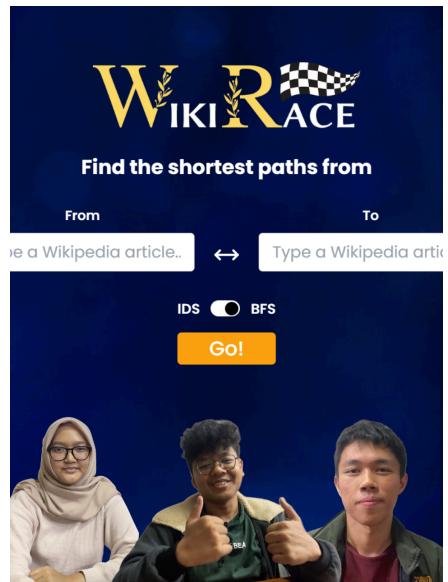


LAPORAN TUGAS BESAR 2 IF2211

STRATEGI ALGORITMA

*Pemanfaatan Algoritma IDS dan BFS
dalam Permainan WikiRace*



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

Disusun oleh:

Amalia Putri	(13522042)
Moh Fairuz Alauddin Yahya	(13522057)
Julian Chandra Sutadi	(13522080)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024**

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Kuasa, karena atas rahmat dan karunia-Nya, kami dapat menyelesaikan makalah ini dengan judul "Pemanfaatan Algoritma IDS dan BFS dalam Permainan WikiRace". Makalah ini disusun sebagai salah satu tugas mata kuliah Strategi Algoritma, dengan fokus pada penerapan konsep Algoritma *Iterative Deepening Search* (IDS) dan *Breadth First Search* (BFS) pada pembuatan permainan WikiRace.

Penyusunan makalah ini tidak lepas dari bantuan berbagai pihak. Kami mengucapkan terima kasih kepada dosen beserta asisten pembimbing mata kuliah Strategi Algoritma yang telah memberikan bimbingan dan panduan selama proses pembuatan makalah ini. Semoga makalah ini dapat memberikan kontribusi positif dalam pemahaman dan pengembangan suatu aplikasi dari konsep IDS dan BFS. Akhir kata, kami menyampaikan permohonan maaf atas segala keterbatasan dalam makalah ini, dan kami menerima dengan terbuka segala kritik dan saran yang bersifat membangun.

Bandung, 27 April 2024

Tim Penulis

DAFTAR ISI

KATA PENGANTAR.....	2
DAFTAR ISI.....	3
DAFTAR PEMBAGIAN TUGAS.....	5
BAB I.....	6
DESKRIPSI TUGAS.....	6
1.1. Abstraksi.....	6
BAB II.....	7
LANDASAN TEORI.....	7
2.1. Dasar Teori.....	7
2.1.1. Penjelajahan Graf.....	7
2.1.2. Iterative Deepening Search (IDS).....	8
2.1.3. Breadth First Search (BFS).....	9
BAB III.....	10
ANALISIS PEMECAHAN MASALAH.....	10
3.1 Langkah-Langkah Pemecahan Masalah.....	10
3.2 Proses Pemetaan Masalah.....	11
3.3 Fitur Fungsional dan Arsitektur Aplikasi.....	13
3.4 Contoh Ilustrasi Kasus.....	14
BAB IV.....	15
IMPLEMENTASI DAN PENGUJIAN.....	15
4.1 Spesifikasi Teknis Program.....	15
a. Iterative Deepening Search (IDS).....	15
b. Breadth First Search (BFS).....	16
c. Multithreading.....	17
4.2 Tata Cara Penggunaan Program.....	19
a. Laman Home (sebelum web scraping).....	19
b. Laman Home (setelah web scraping).....	20
c. Laman About.....	21
d. Laman Authors.....	22
4.3 Langkah-Langkah Penggunaan Website WikiRace.....	23
4.4 Hasil Pengujian.....	24
4.5 Analisis Desain Solusi Algoritma.....	28
BAB V.....	30
KESIMPULAN, SARAN, DAN REFLEKSI.....	30
5.1. Kesimpulan.....	30
5.2. Saran.....	30
5.3. Refleksi.....	31

DAFTAR PUSTAKA.....	32
LAMPIRAN.....	34
Repository.....	34
Youtube.....	34

DAFTAR PEMBAGIAN TUGAS

KEGIATAN		PIC
ALGORITMA	<i>Iterative Deepening Search (IDS)</i>	13522080
	<i>Breadth First Search (BFS)</i>	13522057
WEBSITE	Front-End	13522042
	Back-End	13522057
LAPORAN	Bab 1: Deskripsi Masalah	13522042 13522057 13522080
	Bab 2: Landasan Teori	13522042 13522057
	Bab 3: Analisis Pemecahan Masalah	13522057 13522080
	Bab 4: Implementasi dan Uji Coba	13522042 13522080 13522057
	Bab 5: Kesimpulan, Saran, dan Refleksi	13522042 13522057 13522080
BONUS	Visualisasi	13522042
	Docker	13522057
	Rute terpendek dengan durasi kurang dari satu menit	13522057
	Video	13522042 13522057 13522080

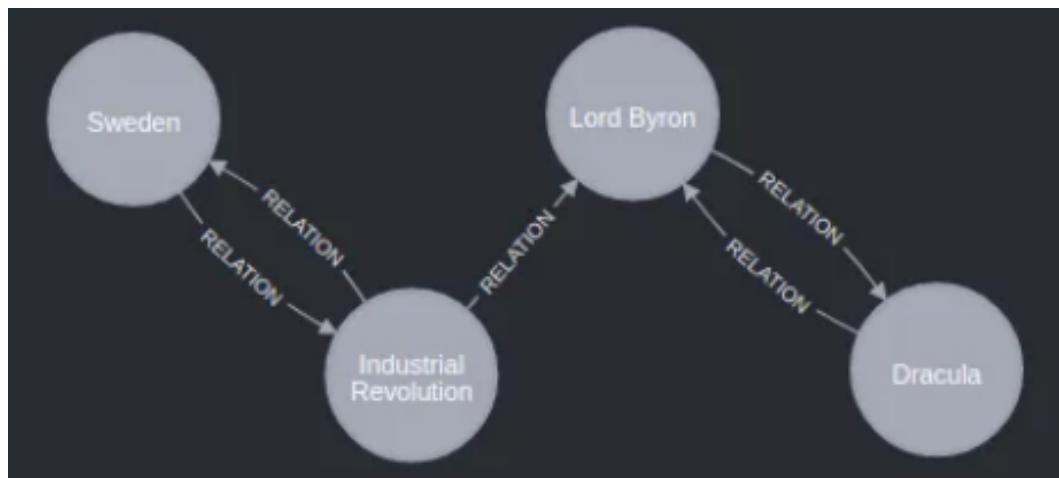
Tabel 1. Pembagian tugas dalam pembuatan Tugas Besar

BAB I

DESKRIPSI TUGAS

1.1. Abstraksi

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 1.1.1. Ilustrasi Graf WikiRace

(Sumber: https://miro.medium.com/v2/resize:fit:1400/1*jxmEbVn2FFWybZsIicJCWO.png)

IF2211 Strategi Algoritma – Tugas Besar 2 1

BAB II

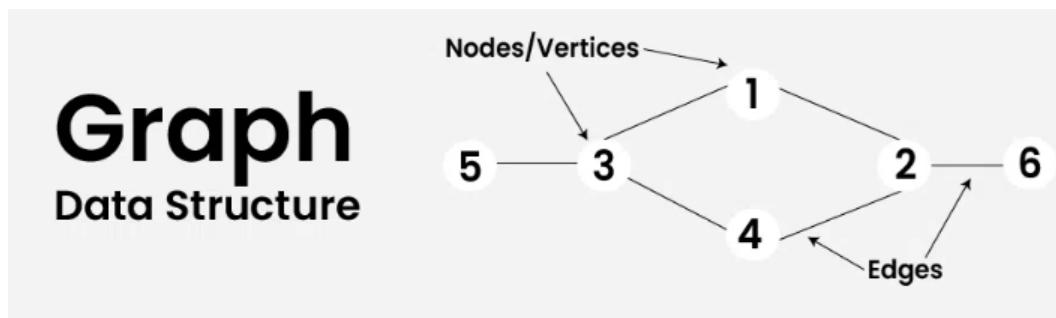
LANDASAN TEORI

2.1. Dasar Teori

2.1.1. Penjelajahan Graf

Graf adalah struktur data non-linear yang terdiri dari vertex (simpul) dan edge (tepi). Vertex terkadang disebut juga simpul, sedangkan edge berupa garis atau lengkung yang menghubungkan dua simpul manapun dalam graf. Secara lebih formal, graf tersusun dari kumpulan vertex (V) dan kumpulan edge (E). Graf dinyatakan dengan $G(V, E)$.

