LAPORAN TUGAS BESAR 1 IF2211
STRATEGI ALGORITMA

Dosen Pengampu : **Dr. Ir. Rinaldi, M.T.**
Dr. Nur Ulfa Maulidevi, S. T, M.Sc

Asisten pembimbing : **Michael Leon Putra W.**

Disusun oleh:

Kelompok - 48 - GoLink

Debrina Veisha Rashika W (13522025)
Angelica Kierra Ninta Gurning (13522048)
Novelya Putri Ramadhani (13522096)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

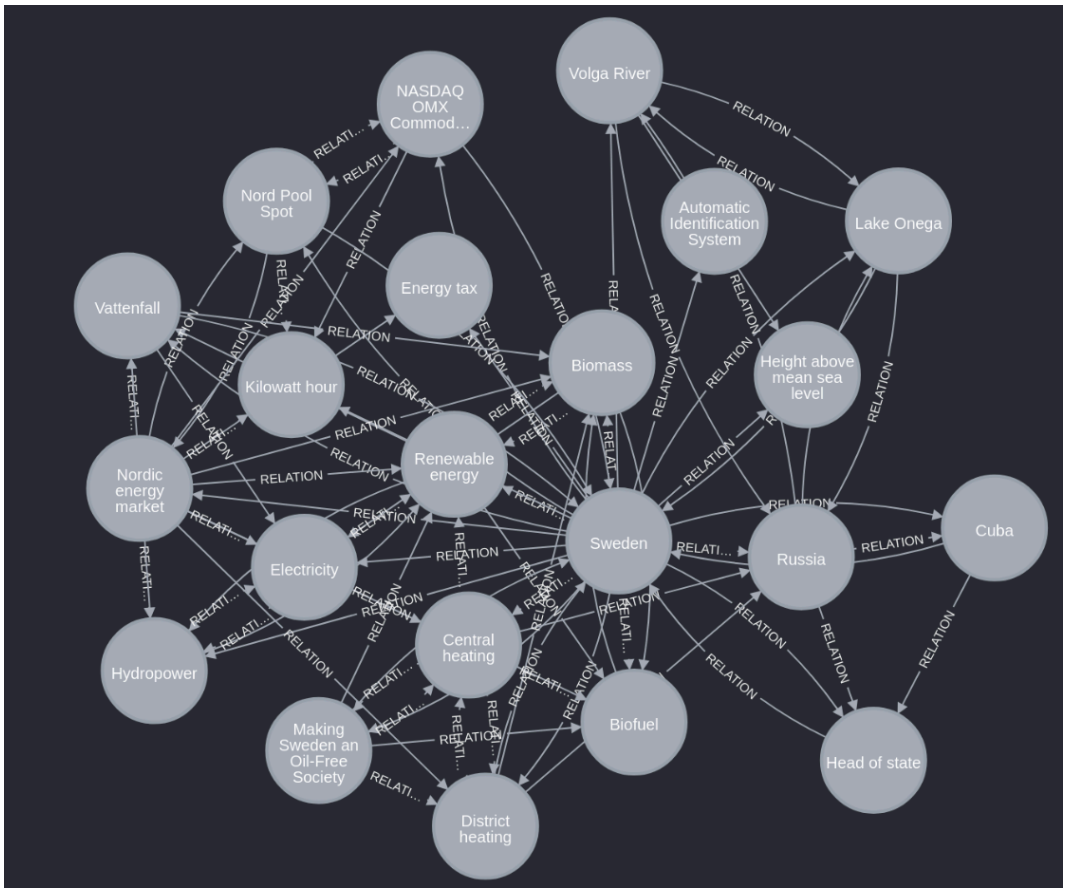
DAFTAR ISI

DAFTAR ISI 1
BAB I 3
BAB II 4
2.1. Penjelajahan Graf 4
2.2. Algoritma Iterative Deepening Search (IDS) 5
2.3. Algoritma Breadth First Search (BFS) 6
2.4. Aplikasi Web yang Dibangun 7
2.4.1. Front End Development 7
2.4.2. Back End Development 8
2.4.3. Docker 8
BAB III 10
3.1. Langkah-Langkah Pemecahan Masalah 10
3.2. Proses Pemetaan Masalah 13
3.3.1. Algoritma Iterative Deepening Search (IDS) 13
3.3.2. Algoritma Breadth First Search (BFS) 14
3.3. Fitur fungsional dan arsitektur aplikasi web yang dibangun 14
3.3.1. Form Input 15
3.3.2. Visualisasi Graf dan Hasil 17
3.3.3. Loading Screen 17
3.3.4. Authors 18
3.3.5. Docker 18
3.3.6. Toast input kosong 18
3.3.7. Toast gagal menerima data 19
3.3.8. Checkbox untuk solusi banyak 19
3.3.9. Link ke Repository Github 19
3.4. Contoh Ilustrasi Kasus 19
BAB IV 21
4.1. Spesifikasi teknis program 21
4.1.1. Struktur data 21
4.1.2. Fungsi dan prosedur 22
4.2. Penjelasan Tata Cara Penggunaan Program 27
4.2.1. Interface program 27
4.2.2. Tata Cara Penggunaan Program 31
4.3. Hasil pengujian 32
1. Test Case 1 32
2. Test Case 2 34
3. Test Case 3 35
4. Test Case 4 36
5. Test Case 5 37
4.4. Analisis hasil pengujian 39

BAB V.....40
5.1. Kesimpulan.....40
5.2. Saran.....40
LAMPIRAN.....42
Repository.....42
Youtube.....42
DAFTAR PUSTAKA.....43

BAB I
DESKRIPSI TUGAS

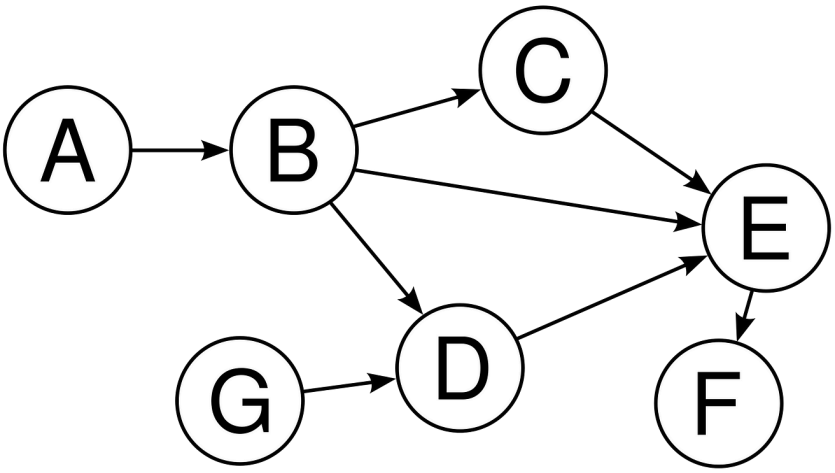
WikiRace atau Wiki Game adalah permainan yang melibatkan Wikipedia, sebuah ensiklopedia daring gratis yang dikelola oleh berbagai relawan di dunia, dimana pemain mulai pada suatu artikel Wikipedia dan harus menelusuri artikel-artikel lain pada Wikipedia (dengan mengeklik tautan di dalam setiap artikel) untuk menuju suatu artikel lain yang telah ditentukan sebelumnya dalam waktu paling singkat atau klik (artikel) paling sedikit.



Gambar 1.1 Ilustrasi Graf Wikirace

BAB II
LANDASAN TEORI

2.1. Penjelajahan Graf



Gambar 2.1.1 Contoh gambar graph

Penjelajahan graf adalah proses mengunjungi setiap simpul dan sisi pada graf tepat satu kali dalam urutan yang terdefinisi dengan baik. Hal ini penting dalam banyak aplikasi, seperti menemukan jalur terpendek antara dua titik, menentukan apakah dua simpul terhubung, dan mewarnai graf.

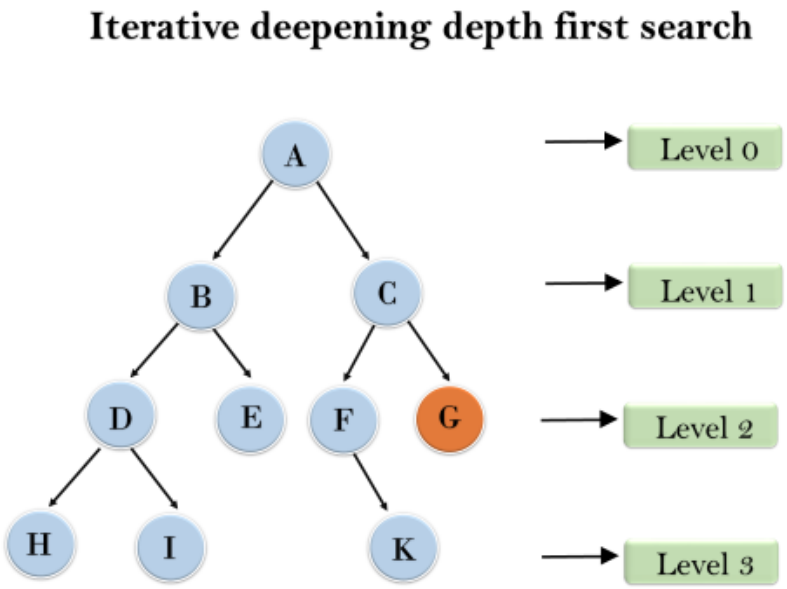
Ada dua algoritma penjelajahan graf yang umum digunakan: pencarian pertama mendalam (DFS) dan pencarian pertama luas (BFS). DFS bekerja dengan menjelajahi setiap jalur dalam graf sedalam mungkin sebelum beralih ke jalur lain. Sebaliknya, BFS bekerja dengan mengunjungi semua simpul pada jarak yang sama dari simpul awal sebelum beralih ke simpul pada jarak berikutnya Algoritma penjelajahan graf dapat dikategorikan menjadi dua jenis utama: tanpa informasi (uninformed/blind search) dan dengan informasi (informed search).

Algoritma tanpa informasi tidak memiliki informasi tambahan tentang graf, dan mereka menjelajahi graf secara sistematis untuk menemukan solusi. Contoh algoritma tanpa informasi termasuk Pencarian Pertama Mendalam (*DFS*), Pencarian Pertama Luas (*BFS*), Pencarian Kedalaman Terbatas (*Depth Limited Search*), Pencarian Pendalaman

Iteratif (*Iterative Deepening Search*), dan Pencarian Biaya Seragam (*Uniform Cost Search*).

Algoritma dengan informasi, di sisi lain, menggunakan informasi tambahan tentang graf, seperti heuristik, untuk memandu pencarian mereka menuju solusi yang lebih menjanjikan. Heuristik adalah fungsi yang memperkirakan jarak dari suatu simpul ke simpul tujuan. Contoh algoritma dengan informasi termasuk Pencarian Terbaik Pertama (*Best First Search*) dan Pencarian A (*A* Search*).

