

LAPORAN TUGAS BESAR 02
IF2211 STRATEGI ALGORITMA

Kelompok Dinasti



Disusun oleh:

Muhammad Yusuf Rafi	(13522009)
Denise Felicia Tiowanni	(13522013)
Rafii Ahmad Fahreza	(10023570)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

DAFTAR ISI

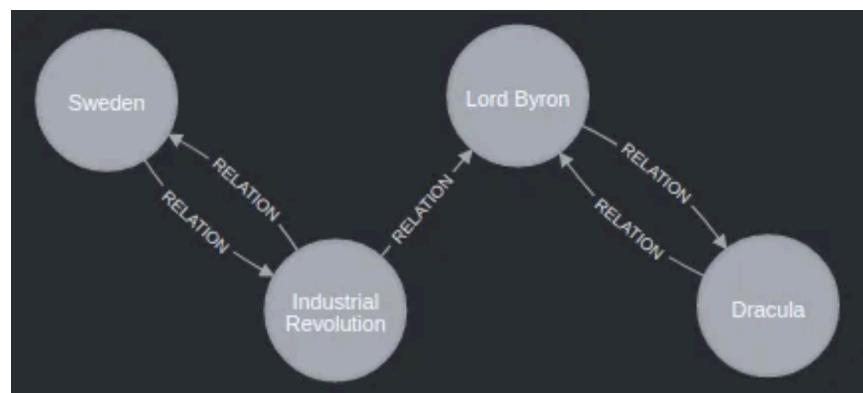
DAFTAR ISI.....	2
BAB I: DESKRIPSI MASALAH.....	4
BAB II: TEORI SINGKAT.....	5
2.1 Penjelajahan Graf dengan Algoritma IDS.....	5
2.2 Penjelajahan Graf dengan Algoritma BFS.....	5
2.3 Bahasa Pemrograman.....	6
2.4 Aplikasi Website: Wikirace by Dinasti.....	7
BAB III: ANALISIS PEMECAHAN MASALAH.....	8
3.1 Langkah-Langkah Pemecahan Masalah.....	8
3.2 Proses Pemetaan Masalah dengan Algoritma IDS.....	8
3.3 Proses Pemetaan Masalah dengan Algoritma BFS.....	9
3.4 Fitur Fungsional dan Arsitektur Aplikasi Website.....	10
3.5 Contoh Ilustrasi Kasus.....	11
BAB IV: IMPLEMENTASI DAN PENGUJIAN.....	12
4.1 Spesifikasi Teknis Program.....	12
4.1.1 Backend.....	12
4.1.1.1 Import.....	12
4.1.1.2 Struktur Data.....	13
4.1.1.3 Fungsi.....	14
4.1.2 Frontend.....	16
4.1.2.1 App.js.....	16
4.1.2.2 index.js.....	16
4.1.2.3 header.js.....	16
4.1.2.4 button.js.....	16
4.1.2.5 chopper-button.js.....	17
4.1.2.6 luffy-button.js.....	17
4.1.2.7 bfs-page.js.....	17
4.1.2.8 ids-page.js.....	17
4.1.2.9 how-to-use-page.js.....	17
4.1.2.10 about-us-page.js.....	18
4.2 Tata Cara Penggunaan Program.....	18
4.2.1 Setup Website.....	18
4.2.1.1 Tanpa Docker.....	18
4.2.1.2 Dengan Docker.....	18
4.2.2 Fitur Utama.....	18
1. Pilih algoritma yang diinginkan, BFS/IDS.....	18
a. Case 1: BFS.....	19
b. Case 2: IDS.....	19

2. Untuk setiap pilihan BFS/IDS, masukkan startURL dan targetURL dan klik ‘Search’	20
3. Tunggu hasilnya keluar.....	20
4. Lihat hasilnya.....	21
4.2.3 Fitur Tambahan.....	21
4.3 Hasil Pengujian.....	23
4.3.1 BFS.....	23
4.3.2 IDS.....	28
4.4 Analisis Hasil Pengujian.....	33
BAB V: PENUTUP.....	35
5.1 Kesimpulan.....	35
5.2 Saran.....	35
5.3 Refleksi.....	35
5.4 Ruang Perbaikan atau Pengembangan.....	36
5.5 Link Repository.....	36
5.6 Link Video.....	36
DAFTAR PUSTAKA.....	37
LAMPIRAN.....	38

BAB I

DESKRIPSI MASALAH

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. Contoh graf WikiRace

Di dalam Tugas Besar 2 ini, dibuat suatu program dalam bahasa Go yang mengimplementasikan algoritma IDS dan BFS untuk menyelesaikan permainan WikiRace. Program kemudian diimplementasikan dalam bentuk sebuah *website* yang menerima masukan berupa jenis algoritma, judul artikel awal, dan judul artikel tujuan. Program kemudian memberikan keluaran berupa jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel (dari artikel awal hingga artikel tujuan), dan waktu pencarian (dalam ms).

BAB II

TEORI SINGKAT

2.1 Penjelajahan Graf dengan Algoritma IDS

Penjelajahan Graf dengan algoritma IDS (*Iterative Deepening Search*) adalah salah satu metode penelusuran (*searching*) yang digunakan untuk mencari jalur atau solusi di dalam sebuah graf. Algoritma ini termasuk dalam kategori algoritma pencarian tak berinformasi atau *blind search* karena tidak memerlukan informasi tambahan tentang keadaan saat ini atau tujuan akhir.

Pada dasarnya, IDS bekerja dengan melakukan penelusuran secara bertahap pada graf dengan memperluas kedalaman penelusuran secara iteratif hingga solusi ditemukan. IDS mencoba untuk memperluas penelusuran mulai dari kedalaman 0 (pencarian hanya pada node awal) hingga kedalaman maksimum yang ditentukan. Algoritma ini menciptakan kesan seperti penelusuran secara mendalam (DFS) dengan tetap membatasi ruang pencarian yang diperlukan, sehingga lebih efisien untuk graf yang sangat besar atau dalam kasus di mana memori terbatas.

Proses penelusuran IDS biasanya dimulai dari kedalaman 0, kemudian naik satu tingkat kedalaman pada setiap iterasi jika solusi belum ditemukan. IDS terus melakukan iterasi hingga mencapai kedalaman maksimum atau menemukan solusi.

2.2 Penjelajahan Graf dengan Algoritma BFS

Penjelajahan Graf dengan algoritma BFS (*Breadth-First Search*) adalah metode penelusuran yang melakukan pencarian secara melebar pada sebuah graf. Algoritma ini juga termasuk dalam kategori algoritma pencarian tak berinformasi atau *blind search*.

BFS bekerja dengan mengeksplorasi semua node tetangga pada kedalaman saat ini terlebih dahulu sebelum bergerak ke kedalaman berikutnya. Algoritma ini menggunakan struktur data antrian (*queue*) untuk menyimpan node-node yang akan dieksplorasi berikutnya. node-node yang dimasukkan ke dalam antrian akan dieksplorasi secara berurutan sesuai dengan urutan masuknya.

Proses penelusuran BFS dimulai dari node awal, kemudian semua node tetangga dari node awal dieksplorasi terlebih dahulu. Setelah semua node pada kedalaman pertama dieksplorasi, BFS akan melanjutkan dengan mengeksplorasi node-node pada kedalaman kedua, dan begitu seterusnya. Algoritma ini terus berlanjut hingga menemukan solusi atau telah mengunjungi semua node yang dapat dijangkau dari node awal.

2.3 Bahasa Pemrograman

a. Go

Pada Tugas Besar II ini, Go digunakan sebagai bahasa pemrograman utama dalam implementasi algoritma penjelajahan graf WikiRace dan backend. Di-*import* pula beberapa *package* yang membantu melengkapi algoritma penjelajahan graf seperti:

1. <github.com/PuerkitoBio/goquery>

Tautan GitHub ini adalah suatu proyek GoQuery yang dikembangkan oleh PuerkitoBio. GoQuery adalah sebuah pustaka Go (bahasa pemrograman) yang memungkinkan kita untuk melakukan *query* pada dokumen HTML menggunakan sintaks yang mirip dengan jQuery. Dengan GoQuery, pencarian dan pemilihan elemen HTML, manipulasi atribut, dll dapat dilakukan.

2. <github.com/rs/cors>

Tautan GitHub ini adalah suatu proyek CORS yang dikembangkan oleh rs. CORS merupakan singkatan dari Cross-Origin Resource Sharing. Singkatnya, proyek ini adalah implementasi Go dari spesifikasi CORS yang memungkinkan untuk mengatasi batasan keamanan browser ketika kita mencoba mengakses sumber daya dari domain yang berbeda. Dengan meng-*import package* ini, permintaan lintas domain di aplikasi web dapat dilakukan.

3. <goroutine>

Goroutine memungkinkan eksekusi konkuren dan paralel dari kode. Go juga memungkinkan untuk menjalankan banyak goroutine secara bersamaan, yang dapat berkomunikasi satu sama lain melalui saluran (channels) sehingga dapat meningkatkan efisiensi dan responsivitas program. Selain itu, goroutine memanfaatkan multiple core CPU secara efisien, karena runtime Go akan mendistribusikan goroutine pada core CPU yang tersedia secara otomatis.

b. React

React digunakan sebagai *library* JavaScript untuk mengembangkan antarmuka pengguna (front-end). Dengan konsep komponen yang dapat digunakan ulang dan kemampuan pembaruan tanpa perlu *re-refresh* halaman, React memastikan antarmuka pengguna responsif dan efisien. Pada Tugas Besar ini, digunakan pula suatu framework bernama Tailwind CSS yang dihubungkan dengan React guna membantu proses *styling*.

2.4 Aplikasi Website: Wikirace by Dinasti

Aplikasi Wikirace by Dinasti adalah suatu aplikasi berbasis *website* yang dikembangkan menggunakan bahasa pemrograman Go, dengan front-end untuk antarmuka pengguna dan back-end untuk logika algoritma dan interaksi dengan Wikipedia API.

Aplikasi ini dibangun dengan tujuan menyediakan *platform* untuk mencari rute terpendek WikiRace dengan algoritma IDS dan BFS. WikiRace sendiri merupakan permainan di mana pemain diminta untuk menavigasi melalui artikel-artikel Wikipedia dari suatu artikel awal ke artikel tujuan yang telah ditentukan sebelumnya, dengan menggunakan sejumlah klik yang minimal dan dalam waktu sesingkat mungkin. Dengan menggunakan algoritma IDS (Iterative Deepening Search) dan BFS (Breadth-First Search), aplikasi ini memungkinkan pemain untuk menemukan rute terpendek dari artikel awal ke artikel tujuan.

Fitur utama aplikasi *website* mencakup kemampuan bagi pengguna untuk memasukkan judul artikel awal dan tujuan, serta memilih algoritma yang ingin digunakan untuk pencarian rute (BFS atau IDS). Setelah proses pencarian selesai, aplikasi akan menampilkan informasi seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel dari artikel awal hingga artikel tujuan, dan waktu pencarian dalam milisekon.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

Pemecahan masalah WikiRace dengan algoritma IDS (*Iterative Deepening Search*) atau BFS (*Breadth First Search*) diawali dengan mendefinisikan titik artikel tujuan awal (startURL) dan tujuan akhir (targetURL). Selanjutnya, dari artikel tujuan yang dimasukkan pengguna, digunakan API Wikipedia dan *package* GoQuery untuk mengambil struktur data yang terkait dengan artikel dan tautan antar-artikel, yang akan digunakan sebagai graf dalam penjelajahan.

Langkah selanjutnya adalah implementasi algoritma penjelajahan graf. Dalam konteks WikiRace, IDS dan BFS diimplementasikan dalam Go untuk menelusuri tautan dari artikel satu ke artikel lain. Selama penjelajahan, kedua algoritma mencatat node dan jalur yang diperiksa dengan menggunakan metode *web-scraping* dari *package* yang telah diimport sebelumnya. Untuk penjelasan mengenai tahapan dan perbedaan algoritma IDS dan BFS akan dijelaskan pada sub-bab berikutnya. Kedua algoritma ini akan membantu mencari rute tersingkat yang dapat dipilih pengguna untuk mencapai targetURL dari startURL.

Selain mengembalikan rute yang perlu dilalui untuk mencapai targetURL, program juga akan memberikan informasi mengenai jumlah artikel yang diperiksa dan dilalui serta waktu yang dibutuhkan untuk mencapai tujuan.