Struktur data graf merupakan alat yang ampuh untuk merepresentasikan dan menganalisis hubungan kompleks antara objek atau entitas. Graf sangat berguna dalam bidang-bidang seperti analisis jaringan sosial, sistem rekomendasi, dan jaringan komputer. Dalam bidang ilmu data olahraga, struktur data graf dapat digunakan untuk menganalisis dan memahami dinamika performa tim dan interaksi antar pemain di lapangan.



Gambar 2.1.1.1. Data Struktur Graf

(Sumber: <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/>)

Konsep ini menjelaskan tentang pengunjungan simpul-simpul di dalam graf dengan cara yang sistematik dengan cara pencarian melebar atau *breadth first search (BFS)* dan pencarian mendalam atau *depth first search (DFS)* dari graf yang diasumsikan terhubung. Terdapat dua pendekatan, yakni graf statis dan graf dinamis, di mana graf statis merupakan graf yang sudah terbentuk sebelum proses pencarian dilakukan dan graf dinamis merupakan graf yang terbentuk saat proses pencarian dilakukan. Dalam konsep penjelajahan graf, digunakan aplikasi

backtracking yang berguna untuk memecahkan persoalan pencarian solusi yang memiliki banyak alternatif pilihan selama pencarian. Untuk mempermudah pemrograman *backtracking* umum digunakan untuk implementasi.

2.1.2. Iterative Deepening Search (IDS)

Algoritma *iterative deepening search* (IDS) merupakan algoritma pencarian untuk menjelajahi graf secara efisien dengan melakukan DFS namun secara iteratif melakukan peningkatan nilai kedalaman-cutoff sampai solusi ditemukan. IDS dijamin menemukan solusi terpendek jika solusi ada dan menggunakan memori yang lebih sedikit dibandingkan DFS dengan kedalaman tak terbatas. Namun, IDS bisa lebih lambat dari BFS untuk graf tertentu dan memerlukan batas kedalaman maksimum yang ditentukan oleh pengguna.

DFS pertama-tama menelusuri simpul dengan melalui satu simpul yang berdekatan dengan akar, kemudian simpul berdekatan berikutnya. Masalah dengan pendekatan ini adalah jika ada simpul yang dekat dengan akar namun tidak dalam beberapa subgraf yang pertama kali dijelajahi oleh DFS, maka DFS mencapai simpul tersebut sangat lambat. Selain itu, DFS mungkin tidak menemukan jalur terpendek ke suatu simpul (dalam hal jumlah tepi). BFS bergerak level demi level, tetapi membutuhkan lebih banyak ruang. Ruang yang dibutuhkan oleh DFS adalah $O(d)$ di mana d adalah kedalaman graf, tetapi ruang yang dibutuhkan oleh BFS adalah $O(n)$ di mana n adalah jumlah simpul dalam graf.

Algoritma IDS diimplementasikan dengan menggunakan struktur data *stack* karena memiliki alasan-alasan sebagai berikut.

- 1) Algoritma IDS adalah variasi dari Depth-First Search (DFS) yang menggunakan pendekatan *Last In First Out* (LIFO). Ini berarti bahwa simpul terakhir yang ditambahkan ke dalam struktur data adalah simpul pertama yang dijelajahi. *Stack* secara alami mendukung operasi LIFO dan memungkinkan IDS untuk mengikuti pola penelusuran ke bawah (ke dalam) hingga batas kedalaman tertentu dengan mudah.
- 2) Algoritma IDS memerlukan *backtracking* yang efisien, yaitu kemampuan untuk mundur ke simpul sebelumnya dan melanjutkan penjelajahan dari sana. *Stack* memungkinkan penambahan dan penghapusan simpul di "atas" tumpukan tanpa

menganggu urutan simpul lain, yang ideal untuk *backtracking* dalam penelusuran DFS.

- 3) Penggunaan *stack* dalam algoritma IDS memudahkan implementasi algoritma yang meniru DFS dengan batasan kedalaman yang meningkat pada setiap iterasi tanpa memerlukan perubahan struktural besar pada cara DFS beroperasi.

2.1.3. Breadth First Search (BFS)

Algoritma BFS (*Breadth First Search*) adalah salah satu algoritma penjelajahan graf yang paling populer. Algoritma BFS melakukan traversal terhadap seluruh simpul dengan pertama mengunjungi semua simpul yang bertetangga dengan simpul pertama kemudian memilih salah satu dari simpul yang bertetangga tersebut untuk diperlakukan sebagai simpul pertama. Pencarian dilakukan sampai ditemukan simpul yang diinginkan. Dalam pemrograman BFS diimplementasikan dengan menggunakan struktur data *queue*.

Algoritma BFS diimplementasikan dengan menggunakan struktur data *queue* karena memiliki alasan-alasan sebagai berikut.

- 1) Algoritma BFS menelusuri simpul berdasarkan urutan kedalaman dari root, mengunjungi semua simpul di satu kedalaman sebelum bergerak ke kedalaman berikutnya. *Queue* mendukung pendekatan *First In First Out* (FIFO), di mana simpul pertama yang ditambahkan adalah simpul pertama yang dijelajahi, memastikan bahwa penelusuran dilakukan secara level demi level.
- 2) Penggunaan *queue* memudahkan manajemen simpul-simpul yang perlu dijelajahi secara berurutan pada kedalaman yang sama. Hal ini penting untuk memastikan bahwa semua simpul pada kedalaman tertentu dijelajahi sebelum melanjutkan ke kedalaman selanjutnya.
- 3) Dengan menyimpan semua simpul pada kedalaman tertentu dalam *queue*, BFS dapat secara efektif dan efisien menelusuri graf atau graf dari atas ke bawah, memastikan bahwa setiap level dijelajahi secara komprehensif sebelum pindah ke level berikutnya.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

3.1.1. Langkah-Langkah Pemecahan Masalah IDS

- 1) Pertama akan dilakukan *pop* terhadap stack untuk didapatkan simpul yang akan ditelusuri saat ini.
- 2) Selanjutnya dilakukan pemeriksaan. Jika simpul yang ditelusuri adalah URL destinasi, maka proses dilanjutkan.
- 3) Jika simpul yang ditelusuri bukan URL destinasi, maka simpul-simpul anak akan dibangkitkan dengan melakukan *scraping*.
- 4) Selanjutnya, untuk setiap anak yang dibangkitkan yang belum pernah dibangkitkan sebelumnya, informasi simpul saat ini akan disimpan sebagai *parent* dari anak tersebut.
- 5) Proses ini dilanjutkan dan berhenti jika semua simpul yang dibangkitkan dan disimpan pada stack sudah diperiksa apakah dapat mencapai destinasi atau tidak.

3.1.2. Langkah-Langkah Pemecahan Masalah BFS

- 1) Mulai dengan mengambil simpul pertama pada suatu kedalaman dari queue yang berisi string yang merupakan link, dan simpul ini disebut sebagai parent simpul.
- 2) Selanjutnya, dilakukan pemeriksaan untuk memastikan apakah hasil deque tersebut sesuai dengan URL yang dituju. Jika hasil deque sesuai, tandai untuk proses pencarian hanya perlu sampai kedalaman yang sama dengan hasil deque.
- 3) Jika hasil deque tidak sesuai dengan URL tujuan, langkah berikutnya adalah melakukan pembangkitan simpul anak dari parent tersebut. Data dari parent simpul akan dibentuk sebagai graf baru untuk menyimpan relasi antara parent dan child.
- 4) Setelah pembangkitan simpul anak, simpul anak tersebut akan ditambahkan ke dalam queue untuk dilanjutkan proses pencarian lagi. Setiap simpul anak yang

ditambahkan ke queue akan mengambil peran sebagai parent simpul baru dalam iterasi berikutnya.

- 5) Proses ini akan terus berlanjut dengan melakukan pemeriksaan dan pembangkitan simpul anak hingga ditemukan jalur yang sesuai dengan tujuan.