2.2. Algoritma Iterative Deepening Search (IDS)



Gambar 2.2.1 Contoh IDS

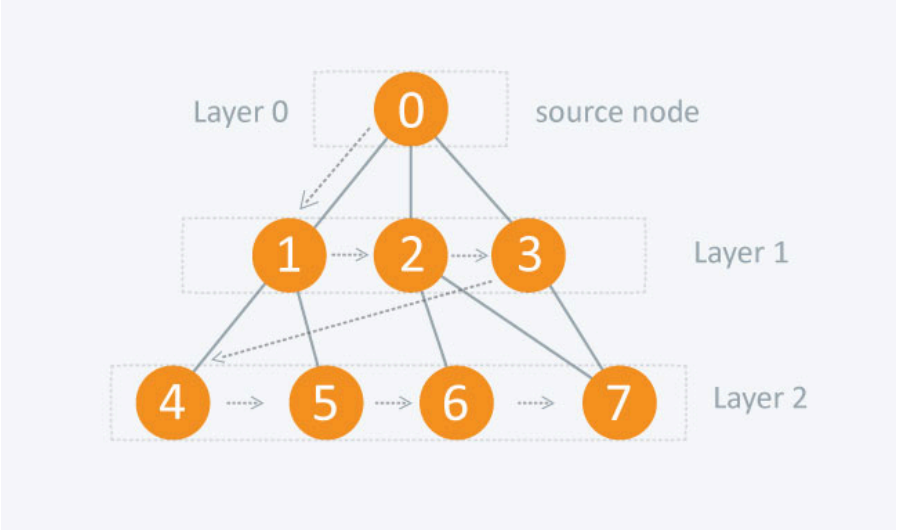
Iterative Deepening Search (IDS) adalah algoritma pencarian yang digunakan untuk mencari solusi dalam ruang pencarian yang memiliki struktur seperti pohon atau graf. IDS merupakan kombinasi dari dua teknik pencarian, yaitu Depth First Search (DFS) dan Breadth First Search (BFS), yang mengatasi kelemahan dari masing-masing teknik tersebut. IDS bekerja dengan melakukan pencarian secara bertahap dengan meningkatkan kedalaman pencarian pada setiap iterasi.

Cara kerja IDS dimulai dengan melakukan pencarian dengan kedalaman terbatas menggunakan DFS. Pada setiap iterasi, IDS akan membatasi kedalaman pencarian maksimumnya. Jika solusi tidak ditemukan pada kedalaman tersebut, IDS akan meningkatkan kedalaman pencarian dan melakukan pencarian kembali. Proses ini akan terus berlanjut hingga solusi ditemukan.

IDS memanfaatkan konsep kedalaman terbatas dari DFS untuk menghemat ruang memori dan mengurangi kompleksitas waktu yang terjadi pada BFS. Dengan membatasi kedalaman pencarian pada setiap iterasinya, IDS dapat mengatasi masalah ruang memori yang sering terjadi pada DFS. Selain itu, IDS juga dapat menemukan solusi optimal dalam ruang pencarian yang memiliki kedalaman tak terbatas dengan cara yang lebih efisien daripada BFS.

Dalam praktiknya, IDS sering digunakan dalam penyelesaian masalah yang melibatkan pencarian solusi optimal dalam ruang pencarian yang kompleks. Contohnya adalah dalam permainan strategi seperti catur atau permainan teka-teki, IDS dapat digunakan untuk mencari solusi optimal dengan membatasi kedalaman pencarian sesuai dengan kompleksitas permainan.

2.3. Algoritma Breadth First Search (BFS)



Gambar 2.3.1 Contoh BFS

Breadth First Search (BFS) merupakan algoritma yang sangat berguna dalam pemrosesan graf, khususnya dalam mencari jalur terpendek atau menjelajahi struktur graf secara sistematis. Dalam dunia pemrograman dan ilmu komputer, BFS sering digunakan dalam berbagai aplikasi, mulai dari pemodelan jaringan komputer hingga permainan video dan kecerdasan buatan.

Algoritma ini bekerja dengan cara yang sangat terstruktur dan sistematis. Pertama-tama, BFS memilih simpul awal sebagai titik awal pencarian. Simpul awal ini menjadi titik referensi dari mana pencarian dimulai. Kemudian, semua tetangga dari simpul awal dieksplorasi secara berurutan. Ini berarti semua simpul yang langsung terhubung dengan simpul awal akan dieksplorasi terlebih dahulu sebelum memeriksa simpul-simpul yang lebih jauh.

Pentingnya menggunakan struktur data queue dalam BFS adalah untuk memastikan bahwa penjelajahan dilakukan secara terurut. Ketika sebuah simpul dieksplorasi, simpul-simpul tetangga yang terhubung dengannya akan dimasukkan ke dalam antrian. Kemudian, simpul-simpul ini akan dieksplorasi satu per satu, dan proses ini akan berlanjut hingga tidak ada lagi simpul yang tersisa untuk dieksplorasi.

Salah satu keunggulan utama dari BFS adalah kemampuannya untuk menemukan jalur terpendek antara dua simpul dalam graf. Misalnya, dalam aplikasi navigasi, BFS dapat digunakan untuk menemukan rute terpendek antara dua lokasi. Selain itu, BFS juga berguna dalam memeriksa keterhubungan atau jarak antara simpul-simpul dalam jaringan komputer atau sosial.

2.4. Aplikasi Web yang Dibangun

2.4.1. Front End Development

Front-end adalah disiplin dalam pengembangan web yang bertanggung jawab untuk membangun tampilan suatu website. Front-End berhubungan dengan interaksi langsung oleh pengguna. Pengembangan front-end melibatkan pemanfaatan beragam teknologi dan bahasa pemrograman untuk menciptakan antarmuka pengguna yang responsif, menarik, dan berfungsi. Pada dasarnya, bahasa yang digunakan adalah HTML (Hypertext Markup Language) yang digunakan untuk membuat struktur dasar halaman web. Untuk membuat tampilan lebih menarik dapat digunakan CSS

(Cascading Style Sheets) untuk mengatur tata letak secara visual, seperti margin, warna, font, padding dan lainnya sesuai dengan halaman web.

Selain HTML dan CSS, front-end development juga sering melibatkan penggunaan JavaScript, yang merupakan bahasa pemrograman yang digunakan untuk membuat halaman web menjadi interaktif. Dengan JavaScript, pengembang dapat menambahkan berbagai fitur dinamis seperti animasi, validasi formulir, efek transisi, dan masih banyak lagi. Selain itu, framework JavaScript seperti React, Angular, dan Vue.js juga sering digunakan untuk mempermudah pengembangan front-end dengan menyediakan komponen-komponen yang dapat digunakan ulang dan menyederhanakan manajemen state aplikasi.

2.4.2. Back End Development

Back-end bertanggung jawab pada pemrosesan data, dan logika dalam bentuk sisi server. Bisa disebut. backend merupakan "otak" dari suatu aplikasi. Di dalam Back-End terdapat istilah "Server-Side Logic" yang mengandung logika kode yang dijalankan di sisi server. Logika mencakup operasi-operasi yang diperlukan dan tidak ditunjukkan kepada *client* (front-end). Terdapat juga database management untuk mengelola data aplikasi. Pada Back-End juga akan menyediakan API untuk berkomunikasi dengan sisi front-end.

Sebagai bagian dari komunikasi antara front-end dan back-end, back-end juga menyediakan API (Application Programming Interface) yang memungkinkan aplikasi front-end untuk berinteraksi dengan sistem back-end. API ini berfungsi sebagai jembatan untuk mentransfer data antara kedua sisi aplikasi, memungkinkan front-end untuk mengakses dan memanipulasi informasi yang tersimpan di sisi server.

2.4.3. Docker

Docker adalah platform open-source yang memungkinkan pengguna untuk menyebarkan aplikasi dalam wadah virtual yang terisolasi. Wadah ini berisi semua komponen yang diperlukan untuk menjalankan aplikasi, termasuk sistem operasi, dependensi, dan konfigurasi. Keuntungan utama Docker adalah portabilitasnya, karena aplikasi dapat dijalankan dengan cara yang sama di berbagai lingkungan, terlepas dari

sistem operasi atau konfigurasi server. Hal ini membantu mengatasi masalah kompatibilitas dan memastikan aplikasi berjalan dengan konsisten.

Docker bekerja dengan menggunakan konsep kontainerisasi. Kontainerisasi adalah proses pengemasan aplikasi ke dalam wadah. Wadah Docker berisi semua komponen yang diperlukan untuk menjalankan aplikasi, termasuk sistem operasi, dependensi, dan konfigurasi. Image Docker adalah template yang berisi instruksi untuk membuat wadah Docker

Container ini menghindarkan user dari masalah kompatibilitas yang mungkin terjadi ketika menjalankan aplikasi di beberapa sistem yang berbeda. Sebab, dengan container ini, software akan berjalan dengan cara yang sama di environment mana pun.

BAB III
ANALISIS PEMECAHAN MASALAH

3.1. Langkah-Langkah Pemecahan Masalah

Untuk menyelesaikan permasalahan ini, diperlukan beberapa dekomposisi masalah guna mempermudah dalam memecahkan masalah. Berikut merupakan hasil dekomposisi serta langkah-langkah pemecahan masalah:

1. Memahami dan mempelajari struktur data yang akan digunakan untuk implementasi penyelesaian masalah
2. Memahami konsep algoritma IDS dan BFS
3. Mempelajari *syntax* bahasa pemrograman Go untuk membuat algoritma dan JavaScript React untuk membuat tampilan aplikasi web
4. Mempelajari tata cara Web Scraping menggunakan bahasa Go
5. Membuat aplikasi web untuk menampilkan hasil pencarian
6. Memetakan permasalahan menjadi elemen - elemen BFS dan DFS
7. Mengimplementasikan algoritma BFS dan IDS pada pencarian rute pada peta persoalan
8. Menghubungkan antara algoritma (*backend)* dan tampilan (*frontend)*

Setelah melakukan dekomposisi permasalahan , beberapa hasil dekomposisi akan dijabarkan menjadi beberapa bagian sebagai berikut,

1. **Pengambilan *link* pada website Wikipedia**

   Pengambilan *link* akan dilakukan dengan menggunakan *scraping* pada web Wikipedia. Scraper akan mengambil tag <a> yang berada pada div dengan id mw-content-text dengan tag <a> yang memiliki attribute title selanjutnya scraper akan mengeluarkan hasil dari *link-link* yang ada pada halaman Wikipedia

2. **Penghubungan data antara *frontend* dan *backend***

   Frontend dan backend akan dihubungkan melalui *endpoint* yang dibuat pada GIN Server. *Frontend* akan melakukan *post request* untuk mengirimkan

data masukkan dan menerima hasil kalkulasi dari *backend*. Aksi tersebut akan di-*trigger* saat menekan tombol pencarian.

3. Penggunaan struktur data untuk BFS dan IDS

Terdapat perbedaan dalam struktur data yang digunakan dalam pencarian BFS dan IDS. Pada algoritma BFS akan digunakan struktur *queue* untuk melakukan pencarian. Sedangkan pada IDS akan digunakan konsep *stack.*

1. Langkah - Langkah Pemecahan dengan Algoritma BFS

Algoritma *Breadth First Search* (BFS) mengutamakan eksplorasi secara melebar. Solusi ini dibangun dengan menggunakan konsep graf dinamis yang digunakan untuk menyimpan rute yang memudahkan langkah pencarian selanjutnya.

1. Menerima input judul awal dan menjadikannya sebagai root dari graf.
2. Akan dilakukan scraping hyperlink dari laman Wikipedia dan menyimpannya sebagai *child node* dari *node* tersebut.
3. Node akan disimpan dalam data menggunakan struktur data queue (melakukan enqueue).
4. Node yang akan di-expand merupakan top of queue.
5. Saat node di-expand, node tersebut akan di keluarkan dari queue (melakukan proses dequeue)
6. Untuk setiap proses akan dilakukan proses yang sama (2-4) hingga halaman Wikipedia yang dikunjungi merupakan End Article
7. Hasil rute-rute yang ditemukan akan disimpan pada matriks yang akan digunakan untuk menampilkan hasil solusi.
8. Jika tidak ada rute yang ditemukan hingga batas waktu yang ditemukan, maka tidak ada solusi.