3.2 Proses Pemetaan Masalah dengan Algoritma IDS

Dalam algoritma IDS, masalah penjelajahan WikiRace dipetakan dengan mencoba kedalaman (*depth*) yang berbeda secara iteratif. Pada setiap iterasi, kedalaman penelusuran ditingkatkan, mulai dari node awal dan mencoba mencapai node tujuan tanpa melebihi batas kedalaman tersebut. Jika jalur tidak ditemukan dalam iterasi saat ini, kedalaman dinaikkan dan penjelajahan diulang.

Jadi, pencarian akan dimulai dari kedalaman terendah (0) yang artinya akan di cek apakah artikel awal (startURL) merupakan targetURL. Apabila bukan, program akan kemudian mencari tautan-tautan artikel apa saja yang terdapat dalam startURL (*scraping*)

dan menjadikan tautan-tautan tersebut sebagai node *child* dari startURL. Program akan kemudian meningkatkan kedalaman (sekarang 1) dan kembali mengecek apakah pada dengan kedalaman saat ini dengan node *child*, targetURL dapat ditemukan. Perlu diingat, program akan mengecek dengan metode DLS, yaitu dia akan mencari sampai tidak ada lagi *child*. Apabila terdapat targetURL, program akan berhenti dan mengembalikan rute yang perlu dilalui untuk mencapai targetURL tersebut. Namun, apabila target belum ditemukan pada kedalaman saat itu, program akan kembali melakukan *scraping* terhadap node-node dari *child* dan meningkatkan kedalaman secara iteratif.

Pada algoritma IDS, digunakan goroutine untuk optimalisasi program sehingga waktu eksekusi berkurang. Pada setiap kedalaman, setiap loop iterasi dalam fungsi IDS akan memanggil fungsi runSearch sebagai goroutine untuk melakukan pencarian solusi pada kedalaman tertentu. Setiap goroutine ini akan melakukan pencarian menggunakan fungsi DLS secara independen. Ketika loop iterasi berjalan, goroutine akan dibuat untuk setiap kedalaman mulai dari 0 hingga 5, dan kemudian dari 5 hingga 9 secara konkuren. Ini berarti bahwa pada suatu waktu, goroutine-goroutine ini akan berjalan bersamaan untuk mencoba menemukan solusi pada kedalaman yang berbeda. Yang mana ketika goroutine-goroutine ini berjalan bersamaan, proses pencarian solusi pada setiap kedalaman akan terjadi secara paralel. Artinya, program tidak perlu menunggu satu pencarian selesai sebelum memulai pencarian pada kedalaman yang lain. Hal ini meningkatkan efisiensi waktu eksekusi keseluruhan program. Selain itu, digunakan pula time.Sleep() sebagai *delay* agar terhindari dari blokir Wikipedia.

3.3 Proses Pemetaan Masalah dengan Algoritma BFS

Dalam algoritma BFS, masalah dipecahkan dengan menelusuri secara melebar dari artikel awal. BFS menggunakan antrian (*queue*) untuk mengelola node yang harus dieksplorasi, memastikan bahwa semua node pada tingkat kedalaman yang sama diperiksa sebelum bergerak ke kedalaman berikutnya.

Program awalnya akan mengecek apakah artikel awal (startURL) merupakan targetURL. Apabila bukan, program akan kemudian mencari tautan-tautan artikel apa saja yang terdapat dalam startURL (*scraping*) dan menjadikan tautan-tautan tersebut sebagai node *child* dari startURL sekaligus langsung mengecek apakah dari tautan-tautan

hasil *scraping* tersebut ada yang merupakan targetURL. Apabila bukan merupakan targetURL, setiap node *child* tersebut akan dimasukkan kedalam *queue*. Apabila *scraping* telah selesai, program akan menelusuri setiap isi *queue* mulai dari yang terawal dan merupakan *scraping* terhadap node *child* tersebut. Program kemudian akan mencari apakah terdapat targetURL dari hasil *scraping* node *child*. Apabila ada, program akan langsung mengembalikan rute. Namun, jika tidak, program akan pindah ke node *child* pada antrian berikutnya dan melakukan *scraping* terhadap node tersebut. Hal ini akan dilakukan terus menerus hingga antrian habis. Dengan demikian solusi yang ditemukan dijamin adalah salah satu yang membutuhkan klik terkecil.

Pada algoritma BFS, juga diimplementasikan goroutine untuk optimalisasi program. Goroutine diimplementasikan agar program dapat bekerja secara konkuren sehingga meningkatkan waktu pencarian rute. Tekniknya adalah *queue* yang dibentuk akan dibagi menjadi beberapa *batch* dan akan diproses per masing-masing *batch*. Goroutine diterapkan untuk masing-masing *batch* tersebut. Diimplementasikan pula `time.Sleep()` sebagai *delay* agar IP Address kita tidak diblokir oleh Wikipedia. Masalahnya adalah, untuk penjelajahan jauh, `time.Sleep()` yang kecil akan menyebabkan pencarian sangat cepat sehingga masih berpotensi terblokir serta terjadi race condition. Namun, apabila `time.Sleep()` yang digunakan besar, mungkin lebih aman dari blokir untuk penjelajahan jauh, hanya saja penjelajahan singkat menjadi lebih lama. Apabila ingin optimal, saat melakukan penjelajahan jauh, `time.Sleep()` bisa dinonaktifkan dahulu.

3.4 Fitur Fungsional dan Arsitektur Aplikasi Website

Terdapat beberapa fitur-fitur fungsional dalam aplikasi *website* Wikirace by Dinasti, diantaranya:

1. Input judul artikel awal dan tujuan

Pengguna dapat menginput judul artikel awal dan tujuan pada *search box* yang tersedia. Program juga menyediakan *suggestion* yang diambil dari Wikipedia API sehingga memungkinkan pengguna untuk memilih saran tersebut dengan klik tunggal tanpa perlu menyelesaikan pengetikan judul artikel secara lengkap.

2. Pemilihan algoritma

Pengguna dapat memilih algoritma pencarian yang digunakan, seperti BFS dan DFS.

3. Visualisasi jalur pencarian.

Program juga memvisualisasikan graf rute dari artikel awal ke artikel tujuan sehingga memudahkan pengguna dalam memvisualisasikan rute tersebut.

4. How to Use

Program memiliki halaman “How to Use” yang secara detail menjelaskan bagaimana penggunaan program secara keseluruhan.

5. About Us

Program memiliki halaman “About Us” yang mempunyai informasi mengenai *developer* aplikasi.

Arsitektur aplikasi juga menggabungkan backend yang ditulis dalam bahasa pemrograman Go, yang bertanggung jawab atas logika algoritma dan komunikasi dengan API Wikipedia, dan frontend yang menggunakan React untuk antarmuka dinamis.

3.5 Contoh Ilustrasi Kasus

Untuk memberikan pemahaman yang lebih baik tentang bagaimana algoritma bekerja, kita bisa melihat contoh kasus penjelajahan dari artikel “Artificial intelligence” ke “Steve Jobs”. Dengan menggunakan baik algoritma BFS ataupun IDS, aplikasi akan menampilkan salah satu jalur yang mungkin, yaitu: “Artificial intelligence” → “Siri” → “Steve Jobs”, dengan menunjukkan jumlah node (artikel) yang diperiksa dan durasi pencarian. Contoh ini membantu mengilustrasikan efisiensi dan efektivitas algoritma dalam menyelesaikan permainan WikiRace dalam setting aplikasi nyata, karena terlihat bahwa Siri memang merupakan salah satu bentuk *artificial intelligence* yang dibuat oleh Steve Jobs pada produk Apple.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

4.1.1 Backend

4.1.1.1 Import

Nama Import	Deskripsi
fmt	Utilitas untuk format I/O. Dengan menggunakan <i>package</i> ini, kita dapat melakukan format input dan output, termasuk mencetak output dan membaca data dari input pengguna.
net/http	Menyediakan fitur HTTP <i>client</i> dan <i>server</i> . Dengan <i>package</i> ini, dapat dibuat <i>server</i> HTTP dengan menggunakan fungsi seperti http.ListenAndServe, dan <i>client</i> HTTP yang dapat melakukan permintaan web dengan fungsi seperti http.Get atau http.Post.
os	Menyediakan cara untuk berinteraksi dengan fungsi sistem operasi pada tingkat yang rendah. Ini termasuk manipulasi file dan direktori, mengambil argumen baris perintah, serta mengelola variabel lingkungan dan proses.
strings	Memanipulasi string UTF-8.
time	Mengukur dan menampilkan waktu (mendapatkan waktu saat ini, etc).
github.com/PuerkitoBio/goquery	Untuk melakukan <i>query</i> dan manipulasi dokumen HTML. Digunakan untuk web <i>scraping</i> , <i>parsing</i> , dan manipulasi dokumen HTML yang diperoleh dari

	permintaan web.
github.com/rs/cors	Implementasi <i>middleware</i> Cross-Origin Resource Sharing (CORS) untuk aplikasi Go. <i>Middleware</i> ini membantu dalam mengatur izin yang memungkinkan sumber daya suatu situs dapat diakses oleh situs lain dari domain yang berbeda.

4.1.1.2 Struktur Data

Struktur Data	Deskripsi
<pre>type node struct { URL string Parent *node Children []*node Depth int }</pre>	<p>Digunakan untuk merepresentasikan setiap simpul dalam graf penjelajahan.</p> <ul style="list-style-type: none"> • URL: String yang menyimpan URL dari halaman web yang direpresentasikan oleh node. • Parent: Pointer ke node <i>parent</i> dari node saat ini. Ini digunakan untuk melacak jalur kembali ke node awal dalam graf. • Children: Slice dari pointer ke node yang merupakan <i>child</i> dari node ini, mewakili halaman yang dapat diakses langsung dari halaman saat ini melalui tautan. • Depth: Integer yang menyatakan kedalaman node ini dari node root atau awal.
<pre>type ShortestPathResult struct { Path []string `json:"path"` ArticlesVisited int `json:"articlesVisited"` ArticlesChecked int `json:"articlesChecked"` }</pre>	<p>Digunakan untuk menyimpan hasil dari algoritma pencarian jalur terpendek, jumlah artikel yang telah dikunjungi, jumlah artikel yang telah diperiksa, dan waktu eksekusi.</p> <ul style="list-style-type: none"> • Path: <i>Slice</i> dari <i>string</i> yang merepresentasikan

<pre> ExecutionTime time.Duration `json:"executionTime"` } </pre>	<p>jalur dari node awal ke node tujuan, diurutkan dari awal ke tujuan. Setiap <i>string</i> dalam <i>slice</i> mewakili URL atau identifikasi lain dari setiap halaman yang dikunjungi dalam jalur.</p> <ul style="list-style-type: none"> • ArticlesVisited: Angka yang menyatakan jumlah artikel yang dikunjungi selama pencarian jalur. Ini melibatkan setiap node yang telah diakses selama proses pencarian. • ArticlesChecked: Angka yang mengukur jumlah artikel yang diperiksa untuk menemukan jalur yang valid. • ExecutionTime: Menyimpan durasi eksekusi pencarian jalur.
---	---

4.1.1.3 Fungsi

Nama Fungsi (parameter)	Deskripsi
func validateURL(url string) bool	Fungsi ini berfungsi untuk memastikan atau memvalidasi apakah startURL dan targetURL yang dimasukkan pengguna adalah URL yang valid atau tersedia.
func BFSHandler(w http.ResponseWriter, r *http.Request)	HTTP handler untuk metode BFS. Fungsi ini mengambil permintaan dari pengguna, memprosesnya untuk menjalankan algoritma BFS, dan mengirimkan hasil pencarian kembali ke pengguna melalui <i>response</i> HTTP.
func IDSHandler(w http.ResponseWriter, r *http.Request, f *os.File)	HTTP handler untuk metode IDS. Handler ini juga mengambil input dari permintaan HTTP, menjalankan

	IDS, dan menulis hasilnya ke <i>response</i> . Ini juga menerima <i>file</i> sebagai parameter untuk log atau keperluan debug.
func min(a, b int) int	Membandingkan untuk mendapatkan nilai minimum.
func BFS(startURL, endURL string) ([]string, int, int, time.Duration)	Menjalankan algoritma <i>Breadth First Search</i> antara URL awal dan URL tujuan. Fungsi ini mengembalikan jalur yang ditemukan, jumlah artikel yang dikunjungi, jumlah artikel yang diperiksa, dan durasi eksekusi.
func IDS(startURL, endURL string, f *os.File) ([]string, int, int, time.Duration)	Menjalankan algoritma <i>Iterative Deepening Search</i> antara URL awal dan URL tujuan. Mengembalikan jalur yang ditemukan, jumlah artikel yang dikunjungi, jumlah artikel yang diperiksa, dan durasi eksekusi. Juga menerima <i>file</i> untuk logging atau debug.
func DLS(stack []*node, endURL string, depthLimit int, f *os.File, visited map[string]bool) ([]string, int, int, bool)	Implementasi dari <i>Depth Limited Search</i> , digunakan sebagai bagian dari IDS. Menggunakan <i>stack</i> untuk mengelola node yang dikunjungi, membatasi pencarian berdasarkan kedalaman. Mengembalikan jalur yang ditemukan, jumlah artikel yang diperiksa dan dikunjungi, serta flag berhasil/tidaknya pencarian.
func extractArticleName(url string) string	Mengambil URL dan mengembalikan nama artikel dari URL tersebut.
func getLinks(URL string) []string	Mengambil sebuah URL dan mengembalikan <i>slice</i> dari URL yang merupakan tautan dalam halaman yang diindikasikan oleh URL awal. Ini digunakan untuk mendapatkan semua tautan dalam sebuah artikel Wikipedia (<i>scraping</i>).

func getPath(endnode *node) []string	Mengembalikan jalur dari node awal ke node yang ditentukan.
func ShortestPathHandler(w http.ResponseWriter, r *http.Request)	Ini adalah HTTP handler yang digunakan untuk memilih antara BFS dan IDS berdasarkan input pengguna dan menulis hasil ke <i>response</i> HTTP.
func recoverMiddleware(next http.Handler) http.Handler	Menangkap <i>panic</i> yang terjadi selama penanganan permintaan HTTP.
func main()	Fungsi utama yang menginisialisasi <i>server</i> HTTP dan menetapkan handler untuk berbagai rute.