3.2 Proses Pemetaan Masalah

3.2.1. Proses Pemetaan Masalah pada Algoritma IDS

- 1) Graf dan Simpul

Persoalan direpresentasikan dalam graf yang terdiri dari simpul-simpul (simpuls) yang merepresentasikan artikel-artikel Wikipedia.

- 2) Parent-Child Relationship:

Setiap simpul dalam IDS WikiRace memiliki relasi parent-child, di mana parent simpul adalah artikel Wikipedia yang sedang dievaluasi, dan child simpul adalah artikel Wikipedia yang terhubung dengan parent simpul melalui link. Yang menjadi parent adalah simpul dengan kedalaman lebih dalam.

- 3) Prioritas Pencarian:

Pencarian jalur dalam IDS WikiRace dilakukan dengan berusaha mencapai solusi paling rendah dengan mengaplikasikan DFS pada tiap tingkat kedalaman.

- 4) Kedalaman Pencarian

Kedalaman pencarian IDS WikiRace bergantung pada kedalaman pertama solusi ditemukan secara DFS namun maksimum sebesar kedalaman awal yang sudah ditentukan.

- 5) Tujuan Pencarian

Tujuan utama dari pencarian dalam IDS WikiRace adalah menemukan jalur terpendek atau paling optimal dari artikel awal (start) ke artikel tujuan (destination) di Wikipedia namun dengan memori yang lebih sedikit dibandingkan dengan BFS.

- 6) Solusi dan Jalur Optimal

Solusi dari IDS WikiRace adalah jalur atau rangkaian artikel Wikipedia yang harus dilewati untuk mencapai artikel tujuan dari artikel awal.

- 7) Kompleksitas Waktu dan Ruang

Misalkan $b = \text{lebar}$ dan $d = \text{kedalaman}$. Kompleksitas waktu IDS adalah $O(b^d)$ dan kompleksitas ruang $O(bd)$. Dalam kasus ini, kedalaman maksimal atau b adalah 6. Umumnya kompleksitas ruang IDS lebih baik, namun karena diimplementasikan untuk mencari solusi banyak maka menjadi sama dengan BFS.

3.2.2. Proses Pemetaan Masalah pada Algoritma BFS

1) Graf dan Simpul

Persoalan direpresentasikan dalam graf yang terdiri dari simpul-simpul (simpuls) yang merepresentasikan artikel-artikel Wikipedia.

2) Parent-Child Relationship

Setiap simpul dalam BFS WikiRace memiliki relasi parent-child, di mana parent simpul adalah artikel Wikipedia yang sedang dievaluasi, dan child simpul adalah artikel Wikipedia yang terhubung dengan parent simpul melalui link.

3) Prioritas Pencarian

Pencarian jalur dalam BFS WikiRace dilakukan dengan prioritas pemrosesan simpul yang memiliki kedalaman yang sama

4) Kedalaman Pencarian

Kedalaman pencarian dalam BFS WikiRace ditentukan oleh jumlah langkah atau kedalaman mencapai link yang dituju.

5) Tujuan Pencarian

Tujuan utama dari pencarian dalam BFS WikiRace adalah menemukan jalur terpendek atau paling optimal dari artikel awal (start) ke artikel tujuan (destination) di Wikipedia.

6) Solusi dan Jalur Optimal

Solusi dari BFS WikiRace adalah jalur atau rangkaian artikel Wikipedia yang harus dilewati untuk mencapai artikel tujuan dari artikel awal.

7) Kompleksitas Waktu dan Ruang

Misalkan $V = \text{jumlah simpul}$, $E = \text{jumlah sisi}$, dan $P = \text{jumlah yang telah dilalui pada graf}$, maka kompleksitas waktu adalah $O(|V|)$, yakni menunjukkan

bahwa dalam kasus terburuk, setiap vertex akan dikunjungi sekali. Sedangkan kompleksitas ruangnya adalah $O(V + E + P)$, di mana $O(V)$ untuk menyimpan informasi tentang status (dikunjungi atau belum), jarak dari sumber, dan pendahulu dalam P . Sedangkan $O(E)$ untuk menyimpan informasi tentang koneksi antar vertex. Terakhir, $O(P)$ untuk menyimpan semua jalur yang mungkin dari simpul sumber ke simpul lain.

3.3 Fitur Fungsional dan Arsitektur Aplikasi

3.3.1. *Front-End*

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.

Pada sistem ini akan digunakan framework untuk memudahkan pengembangan web. 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.

3.3.2. *Back-End*

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. Pada sistem ini, API digunakan untuk melakukan integrasi antar sistem Next.js (front-end) dan Gin (framework web Golang yang digunakan).

Pada sistem ini framework yang digunakan adalah Gin yang merupakan berbasis Golang. Gin dipilih karena mendukung *tech stack* yang menggunakan bahasa pemrograman Golang. Gin adalah kerangka kerja web yang kuat dan ringan untuk membangun aplikasi backend dalam bahasa pemrograman Go (Golang). Dirancang untuk cepat, efisien, dan minimalis sambil menyediakan fitur-fitur esensial untuk mengembangkan aplikasi web dan API yang tangguh.

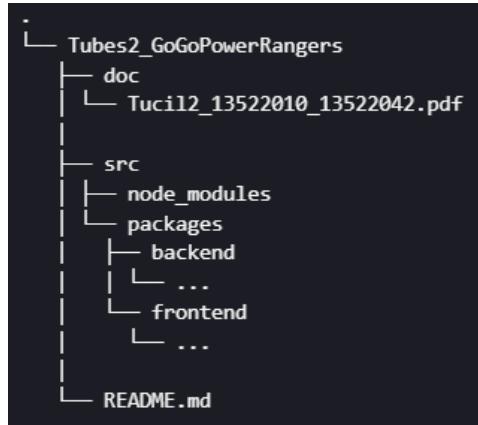
3.4 Contoh Ilustrasi Kasus

Julian adalah pengguna yang ingin mengetahui hubungan antara dua hal yang ia ketahui, misalnya hubungan antara Joko Widodo dengan Philosophy melalui Wikipedia. Ternyata, hubungan dari kedua hal tersebut dihubungkan oleh Gadjah Mada University. Setelah Julian telusuri, ia menjadi tahu bahwa Joko Widodo merupakan alumni dari Gadjah Mada University dan Philosophy merupakan disiplin ilmu yang ada di Gadjah Mada University. Selain itu, Julian tidak perlu berkeberatan untuk mencari atau menelusuri satu per satu hal di Wikipedia secara manual karena sudah ada website WikiRace yang memudahkan Julian dalam melakukan pencarian hubungan antar hal yang ingin diketahuinya.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program



Gambar 4.1.1. Struktur Folder

a. *Iterative Deepening Search (IDS)*

Algoritma IDS diimplementasikan dengan struktur data stack. Untuk efisiensi, stack tidak akan menampung seluruh link sampai tahap tersebut, melainkan disimpan pada variabel terpisah yaitu parents yang merupakan map dengan value semua link yang berada pada satu tingkat kedalaman di atas link yang menjadi key. Selain itu, juga disimpan daftar parent simpul yang pernah dikunjungi.

Setelah semua link dikunjungi dengan metode IDS, dari kumpulan simpul dan parents yang telah dibangkitkan akan dilakukan rekonstruksi untuk menemukan semua path yang berhasil menuju *destination*. Rekonstruksi ini dilakukan secara rekursif.

Tabel 4.1.1. Struktur program *Iterative Deepening Search (IDS)*

Fungsi	Penjelasan
func IDSHandlers(source string, destination string, maxDepth int) ([][]string, int, error)	Memanggil fungsi DFSHelper dengan kedalaman yang bertambah secara iteratif lalu mengembalikan sekumpulan <i>paths</i> yang adalah kumpulan link, beserta jumlah link yang ditelusuri.
func DFSHelper(source string,	Melakukan proses DFS dengan

destination string, maxDepth int) ([][string, int, error)	kedalaman maksimal berdasarkan yang dipanggil oleh fungsi IDSHandlers
func IDSHTTPHandler(c *gin.Context)	Menerima permintaan JSON yang berisi informasi sumber dan tujuan kemudian menjalankan algoritma IDS sambil mengukur waktu eksekusi. Kemudian, fungsi akan mengembalikan kumpulan link, <i>runtime</i> , dan jumlah link yang ditelusuri.

b. **Breadth First Search (BFS)**

Algoritma BFS dapat ditingkatkan dengan menggabungkan struktur data graf dan queue. Queue digunakan untuk menampung simpul saat ini yang dievaluasi setiap iterasi. Dalam upaya efisiensi, kami menggunakan variabel closestDistance untuk mencegah penambahan simpul pada kedalaman yang lebih dalam yang telah dievaluasi sebelumnya pada kedalaman yang lebih dangkal. Kami juga menggunakan variabel parent sebagai representasi graf, yang menyimpan array path parents dari currentsimpul untuk rekonstruksi paths yang diinginkan. Untuk memastikan bahwa array parents adalah unik, kami menggunakan variabel parentsVisited untuk menandai apakah parent telah ditambahkan. Semua variabel sebagai langkah optimasi kami susun menjadi dictionary dengan menggunakan currentsimpul sebagai key untuk mengakses valuenya.