2. Langkah - Langkah Pemecahan dengan Algoritma IDS

Algoritma *Iterative Deepening Search* (IDS) menggunakan konsep DFS yang mengutamakan pencarian secara mendalam pada simpul yang sedang dikunjungi dengan peningkatan nilai kedalaman-cutoff, sampai solusi ditemukan. Pencarian DFS dilakukan secara *pre order* oleh sebab itu proses pencarian bisa digunakan dengan menggunakan stack atau rekursif. Untuk menghindari masalah memori yang disimpan, kami menggunakan metode rekursif yang masih memiliki konsep yang sama dalam melakukan pencarian secara DFS. Solusi ini dibangun dengan menggunakan konsep graf dinamis yang digunakan untuk menyimpan rute yang memudahkan langkah pencarian selanjutnya. Berikut adalah prosedurnya.

1. Menerima input judul awal dan menjadikannya sebagai root dari graf.
2. Proses pencarian akan dimulai dari kedalaman 0 terlebih dahulu.
3. Akan dilakukan scraping hyperlink dari laman Wikipedia dan menyimpannya sebagai child node dari node tersebut.
4. Dilakukan penelusuran secara *pre order* dengan menggunakan konsep stack yang disimulasikan dengan rekursif hingga ditemukan End Article serta menandai setiap node yang telah dikunjungi.
5. Jika pada kedalaman tersebut belum ditemukan, maka kedalaman pencarian akan bertambah satu dan proses akan pencarian akan diulang kembali hingga ditemukan End Article yang dicari.
6. Hasil rute-rute yang ditemukan akan disimpan pada matriks yang akan digunakan untuk menampilkan hasil solusi.
7. Jika tidak ada rute yang ditemukan hingga batas waktu yang ditemukan, maka tidak ada solusi.

3.2. Proses Pemetaan Masalah

3.3.1. Algoritma Iterative Deepening Search (IDS)

Pencarian menggunakan algoritma IDS akan dilakukan sebagai berikut:

1. Menerapkan algoritma IDS pada setiap pencarian artikel
2. Akan dilakukan scraping hyperlink dari laman Wikipedia
3. Masukkan artikel - artikel yang ada pada halaman Wikipedia ke dalam *stack* yang dimiliki (*push)*
4. Pencarian dimulai dari *depth 0* lalu akan bertambah jika artikel akhir tidak ditemukan
5. Mulai dari *top* of *stack* untuk memeriksa, lakukan scraping kembali dan masukkan (*push)* link yang didapatkan pada *stack*
6. *Node* yang akan di *expand* selalu merupakan *top of stack*
7. Saat *node* di-*expand,* node tersebut akan di keluarkan dari *stack* (melakukan proses *pop)*
8. Jika *depth* yang ditentukan sudah dicapai, namun artikel akhir belum ditemui makan *depth* akan bertambah dan pencarian akan dimulai dari *root node*
9. Pencarian akan dilakukan sampai *node* artikel tujuan ditemukan

Pemanggilan stack pada program IDS disimulasikan dengan pemanggilan fungsi rekursif untuk menghindari masalah memory.

Nodes	Judul artikel-artikel yang ada di Wikipedia
Edge	Merupakan yang menghubungkan antara *link* artikel Wikipedia, Jika pada artikel A terdapat hyperlink untuk menuju artikel B, maka dapat disimpulkan bahwa terdapat *edge* untuk A dan B
Goal State	Mencari rute terpendek antara *Start Article* dan *End Article*
Path Cost	Jumlah artikel yang ditempuh untuk mencapai tujuan.

Untuk pemetaan saat mencari solusi banyak, algoritma IDS tidak berhenti saat menemukan satu solusi saja. Algoritma akan terus mencari alternatif rute lain sampai solusi-solusi ditemukan. Apabila alternatif yang ditemukan kedalamannya lebih panjang daripada alternatif lainnya, maka program akan langsung keluar. Artinya

solusi tersebut bukan lagi solusi rute terbaik, program hanya akan mengeluarkan rute-rute yang terbaik.

3.3.2. Algoritma Breadth First Search (BFS)

Pencarian menggunakan algoritma BFS akan dilakukan sebagai berikut :

1. Menerapkan algoritma BFS pada setiap pencarian artikel
2. Akan dilakukan scraping hyperlink dari laman Wikipedia
3. Hasil dari scraping akan disimpan dalam data menggunakan struktur data *queue* (melakukan *enqueue*)
4. *Node* yang akan di-*expand* merupakan *top of queue.*
5. Saat *node* di-*expand,* node tersebut akan di keluarkan dari *queue* (melakukan proses *dequeue)*
6. Untuk setiap proses akan dilakukan proses yang sama (2-4) hingga halaman Wikipedia yang dikunjungi merupakan *End Article*

Nodes	Judul artikel-artikel yang ada di Wikipedia
Edge	Merupakan yang menghubungkan antara *link* artikel Wikipedia, Jika pada artikel A terdapat hyperlink untuk menuju artikel B, maka dapat disimpulkan bahwa terdapat *edge* untuk A dan B
Goal State	Mencari rute terpendek antara *Start Article* dan *End Article*
Path Cost	Jumlah artikel yang ditempuh untuk mencapai tujuan.

Untuk pemetaan saat mencari solusi banyak, algoritma BFS tidak berhenti saat menemukan satu solusi saja. Algoritma akan terus mencari alternatif rute lain sampai solusi-solusi ditemukan. Apabila alternatif yang ditemukan kedalamannya lebih panjang daripada alternatif lainnya, maka program akan langsung keluar. Artinya solusi tersebut bukan lagi solusi rute terbaik, program hanya akan mengeluarkan rute-rute yang terbaik. Tahapan ini mirip seperti algoritma IDS untuk banyak solusi

3.3. Fitur fungsional dan arsitektur aplikasi web yang dibangun

Aplikasi Web yang dibangun menggunakan *package manager* npm dan NextJS untuk mempermudah pengembangan. Next.js adalah sebuah framework React yang dirancang untuk memudahkan pengembangan aplikasi web dengan menyediakan fitur-fitur seperti Server-Side Rendering (SSR), Static Site Generation (SSG), routing berbasis file system, dan banyak lagi. Dikembangkan oleh tim Vercel, Next.js memungkinkan pengembang untuk membuat aplikasi web yang responsif, efisien, dan mudah diatur. Bahasa yang digunakan untuk pengembangan merupakan JavaScript React (JSX).

Untuk pengembangan Backend, pada sistem ini akan digunakan Framework GIN yang menggunakan bahasa Go. Framework GIN digunakan untuk mempermudah proses pengembangan. GIN memiliki validasi JSON bawaan yang akan mempermudah proses pengambilan dan pengiriman data kepada frontend. Selain itu, GIN menyediakan utilitas untuk pengujian handler HTTP, menyederhanakan proses pengujian untuk aplikasi web.

3.3.1. Form Input

3.3.1.1. Toggle Switch

Gambar 3.3.1.1.1 Toggle

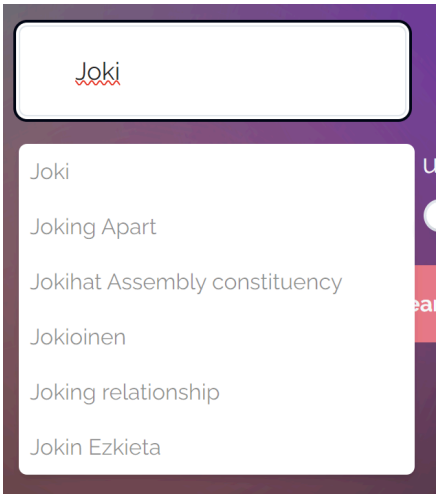
Toggle Switch berfungsi untuk menerima masukkan dari pengguna. Pengguna akan memilih algoritma IDS ataupun BFS untuk menelusuri Wikipedia. Toggle switch akan memberikan nilai boolean untuk menentukan algoritma yang dipakai. Jika pengguna memilih untuk menggunakan algoritma IDS maka Toggle akan mengembalikan nilai ‘false’, jika pengguna akan memilih algoritma BFS makan Toggle akan mengembalikan nilai ‘true’.

3.3.1.2. Input Box

Gambar 3.3.1.2.1 Input

Pengguna dapat memasukkan nama artikel Wikipedia yang akan ditelusuri. Pengguna dapat menukar *Start Article* dan *End Link* dengan menekan tombol yang di antara input box.

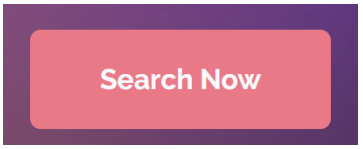
3.3.1.3. Suggestion



Gambar 3.3.1.3.1 Suggestion

Website akan memberikan sugesti nama Artikel berdasarkan masukkan oleh pengguna. Website akan menggunakan API dari Wikipedia untuk memberikan rekomendasi judul artikel.

3.3.1.4. Tombol Pencarian



Gambar 3.3.1.4.1 Tombol Pencarian

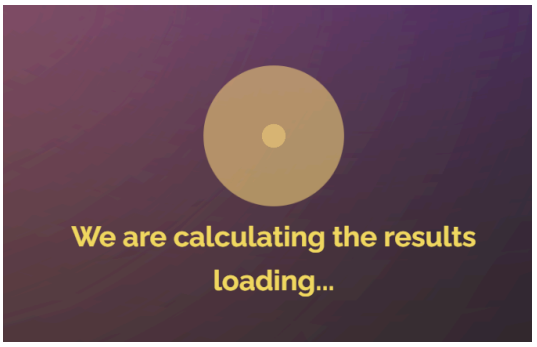
Ketika pengguna menekan tombol pencarian, website akan mengirimkan data *Start Article*, *End Article*, dan data dari Toggle kepada server GIN (GO) yang kemudian akan memproses dan melakukan pencarian rute. Setelah pencarian selesai maka, hasil pencarian akan kembali dikirimkan ke *client* (website)

3.3.2. Visualisasi Graf dan Hasil

Gambar 3.3.2.1 Visualisasi graf dan hasil

Hasil divisualisasikan menggunakan teks sesuai dengan data yang diterima. Rute dapat divisualisasikan menggunakan DAG (Directed Acyclic Graph). Server akan mengembalikan artikel yang menjadi rute dalam penelusuran. Artikel-artikel tersebut akan dipetakan menjadi Node dan Link dan ditampilkan dalam bentuk graf. Node-node pada graf merupakan judul dari artikel-artikel dan dapat ditekan untuk mengunjungi laman artikel tersebut.

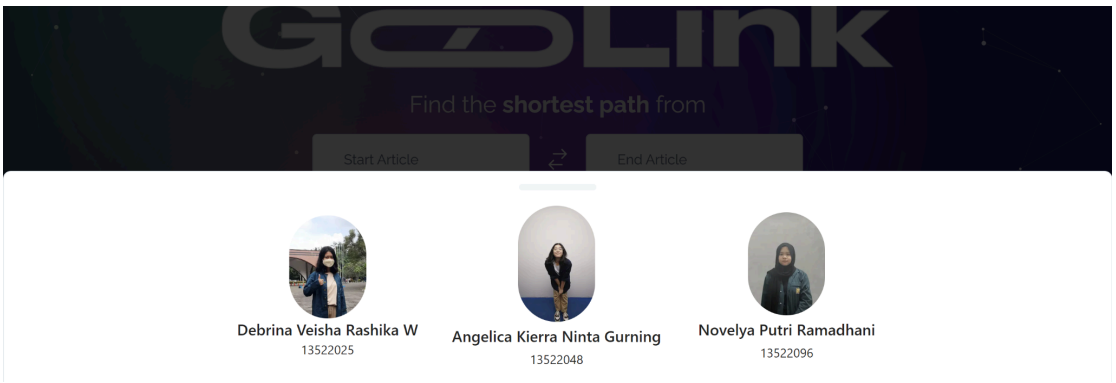
3.3.3. Loading Screen



Gambar 3.3.3.1 Loading Screen

Pada saat proses pencarian, website akan menampilkan *loading screen*. Apabila website sudah menerima data dari hasil pencarian, maka *loading screen* akan dimatikan.