4.1.2 Frontend

4.1.2.1 App.js

‘App.js’ adalah *file* utama dalam tampilan pengguna yang menggunakan React Router untuk menangani navigasi. Terdapat beberapa komponen utama dalam file ini, diantaranya Header, Button, IDSPage, BFSPage, HowToUsePage, AboutUsPage.

4.1.2.2 index.js

‘index.js’ menyiapkan aplikasi React dengan mengimpor modul, membuat elemen root, serta merender komponen utama App ke dalam elemen root yang telah ditentukan.

4.1.2.3 header.js

File ‘header.js’ mengatur tampilan header dari *website*. File ini mencakup beberapa komponen yang terdapat pada *website*, seperti judul dan nama *website*.

4.1.2.4 button.js

File ‘button.js’ mengatur tampilan button dari *website*. File ini mencakup beberapa komponen yang terdapat pada *website*, seperti LuffyButton yang mengarah pada IDS dan ChopperButton yang mengarah pada BFS.

4.1.2.5 chopper-button.js

File ‘chopper-button.js’ mengatur tampilan chopper button. File ini mencakup beberapa komponen yang nantinya dapat menavigasi halaman ke bfs-page apabila ditekan.

4.1.2.6 luffy-button.js

File ‘luffy-button.js’ mengatur tampilan luffy button. File ini mencakup beberapa komponen yang nantinya dapat menavigasi halaman ke ids-page apabila ditekan.

4.1.2.7 bfs-page.js

File ‘bfs-page.js’ mengatur tampilan pencarian dengan metode BFS. File ini mencakup beberapa komponen yang menerima input pengguna, tombol *search*, serta menampilkan hasil dari pencarian, termasuk menampilkan graf.

4.1.2.8 ids-page.js

File ‘ids-page.js’ mengatur tampilan pencarian dengan metode IDS. File ini mencakup beberapa komponen yang menerima input pengguna, tombol *search*, serta menampilkan hasil dari pencarian, termasuk menampilkan graf.

4.1.2.9 how-to-use-page.js

File ‘how-to-use-page.js’ mengatur isi atau konten dari halaman How to Use. Pada file ini juga terdapat tombol *back* untuk kembali ke halaman awal.

4.1.2.10 about-us-page.js

File ‘about-us-page.js’ mengatur isi atau konten dari halaman About Us. Pada file ini juga terdapat tombol *back* untuk kembali ke halaman awal.

4.2 Tata Cara Penggunaan Program

4.2.1 Setup Website

4.2.1.1 Tanpa Docker

Run file Api.go dalam <i>directory</i> backend (sebelumnya, cd src/backend terlebih dahulu)
>> go run ./Api.go
Run React <i>website</i> dalam <i>directory</i> frontend/wikirace (sebelumnya, cd src/frontend/wikirace terlebih dahulu)
>> npm run start

4.2.1.2 Dengan Docker

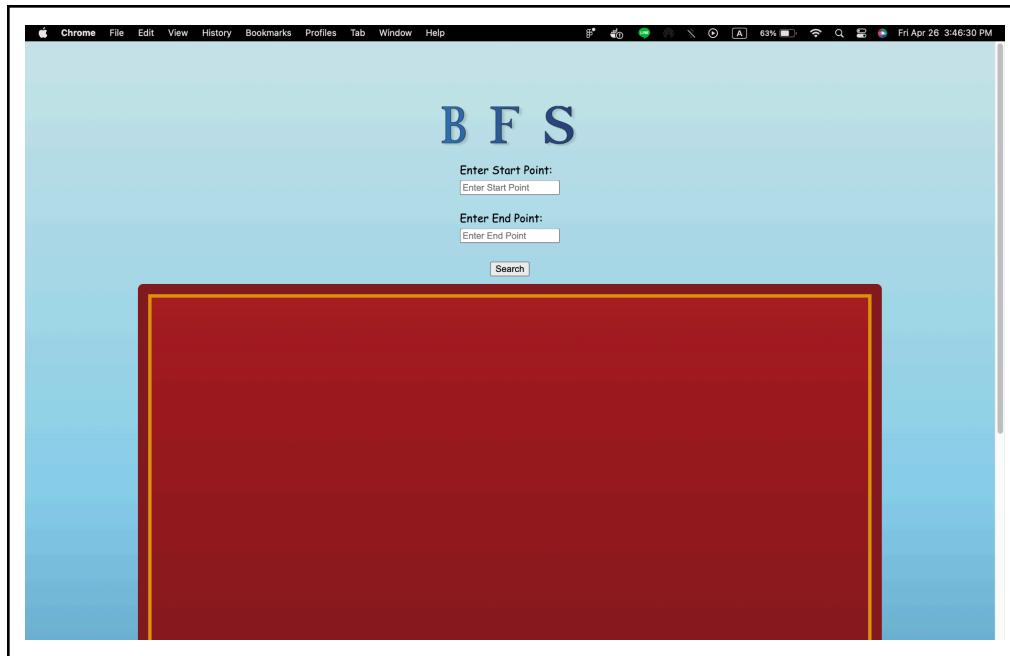
Run dockerfile dalam <i>directory</i> tugas besar
>> docker-compose up --build

4.2.2 Fitur Utama

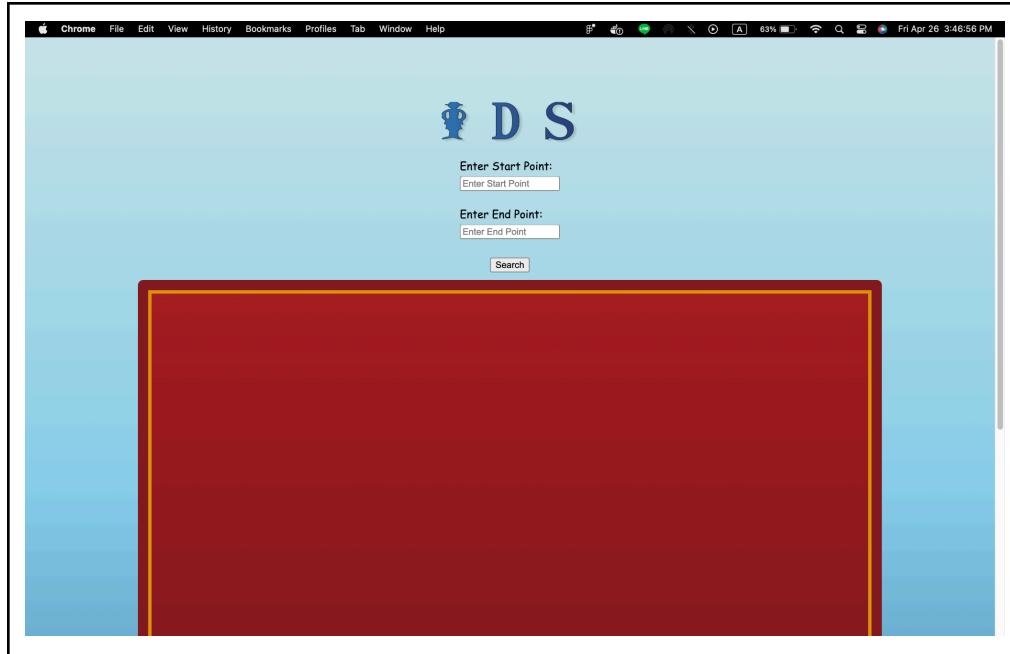
- 1. Pilih algoritma yang diinginkan, BFS/IDS.**



