa menggunakan *reverse-path* atau suatu sisi yang arahnya berlawanan dengan arah yang sebenarnya pada graf. Hal ini memungkinkan untuk melakukan pengulangan pada suatu sisi apabila hasil yang diberikannya tidak optimal atau tidak mencapai nilai maksimum seperti yang diharapkan.

Berikut adalah contoh penyelesaian *maximum flow problem* dengan metode *Ford-Fulkerson*:

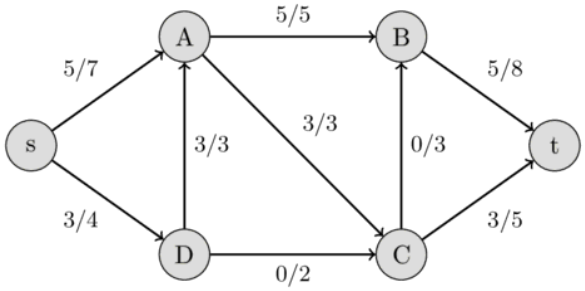
Kita akan menggunakan graf yang telah dipakai sebelumnya,



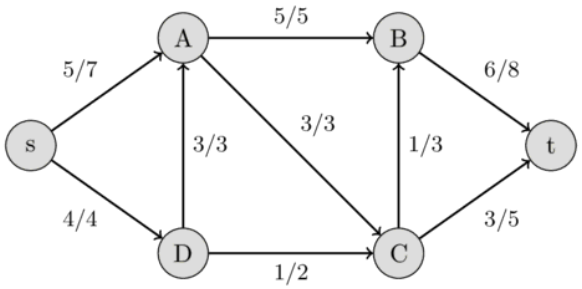
Pada graf di atas, kita cari jalur yang mungkin dari s ke t. Salah satu jalur yang didapatkan adalah s-A-B-t, dengan *residual capacity* yaitu 7, 5, dan 8. Karena kita memilih yang paling minimum, maka kita akan menambahkan 5 *flow* ke masing-masing sisi pada jalur yang sedang diproses.



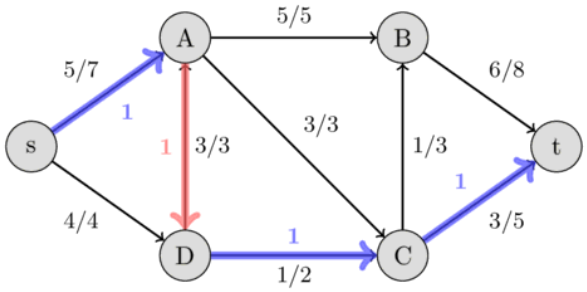
Kita kembali mencari jalur yang memungkinkan, contohnya adalah s-D-A-C-T dengan *residual capacity* 4, 3, 3, dan 5 sehingga kita akan menambahkan 3 *flow* pada masing-masing sisi. Sampai saat ini, nilai *flow* pada jaringan tersebut sudah 8.



Selanjutnya adalah jalur s-D-C-B-T dengan *residual capacity* 1, 2, 3, 3 sehingga kita akan menambahkan 1 *flow* pada setiap sisi.



Selanjutnya adalah jalur s-A-D-C-B-T dengan *residual capacity* 2, 3, 1, 2, 2. Pada jalur ini, kita menemukan sisi A-D yang berlawanan arah dengan yang sebenarnya. Akan tetapi, seperti yang telah disebutkan di atas, kita dapat memilih sisi yang berlawanan pada jalur tersebut dengan syarat bahwa pada sisi tersebut telah terdapat sejumlah *flow* yang melewatinya. Untuk sisi yang berlawanan, *capacity* dianggap 0 dan jumlah *flow* yang melewatinya adalah nilai negatif dari *flow* pada arah yang sebenarnya, sehingga pada kasus ini, *residual capacity* dari sisi A-D adalah 0-(-3) = 3. Lalu kita menambahkan 1 pada jalur tersebut sebagai nilai yang minimal.



Graf akhirnya akan terlihat seperti pada Gambar 2.2. Dengan *maximum flow* adalah 10. Kompleksitas dari Ford-Fulkerson adalah O(|E|*f), dengan f adalah jumlah *flow* maksimum dari graf tertentu.