3.3.4. Authors



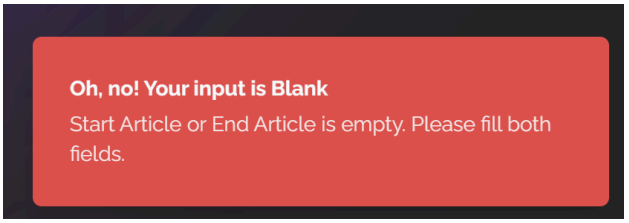
Gambar 3.3.4.1 Drawer Authors

Merupakan komponen *drawer* yang berisi foto, nama, serta nim anggota kelompok

3.3.5. Docker

Baik *Front End* maupun *Back End* dapat dijalankan dalam satu Docker container. Terdapat dua *docker image* yang dibuat yaitu untuk *Front End* dan juga *Back End*, yang kemudian dikemas dalam satu *container*.

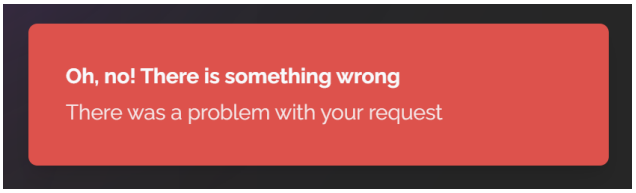
3.3.6. Toast input kosong



Gambar 3.3.6.1 Error input kosong

Program akan memeriksa apakah kedua *input box* sudah memiliki nilai atau belum, jika salah satu atau keduanya belum terisi, maka website akan memberikan pesan error.

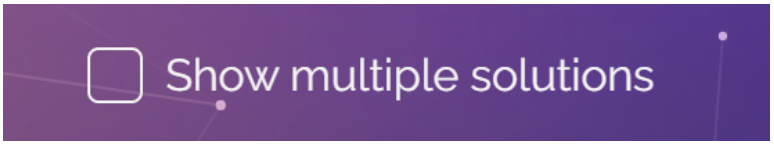
3.3.7. Toast gagal menerima data



Gambar 3.3.7.1 Error menerima data

Website akan menampilkan sebuah pesan error apabila gagal untuk mengirim atau menerima data dari *server* (GIN).

3.3.8. Checkbox untuk solusi banyak



Gambar 3.3.8.1 Checkbox solusi banyak

Terdapat pilihan untuk menampilkan solusi banyak yang direpresentasikan dengan dengan komponen *checkbox*. Apabila *checkbox* ditekan maka, server akan menerima boolean dan memanggil fungsi IDS/BFS untuk banyak solusi.

3.3.9. Link ke Repository Github

Gambar 3.3.9.1 Hyperlink github

Terdapat sebuah hyperlink yang akan membuka *tab* baru saat ditekan. Hyperlink tersebut akan membawa ke laman Repository Github tugas besar ini.

3.4. Contoh Ilustrasi Kasus

Berikut merupakan contoh kasus dari penggunaan website Go Link Wiki Race. Pada saat pertama kali *website* ditampilkan, pengguna dapat langsung memasukan artikel yang ingin dicari rutenya, atau menekan bagian ‘Authors’, maupun ‘Github’.

1. Jika menekan ‘Authors’ sebuah komponen *drawer* akan muncul dan menampilkan nama, NIM, dan foto anggota kelompok
2. Jika menekan ‘Github’ akan dibawa ke halaman baru, yaitu *repository* Github Tugas Besar Ini
3. Jika memilih untuk langsung melakukan pencarian maka, ada beberapa langkah yang harus dilakukan.
    a. Setelah pengguna memasukkan *input* ‘Start Article’ dan ‘End Article’ pada *input box*, pengguna dapat memilih untuk melakukan pencarian menggunakan algoritma BFS ataupun IDS menggunakan *toggle switch* yang tersedia, setelah itu pengguna dapat memilih untuk mengeluarkan solusi banyak atau tidak dengan menekan *checkbox*.
    b. Setelah pengguna memasukkan semua *input*, pengguna dapat menekan tombol *‘Search Now’*. Saat tombol ditekan, bagian *frontend* akan berkomunikasi ke *backend* dengan melakukan *fetching* dari API dan mengirimkan data input berupa , *start article, end article, input toggle, checkbox*.
    c. Bagian *backend* akan menerima data dari *frontend* dan melakukan pemrosesan data, yaitu mencari rute terbaik menggunakan algoritma IDS maupun BFS
    d. Setelah pencarian rute ditemukan, *backend* akan mengirimkan data hasil pencarian berupa, jumlah artikel yang dikunjungi, jumlah artikel yang dicari, waktu eksekusi, dan juga rute terbaik
    e. *Frontend* akan menerima data tersebut dan menampilkannya dalam bentuk *graf* dan juga teks. Tiap simpul pada graf dapat ditekan dan akan membawa pengguna ke halaman artikel Wikipedia tersebut.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi teknis program

4.1.1. Struktur data

1. Simpul (Node)

Struktur data simpul digunakan untuk membuat graf yang digambarkan dalam bentuk pohon. Struktur ini digunakan dalam melakukan proses pencarian solusi. Simpul memiliki atribut *value* yang merupakan judul wikipedia, *depth* yang digunakan untuk mencatat kedalaman graf, *parent* untuk menyimpan simpul induk, dan *children* untuk menyimpan simpul anak. Selain itu terdapat status pada simpul yakni, *Visited* yang berarti simpul tersebut telah dikunjungi dan diperiksa serta *MainRoute* yang menandakan bahwa node tersebut adalah tujuan yang dicari.

2. List

Struktur data list digunakan untuk menyimpan suatu elemen data yang digunakan dalam beberapa bagian program, sebagai berikut:

1) List of *string*

   Larik dengan tipe *string* digunakan untuk menyimpan rute yang dilewati dari *root* menuju simpul tujuan. Larik ini akan digunakan dalam melakukan visualisasi hasil pencarian rute.

2) List of Tree

   Larik dengan tipe Tree digunakan dalam menyimpan simpul-simpul anak yang dimiliki suatu induk. Simpul anak sendiri merupakan judul-judul link yang terdapat pada halaman wikipedia dari hasil *scraping*. Larik tipe ini juga digunakan dalam menyimpan simpul dalam melakukan proses pencarian menggunakan BFS.

3) List of Link

   Larik dengan tipe Link digunakan dalam menyimpan hasil *scraping* dari suatu halaman wikipedia. Hasil scraping akan disimpan dalam struktur Link yang memiliki atribut judul beserta Urlnya.

3. **Matriks**

Struktur data matriks digunakan untuk menyimpan list-list hasil dari pencarian menggunakan BFS dan IDS. Semua rute yang digunakan untuk mencapai solusi yang ditemukan akan disimpan pada matriks. Matriks akan digunakan untuk menampilkan solusi baik itu *single solution* ataupun *multiple solution*.

4. **Queue**

Struktur data Queue digunakan untuk menyimpan rute yang akan dikunjungi pada graf pada proses pencarian *Breadth First Search* (BFS) . Sehingga elemen akan tersimpan dalam FIFO (first-in first-out) yang memudahkan skema pengecekan semua rute yang dilewati selama pencarian.

4.1.2. Fungsi dan prosedur

Pada folder src/backend terdapat beberapa *package* yang digunakan dalam membangun program. Sebagai berikut:

1. **tree**

tree.go	
Struct	type Tree struct { ○ String value {menyimpan judul wikipedia} ○ Int Depth {menyimpan kedalaman pohon} ○ Tree Parent {menyimpan node induk} ○ Tree[] Children {menyimpan anak} ○ Bool Visited {status apakah sudah diperiksa atau belum} ○ Bool MainRoute {status apakah merupakan node tujuan}
Method	○ func NewNode(value string) -> *Tree { kontruktor dalam membuat tree } ○ func (n *Tree) NewNodeLink(link []scraper.Link) { method untuk membuat node dari list link } ○ func (n *Tree) AddChild(child *Tree) { method untuk menambah anak pada node } ○ func (n *Tree) GetDepth() -> int

	{ fungsi untuk mendapat nilai kedalaman dari node} ○ func (n *Tree) AddVisitedNode() { method untuk menandai node telah diperiksa } ○ func (n *Tree) AddMainRoute() { method untuk menandai bahwa node adalah rute tujuan } ○ func (n *Tree) UndoMainRoute() { method untuk menghapus node dari rute tujuan } ○ func IsGoalFound(s1 string, s2 string) -> bool { fungsi untuk menentukan apakah goal ditemukan / tidak } ○ func GoalRoute (n *Tree) -> [] string { fungsi untuk menentukan rute dari root ke goal }

2. golink

golink.go	

Struct	type GoLinkStats struct {
	Route []\[]string
	LinksTraversed int
	LinksChecked int
	Runtime time.Duration
	}

Atribut	○ string[][] Route {menyimpan list rute pencarian} ○ int LinksTraversed {menyimpan jumlah artikel yang telah dilalui} ○ int LinksChecked {menyimpan jumlah artikel yang telah diperiksa} ○ time.Duration Runtime {menyimpan durasi runtime pencarian}
Method	○ func NewGoLinkStats() *GoLinkStats {inisialisasi} ○ func (g *GoLinkStats) AddTraversed()

	{menambah jumlah artikel yang telah dilalui} ○ func (g *GoLinkStats) AddChecked() {menambah jumlah artikel yang telah diperiksa} ○ func (g *GoLinkStats) AddRoute(rute []string) {menambah rute ke array} ○ func (g *GoLinkStats) SetRuntime(duration time.Duration) {mengatur runtime pencarian} ○ func SameList(listOfLists [][]string, checkList []string) -> bool {memeriksa apakah list sudah ada di rute} ○ func (g *GoLinkStats) PrintStats() {mencetak statistik pencarian}

3. Scraper

scraper.go	

Struct	type Tree struct {
	○ String name {menyimpan judul wikipedia}
	○ String Url {menyimpan link wikipedia}
	}

Method	○ func PrintLink(links []Link) { method untuk print semua link yang didapat } ○ func StringToWikiUrl(name string) -> string { method untuk mengubah string menjadi url } ○ func contains(arr []string, str string) -> bool { memeriksa apakah link terdapat judul yang tidak perlu } ○ func Scraper(linkName string) ([]Link, error) { method utama untuk melakukan scraper pada web}

4. BFS

bfs.go	
Method	○ func Bfsfunc(value string, goal string, multisol bool) -> *golink.GoLinkStats { fungsi utama untuk melakukan pencarian dengan BFS } ○ func SearhForGoalBfsMT(n *tree.Tree, goal string, stats *golink.GoLinkStats) -> bool { fungsi untuk mencari judul tujuan menggunakan BFS } ○ func SearchForGoalBfsMTMS(root *tree.Tree, goal string, stats *golink.GoLinkStats) -> bool { fungsi untuk mencari multisolution dari judul tujuan menggunakan BFS } ○ func PrintTreeBfs(n *tree.Tree) { method untuk print pohon secara bfs }

