TUGAS PROJECT ANALISIS ALGORITMA

"PENERAPAN ALGORITMA BREADTH FIRST SEARCH"

Dosen Pengampu :

Muhaimin Hasanudin, St., M. Kom

Disusun Oleh :

1.	Alya Farhania	202331005
2.	Rafifah Nur Rifiyati	202331003
3.	Giska Rahayu	202331059

PROGRAM STUDI S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI PLN JAKARTA
2025

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat, taufik, dan hidayah-Nya kepada kami semua. Berkat pertolongan-Nya, kami dapat menyusun dan menyelesaikan makalah yang berjudul *"Penerapan Algoritma Breadth First Search"* dengan baik dan tepat waktu. Makalah ini disusun sebagai bagian dari Tugas Project pada mata kuliah **Analisis Algoritma** yang dibimbing oleh Bapak **Muhaimin Hasanudin, S.T., M.Kom**.

Kami menyadari bahwa makalah ini masih jauh dari sempurna, baik dari segi isi maupun penyajian. Oleh karena itu, dengan segala kerendahan hati kami membuka diri terhadap kritik dan saran yang membangun guna perbaikan di masa mendatang. Kami percaya bahwa kritik yang membangun dapat mendorong kami untuk terus belajar dan berkembang.

Ucapan terima kasih kami sampaikan kepada Bapak Muhaimin Hasanudin, S.T., M.Kom atas ilmu, bimbingan, dan motivasi yang telah diberikan selama proses perkuliahan dan penyusunan proyek UAS ini. Kami juga mengucapkan terima kasih kepada semua pihak yang telah membantu dan memberikan dukungan, baik secara langsung maupun tidak langsung. Bantuan dan dukungan yang diberikan sangat berarti bagi kami.

Semoga makalah ini dapat memberikan manfaat dan menjadi referensi awal bagi pembaca dalam memahami penerapan algoritma *Breadth First Search* dalam penyelesaian permasalahan pencarian pada graf. Akhir kata, semoga makalah ini dapat menjadi motivasi bagi kita semua untuk terus belajar dan mengikuti perkembangan ilmu pengetahuan dan teknologi informasi di masa depan.

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam dunia informatika dan pemrograman, algoritma merupakan inti dari solusi pemecahan masalah. Sebuah algoritma dirancang untuk menyelesaikan permasalahan secara efisien dan optimal, baik dari sisi waktu maupun sumber daya yang digunakan. Salah satu kelas permasalahan yang sering ditemui adalah pencarian jalur (*pathfinding*) pada suatu struktur data yang berbentuk graf, seperti pada jaringan komputer, peta jalan, atau relasi antar objek.

Salah satu algoritma pencarian jalur yang paling populer dan fundamental adalah *Breadth First Search* (BFS)**.** BFS merupakan algoritma penelusuran yang bekerja dengan menelusuri graf secara melebar dari simpul awal ke semua tetangganya terlebih dahulu, sebelum melanjutkan ke simpul yang lebih dalam. Algoritma ini cocok digunakan pada graf tak berbobot untuk mencari jalur terpendek dalam jumlah langkah (simpul).

Dalam studi kasus ini, algoritma *Breadth First Search* (BFS) diterapkan untuk menyelesaikan persoalan pencarian jalur terpendek pada peta jaringan stasiun dari titik awal Stasiun ITPLN menuju titik tujuan Plaza Blok M. Masalah ini sangat relevan untuk menggambarkan bagaimana algoritma BFS bekerja secara nyata dan bagaimana optimasi traversal simpul bisa diterapkan dalam kehidupan sehari-hari, seperti sistem navigasi, sistem informasi geografis, maupun aplikasi transportasi publik digital.

1.2 Rumusan Masalah

1. Bagaimana prinsip kerja algoritma *Breadth First Search* (BFS) dalam menelusuri graf?
2. Bagaimana algoritma *Breadth First Search* (BFS) dapat digunakan untuk mencari jalur terpendek pada graf tidak berbobot?
3. Bagaimana penerapan algoritma *Breadth First Search* (BFS) pada kasus jaringan stasiun dari ITPLN menuju Plaza Blok M?
4. Apa hasil traversal dan jalur terpendek yang dihasilkan dari algoritma ini?

1.3 Tujuan Masalah

1. Untuk memahami konsep dasar dan prinsip kerja algoritma *Breadth First Search* (BFS).
2. Untuk mengimplementasikan algoritma *Breadth First Search* (BFS) dalam bentuk kode program menggunakan bahasa Python.
3. Untuk menganalisis hasil traversal dan jalur terpendek yang diperoleh dari algoritma *Breadth First Search* (BFS).

4. Untuk mengevaluasi efektivitas *Breadth First Search* (BFS) dalam menyelesaikan permasalahan pencarian jalur.

1.4 Manfaat Penelitian

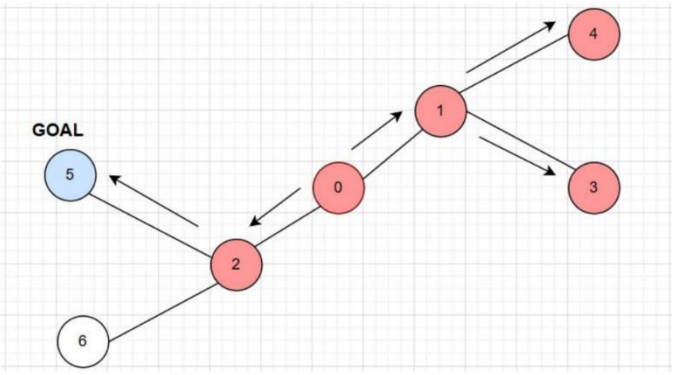
1. Menambah wawasan dan pemahaman tentang konsep algoritma pencarian jalur, khususnya algoritma *Breadth First Search* (BFS).
2. Memberikan contoh penerapan nyata algoritma graf dalam bentuk simulasi rute transportasi yang sering ditemui di kehidupan sehari-hari.
3. Menjadi referensi bagi mahasiswa atau peneliti lain yang ingin mengkaji lebih dalam tentang strategi traversal graf dan pencarian jalur terpendek.
4. Menunjukkan bagaimana algoritma *Breadth First Search* (BFS) dapat diimplementasikan dalam bahasa pemrograman Python secara efisien dan mudah dipahami.