D. Edmonds-Karp Algorithm

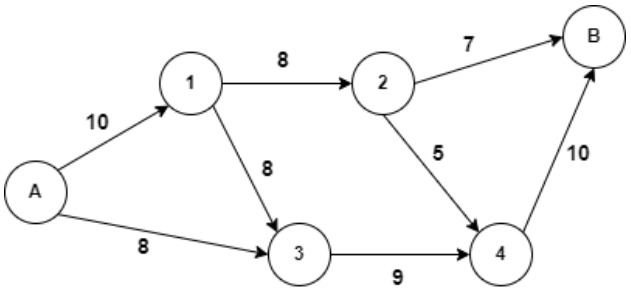
Edmons-Karp algorithm adalah algoritma yang merupakan implementasi dari *Ford-Fulkerson Method*. Algoritma ini menggunakan pendekatan Breadth First Search (BFS) pada pencarian *augmented path*-nya.

Algoritma ini dipublikasikan oleh Jack Edmonds dan Richard Karp pada tahun 1972 pada sebuah *paper* yang berjudul "*Theoretical improvements in algorithmic efficiency for network flow problems"*. Kompleksitas waktu dari algoritma ini adalah $O(VE^2)$.

III. STUDI KASUS: *MAXIMUM FLOW PROBLEM* PADA PEMODELAN LALU LINTAS

Berikut adalah contoh pemodelan lalu lintas dari kota A ke kota B yang bisa melewati beberapa jalur. Jalur-jalur tersebut merupakan sisi dengan angka yang merepresentasikan jumlah

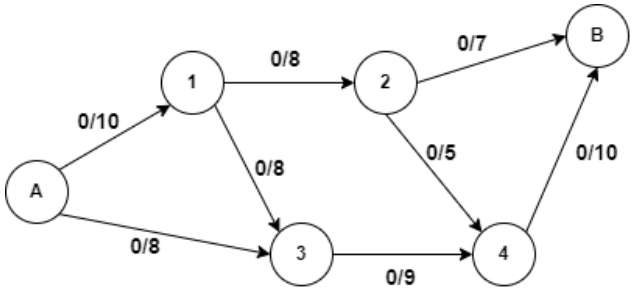
mobil maksimum yang dapat melewati jalur tersebut pada suatu waktu.



Gambar 3.1 Pemodelan lalu lintas dalam graf
Sumber: Dokumentasi Pribadi

Di sini kita akan berusaha memaksimalkan jumlah mobil yang lewat dari kota A ke kota B dengan memanfaatkan Edmonds-Karp Algorithm, tanpa menggunakan program.

Pertama, kita akan menginisiasi *flow* pada setiap sisi menjadi 0.

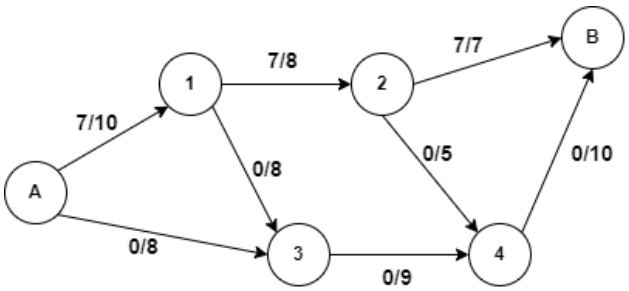


Pada graf di atas, kita akan mencari semua jalur yang mungkin dari kota A ke kota B dengan menggunakan algoritma BFS.

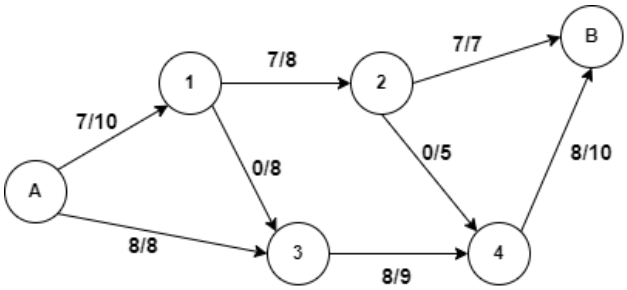
Iterasi	Jalur yang ditemukan
1	A-1 A-3
2	A-1-2 A-1-3 A-3-4
3	**A-1-2-B** A-1-2-4 A-1-3-4 **A-3-4-B**
4	**A-1-2-4-B** **A-1-3-4-B**

Dari pencarian BFS tersebut, kita mendapatkan setidaknya 4 jalur yang akan diperiksa di awal.