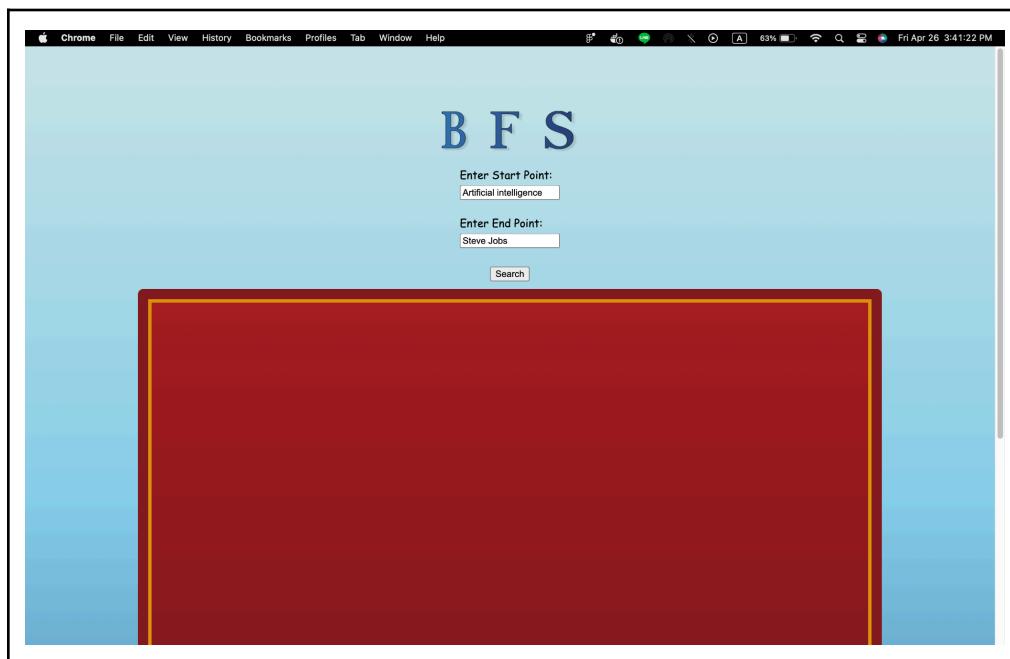
a. Case 1: BFS



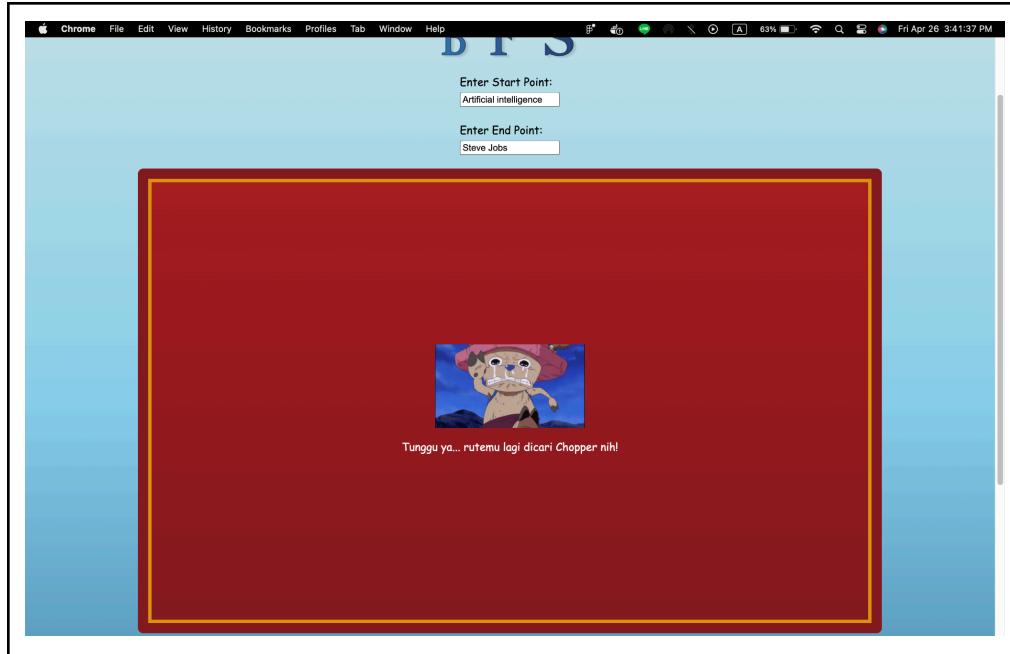
b. Case 2: IDS



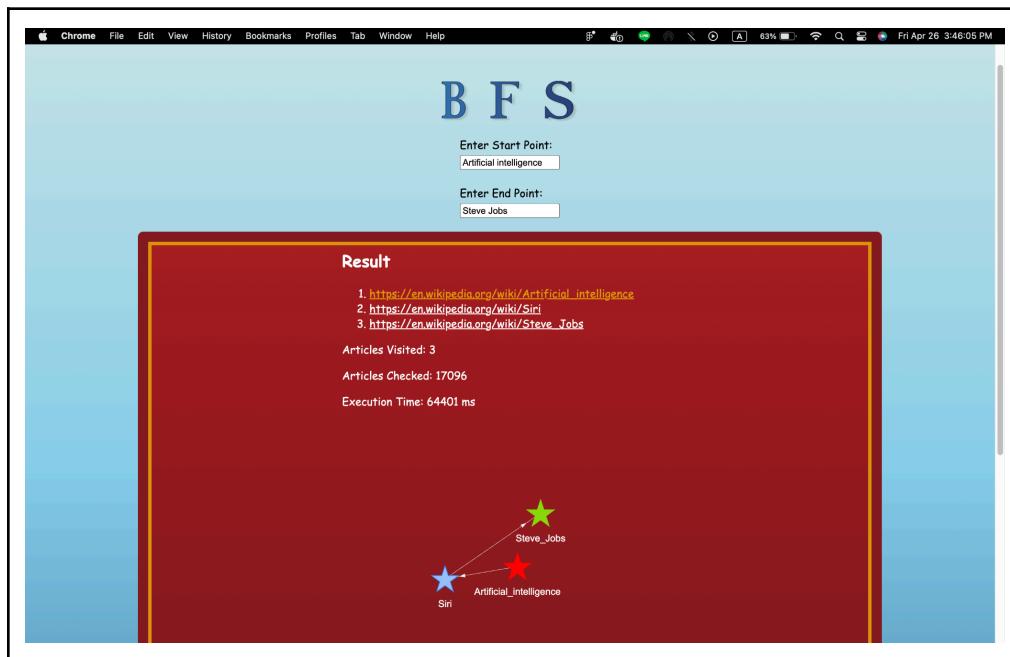
2. Untuk setiap pilihan BFS/IDS, masukkan startURL dan targetURL dan klik 'Search'.



3. Tunggu hasilnya keluar.

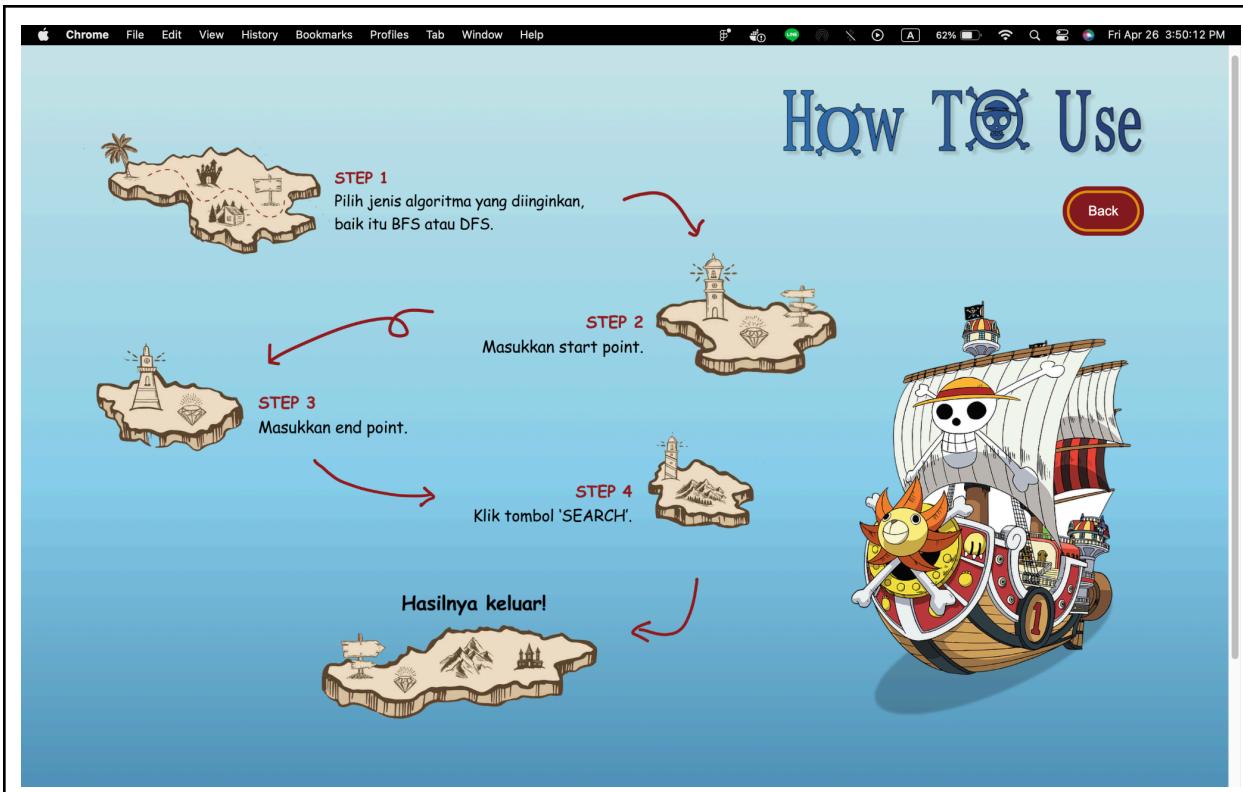


4. Lihat hasilnya.



4.2.3 Fitur Tambahan

How to Use



About Us

The screenshot displays the 'About Us' page. It includes a title, a welcome message, a crew introduction section, and three member profiles.

About Us

Welcome to Wikirace by Dinasti!

Wikirace by Dinasti merupakan sebuah website Wikirace dengan algoritma BFS dan IDS. Ayo temukan rute tersingkat dari suatu keyword ke keyword lain!

Meet Our Crew

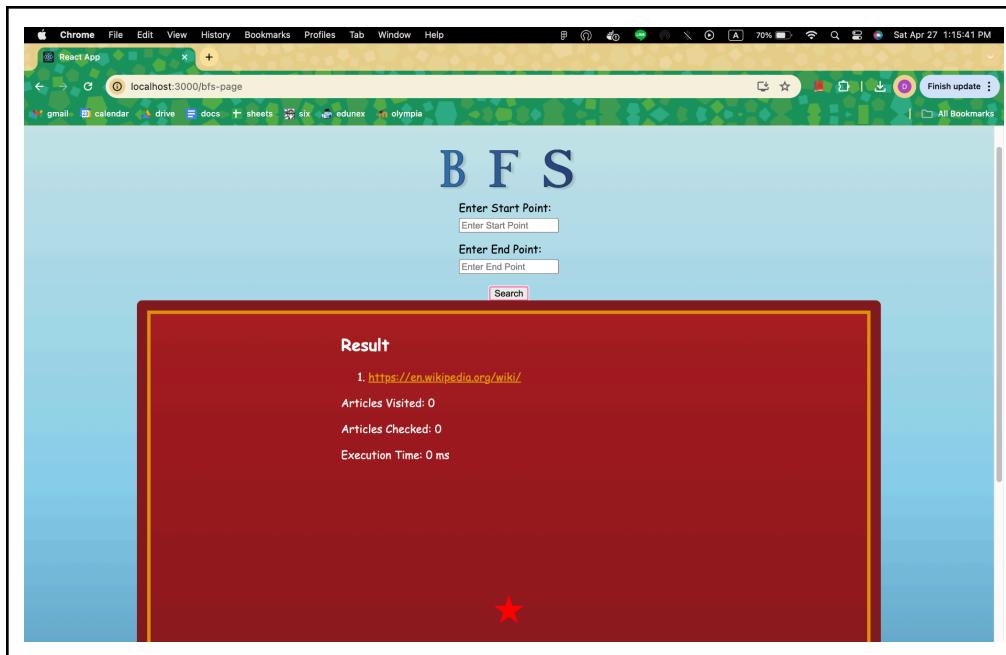
Denise Felicia Tiovanni NIM: 13522013	Muhammad Yusuf Rafi NIM: 13522009	Rafii Ahmad Fahreza NIM: 10023570

4.3 Hasil Pengujian

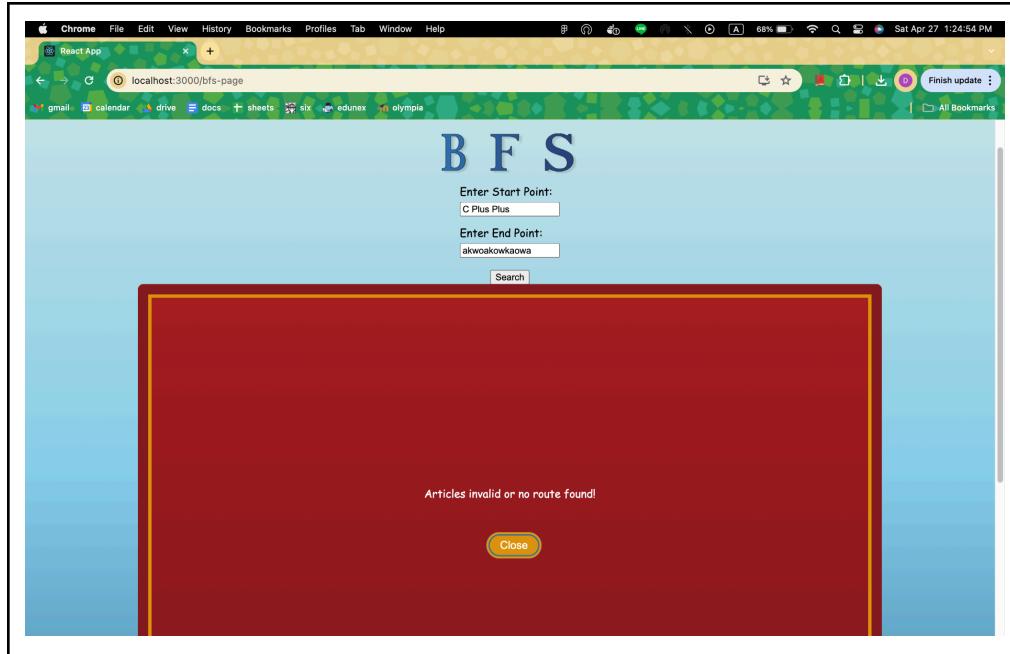
/* screenshot antarmuka dan skenario yang memperlihatkan berbagai kasus yang mencakup seluruh fitur pada aplikasi Anda */