BAB II
LANDASAN TEORI

2.1 Pengertian Breadth First Search (BFS)

Breadth First Search (BFS) adalah algoritma yang digunakan untuk menelusuri atau menjelajahi semua node pada graf secara berurutan, dimulai dari node awal dan menyebar ke node tetangga sebelum melanjutkan ke node yang lebih jauh. Algoritma ini menggunakan pendekatan lebar, sehingga disebut juga dengan *Breadth First*. Berikut ini merupakan langkah-langkah algoritma *Breadth First Search* (BFS) :

1. Masukkan node awal ke dalam queue
   - Node awal (start node) merupakan titik awal pencarian.
   - Tambahkan node ini ke queue dan tandai sebagai telah dikunjungi.
2. Selama queue tidak kosong, lakukan :
   a. Ambil node dari depan queue
      - Gunakan operasi dequeue (ambil node paling depan dari antrian).
      - Node ini menjadi node yang sedang dieksplorasi.
   b. Tandai node sebagai dikunjungi
      - Simpan node tersebut dalam struktur visited, misalnya list atau set, agar tidak dikunjungi dua kali.
   c. Tambahkan semua tetangga yang belum dikunjungi ke dalam queue
      - Lihat semua tetangga dari node saat ini.
      - Jika tetangga belum dikunjungi, tambahkan ke queue.
      - Tetangga-tetangga ini akan dikunjungi pada putaran selanjutnya.
3. Ulangi sampai menemukan node tujuan
   - Jika selama eksplorasi ditemukan node tujuan, maka algoritma berhenti.
   - Jika queue habis dan tujuan belum ditemukan, maka tidak ada jalur dari node awal ke tujuan.



Gambar Ilustrasi Proses Pencarian Jalur Algoritma *Breadth-First Search*

2.2 Struktur Data Queue

Queue (antrian) adalah struktur data linear yang mengikuti prinsip *First In First Out* (FIFO), yang berarti elemen yang masuk lebih dulu akan keluar lebih dulu. Dalam implementasi BFS, queue digunakan untuk menyimpan jalur sementara yang akan dikembangkan. Simpul yang masuk lebih awal akan dikunjungi lebih dahulu, menjamin traversal level demi level.

2.3 Pencarian Jalur pada Graf

Pencarian jalur (*pathfinding*) adalah proses menemukan rute dari satu titik ke titik lainnya dalam struktur graf. Dalam implementasinya, graf digunakan untuk merepresentasikan jaringan seperti jalan, stasiun, atau bahkan koneksi dalam sistem komputer. Terdapat dua jenis graf yaitu :

- Graf berbobot (*weighted graph*) setiap sisi memiliki nilai bobot tertentu.
- Graf tak berbobot (*unweighted graph*) setiap sisi dianggap memiliki bobot yang sama.

Algoritma *Breadth First Search* (BFS) sangat cocok untuk digunakan pada graf tak berbobot, karena BFS menjamin bahwa simpul yang pertama kali ditemukan adalah simpul dengan jarak langkah minimum (*minimum number of hops*) dari titik awal.

2.4 Kelebihan dan Kekurangan BFS

a. Kelebihan *Breadth First Search* (BFS)
   - Tidak akan menemui jalan buntu.
   - Jika ada satu solusi, maka BFS akan menemukannya dan jika ada lebih dari satu solusi, maka solusi minimum akan ditemukan.
   - BFS berguna untuk menganalisis node dalam grafik dan membangun jalur terpendek untuk melintasinya.
   - BFS dapat melintasi grafik dengan jumlah iterasi terkecil.
   - Arsitektur algoritma BFS sederhana dan kuat.

b. Kekurangan *Breadth First Search* (BFS)
   - Membutuhkan memori yang cukup banyak, karena menyimpan semua node dalam satu pohon.
   - Membutuhkan memori yang cukup banyak, karena menyimpan semua node dalam satu pohon.

2.5 Aturan Algoritma BFS

Berikut adalah aturan penting untuk menggunakan algoritma BFS yaitu

1. Antrian *First In First Out* (FIFO) struktur data digunakan oleh BFS.
2. Dengan menandai node mana pun dalam grafik sebagai root dan mulai melintasi data dari node tersebut.
3. BFS melintasi semua node dalam grafik dan terus menghapusnya setelah selesai.

4. BFS mengunjungi node berdekatan yang belum dikunjungi, menandainya sebagai selesai, dan memasukkannya ke dalam antrian.
5. Menghapus simpul sebelumnya dari antrian jika tidak ditemukan simpul yang berdekatan.
6. Algoritma BFS melakukan iterasi hingga semua simpul pada grafik berhasil dilintasi dan ditandai sebagai selesai.