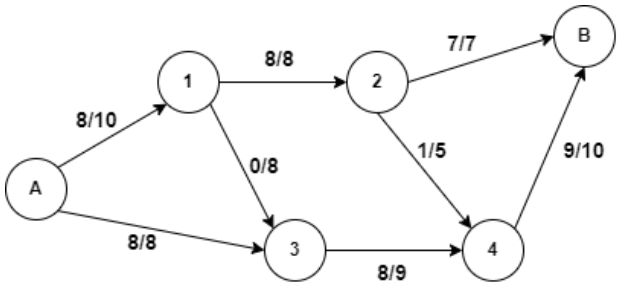
Yang pertama adalah jalur A-1-2-B, dengan *residual capacity*-nya secara berurutan adalah 10, 7, 8. Dengan mengambil nilai minimum (*bottleneck*), kita akan menambahkan 7 *flow* pada setiap sisi jalur tersebut.



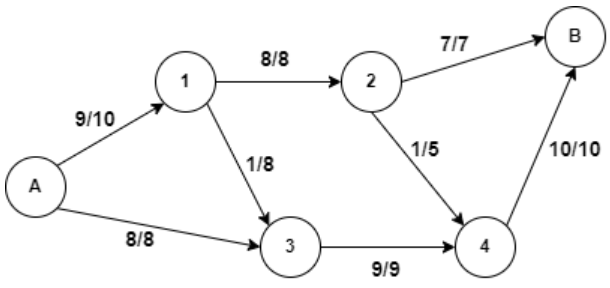
Jalur yang kedua adalah A-3-4-B, dengan *residual capacity* 8, 9, dan 10. *Bottleneck* adalah 8 sehingga kita menambahkan 8 *flow* pada setiap sisi pada jalur tersebut.



Jalur yang ketiga adalah A-1-2-4-B, dengan *resdual capacity* 3, 1, 5, 2. *Bottleneck* adalah 1 sehingga kita menambahkan 1 *flow* pada setiap sisi pada jalur tersebut.



Jalur terakhir yang kita dapatkan pada pencarian BFS adalah A-1-3-4-B dengan *residual capacity* 2, 8, 1, 1. Dengan *bottleneck* 1 kita menambahkan 1 *flow* pada setiap sisi pada jalur tersebut.



Pada graf tersebut, jumlah aliran (*flow*) telah maksimal dengan jumlah 17. Model lalu lintas seperti ini dapat dikatakan kurang baik karena terdapat beberapa sisi yang masih memiliki sisa *residual capacity* yang cukup banyak, tetapi tidak optimal untuk dilewati yaitu sisi 1-3 dan sisi 2-4.

IV. Menyelesaikan Berbagai Maximum Flow Problem Dengan Program

Berikut adalah hasil implementasi algoritma Edmonds-Karp dalam bahasa pemrograman python:

printPath

```
def printPath(path):
    for i in range(len(path)):
        print(path[i][0], end="->")
    print(path[-1][1])
```

BFS

```
def bfs(C, F, s, t):
    queue = [s]
    paths = {s: []}
    if s == t:
        printPath(paths[s])
        return paths[s]
    while queue:
        u = queue.pop(0)
        for v in range(len(C)):
            if(C[u][v]-F[u][v]  >  0)  and  v  not  in
paths:
                paths[v] = paths[u]+[(u, v)]
                if v == t:
                    printPath(paths[v])
                    return paths[v]
                queue.append(v)
    return None
```

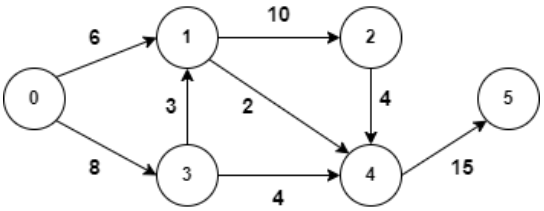
Edmons-Karp

```
def max_flow(C, s, t):
    n = len(C)  # C is the capacity matrix
    F = [[0] * n for i in range(n)]
    print("\nAvailable route (BFS)")
    path = bfs(C, F, s, t)
    while path != None:
        flow  =  min(C[u][v]  -  F[u][v]  for  u,  v  in
path)
        for u, v in path:
            F[u][v] += flow
            F[v][u] -= flow
        path = bfs(C, F, s, t)
    return sum(F[s][i] for i in range(n))
```

Berikut adalah hasil pengujian program *maximum flow solver* yang telah dibuat pada beberapa kasus:

1. Kasus 1

   Representasi graf:



   Representasi dalam matriks (input):