5. IDS

ids.go	
Atribut	○ Bool found {menandai goal sudah ditemukan} ○ Bool single {menandai dilakukan pencarian satu solusi}
Method	○ func Idsfunc(value string, goal string, multisol bool) -> *golink.GoLinkStats { fungsi utama untuk melakukan pencarian dengan IDS} ○ func SearchForGoal(ctx context.Context, node *tree.Tree, goal string, currentLevel int, stats *golink.GoLinkStats, onesolution bool) { fungsi untuk mencari melakukan pencarian tujuan secara IDS} ○ func PrintTreeIds(n *tree.Tree) { method untuk print pohon secara dfs}

6. Server

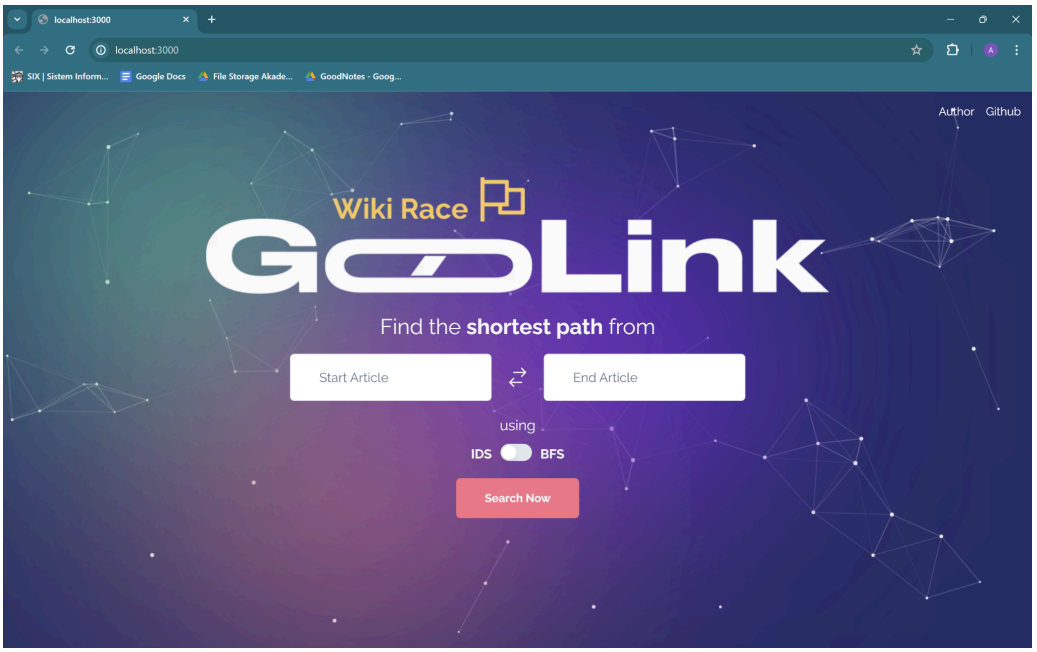
server.go	

struct	type InputData struct {
	○ String startlink {Link start }
	○ String endlink {Link tujuan yang dicari}
	○ Bool UseToggle {Menandai untuk BFS / IDS}
	}
	type ResponseData struct {
	○ String[][] Articles {Rute yang ditemukan}
	○ Int ArticlesVisited {Artikel yang dikunjungi}
	○ Int AticlesSearched {Link tujuan yang dilewati}
	○ Int64 TimeNeeded {Waktu yang dibutuhkan}
	}

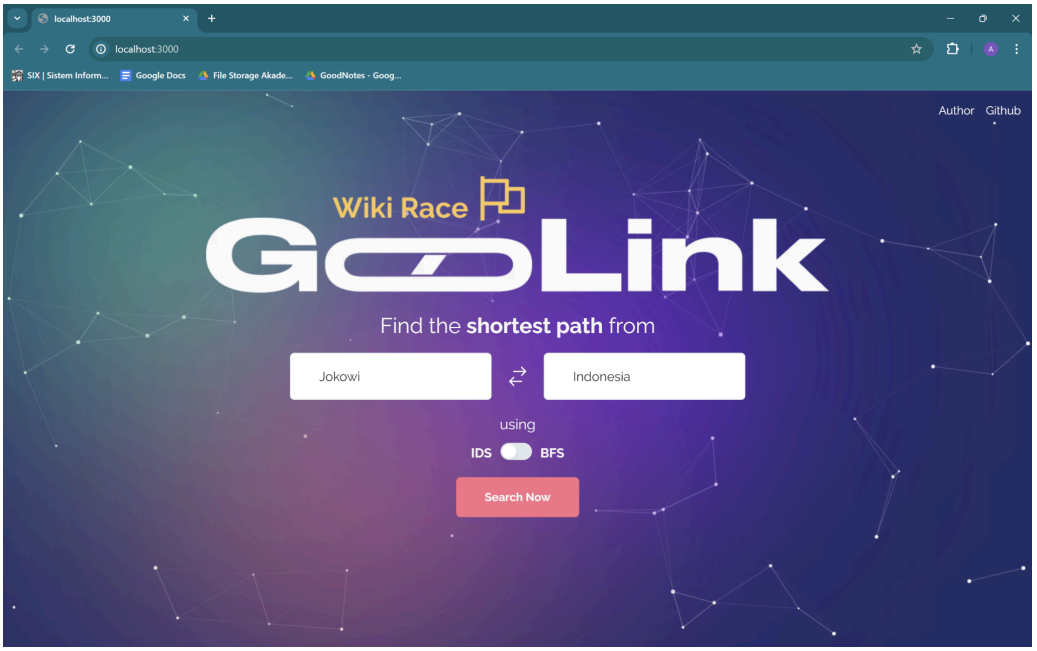
Method	○ func main() { method utama dalam menerima masukan pengguna, memproses nya, dan mengirimkan hasil kepada server}