4.3.1 BFS

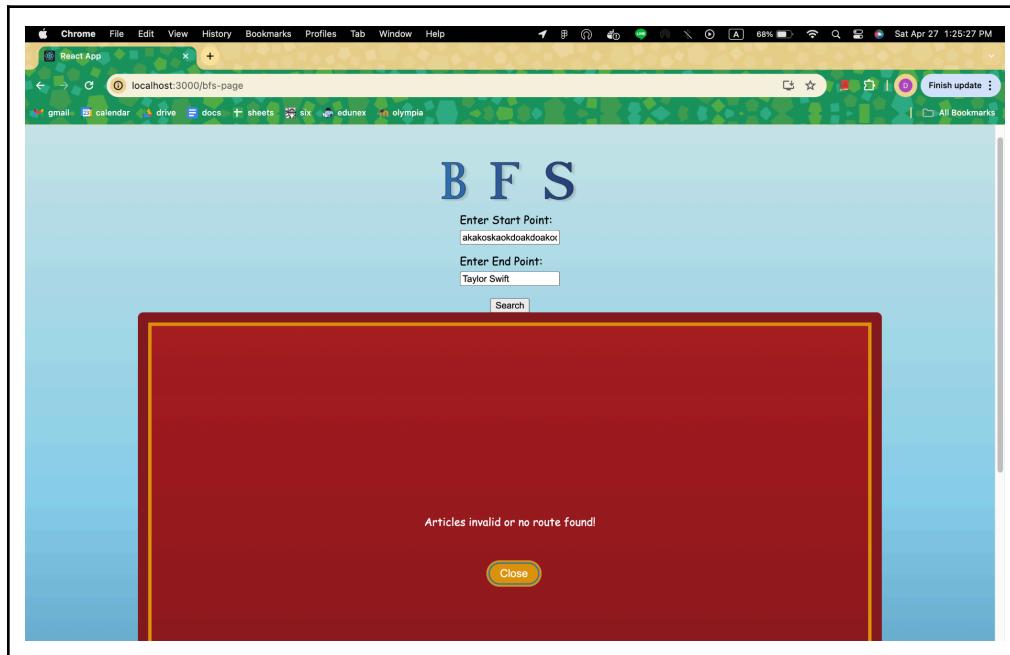
1. StartURL dan TargetURL kosong (*searching* dengan laman /wiki/)



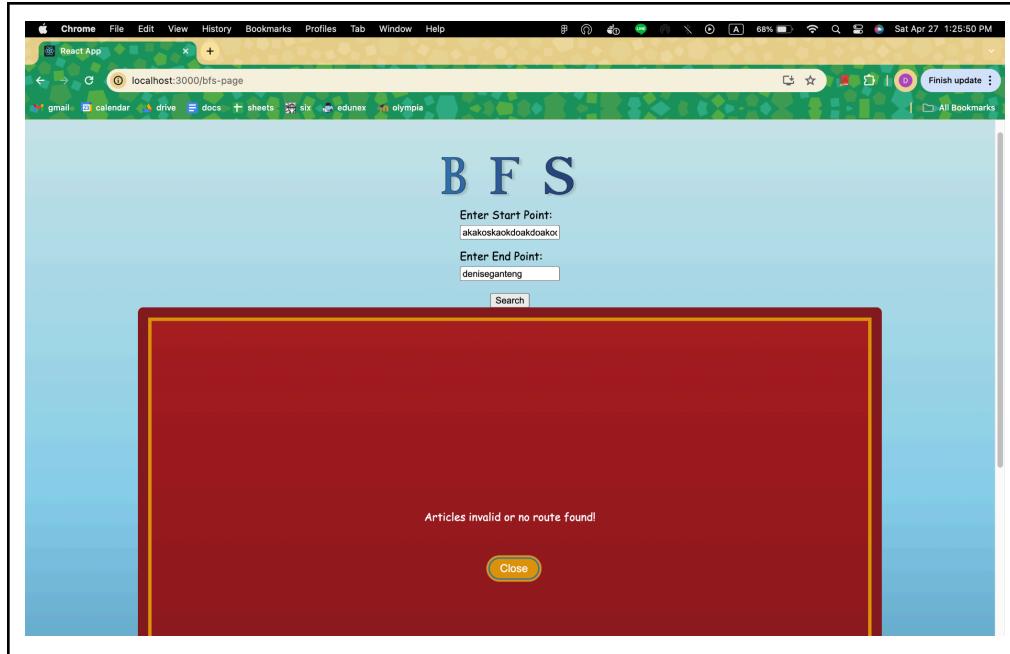
2. StartURL valid dan TargetURL tidak valid



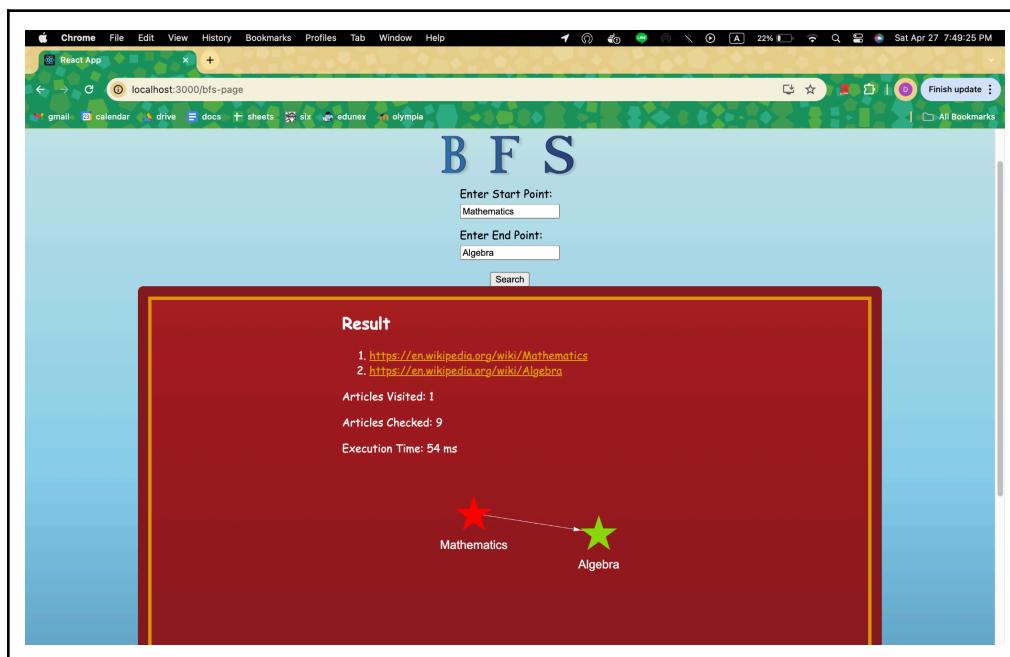
3. StartURL tidak valid dan TargetURL valid



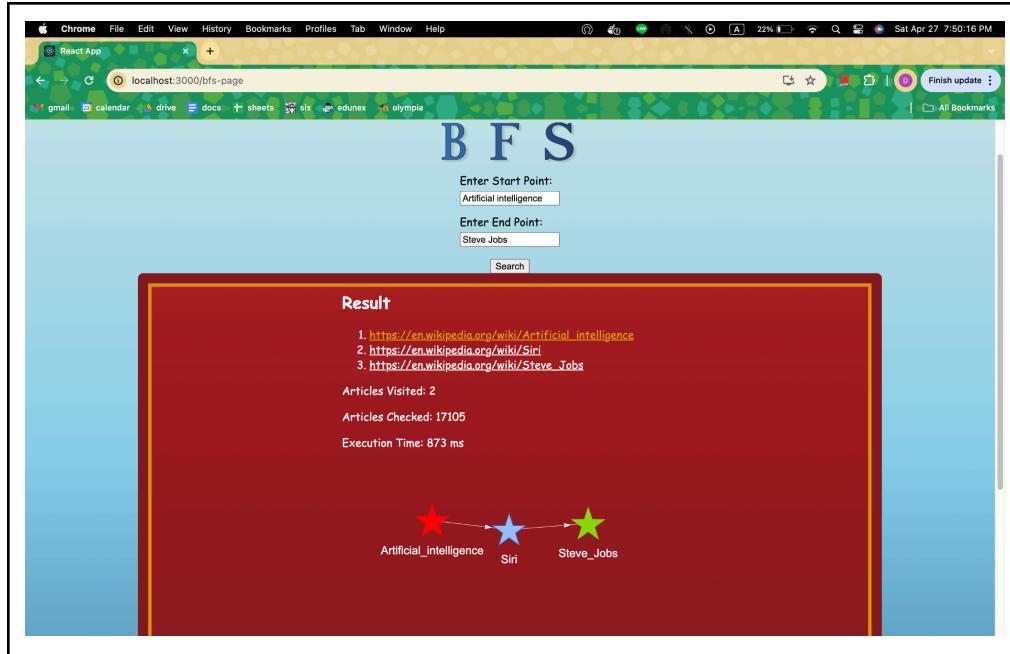
4. StartURL dan TargetURL tidak valid



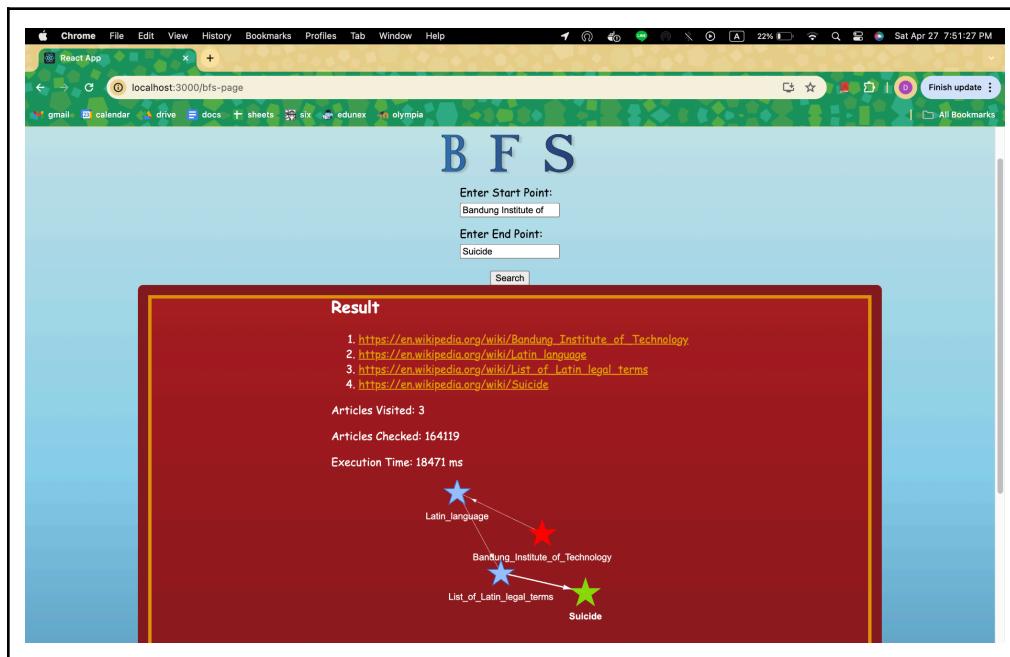
5. Pencarian 1 derajat: Mathematics → Algebra



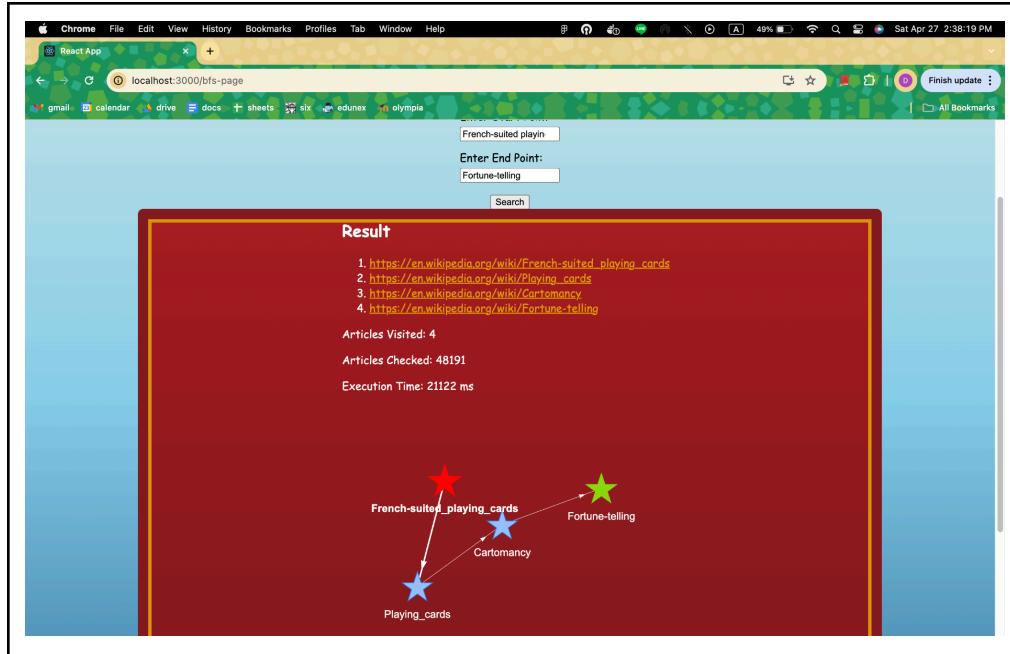
6. Pencarian 2 derajat: Artificial intelligence → Steve Jobs