2.6 Perbandingan BFS (Breadth First Search) dengan DFS (Depth First Seacrh)

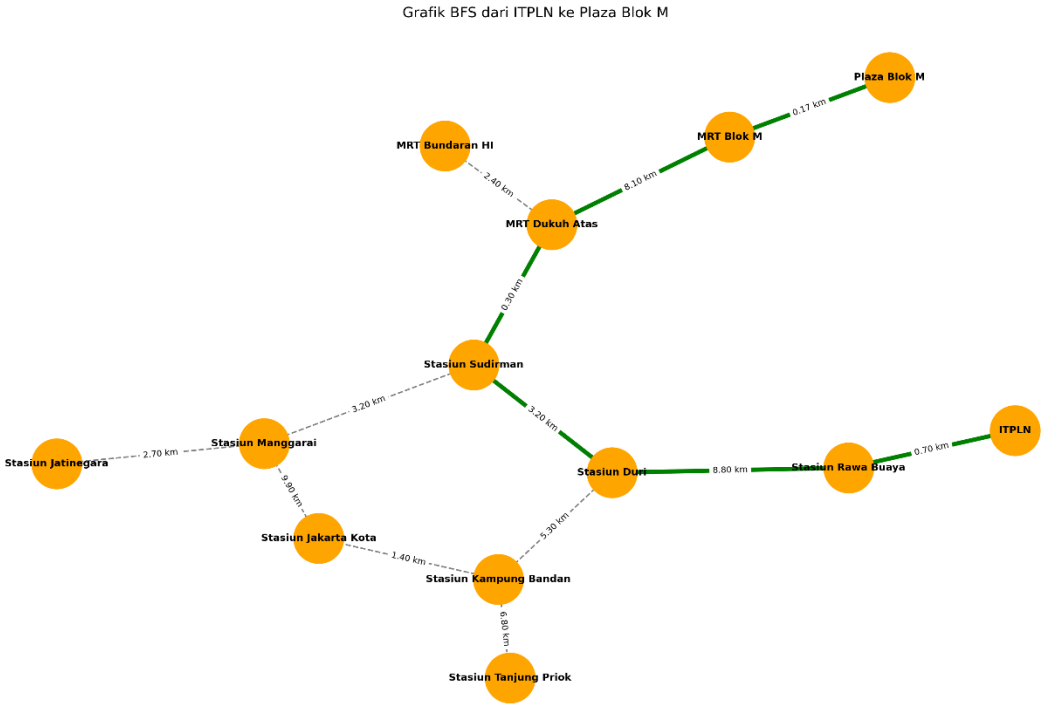
Algoritma *Breadth First Search* (BFS) dan *Depth First Search* (DFS) merupakan dua metode dasar dalam penelusuran graf. Keduannya digunakan untuk mengeksplorasi node atau simpul dalam graf, baik untuk keperluan pencarian, analisis jalur, maupun penyelesaian permasalahan yang melibatkan struktur data graf. Meskipun tujuan dasarnya sama yaitu menjelajahi graf, pendekatan dan karakteristik dari kedua algoritma ini berbeda secara signifikan.

1. Strategi Traversal
   - BFS melakukan penelusuran secara *breadth-wise* yaitu dengan mengunjungi seluruh tetangga dari simpul awal terlebih dahulu sebelum melanjutkan ke tingkat berikutnya.
   - DFS melakukan penelusuran secara *depth-wise* yaitu mengikuti satu cabang graf sedalam mungkin sebelum kembali dan menjelajahi cabang lain.
2. Struktur Data yang Digunakan
   - BFS menggunakan Queue (antrian) dengan prinsip FIFO (*First In First Out*) untuk menyimpan simpul yang akan dikunjungi.
   - DFS menggunakan Stack (tumpukan) denfan prinsip LIFO (*Last In First Out*) untuk melakukan penelusuran.
3. Jalur Terpendek
   - BFS sangat efektif dalam menemukan jalur terpendek pada graf tak berbobot.
   - DFS tidak menjamin pencarian jalur terpendek karena bisa masuk ke cabang yang dalam terlebih dahulu.

Selain DFS, algoritma BFS juga sering dibandingkan dengan algoritma lain seperti Dijkstra dan A*. Namun, perbandingan yang paling relevan dalam konteks graf tak berbobot adalah antara BFS dan DFS karena keduanya tidak memperhitungkan bobot sisi dan umum digunakan untuk eksplorasi graf dasar.

BAB III
PEMBAHASAN

3.1 Grafik BFS ITPLN - Plaza Blok M

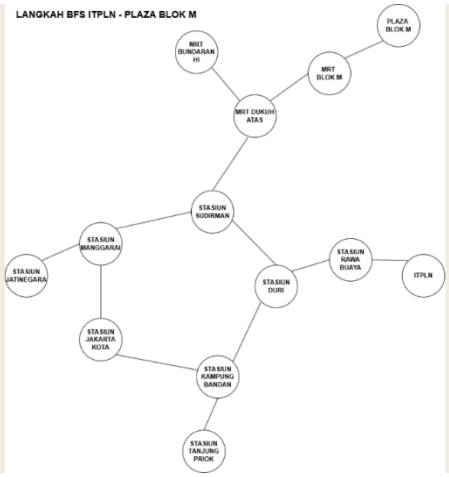


Gambar tersebut menunjukkan peta jaringan stasiun yang merepresentasikan proses BFS dari stasiun ITPLN menuju Plaza Blok M. Traversal BFS dimulai dari ITPLN, kemudian menjelajah secara melebar ke stasiun-stasiun terdekat yang belum dikunjungi, sesuai dengan urutan abjad karena penggunaan fungsi *sorted()*. Proses BFS ini bertujuan untuk menemukan jalur terpendek (dalam jumlah simpul) dari titik awal ke tujuan. Jalur terpendek yang ditemukan adalah:

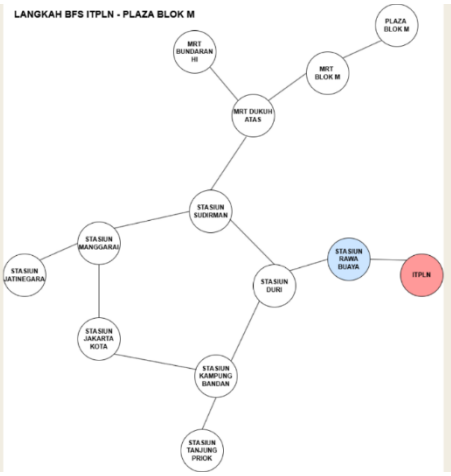
ITPLN → Stasiun Rawa Buaya → Stasiun Duri → Stasiun Sudirman → MRT Dukuh Atas → MRT Blok M → Plaza Blok M

3.2 Langkah-langkah BFS dari ITPLN ke Plaza Blok M dengan Urutan Abjad Saat Memasukkan Tetangga ke Antrian

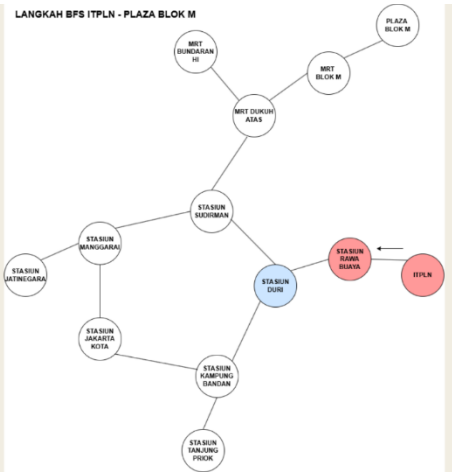
1. Menentukan Titik Awal = ITPLN | Titik Akhir = Plaza Blok M



2. Kunjungi ITPLN | Tambahkan Stasiun Rawa Buaya ke antrian **Queue Stasiun Rawa Buaya**



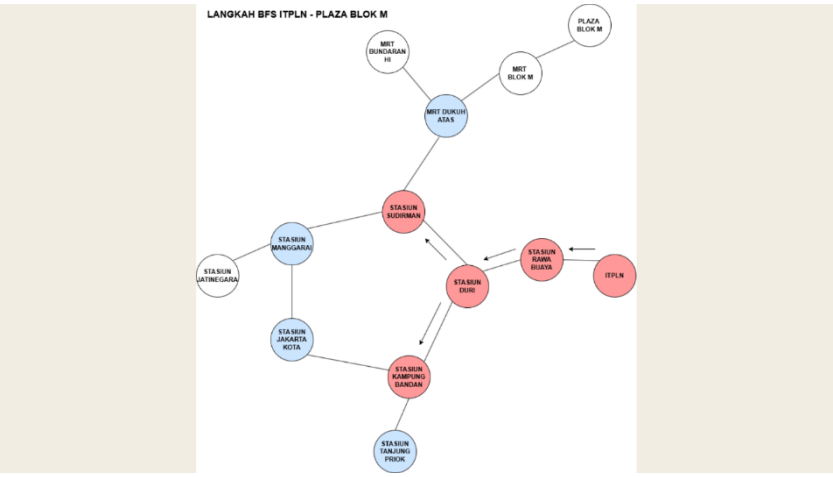
3. ITPLN sudah dikunjungi | Kunjungi Stasiun Rawa Buaya | Tambahkan Stasiun Duri ke antrian **Queue Stasiun Duri**



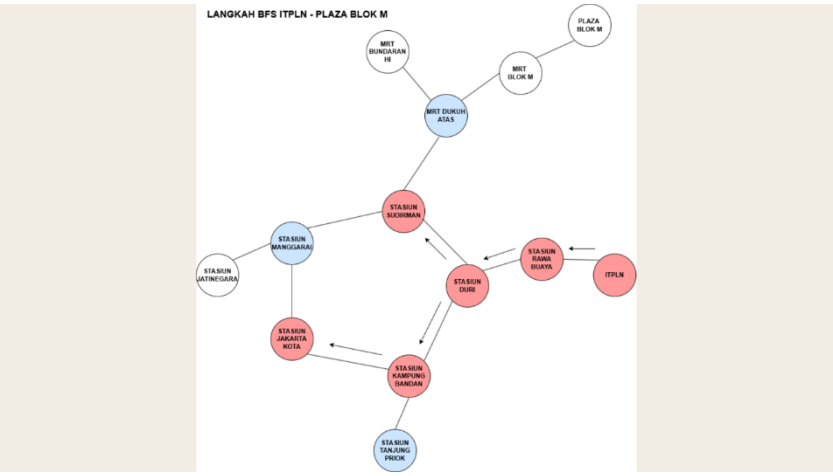
4. Stasiun Rawa Buaya sudah dikunjungi | Kunjungi Stasiun Duri | Tambahkan Stasiun Kampung Bandan & Stasiun Sudirman ke antrian Queue Stasiun Kampung Bandan dan Stasiun Sudirman

5. Stasiun Duri sudah dikunjungi | Kunjungi Stasiun Kampung Bandan | Tambahkan Stasiun Jakarta Kota & Stasiun Tanjung Priok ke antrian **Queue Stasiun Sudirman, Stasiun Jakarta Kota dan Stasiun Tanjung Priok**