4.2. Penjelasan Tata Cara Penggunaan Program

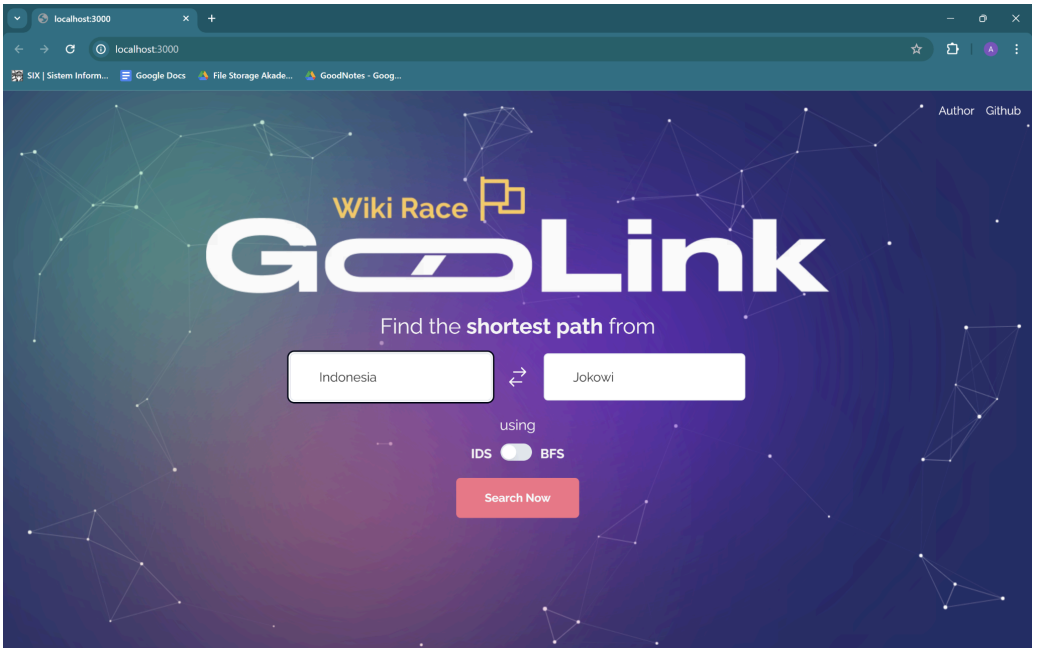
4.2.1. Interface program



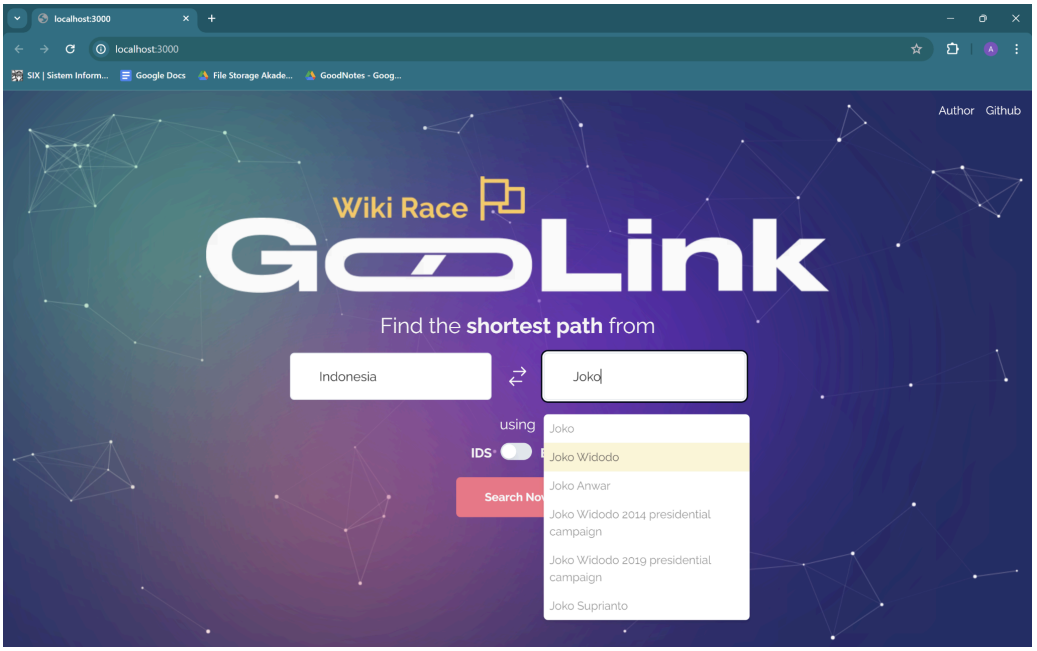
Gambar 4.2.1.1 Tampilan awal



Gambar 4.2.1.2 Tampilan menerima masukan

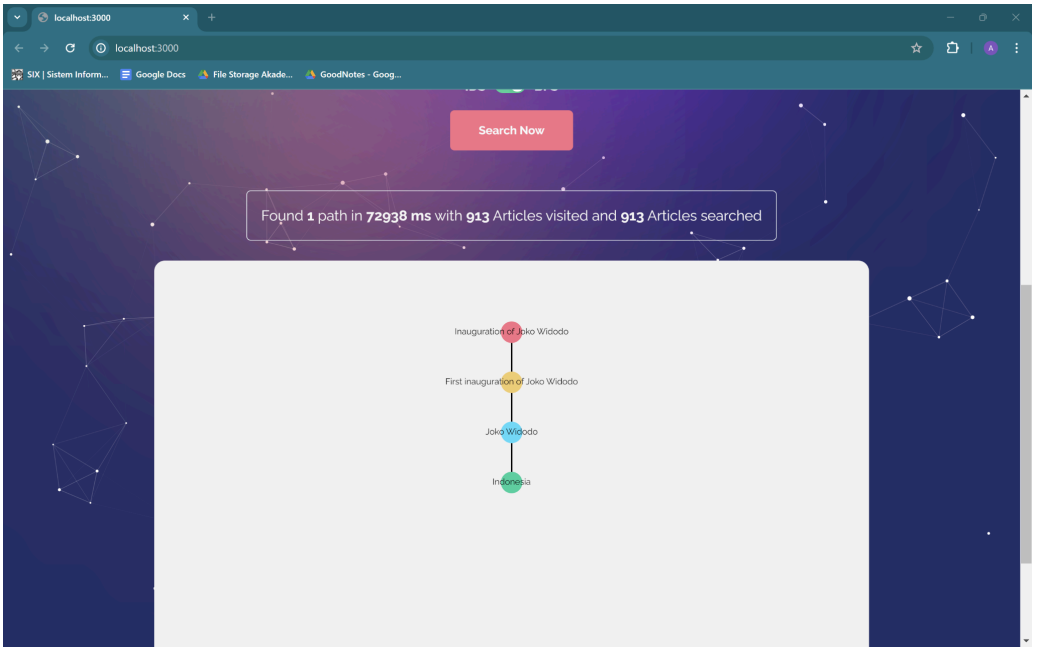


Gambar 4.2.1.3 Tampilan penukaran input

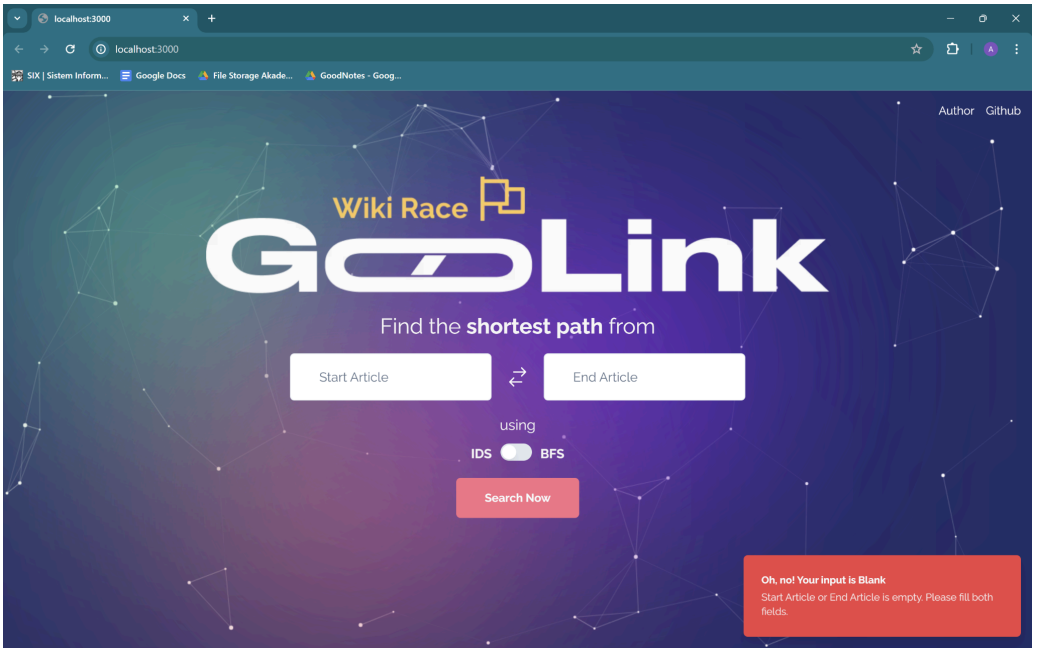


Gambar 4.2.1.4 Tampilan rekomendasi input

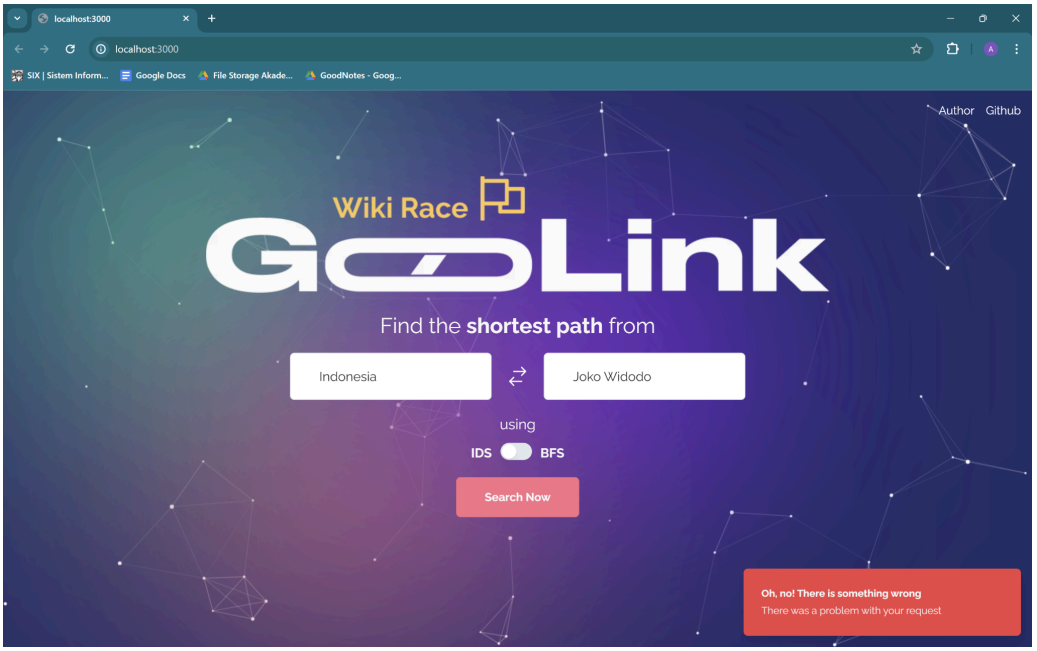
Gambar 4.2.1.5 Tampilan loading screen



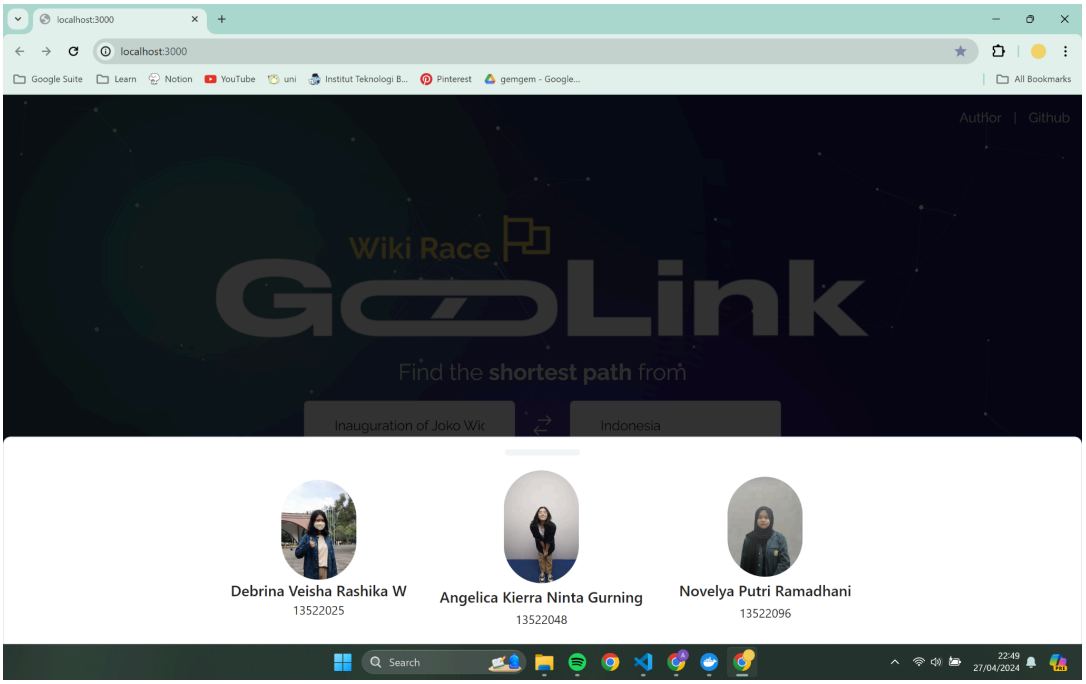
Gambar 4.2.1.6 Tampilan visualisasi hasil



Gambar 4.2.1.7 Tampilan input kosong



Gambar 4.2.1.8 Tampilan gagal mengambil data



Gambar 4.2.1.9 Tampilan authors

4.2.2. Tata Cara Penggunaan Program

Program dapat dijalankan pada sistem operasi berbasis linux maupun windows. Langkah pertama yang harus dilakukan adalah untuk melakukan *clone* dari repository github tugas besar ini : https://github.com/novelxv/Tubes2_GoLink Program dapat dijalankan dengan menggunakan Docker ataupun tanpa Docker.

Berikut merupakan tata cara menjalankan program dengan menggunakan Docker :

1. Pastikan ada aplikasi Docker sudah ter-*install* pada komputer.
2. Buka *file* tugas besar ini
3. Pindah ke directory 'src'
4. Pada terminal, masukkan perintah `'docker compose build'`, untuk membuat sebuah docker image baru
5. Jika program sudah selesai proses 'build', masukkan perintah `'docker compose up'` pada terminal untuk menjalankan aplikasi web

6. Setelah selesai proses *compose up*, klik *link* yang telah disediakan dan akan diarahkan pada '`http://localhost:3000/`'
7. Jika aplikasi web sudah berjalan, pencarian IDS dan BFS sudah bisa digunakan
8. Untuk mematikan aplikasi, masukkan perintah '`docker compose down`' untuk menghilangkan *container Docker*

Berikut merupakan tata cara menjalankan program dengan **tanpa** menggunakan Docker :

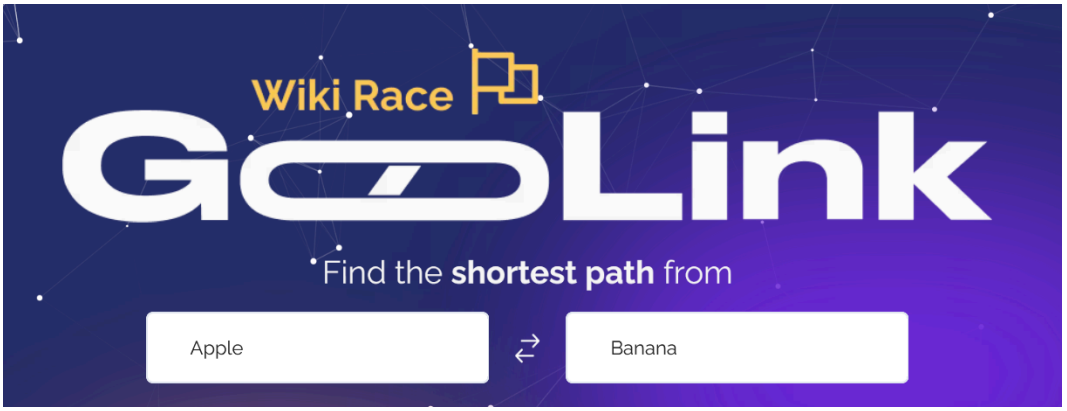
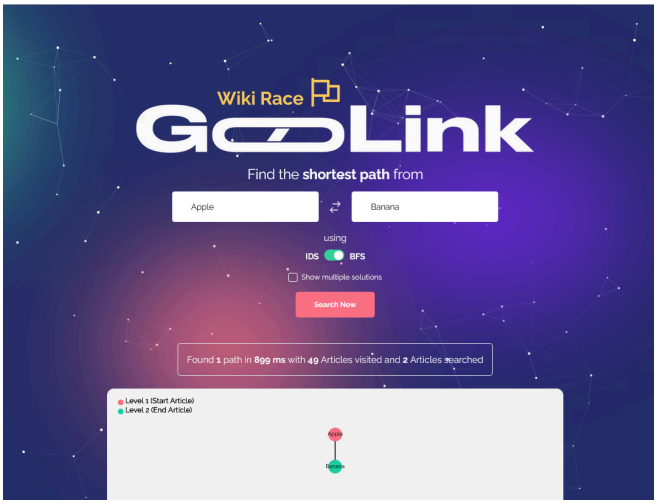
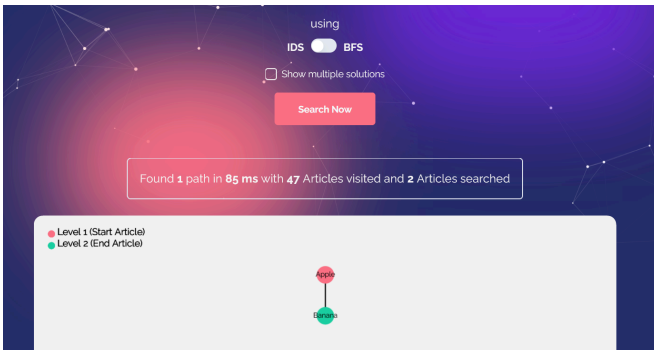
1. Pastikan ada aplikasi Docker sudah ter-*install* pada komputer.
2. Buka *file* tugas besar ini
3. Pindah ke directory 'src'
4. Untuk menjalankan bagian *frontend*, pindah ke directory 'frontend'
5. Pastikan *npm* sudah terinstall, lalu buka terminal baru dan masukkan perintah '`npm run dev`'
6. Setelah selesai proses kompilasi, klik *link* yang telah disediakan dan akan diarahkan pada '`http://localhost:3000/`'
7. Untuk menjalankan bagian *backend*, pindah ke directory 'backend
8. Buka terminal dan jalankan perintah '`go run server.go`'