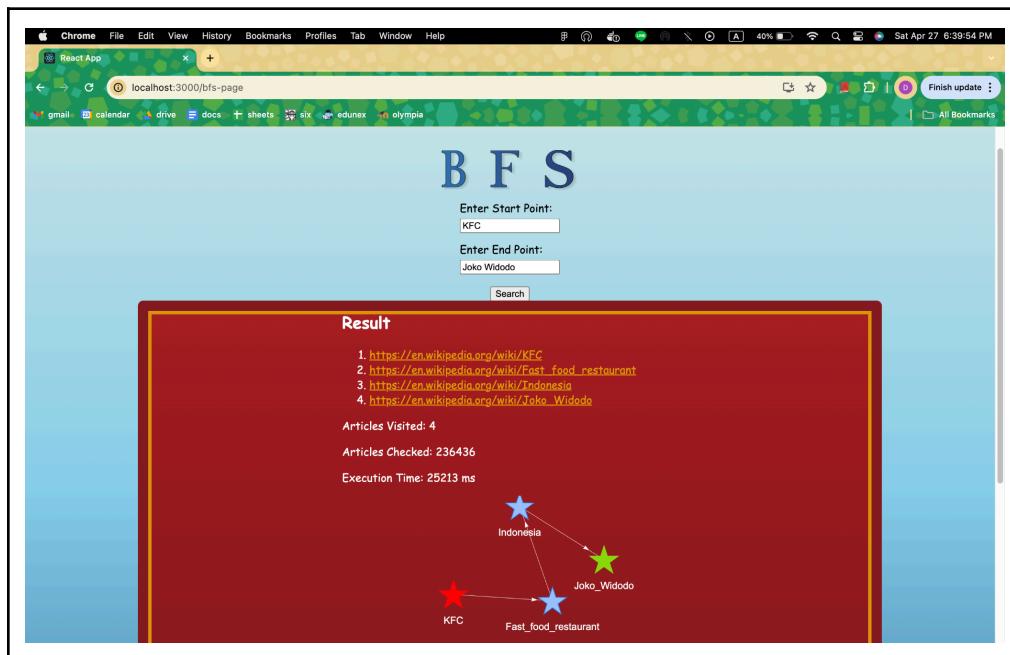
7. Pencarian 3 derajat: Bandung Institute of Technology → Suicide



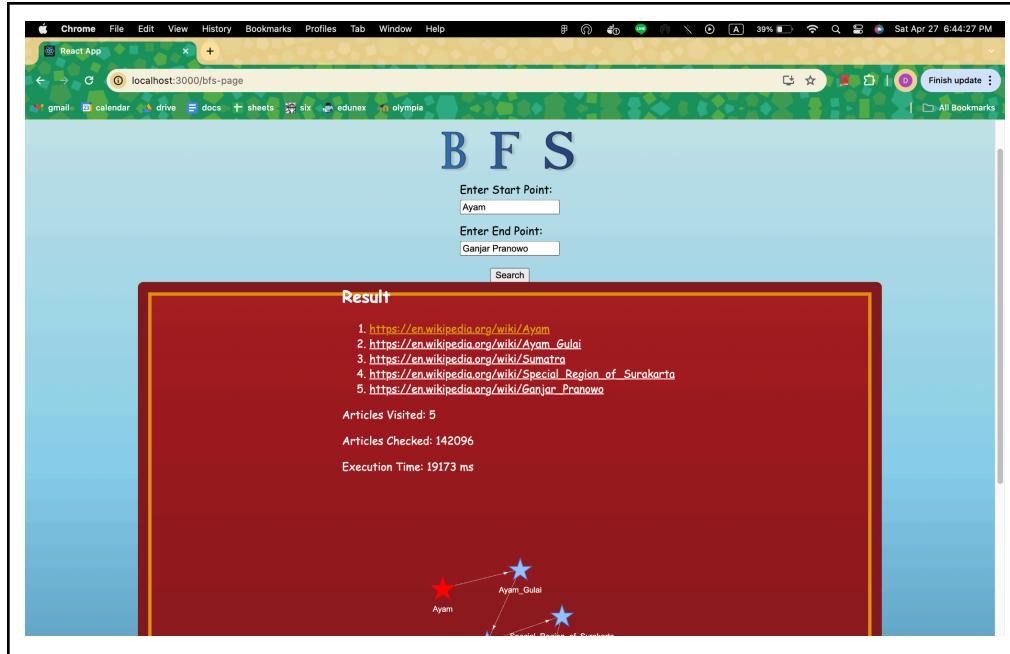
8. Pencarian 3 derajat: French-suited playing cards → Fortune-telling



9. Pencarian 3 derajat: KFC → Joko Widodo

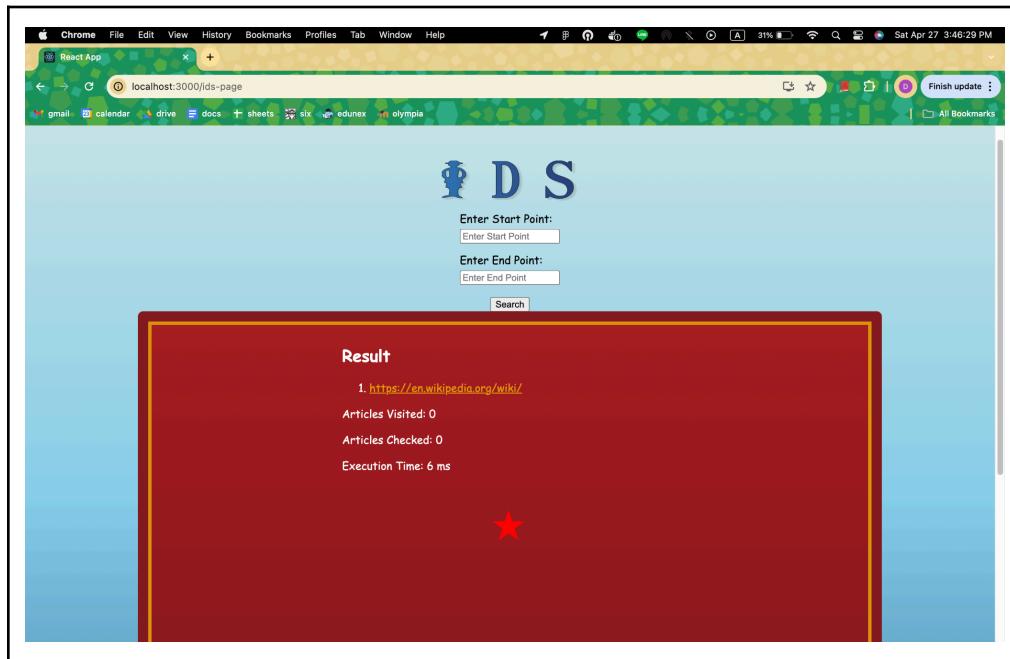


10. Pencarian 4 derajat: Ayam → Ganjar Pranowo

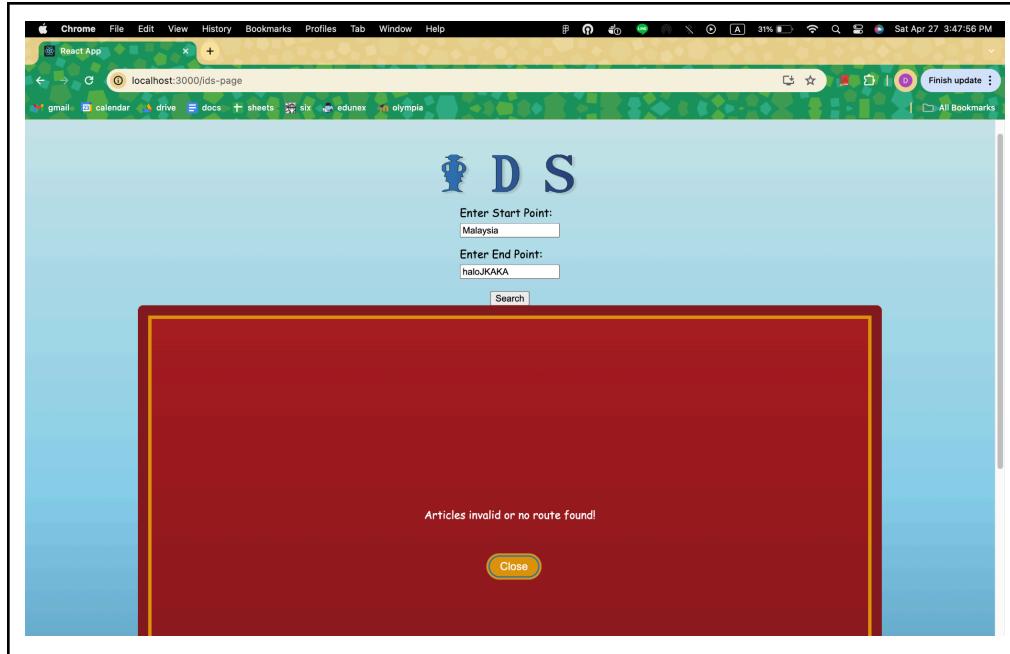


4.3.2 IDS

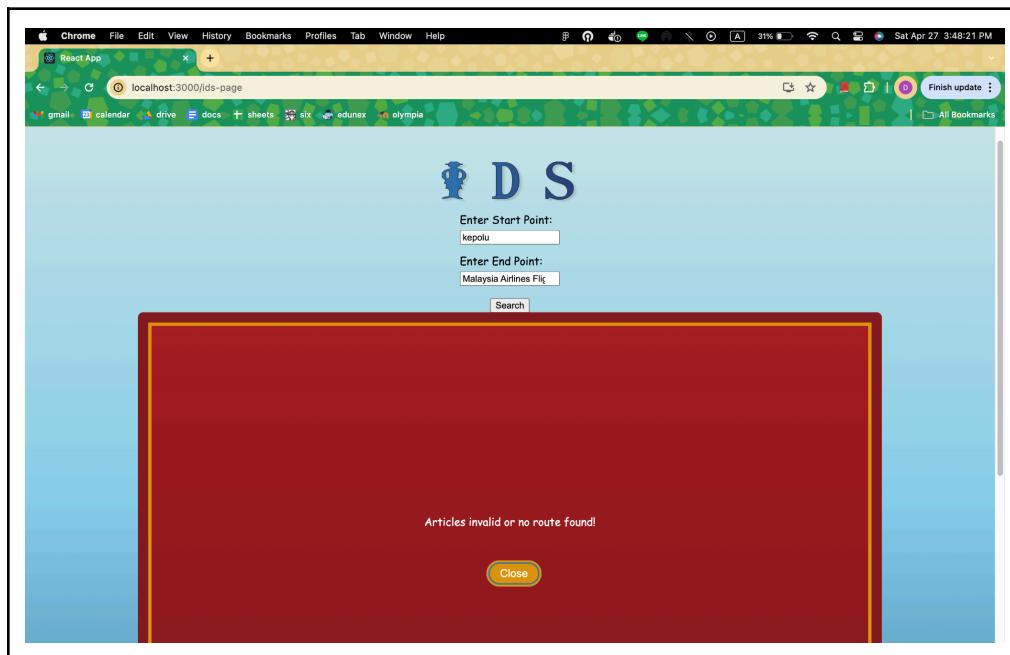
1. StartURL dan TargetURL kosong (*searching* dengan laman /wiki/)