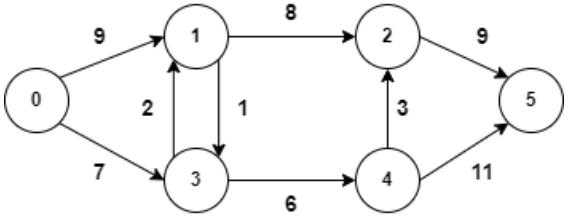
0	6	0	8	0	0
0	0	10	0	2	0
0	0	0	0	4	0
0	3	0	0	4	0
0	0	0	0	0	15
0	0	0	0	0	0

   Output program:

```
Available route (BFS)
0->1->4->5
0->3->4->5
0->1->2->4->5
Maximum flow: 10
```

2. Kasus 2

   Representasi graf:



   Representasi dalam matriks (input):

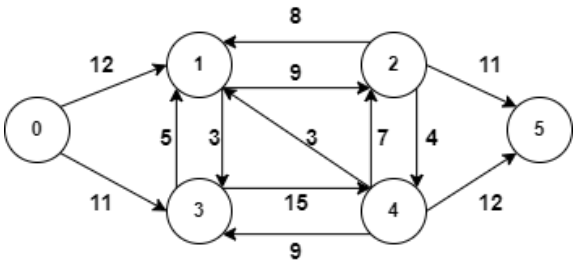
0	9	0	7	0	0
0	0	8	1	0	0
0	0	0	0	0	9
0	2	0	0	6	0
0	0	3	0	0	11
0	0	0	0	0	0

   Output program:

```
Available route (BFS)
0->1->2->5
0->3->4->5
Maximum flow: 14
```

3. Kasus 3

   Representasi graf:



   Representasi dalam matriks (input):

0	12	0	11	0	0
0	0	9	3	0	0
0	8	0	0	4	11
0	5	0	0	15	0
0	3	7	9	0	12
0	0	0	0	0	0

   Output program:

```
Available route (BFS)
0->1->2->5
0->3->4->5
0->1->3->4->5
0->1->3->4->2->5
Maximum flow: 23
```

V. KESIMPULAN

Algoritma pencarian seperti Breadth First Search memiliki manfaat yang cukup banyak, terutama dalam menyelesaikan permasalahan sehari-hari, bahkan permasalahan yang cukup rumit.

Pada makalah ini, kita telah membahas penggunaan algoritma BFS dalam penyelesaian masalah *maximum flow* yang memiliki penerapan yang cukup berguna. *Maximum flow problem* dapat diselesaikan dengan algoritma Edmonds-Karp yang menggunakan metode Ford-Fulkerson pada implementasinya. Algoritma ini juga memanfaatkan pencarian BFS untuk mencari semua jalur yang dapat dilalui dari simpul awal ke simpul tujuan. Untuk lebih memudahkan pekerjaan, kita dapat mengimplementasikan metode tersebut menjadi sebuah program sederhana. Dengan merepresentasikan permasalahan menjadi sebuah graf, program dapat mengeluarkan output jumlah *maximum flow* dari graf yang kita berikan.

Kita juga bisa menggunakan beberapa algoritma alternatif untuk penyelsaian *maximum flow problem* ini, diantaranya adalah *Dinic's Algorithm* dan *Push-Relabel Algorithm.*

VI. UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga ingin mengucapkan terimakasih yang sebesar-besarnya kepada Pak Dr. Ir. Rinaldi Munir yang telah memberikan banyak ilmu bermanfaat selama satu semester yang dapat membantu penulis dalam menyusun makalah ini. Terakhir, tetapi tidak kalah penting, penulis ucapkan terima kasih kepada orangtua dan teman-teman saya atas cinta, pengertian, dan dukungan selama ini.

VIDEO LINK AT YOUTUBE

https://youtu.be/wiMBM8BrYtg

REFERENCES

[1] https://www.guru99.com/breadth-first-search-bfs-graph-example.html
[2] https://cp-algorithms.com/graph/edmonds_karp.html
[3] https://jamieheller.github.io/theory.html
[4] https://www.programiz.com/dsa/ford-fulkerson-algorithm#:~:text=Ford%2DFulkerson%20algorithm%20is%20a,amount%20of%20stuff%20through%20it.
[5] https://brilliant.org/wiki/ford-fulkerson-algorithm/
[6] https://www.scaler.com/topics/data-structures/ford-fulkerson-algorithm-for-maximum-flow-problem/
[7] https://github.com/anxiaonong/Maxflow-Algorithms
[8] https://algorithms.tutorialhorizon.com/max-flow-problem-ford-fulkerson-algorithm/
[9] https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Mei 2022

Jova Andres Riski Sirait 13520072