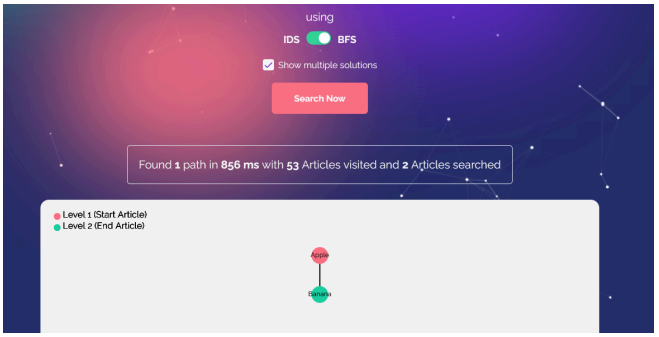
Berikut merupakan tata cara untuk menggunakan website:

1. Masukkan judul artikel Awal dan Akhir , pada *input box* yang tersedia
2. Pilih mode pencarian IDS atau BFS dengan menggunakan *toggle switch* yang tersedia
3. Tekan tombol pencarian 'Search Now' dan tunggu hasil ditampilkan

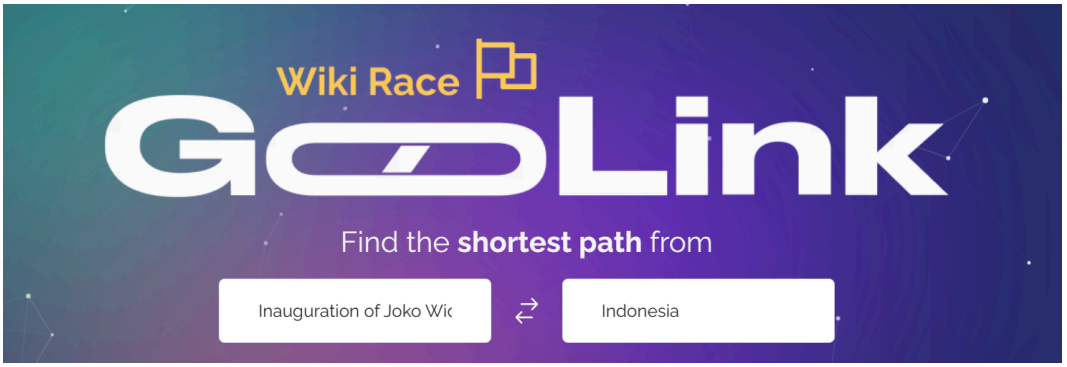
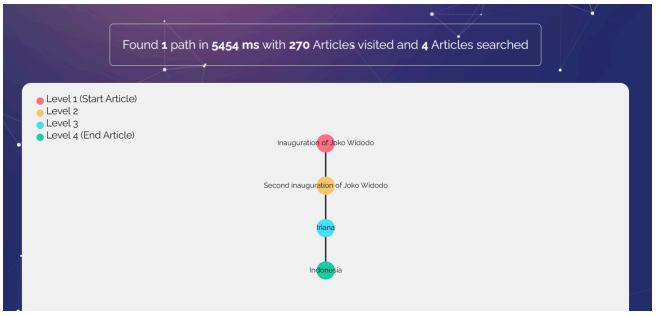
4.3. Hasil pengujian

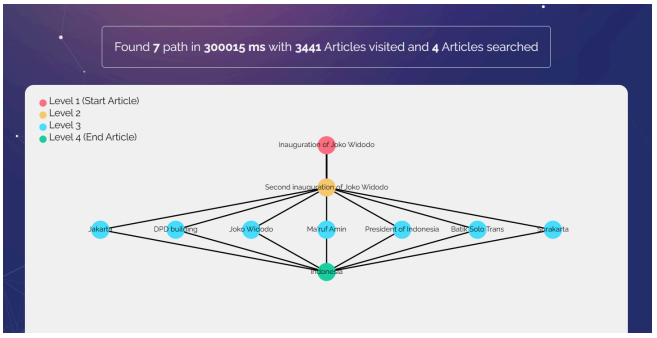
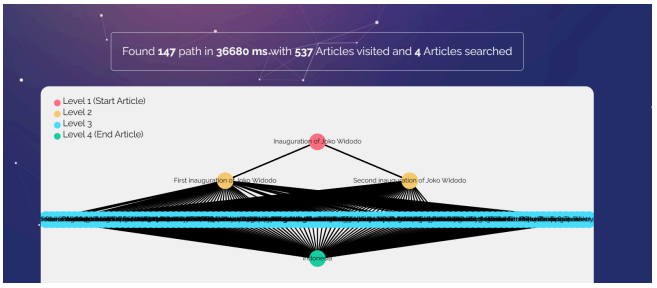
1. Test Case 1

Input	
	
Algoritma	**Hasil**
BFS	
IDS	

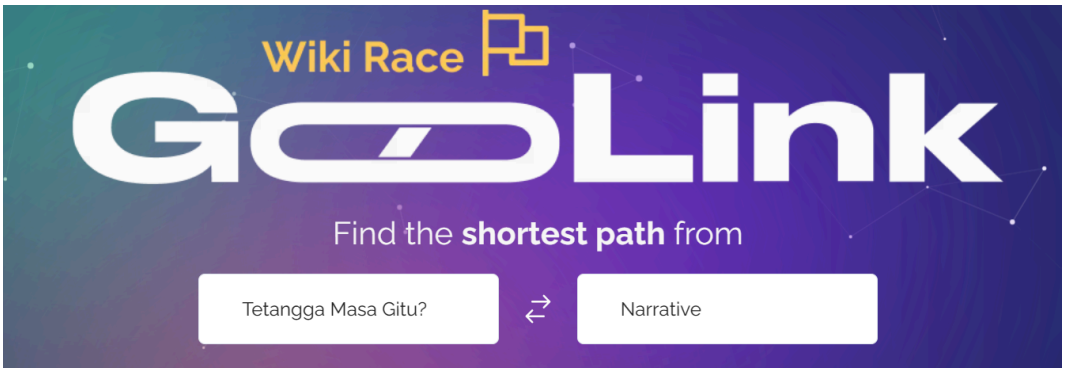
BFS Multiple Solution	
IDS Multiple Solution	

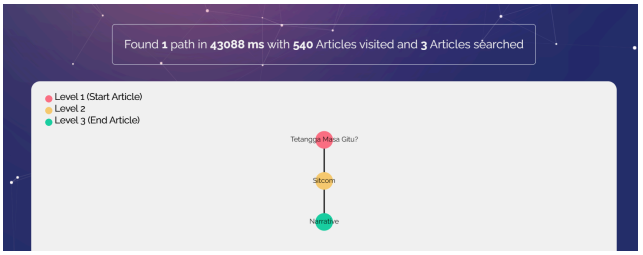
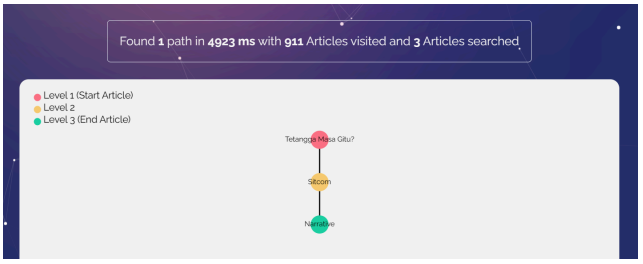
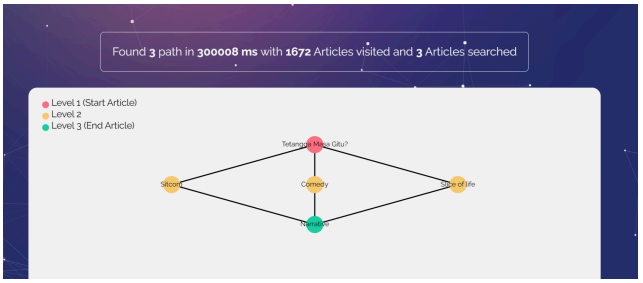
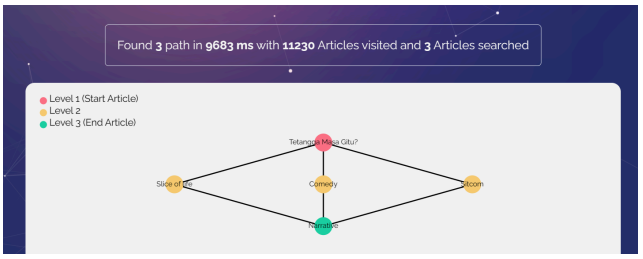
2. Test Case 2