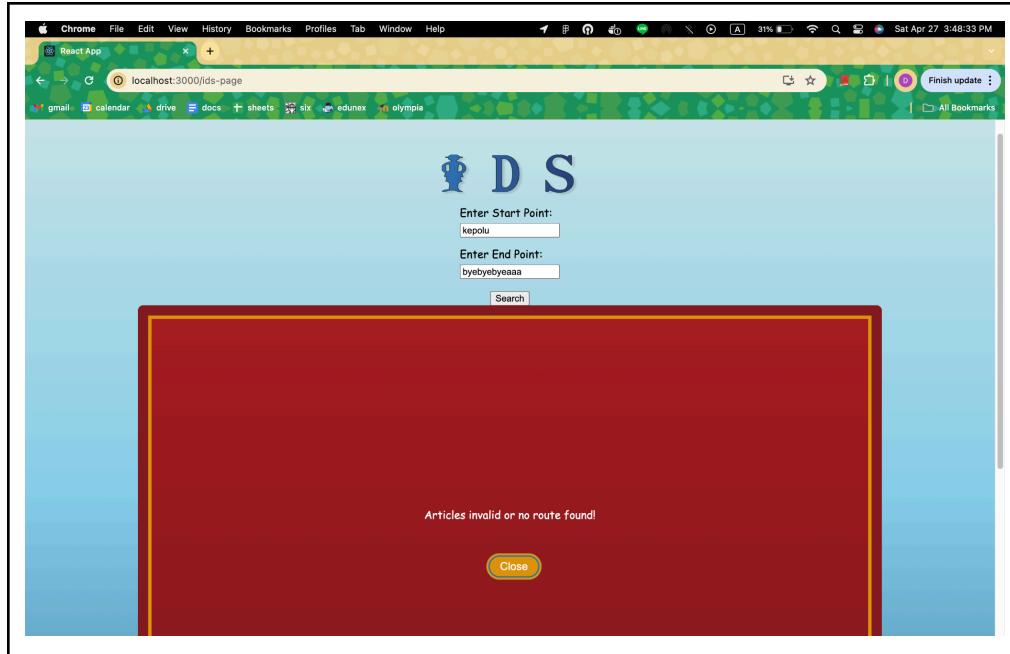
2. StartURL valid dan TargetURL tidak valid



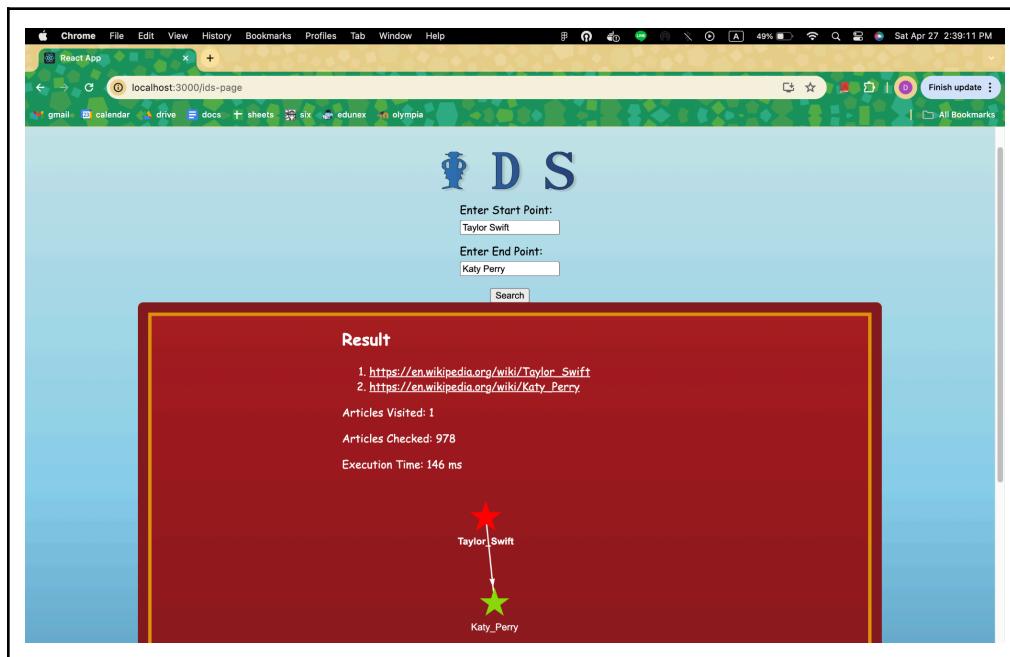
3. StartURL tidak valid dan TargetURL valid



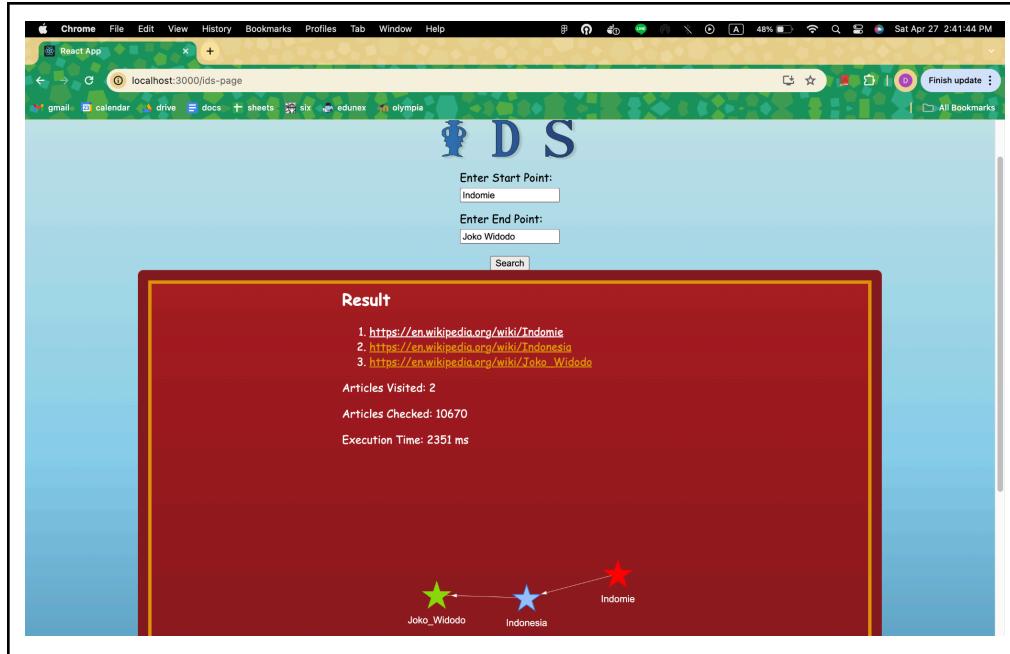
4. StartURL dan TargetURL tidak valid



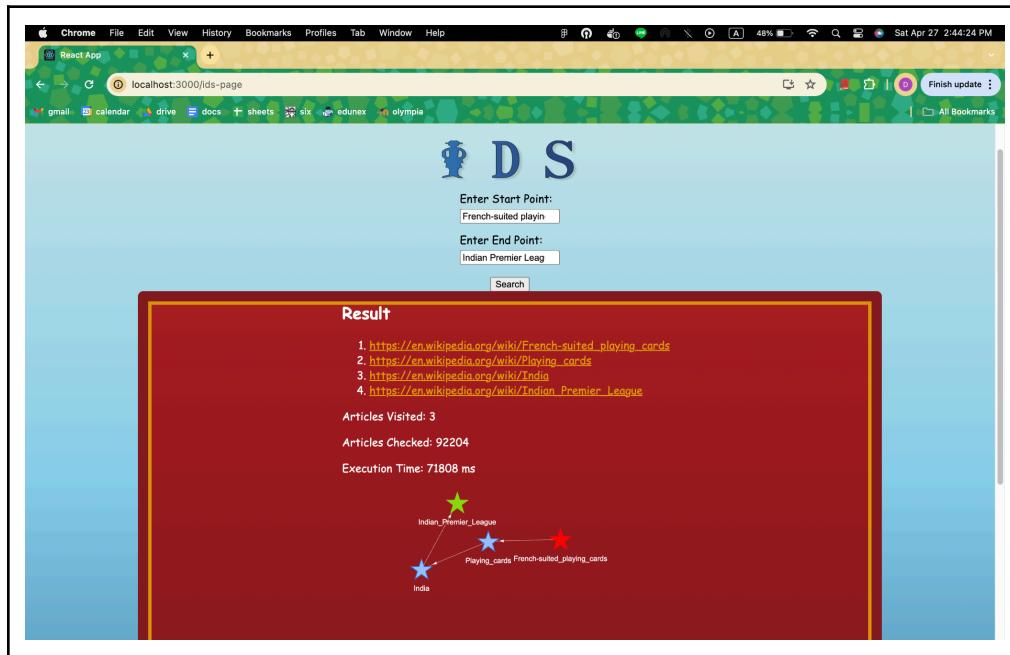
5. Pencarian 1 derajat: Taylor Swift → Katy Perry



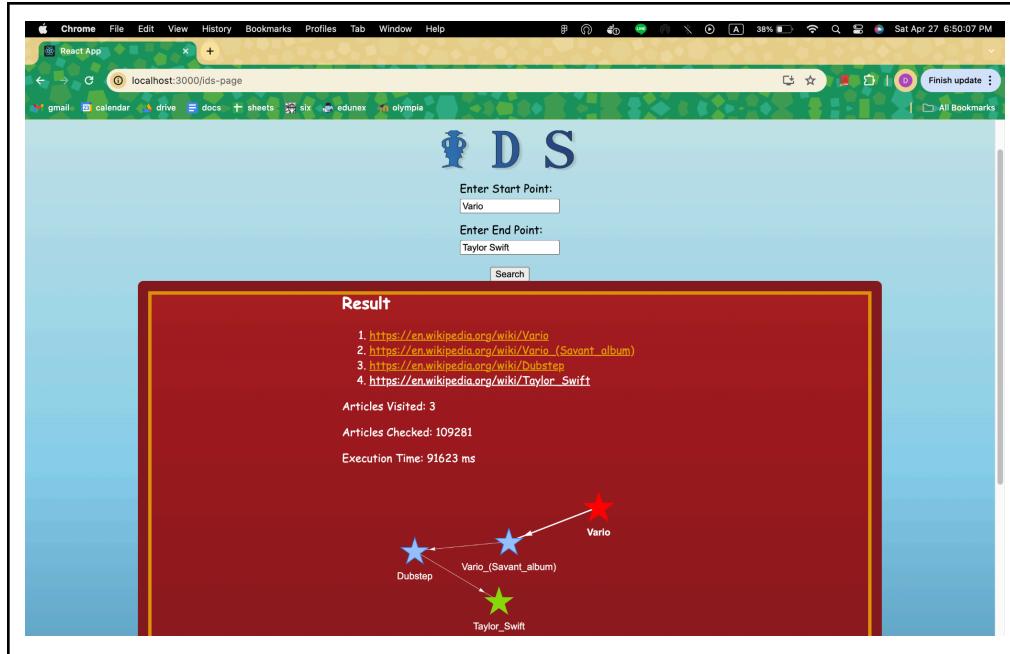
6. Pencarian 2 derajat: Indomie → Joko Widodo



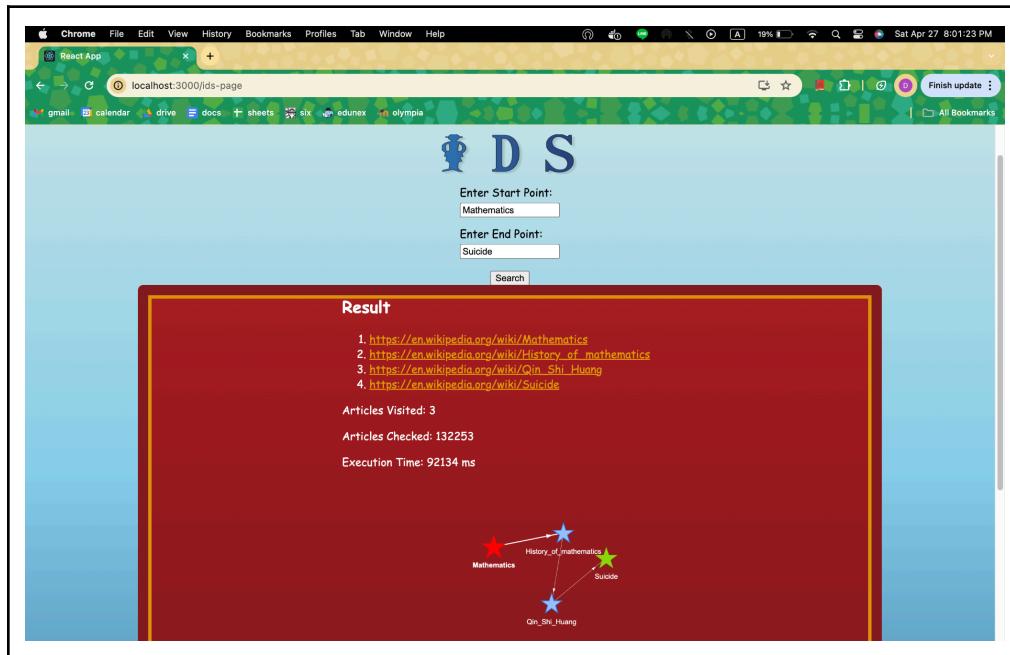
7. Pencarian 3 derajat: French-suited playing cards → Indian Premier League