Kami memilih representasi graf ini karena lebih optimal daripada tipe data matrix of paths atau tipe data lain yang kami uji, yang kami temukan lebih lambat dan kurang efisien. Sebagai langkah optimasi tambahan, kami menggunakan multithreading dalam pembentukan child simpuls dengan membagi tugas thread ke currentsimpul yang berbeda. Untuk mempertahankan prinsip BFS, kami menggunakan waiting group untuk setiap thread concurrent pada suatu kedalaman sebelum melanjutkan ke kedalaman yang lebih dalam. Multithreading juga diterapkan untuk meningkatkan kinerja saat rekonstruksi path menjadi susunan paths solusi secara rekursi.

Tabel 4.1.2. Struktur program *Breadth First Search* (BFS)

Fungsi	Penjelasan
--------	------------

func BFSHandlers(source string, destination string, maxDepth int) ([][]string, error)	Pemrosesan secara BFS dalam representasi queue dan graf serta mengembalikan sekumpulan <i>paths</i> berupa string yang merupakan link sebagai simpul dari graf yang terhubung
func BFSHTTPHandler(c *gin.Context)	Memproses nilai yang dikembalikan oleh fungsi BFSHandlers serta mengonversi <i>message</i> , <i>paths</i> , dan <i>runtime</i> dalam bentuk JSON untuk mengelola <i>back-end</i> .
func reconstructPath(parents *map[string][]string, source string, destination string) [][]string	Merekonstruksi hasil solusi path dengan memanfaatkan info dari parents secara rekursi hingga mencapai source url.

c. Multithreading

Dalam implementasi multithreading, kami membagi tugas-tugas pemrosesan ke dalam goroutine untuk meningkatkan efisiensi. Goroutine ini dapat berjalan secara bersamaan, memungkinkan pemrosesan data lebih cepat daripada menggunakan pendekatan single-threaded. Namun, untuk memastikan konsistensi data dan menghindari race condition, kami menggunakan prinsip mutex yang digunakan untuk mengunci akses ke data yang bersamaan agar tidak ada thread yang mengubah data tersebut ketika thread lain sedang mengaksesnya.

Selain itu, kami menggunakan waitGroup yang memungkinkan untuk menambahkan dan menahan thread agar tidak melanjutkan eksekusi sebelum tugas-tugas yang seharusnya dilakukan oleh thread-thread lain selesai dieksekusi dalam rangka menjaga prinsip dari algoritma yang ada.

Dengan menggunakan semaphore, kami juga dapat membatasi jumlah thread yang dapat berjalan secara bersamaan dengan sweet spot 200-300. Hal ini membantu dalam mengelola sumber daya dan mencegah overload pada sistem, serta mengurangi risiko pembatasan akses dari layanan Wikipedia yang membatasi akses bersamaan dari satu IP.

d. Scraping

Scraping adalah proses otomatis untuk mengambil informasi dari halaman web. Pada program ini, scraping dilakukan untuk mengunjungi child simpul dari

suatu link yang ditekan. Penggunaan GoQuery sebagai library utama memungkinkan pencarian tag '`<a>`' yang mengandung link dengan 'href' yang diawali dengan '/wikipedia' dan tidak mengandung karakter ":" (titik dua). Hal ini membatasi scraping pada link-link yang relevan dengan Wikipedia dan menghindari link ke halaman "Main_Page". Konten yang diambil dibatasi pada tag 'main' dalam '`<div>`' dengan id 'mw-content-text', sehingga fokus pada informasi yang penting dan relevan untuk proses WikiRace. Dengan menerapkan constraint dan fokus pada bagian-bagian yang relevan, program dapat melakukan scraping dengan lebih efisien dan akurat, sesuai dengan kebutuhan dalam mencari jalur antara dua artikel Wikipedia.

Fungsi	Penjelasan
<code>func ScrapperHandlerLinkBuffer(url string) ([]string, error)</code>	Melakukan scraping dengan library goquery dengan constraint yang telah disebutkan diatas.

e. Gin

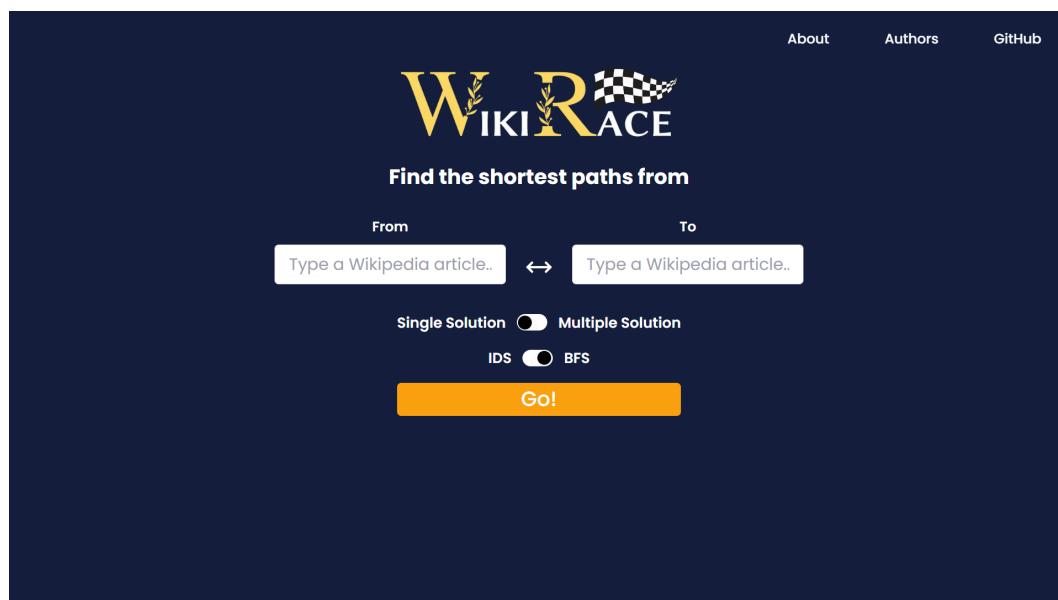
Tabel 4.1.3. Struktur program *GIN*

Fungsi	Penjelasan
<code>func SetupRouter() *gin.Engine</code>	Mengkonfigurasikan <i>method</i> GET dan POST pada beberapa <i>endpoint</i> , yakni untuk: <ol style="list-style-type: none"> 1) Mengetahui informasi Packet Internet Gopher (PING) 2) Mengirim data ke server: <i>link</i> Wikipedia, <i>autocomplete</i>, hasil kalkulasi dengan BFS dan IDS, dan pengelola <i>web scraping</i>.
<code>func CORSMiddleware() gin.HandlerFunc</code>	Menerima permintaan dari domain atau subdomain yang berbeda, memastikan bahwa aplikasi tersebut dapat berinteraksi dengan aman melalui browser dari berbagai asal tanpa melanggar kebijakan keamanan yang sama asal dari browser.

4.2 Tata Cara Penggunaan Program

a. Laman *Home* (sebelum *web scraping*)

Ketika memasuki aplikasi WikiRace, pengguna akan berada pada laman *home* atau laman utama. Pada laman tersebut, pengguna dapat memasukkan *input* berupa judul *link Wikipedia* dari simpul *start (source)* dan *end (destination)*. Serta terdapat komponen *switch* sehingga pengguna dapat memilih antara algoritma IDS atau BFS untuk melakukan *searching* pada *web scraping*.