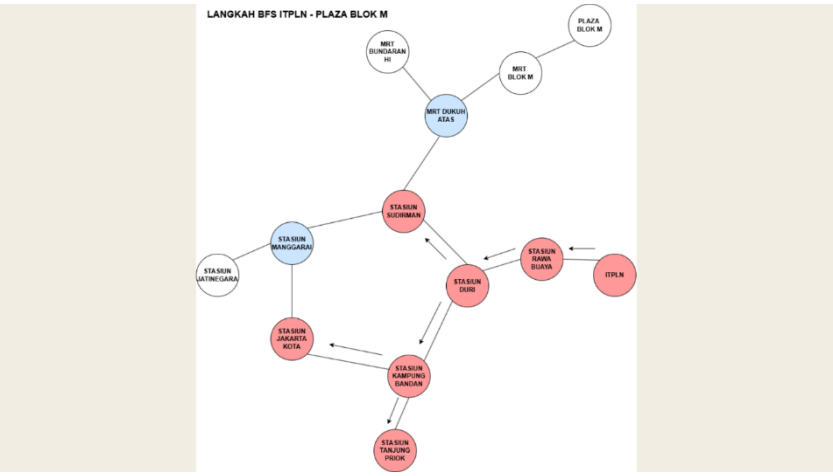
6. Stasiun Kampung Bandan sudah dikunjungi | Kunjungi Stasiun Sudirman | Tambahkan MRT Dukuh Atas & StasiunManggarai ke antrian **Queue Stasiun Jakarta Kota, Stasiun Tanjung Priok, MRT Dukuh Atas dan Stasiun Manggarai**



7. Stasiun Sudirman sudah dikunjungi | Kunjungi Stasiun Jakarta Kota | Stasiun Manggarai sudah masuk ke antrian **Queue Stasiun Tanjung Priok, MRT Dukuh Atas dan Stasiun Manggarai**



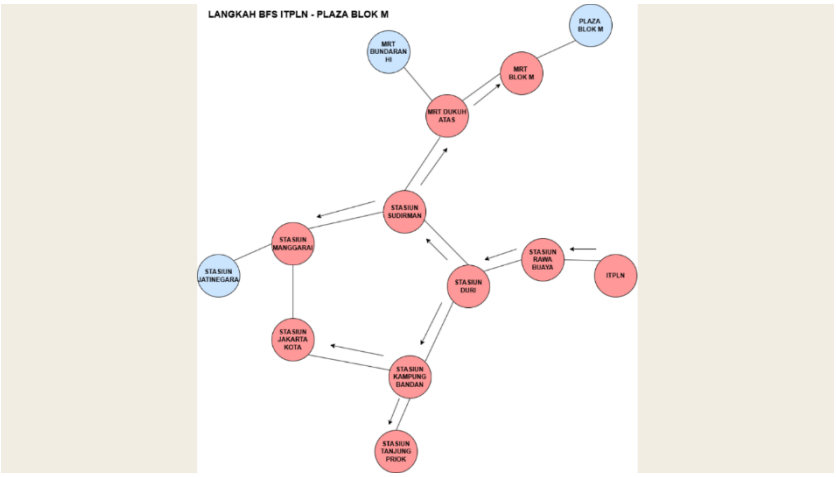
8. Stasiun Jakarta Kota sudah dikunjungi | Kunjungi Stasiun Tanjung Priok | Tidak ada tetangga baru **Queue MRT Dukuh Atas dan Stasiun Manggarai**



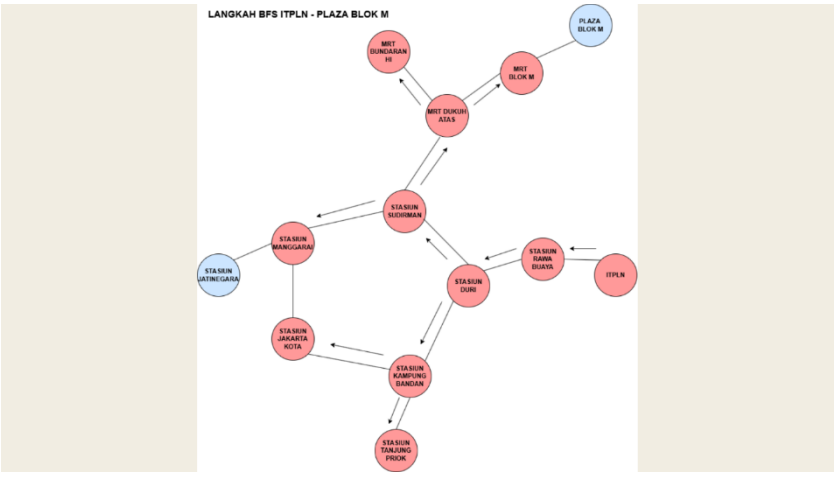
9. Stasiun Tanjung Priok sudah dikunjungi | Kunjungi MRT Dukuh Atas | Tambahkan MRT Blok M & MRT Bundaran HI ke antrian **Queue Stasiun Manggarai, MRT Blok M dan MRT Bundaran HI**

10. MRT Dukuh Atas sudah dikunjungi | Kunjungi Stasiun Manggarai | Tambahkan Stasiun Jatinegara ke antrian Queue MRT Blok M, MRT Bundaran HI dan Stasiun Jatinegara **Queue MRT Blok M, MRT Bundaran HI dan Stasiun Jatinegara**