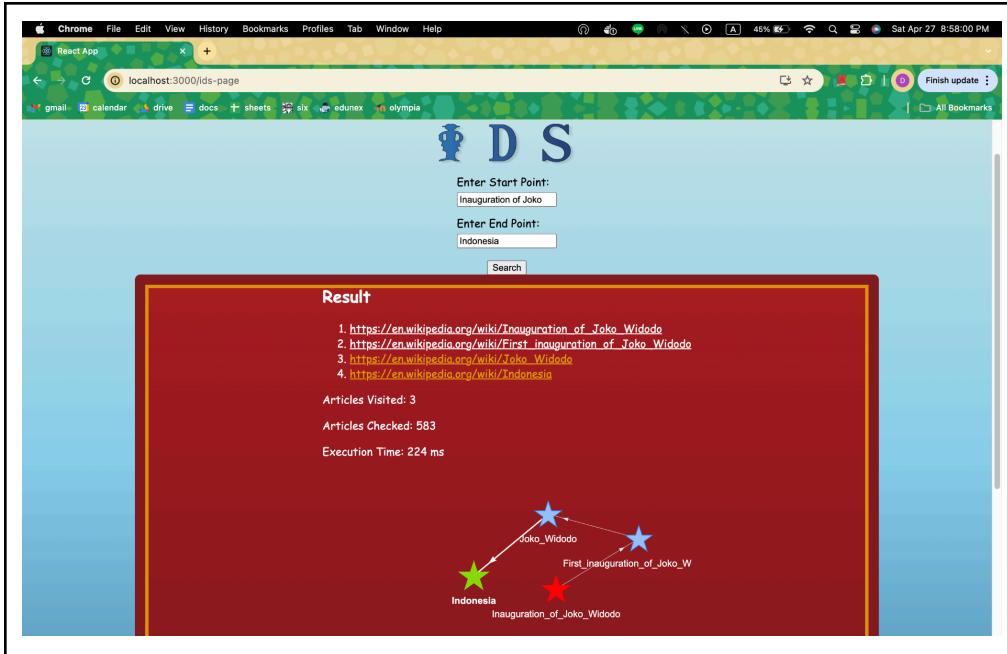
8. Pencarian 3 derajat: Vario → Taylor Swift



9. Pencarian 3 derajat: Mathematics → Suicide



10. Pencarian 3 derajat: Inauguration of Joko Widodo → Indonesia



4.4 Analisis Hasil Pengujian

Menurut kami, penjelajahan graf untuk menyelesaikan permasalahan WikiRace lebih efektif dengan menggunakan algoritma BFS atau *Breadth First Search*. Hal ini dikarenakan algoritma BFS melakukan pencarian secara melebar, yang berarti ia akan mengeksplorasi semua node pada kedalaman tertentu sebelum melanjutkan ke kedalaman berikutnya. Dengan demikian, BFS cenderung menemukan solusi dengan jumlah langkah minimum, karena ia mengeksplorasi terlebih dahulu node-node yang lebih dekat dengan node awal.

Namun, secara teoritis, keefektifan sebuah algoritma juga tergantung pada sifat dari graf ataupun pencarian yang sedang diproses. Jika graf memiliki sifat yang sangat kompleks atau mengandung banyak cabang yang bercabang (derajat yang besar), yang berarti jarak antara startURL dengan targetURL sangat jauh, algoritma IDS atau *Iterative Deepening Search* juga dapat memberikan hasil yang baik. Hal ini karena IDS dapat bekerja efektif dalam graf yang besar dengan batasan kedalaman yang tidak pasti sementara BFS akan memerlukan waktu yang lebih lama karena ia harus memeriksa lebih banyak jumlah artikel untuk setiap node *child*.

Hanya saja, pada program kami, pencarian dengan BFS tetap lebih cepat dibandingkan IDS meskipun jarak antara startURL dengan targetURL cukup jauh. Hal ini karena pada algoritma BFS, diimplementasikan *batch searching* yang mempercepat proses pencarian.

Terdapat juga kendala dalam pengujian seperti apabila jarak penjelajahan antar startURL dan targetURL sangat jauh dan menyebabkan harus mengecek sangat banyak artikel dalam waktu cepat, IP Address seringkali terblokir yang membuat *searching* terputus. Namun, apabila time.Sleep() pada kode dinaikkan, kode akan menjadi lambat dalam memproses *searching*. Selain itu, juga terjadi race condition pada jumlah artikel yang diperiksa meskipun sudah diimplementasikan mutex.Lock() dan mutex.Unlock().

Selain itu, karena diimplementasikan *caching* untuk artikel-artikel yang telah dikunjungi, biasanya untuk pencarian pertama akan memakan waktu lebih lama dibanding pencarian selanjutnya, terutama apabila pencarian selanjutnya melalui artikel (node) yang sama seperti sebelumnya.

BAB V

PENUTUP

5.1 Kesimpulan

Dalam pembelajaran mata kuliah Strategi Algoritma IF2211, permainan Wikirace telah berhasil kami buat dengan mengimplementasikan metode penelusuran (searching) yaitu algoritma Breadth First Search (BFS) dan Iterative Deepening Search (IDS). Permainan Wikirace ini untuk dimulai dengan suatu artikel dan menelusuri artikel-artikel lain untuk menuju ke artikel tujuan dalam waktu yang paling singkat atau klik (artikel) paling sedikit.

Algoritma IDS memiliki cara kerja yang bertahap pada graf dengan memperluas kedalaman penelusuran secara iteratif hingga solusi ditemukan. Algoritma ini lebih efisien untuk graf yang sangat besar atau dalam kasus di mana memori terbatas, karena melakukan penelusuran secara mendalam tapi tetap membatasi ruang pencarian yang diperlukan. Sedangkan BFS, bekerja dengan mengeksplorasi semua node tetangga pada kedalaman saat ini terlebih dahulu sebelum bergerak ke kedalaman berikutnya.

Implementasi algoritma IDS dan BFS pada permainan Wikirace ini dibuat dalam bentuk website yang menggunakan bahasa pemrograman Go dengan bantuan pustaka React. Website menerima masukan berupa jenis algoritma, judul artikel awal, dan judul artikel tujuan. Program kemudian memberikan keluaran berupa jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel (dari artikel awal hingga artikel tujuan), dan waktu pencarian (dalam ms).

5.2 Saran

Kami memiliki beberapa saran dalam penggeraan proyek yang sejenis dengan tugas besar ini, seperti meningkatkan kecepatan kode, efisiensi algoritma, serta eksplorasi yang lebih lanjut mengenai bahasa pemrograman Go.

5.3 Refleksi

Pengalaman berharga yang kami dapat dari tugas besar ini yang memiliki tantangan yang meningkatkan kemampuan kami. Meskipun terdapat kendala dalam menghadapi

tugas ini seperti waktu pengerjaan yang kurang efektif dan efisien serta tenaga kerja yang terbatas, dengan kemampuan adaptasi, khususnya dalam memahami bahasa yang belum pernah kami coba (Golang) dan pengembangan *website* yang masih belum cukup baik, kami dapat menyelesaikan tugas ini. Pada akhirnya, kemampuan *problem solving*, *computational thinking*, dekomposisi masalah, komunikasi, serta kerjasama kami dapat meningkat. Selain itu, masing-masing dari kami juga bisa mengetahui lebih dalam mengenai pemrograman.

5.4 Ruang Perbaikan atau Pengembangan

Program yang kami buat masih jauh dari kata sempurna. Untuk kedepannya, perlu dilakukan pengembangan optimasi kinerja sistem, terutama dalam hal waktu eksekusi program, agar sistem dapat memberikan respons yang lebih cepat namun tetap akurat. Selain itu, perlu juga dilakukan penyempurnaan antarmuka pengguna (frontend) untuk meningkatkan kegunaan dan daya tarik visual sistem. Terakhir, perlu dilakukan uji coba lebih lanjut untuk memastikan keandalan sistem dalam berbagai kasus penggunaan guna memperbaiki *bug* atau kesalahan yang mungkin terjadi.

5.5 Link Repository

Link Github *repository* Tugas Besar 2 Mata Kuliah IF2211 Strategi Algoritma Kelompok Dinasti dapat diakses pada tautan berikut:
https://github.com/cupski/Tubes2_Dinasti.

5.6 Link Video

Link video demo Tugas Besar 2 Mata Kuliah IF2211 Strategi Algoritma Kelompok Dinasti dapat diakses pada tautan berikut:
https://drive.google.com/file/d/1yQrhTbveGRLPSVIawmXXAJzZn-CYjcRz/view?usp=s_haring.

DAFTAR PUSTAKA

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/BFS-DFS-2021-Bag1-2024.pdf>

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf>

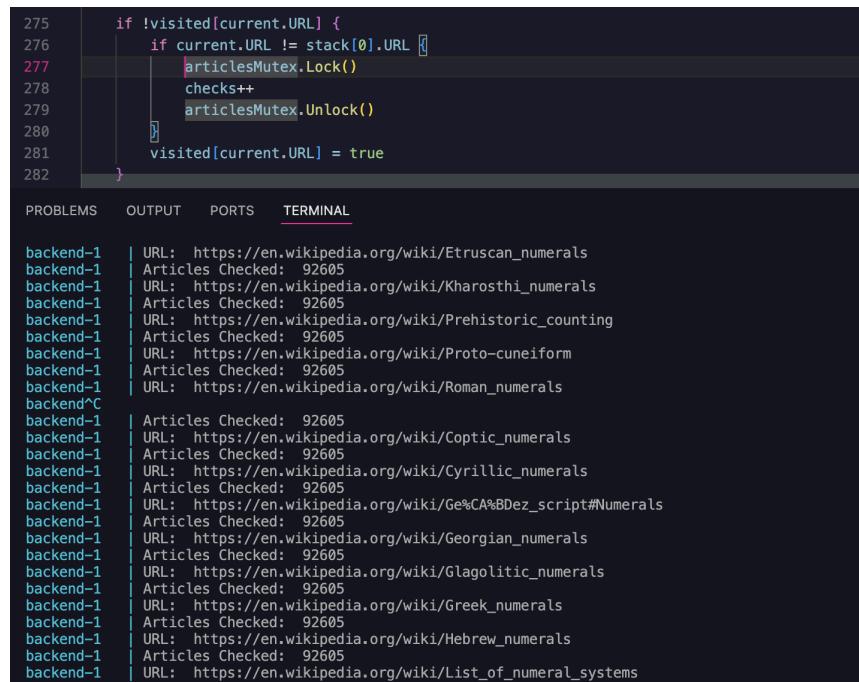
<https://dasarpemrogramangolang.novalagung.com/A-goroutine.html>

<https://www.tutorialspoint.com/race-condition-in-golang>

LAMPIRAN

```
URL: https://en.wikipedia.org/wiki/Iraqi_invasion_of_Iran
Articles Checked: 989147
URL: https://en.wikipedia.org/wiki/Iraqi_invasion_of_Kuwait
Articles Checked: 989147
URL: https://en.wikipedia.org/wiki/Bill_Clinton
Articles Checked: 989147
URL: https://en.wikipedia.org/wiki/European_Economic_Community
Articles Checked: 989147
signal: killed
```

Gambar 2. Bukan aku aja yang terbunuh, terminalnya juga :)



```
275     if !visited[current.URL] {
276         if current.URL != stack[0].URL [
277             articlesMutex.Lock()
278             checks++
279             articlesMutex.Unlock()
280         ]
281         visited[current.URL] = true
282     }
```

backend-1 | URL: https://en.wikipedia.org/wiki/Etruscan_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Kharosthi_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Prehistoric_counting
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Proto-cuneiform
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Roman_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Coptic_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Cyrillic_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Ge%C4%8Dez_script#Numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Georgian_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Glagolitic_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Greek_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/Hebrew_numerals
backend-1 | Articles Checked: 92605
backend-1 | URL: https://en.wikipedia.org/wiki/List_of_numeral_systems

Gambar 3. Aneh + bingung BGT, udah pakai mutex aja masih race condition.

Kak, susah banget kurang tenaga kerja :))