Gambar 4.2.1. Tampilan laman *home*

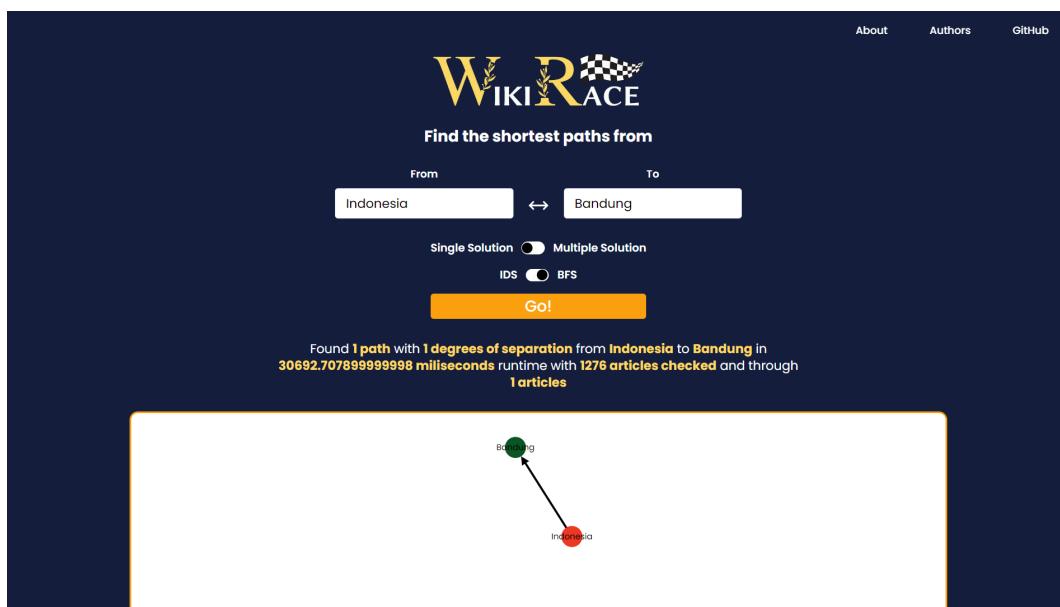
Tabel 4.2.1. Detail Komponen Halaman Utama

No	Nama	Tipe	Deskripsi
1	AutoComplete From → To	Input	Pengguna dapat memasukkan judul Wikipedia
2	IDS-BFS Algorithm Option	Switch	Pengguna dapat memilih jenis algoritma yang ingin digunakan
3	Single-Multiple Algorithm Solution Option	Switch	Pengguna dapat menerima hasil jika <i>single</i> hanya satu rute, sedangkan <i>multiple</i> bisa jadi

			ada lebih dari satu rute
4	Searching & Save	Button	Pengguna dapat melakukan proses pencarian dan menyimpan data (<i>cache</i>)
5	Navbar: About, Authors, GitHub	Link/Button	Pengguna dapat melihat informasi tentang cara penggunaan WikiRace, profil pengguna, dan GitHub WikiRace

b. Laman *Home* (setelah *web scraping*)

Pertama-tama, pengguna akan memasukkan judul *Wikipedia* pada *link source (start)* dan *link destination (end)*. Setelah itu, pengguna menekan tombol “Go!” dan akan ditampilkan hasil (jika ditemukan) akan ditampilkan visualisasi graf antar simpul serta *path(s)* yang cocok dan ditemukan. Selain itu, ditampilkan juga waktu eksekusi, jumlah artikel yang telah dicek, serta keterangan *path* awal dan akhir yang telah diproses.



Gambar 4.2.3. Visualisasi Graf antar Simpul



Gambar 4.2.4. *Individual Paths*

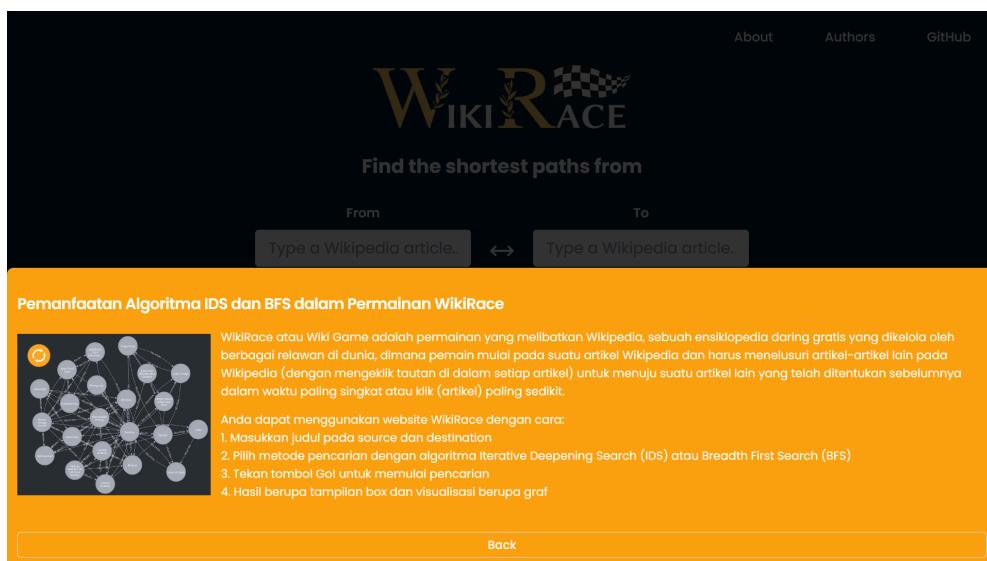
Tabel 4.2.2. Detail Komponen Pencarian

No	Nama	Tipe	Deskripsi
1	Deskripsi Hasil	Label	Pengguna dapat mengetahui informasi tentang simpul asal dan tujuan, waktu eksekusi, jumlah kedalaman, jumlah artikel yang telah dicek, dan jumlah artikel yang ditemukan
2	Visualisasi Graf	Node	Pengguna dapat melihat visualisasi graf dari antar <i>link</i> untuk menunjukkan hubungan antar topik yang dicari
3	Individual Paths	Link	Pengguna dapat mengklik <i>link</i> yang tertera dari semua simpul terhubung yang ditemukan

c. **Laman *About***

Pada laman ini, pengguna dapat mendapatkan informasi tentang cara memanfaatkan website WikiRace dari deskripsi dan langkah-langkah yang tertera.

Selain itu, terdapat visualisasi sedikit konsep dari graf sehingga pengguna dapat mengetahui sedikit konsep dari cara kerja WikiRace.



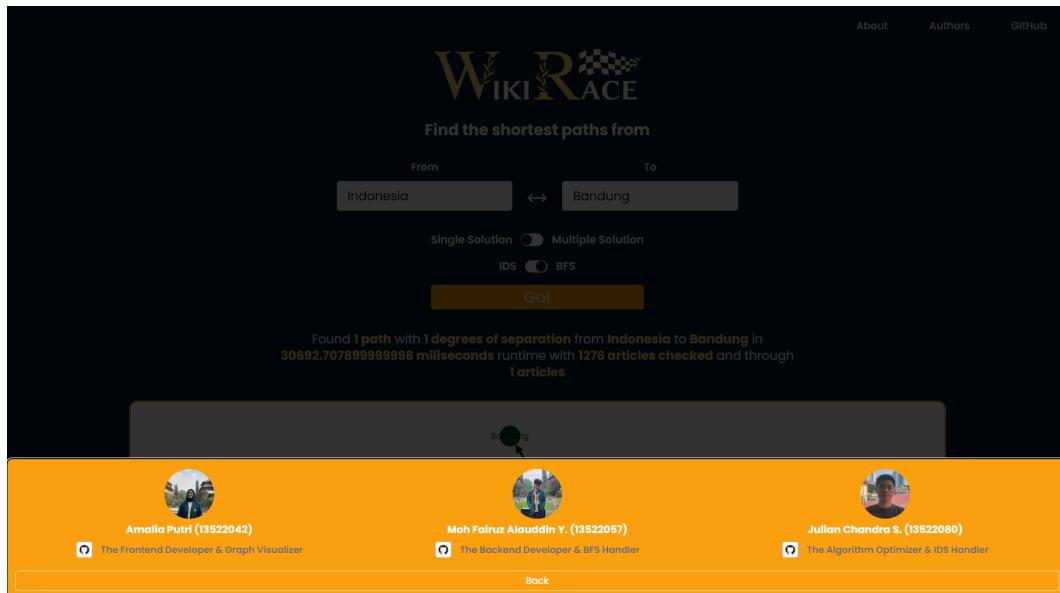
Gambar 4.2.5. Laman *About*

Tabel 4.2.3. Detail Komponen *About*

No	Nama	Tipe	Deskripsi
1	Deskripsi WikiRace	Label	Pengguna dapat mengetahui informasi tentang WikiRace terutama fungsi penggunaannya
2	Deskripsi Cara Penggunaan	Label	Pengguna dapat melihat langkah-langkah penggunaan WikiRace
3	Gambar Graf Interaktif	Image	Pengguna dapat melihat melakukan <i>zoom in-out</i> untuk memahami konsep dasar graf
4	Back	Button	Pengguna dapat kembali ke laman utama

d. Laman *Authors*

Pada laman ini, pengguna dapat mengakses informasi tambahan terkait profil singkat *authors*, berupa nama lengkap, NIM, peran, dan GitHub.