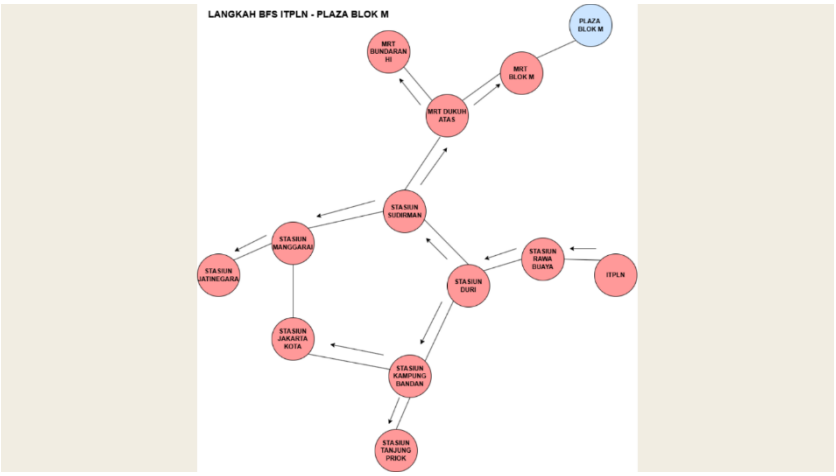
11. Stasiun Manggarai sudah dikunjungi | Kunjungi MRT Blok M | Tambahkan Plaza Blok M **Queue MRT Bundaran HI, Stasiun Jatinegara dan Plaza Blok M**



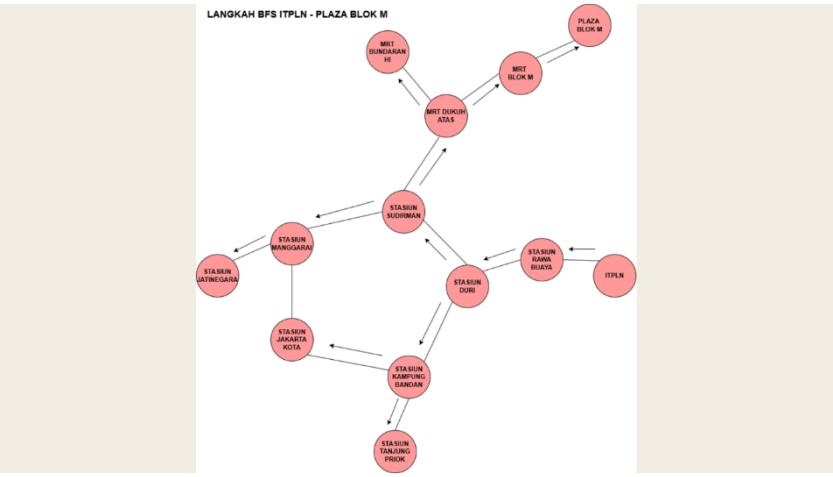
12. MRT Blok M sudah dikunjungi | Kunjungi MRT Bundaran HI | Tidak ada tetangga baru **Queue Stasiun Jatinegara dan Plaza Blok M**



13. MRT Bundaran HI sudah dikunjungi | Kunjungi Stasiun Jatinegara | Tidak ada tetangga baru **Queue Plaza Blok M**



14. Stasiun Jatinegara sudah dikunjungi | Kunjungi Plaza Blok M | Solusi Ditemukan



- Urutan BFS traversal
  ITPLN → Stasiun Rawa Buaya → Stasiun Duri → Stasiun Kampung Bandan → Stasiun Sudirman → Stasiun Jakarta Kota → Stasiun Tanjung Priok → MRT Dukuh Atas → Stasiun Manggarai → MRT Blok M → MRT Bundaran HI → Stasiun Jatinegara → Plaza Blok M

- Jalur terpendek dari ITPLN ke Plaza BlokM
  ITPLN → Stasiun Rawa Buaya → Stasiun Duri → Stasiun Sudirman → MRT Dukuh Atas → MRT Blok M → Plaza Blok M

3.3 PseudoCode

1. Buat graf kosong G
2. Definisikan route_optimal sebagai daftar:

   ["ITPLN", "Stasiun Rawa Buaya", "Stasiun Duri",

   "Stasiun Sudirman", "MRT Dukuh Atas", "MRT Blok M", "Plaza Blok M"]
3. Definisikan route_alternatif sebagai daftar:

   ["ITPLN", "Stasiun Rawa Buaya", "Stasiun Duri", "Stasiun Kampung Bandan",

   "Stasiun Jakarta Kota", "Stasiun Manggarai", "Stasiun Sudirman",

   "MRT Dukuh Atas", "MRT Blok M", "Plaza Blok M"]
4. Fungsi add_route_edges(route):

   Untuk i dari 0 hingga panjang(route) - 2:

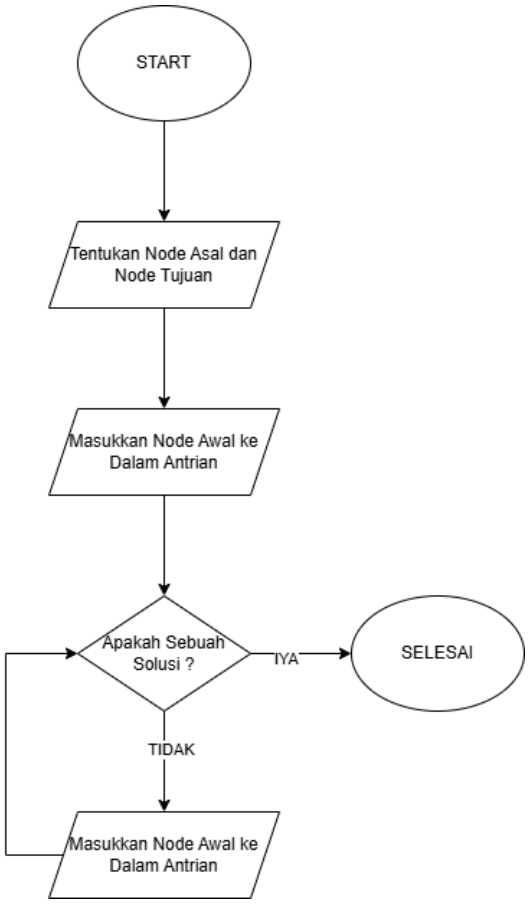
   Tambahkan edge dari route[i] ke route[i + 1] ke dalam G
5. Panggil add_route_edges(route_optimal)

6. Panggil add_route_edges(route_alternatif)
7. Tambahkan cabang tambahan ke graf G:
   - Stasiun Kampung Bandan ↔ Stasiun Tanjung Priok
   - Stasiun Manggarai ↔ Stasiun Jatinegara
   - Stasiun Manggarai ↔ Stasiun Cawang
   - MRT Dukuh Atas ↔ MRT Bundaran HI
8. Jika node "Stasiun Cawang" ada di graf:
   - Hapus node tersebut dari graf G
9. Fungsi bfs_traversal_and_shortest_path(graf, start, goal):
   a. Buat himpunan visited kosong
   b. Buat queue berisi list [start]
   c. Buat traversal_order sebagai list kosong
   d. Selama queue tidak kosong:
      - Ambil path dari depan queue
      - Ambil node terakhir dari path
      - Jika node belum dikunjungi:
        - Tambahkan node ke traversal_order
        - Tandai node sebagai dikunjungi
        - Jika node == goal:
      - Kembalikan traversal_order dan path
      - Untuk setiap neighbor dari node (urut abjad):
        - Jika neighbor belum dikunjungi:
        - Salin path → new_path
        - Tambahkan neighbor ke new_path
        - Masukkan new_path ke queue
   e. Jika tidak ditemukan:
      - Kembalikan traversal_order dan None
10. Panggil bfs_traversal_and_shortest_path(G, "ITPLN", "Plaza Blok M")

→ Simpan hasil ke traversal_order dan shortest_path

11. Cetak tetangga dari Stasiun Manggarai
12. Cetak traversal_order hasil BFS
13. Cetak shortest_path dari ITPLN ke Plaza Blok M
14. Gambar graf:
    - Node warna oranye
    - Jalur shortest_path dengan garis hijau tebal
    - Edge lain digambar putus-putus
    - Tampilkan label nama stasiun
    - Tambahkan judul: "Grafik BFS dari ITPLN ke Plaza Blok M"