Input	
	
Algoritma	**Hasil**
BFS	

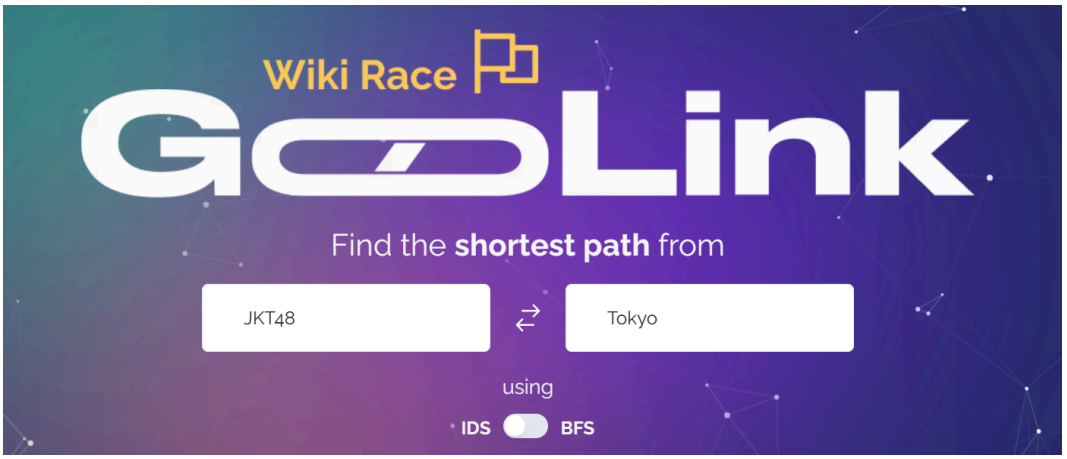
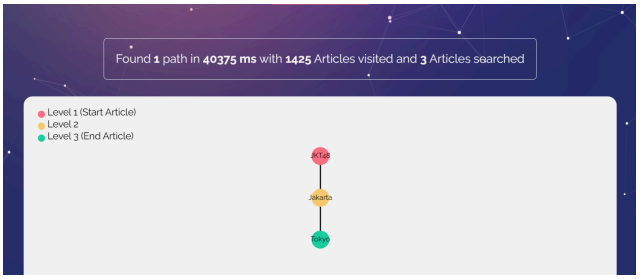
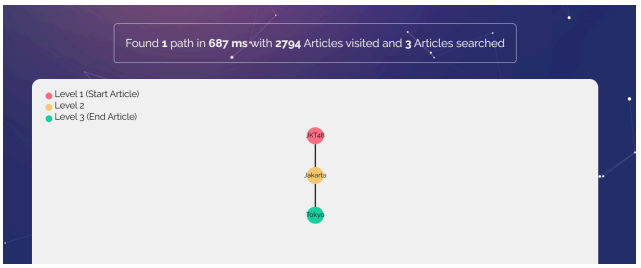
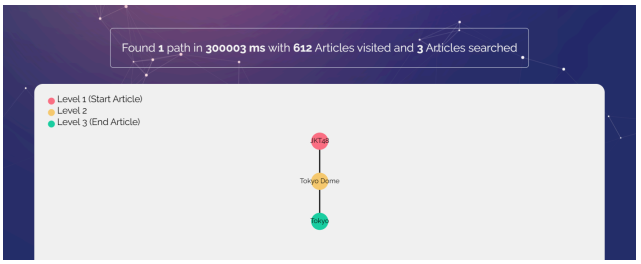
IDS	
BFS Multiple Solution	
IDS Multiple Solution	

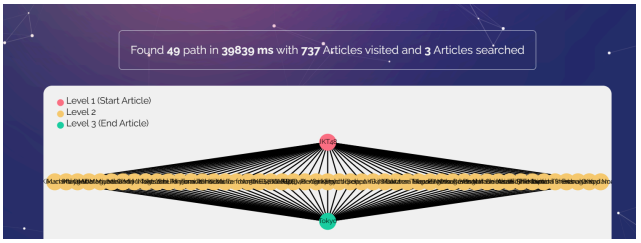
3. Test Case 3

Input	
	
Algoritma	**Hasil**

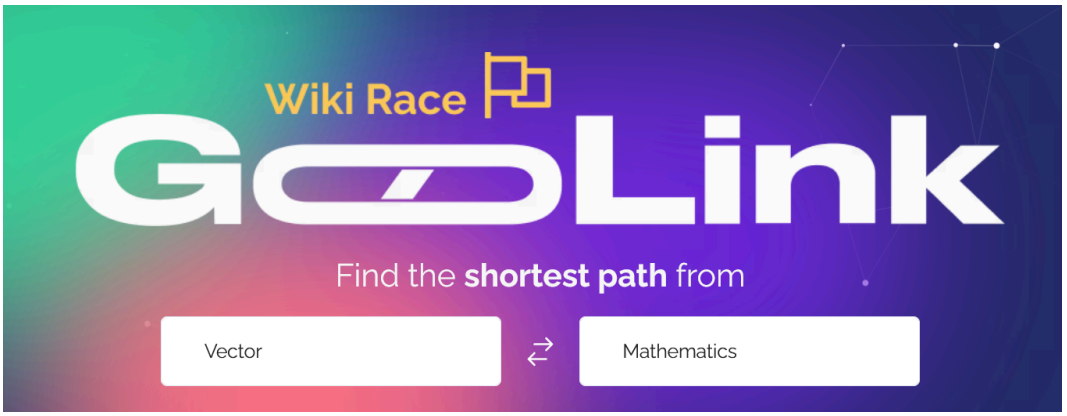
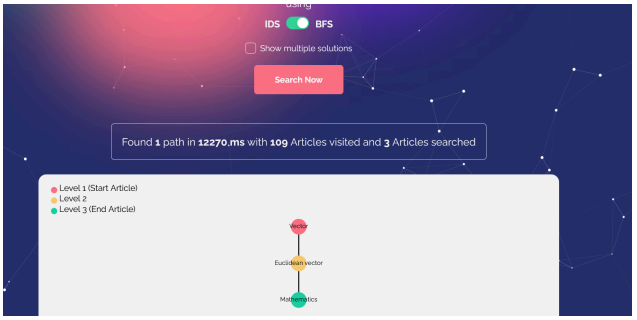
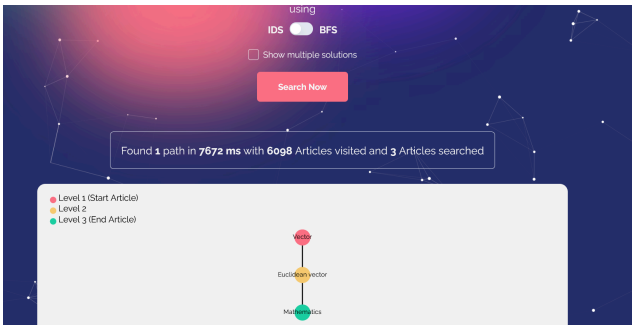
BFS	
IDS	
BFS Multiple Solution	
IDS Multiple Solution	

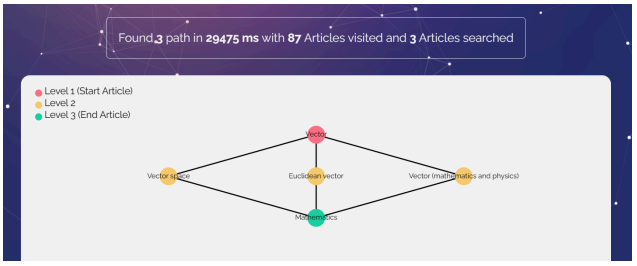
4. Test Case 4

Input	
	
Algoritma	**Hasil**
BFS	
IDS	
BFS Multiple Solution	

IDS Multiple Solution	

5. Test Case 5

Input	
	
Algoritma	**Hasil**
BFS	
IDS	

BFS Multiple Solution	
IDS Multiple Solution	

4.4. Analisis hasil pengujian

Dari hasil pengujian yang sudah dilakukan, algoritma IDS dan BFS sudah menjalankan tugasnya dengan baik dan memberikan hasil yang sesuai dalam mencari semua rute untuk mencapai artikel tujuan. Algoritma BFS memiliki kompleksitas waktu O(bd) dimana b adalah percabangan maksimum dan d adalah kedalaman dari simpul tujuan. Sedangkan, algoritma IDS memiliki kompleksitas waktu O(b^d) dimana b adalah percabangan maksimum dan d adalah kedalaman simpul tujuan.

Dari desain solusi, diperoleh bahwa pencarian menggunakan IDS memiliki solusi yang lebih sangkil dari segi kompleksitas ruang terutama jika simpul tujuan berada pada kedalaman yang dalam. IDS sendiri merupakan kombinasi dari DFS dan BFS yang membatasi kedalaman pencarian pada setiap iterasi. IDS memungkinkan pencarian dalam kedalaman bertahap, sehingga menggabungkan keunggulan BFS dalam menemukan jalur terpendek dengan efisiensi memori DFS.

BFS sendiri efektif untuk menemukan jalur terpendek jika kedalaman simpul tujuan yang dicari pendek (tidak dalam). Jika dibandingkan IDS, saat melakukan pencarian pada kedalaman yang rendah simpul-simpul akan dibangkitkan kembali sehingga terdapat simpul redundant karena bekerja secara iteratif. BFS menjamin menemukan jalur terpendek karena menelusuri level per level dari simpul awal hingga simpul tujuan. Namun, BFS memerlukan lebih banyak ruang memori untuk menyimpan simpul yang

belum dieksplorasi. Oleh karena itu, BFS lebih tidak sangkil adalah karena harus menyimpan rute dalam queue untuk mencari simpul yang akan dianalisis berikutnya.

Dari hasil uji coba tersebut, dapat disimpulkan bahwa waktu yang dibutuhkan untuk IDS (*Iterative Deepening Search*) lebih cepat dibandingkan dengan BFS (*Breadth-First Search*). Perbedaan ini tidak hanya disebabkan oleh efisiensi memori, tetapi juga karena penggunaan go routine yang berbeda untuk melakukan pencarian secara paralel. Semakin banyak node yang harus dicari, waktu yang dibutuhkan juga akan semakin banyak. Proses mencari secara *multisolution* (banyak solusi) membutuhkan waktu lebih lama daripada pencarian *single solution* (solusi tunggal), karena pencarian multisolution akan terus berlanjut hingga semua solusi ditemukan atau batas pencarian tercapai, sementara pencarian single solution akan berhenti segera setelah solusi pertama ditemukan.

Dengan demikian, IDS memiliki keunggulan dalam beberapa aspek dibandingkan dengan BFS dalam hal kecepatan pencarian, terutama dalam konteks pencarian solusi pada graf yang kompleks dan besar.

BAB V
KESIMPULAN, SARAN

5.1. Kesimpulan

Permainan WikiRace merupakan permainan untuk mencari rute dari masukan judul artikel awal ke artikel akhir dengan membentuk hubungan seperti Graf. Dalam tugas besar ini dilakukan pencarian rute dengan menggunakan algoritma *Breadth First Search* (BFS) dan *Iterative Deepening Search* (IDS). Keduanya menjamin ditemukannya suatu solusi karena sama-sama menelusuri setiap node pada tiap levelnya. Namun, algoritma BFS membutuhkan lebih banyak memori karena harus menyimpan semua semua node pada graf sedangkan IDS tidak melakukan penyimpanan memori. Semakin banyak kedalaman yang dibutuhkan maka algoritma IDS akan lebih efektif, sedangkan jika jumlah kedalaman yang dibutuhkan tidak terlalu dalam maka BFS akan memberikan hasil yang lebih optimal.

5.2. Saran

Terdapat beberapa saran yang dapat dilakukan untuk pengembangan lebih lanjut, antara lain:

1. Aplikasi Web dibuat lebih responsive sehingga meningkatkan pengalaman pengguna
2. Membuat algoritma menjadi lebih efisien agar bisa mempercepat waktu pencarian program.
3. Selain itu, batasan spek yang berubah-ubah juga membingungkan sehingga kedepannya bisa menjadi pertimbangan asisten untuk membuat batasan yang jelas di awal.
4. Dari pada fokus dengan IDS dan BFS, kami lebih bingung dengan cara mempercepat dan banyak drama scraping, mulai diblokir dari wikipedia berkali-kali, out of memory, yang awalnya tidak boleh menyimpan di database menjadi boleh, dan masih banyak lagi.

LAMPIRAN

Repository

Link Repository dari Tugas Besar 02 IF2211 Strategi Algoritma kelompok 48 “GoLink” adalah sebagai berikut.

https://github.com/novelxv/Tubes2_GoLink

Youtube

Link video Youtube dari Tugas Besar 02 IF2211 Strategi Algoritma kelompok 48 “GoLink” adalah sebagai berikut.

https://youtu.be/Jxm33AYBIxU

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/BFS-DFS-2021-Bag1-2024.pdf, terakhir diakses 20 April 2024.

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf, terakhir diakses 21 April 2024.