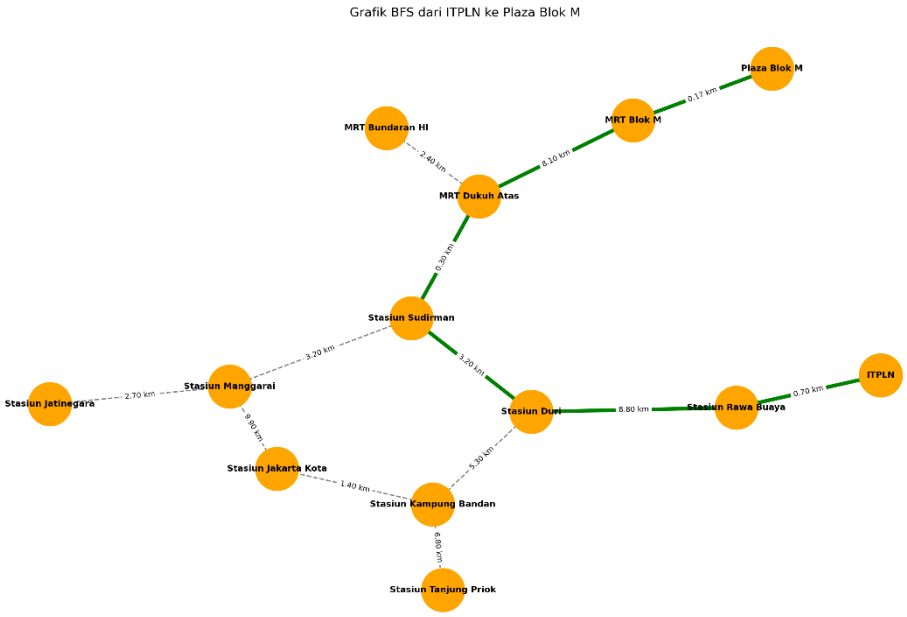
3.4 Gambar Flowchart



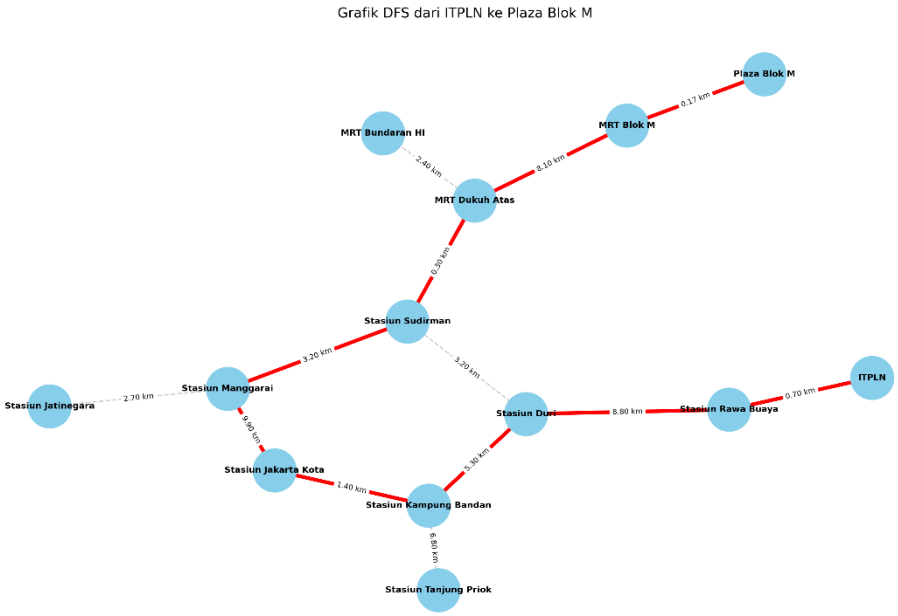
3.5 Hasil Perbandingan BFS dan DFS

Berikut adalah perbandingan hasil visual dan traversal antara algoritma BFS dan DFS berdasarkan dua gambar graf berikut :

1. Perbandingan Hasil Visual



Gambar 1 : Grafik BFS dari ITPLN ke Plaza Blok M



Gambar 2 : Grafik DFS dari ITPLN ke Plaza Blok M

2. Perbandingan Hasil Traversal
    a. BFS (Breadth First Search)
        - Jalur = ITPLN → Stasiun Rawa Buaya → Stasiun Duri → Stasiun Sudirman → MRT Dukuh Atas → MRT Blok M → Plaza Blok M
        - Total simpul yang dilalui = 7 simpul
        - Jalur ini langsung menuju simpul tujuan dengan jumlah langkah yang menimum.
        - BFS menyelesaikan pencarian dengan lebih efisisen karena tidak menelusuri jalur lain yang tidak perlu.
    b. DFS (Depth First Search)
        - Jalur = ITPLN → Stasiun Rawa Buaya → Stasiun Duri → Stasiun Kampung Bandan → Stasiun Jakarta Kota → Stasiun Manggarai Stasiun Sudirman → MRT Dukuh Atas → MRT Blok M → Plaza Blok M
        - Total simpul yang dilalui = 10 simpul
        - Jalur ini langsung menuju simpul tujuan dengan jumlah langkah yang menimum.
        - DFS menjelajahi lebih banyak simpul dan mengambil rute lebih penjang karena bersifat mendalam dan tidak mempertimbangkan efisiensi jarak.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan hasil analisis dan implementasi yang telah dilakukan, dapat disimpulkan bahwa algoritma *Breadth First Search* **(BFS)** merupakan metode pencarian jalur yang sangat efektif untuk graf tak berbobot, khususnya dalam menemukan jalur terpendek berdasarkan jumlah simpul yang dilalui. BFS bekerja dengan menjelajahi graf secara melebar, memastikan bahwa setiap simpul pada tingkat yang sama dikunjungi terlebih dahulu sebelum melangkah lebih dalam. Dalam penerapannya pada kasus jaringan stasiun dari ITPLN ke Plaza Blok M, BFS berhasil menemukan rute terpendek secara efisien dengan traversal minimal. Dibandingkan dengan *Depth First Search* (DFS), BFS menunjukkan keunggulan dari segi efisiensi dan akurasi jalur, karena DFS cenderung menjelajah lebih dalam tanpa mempertimbangkan optimalitas lintasan. Selain memberikan hasil yang akurat, struktur data queue dan prinsip FIFO pada BFS juga membuat algoritma ini lebih mudah diimplementasikan dan dipahami dalam bahasa pemrograman seperti Python. Dengan demikian, algoritma BFS sangat relevan digunakan dalam berbagai permasalahan pencarian jalur dalam sistem informasi, navigasi, dan transportasi digital.

DAFTAR PUSTAKA

Chandra, D., Ramaningtyas, A. T., & Hakim, L. (2021). Penerapan Breadth First Search untuk Mengelola Keuangaan dengan Menentukan Karakteristik Investasi Individu. *Jurnal Teknik Informatika UNIKA Santo Thomas*, 395-402.

Muhardono, A. (2023). Penerapan Algoritma Breadth First Search dan Depth First Search pada Game Angka. *Jurnal Minfo Polgan*, *12*(1), 171-182.

Suryani, S., Nurdiansah, N., Faizal, F., Nasaruddin, N., & Alam, S. (2023). Implementasi Algoritma BFS pada Desain Sistem Pengolahan Temu Kembali Berkas. *REMIK: Riset dan E-Jurnal Manajemen Informatika Komputer*, *7*(1), 675-685.

Wibowo, A., & Fathurrahman, F. (2011). Implementasi Algoritma Breadth First Search Dan Obstacle Detection Dalam Penelusuran Labirin Dinamis Menggunakan Robot Lego. *Jurnal Ilmu Komputer dan Informasi*, *4*(1), 15-22.

Yuliana, Y., Noviyanti, N., & Qulub, M. (2024). IMPLEMENTASI ALGORITMA DEPTH-FIRST SEARCH DAN BREADTH-FIRST SEARCH PADA DOKUMEN AKREDITASI. *JOURNAL OF SCIENCE AND SOCIAL RESEARCH*, *7*(1), 197-204.