Gambar 4.2.6. Tampilan laman *Authors*

Tabel 4.2.3. Detail Komponen *Authors*

No	Nama	Tipe	Deskripsi
1	Profil <i>Author</i>	Foto, Label, Icon	Pengguna dapat mengetahui informasi <i>author</i> dan menelusuri lebih lanjut profil GitHub <i>author</i>
2	Back	Button	Pengguna dapat kembali ke laman utama

4.3 Langkah-Langkah Penggunaan Website WikiRace

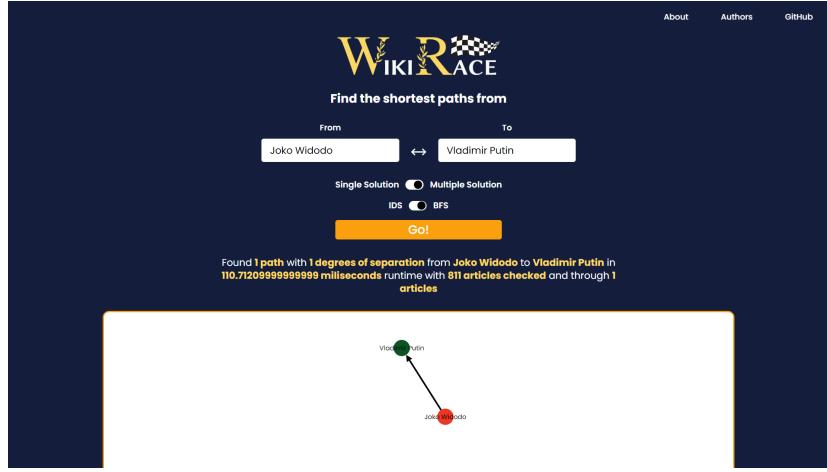
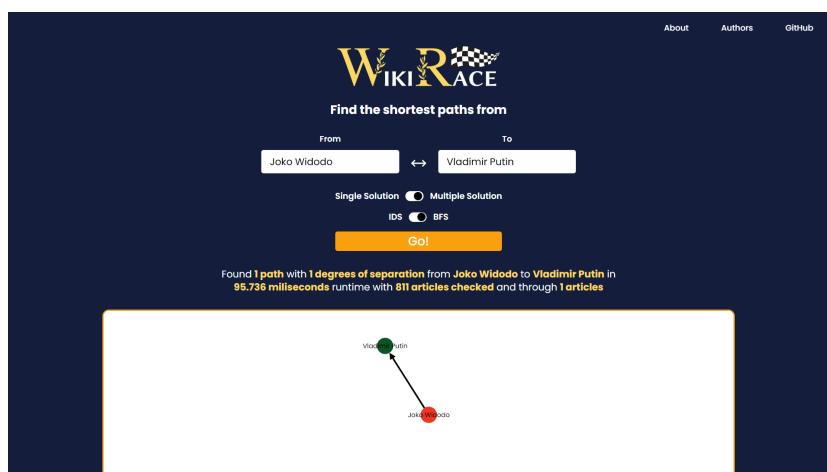
Berdasarkan erdapat panduan atau langkah-langkah penggunaan Website WikiRace, yakni sebagai berikut.

- 1) Klik menu *About* untuk mengetahui panduan dari Website WikiRace.

- 2) Masukkan judul topik atau suatu objek yang ingin dicari, misalnya Indonesia, ayam, kucing, dan sebagainya pada kedua *input* Autocomplete dari asal ke tujuan.
- 3) Klik tombol “Go!” untuk memproses pencarian dan mendapatkan hasil.
- 4) Lihat hasil dalam bentuk deskripsi dan visualisasi graf.
- 5) Klik link pada *Individual Paths*, jika ingin melihat artikelnya lebih lanjut.

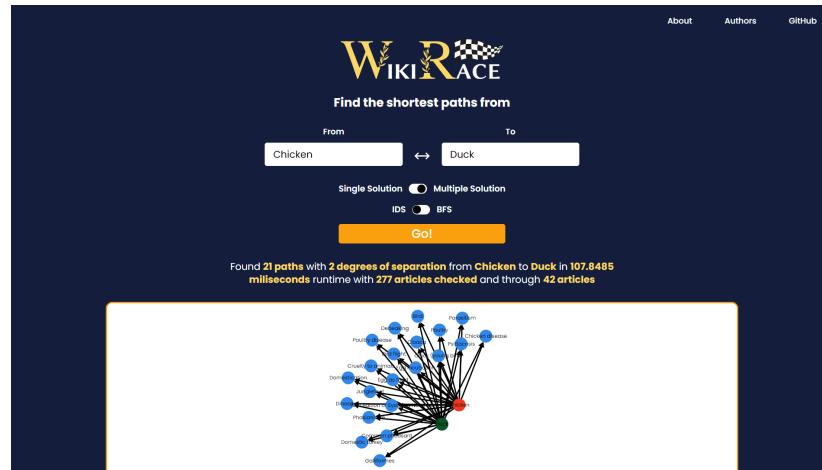
4.4 Hasil Pengujian

Tabel 4.3.1 Hasil Pengujian Komponen

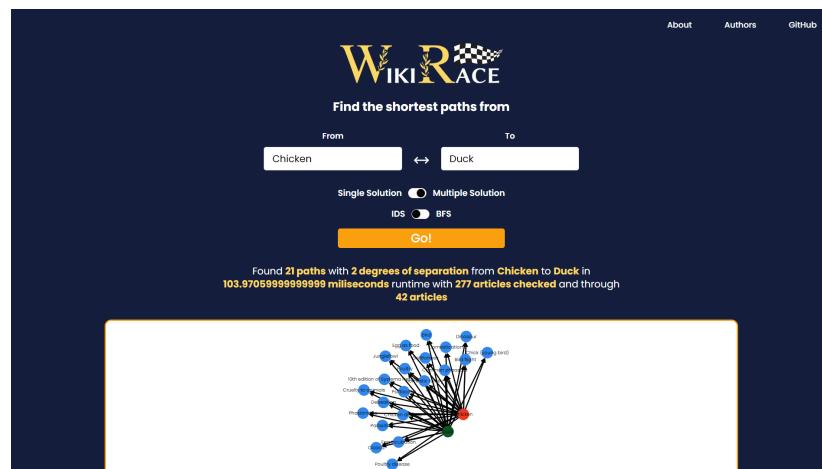
Fitur	Hasil Pengujian
Pencarian dengan Algoritma Iterative Deepening Search (IDS)	<p>1. Jokowi → Putin</p> <p>Tanpa Cache:</p>  <p>Dengan Cache:</p> 

2. Chicken → Duck

Tanpa Cache:

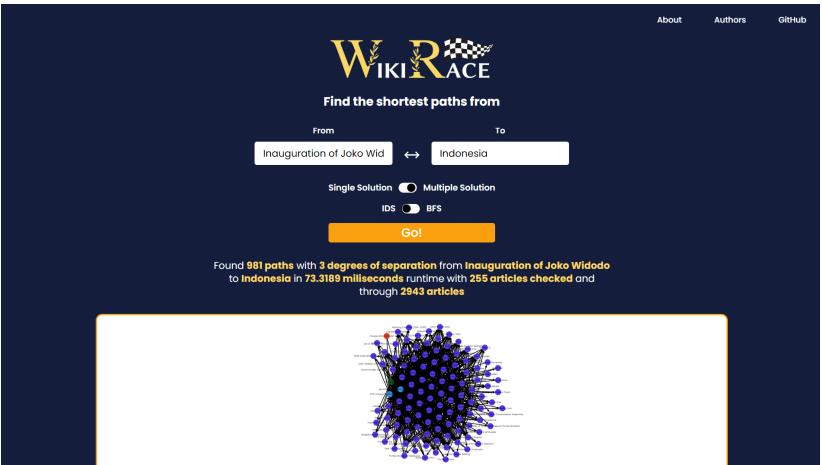
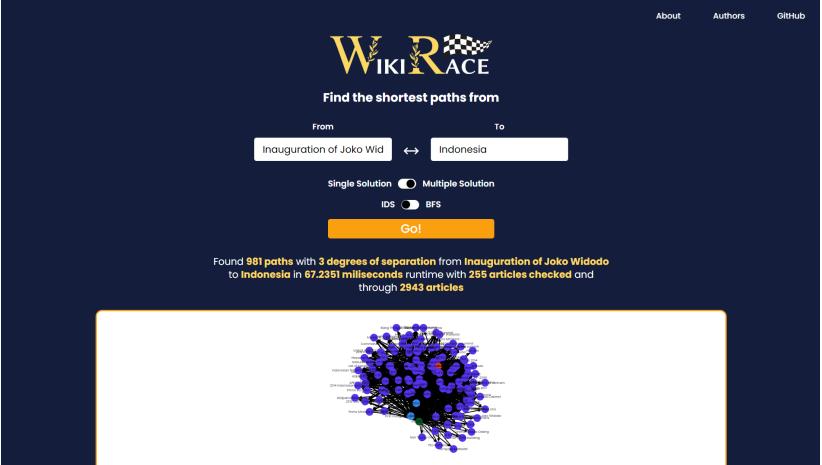
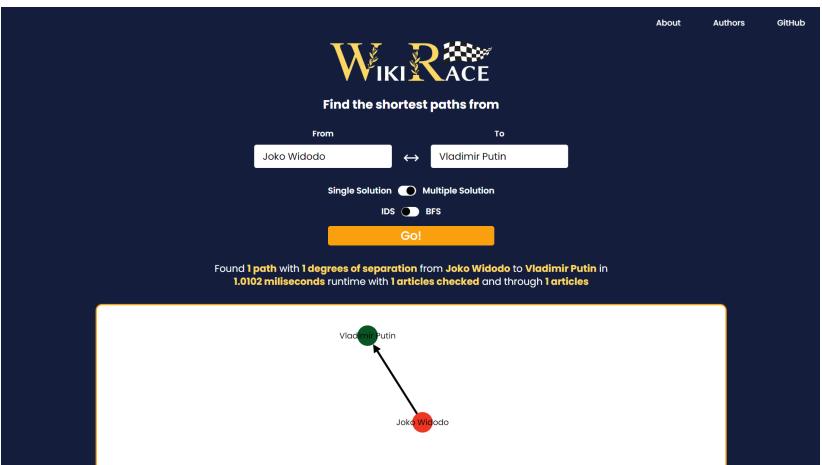


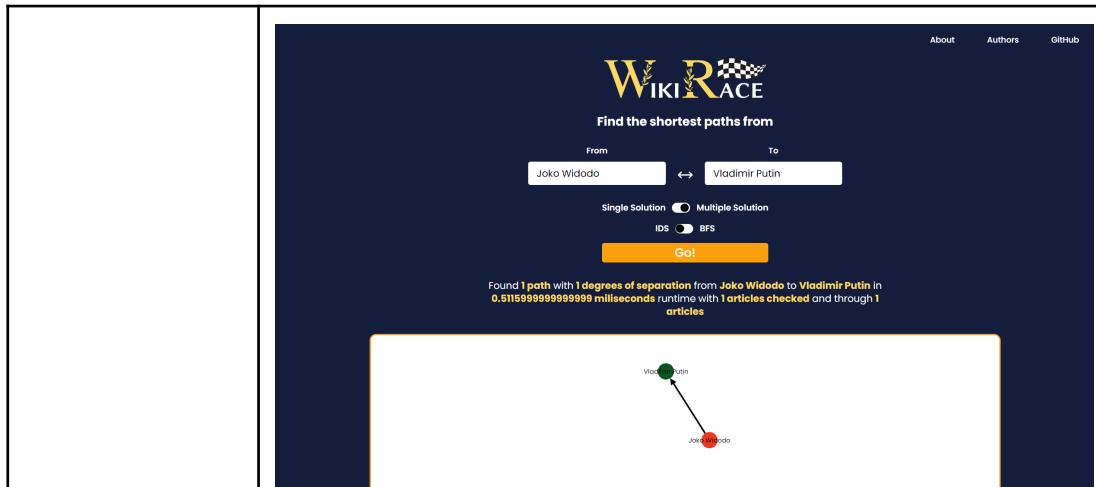
Dengan Cache:



3. Inauguration of Joko Widodo → Indonesia

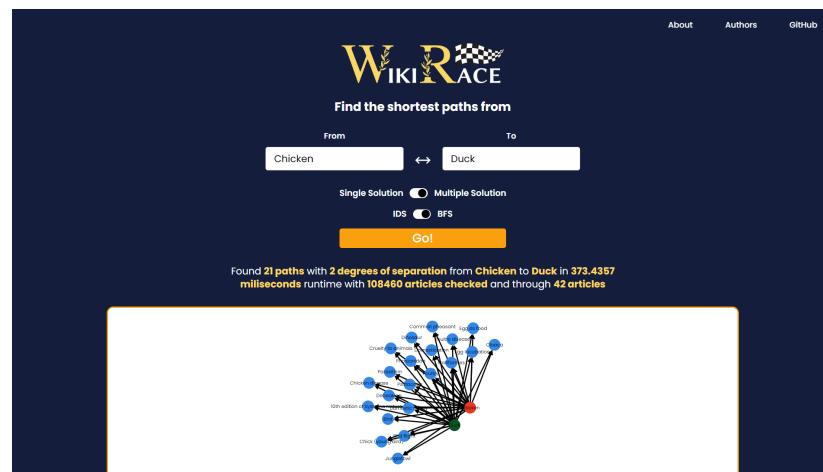
Tanpa Cache:

	 <p>Dengan Cache:</p> 
<p>Pencarian dengan Algoritma <i>Breadth First Search (BFS)</i></p>	<p>1. Jokowi → Putin</p> <p>Tanpa Cache:</p>  <p>Dengan Cache:</p>

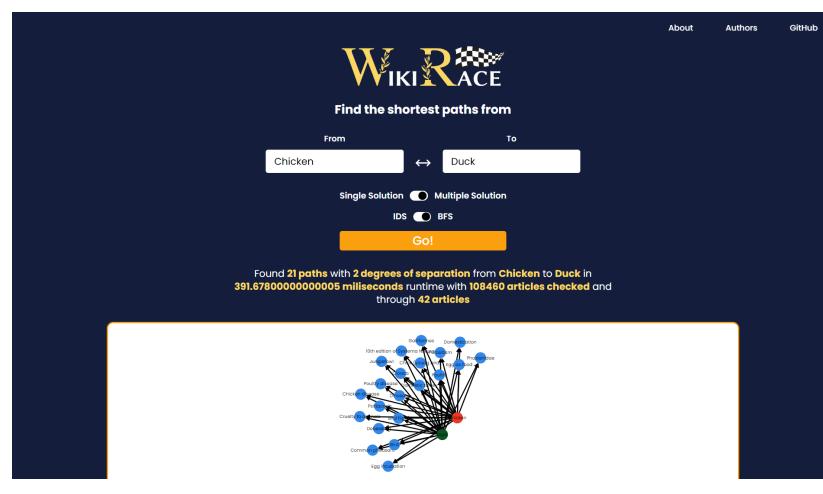


2. Chicken → Duck

Tanpa Cache:

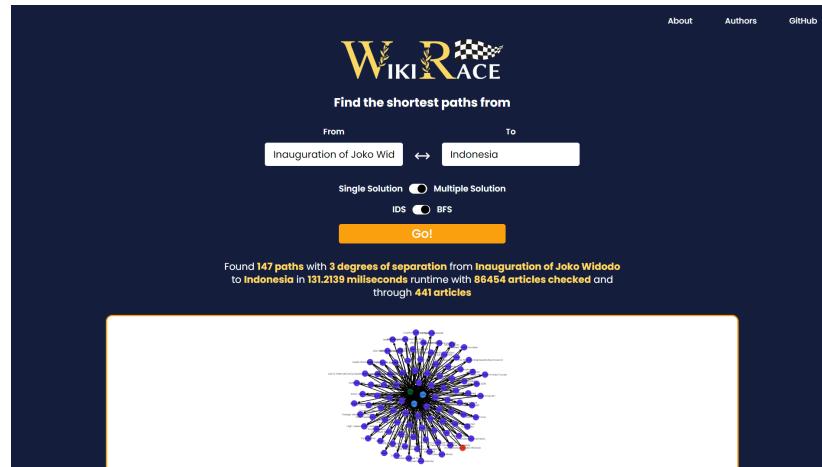


Dengan Cache:

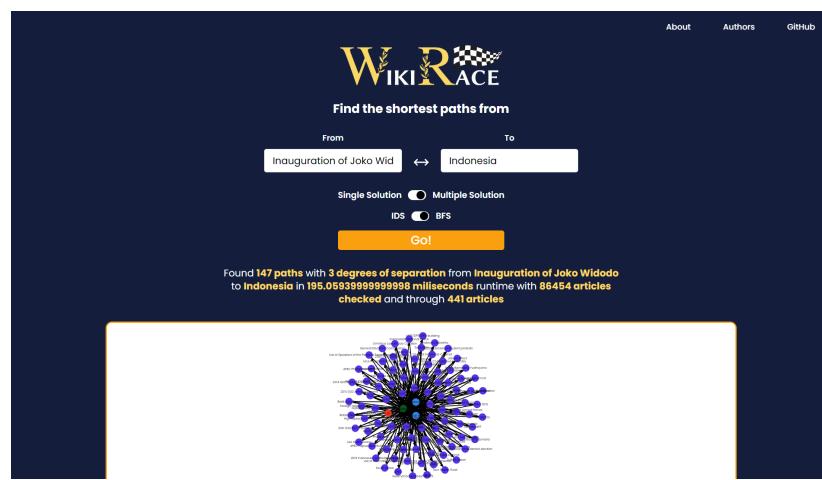


3. Inauguration of Joko Widodo → Indonesia

Tanpa Cache:

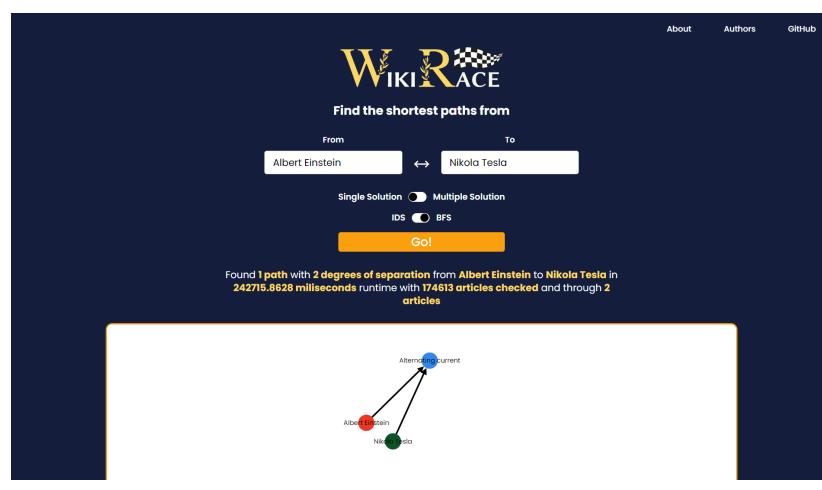


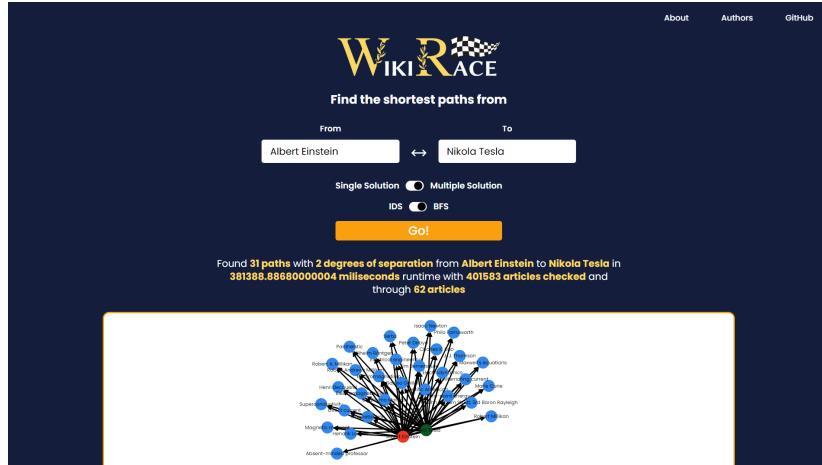
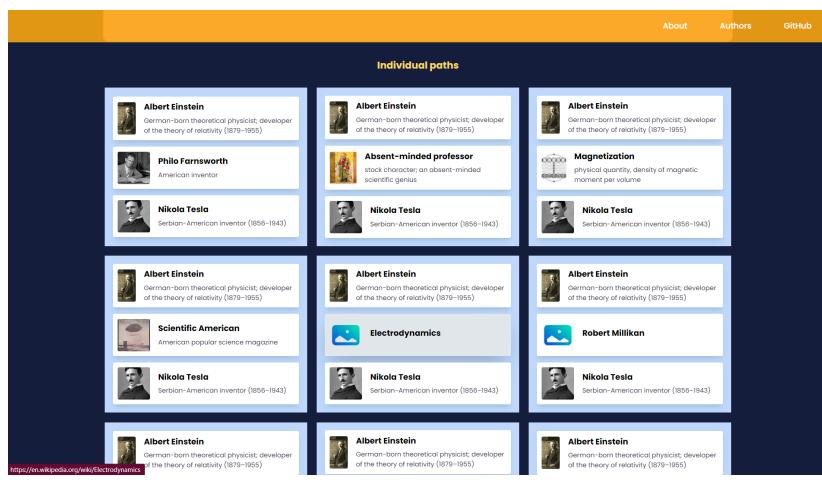
Dengan Cache:



Single Solution VS Multiple Solution

Single:



	<p>Multiple:</p> 
Card Solution Component	

4.5 Analisis Desain Solusi Algoritma

4.5.1. Pencarian dengan Algoritma *Iterative Deepening Search (IDS)* VS *Breadth First Search (BFS)*

Berikut analisis terkait pencarian dengan algoritma IDS dari percobaan subbab 4.4.

- 1) Berdasarkan percobaan Jokowi → Putin, algoritma BFS memiliki waktu eksekusi yang lebih sangkil daripada IDS dan hanya perlu menelusuri 1 buah artikel, artinya langsung menemukan *goal node*, sedangkan IDS menelusuri sebanyak 811 artikel. Hal ini disebabkan oleh sifat IDS yang menelusuri ke kedalaman tertentu dahulu sebelum menelusuri simpul tetangga pada kedalaman yang sama.

- 2) Berdasarkan percobaan Chicken → Duck dan Inauguration of Joko Widodo → Indonesia, algoritma IDS memiliki waktu eksekusi yang lebih sangkil daripada BFS dan hanya perlu menelusuri 277 buah artikel daripada 108460 artikel dengan BFS, serta 86454 artikel dicek dalam 131ms (BFS) dan 255 artikel dicek dalam 73ms artinya menemukan *goal node* pada kedalaman 2 di suatu lintasan tanpa ke tetangganya.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1. Kesimpulan

Konsep *Iterative Deepening Search* (IDS) dan *Breadth First Search* (BFS) dapat diaplikasikan pada WikiRace, yakni dengan cara menganggap bahwa tiap simpul graf adalah *path* dari tiap *link* Wikipedia yang dilakukan *scrapping* dan tersambung melalui sisi antar simpul atau *path* yang berhubungan atau cocok karena mempunyai simpul perantara yang sama dari simpul *start* dan *end*.

Algoritma IDS menggunakan struktur data *stack* dan algoritma BFS menggunakan struktur data *queue*. Secara hipotesis dan eksperimen dari implementasi dari konsep IDS dan BFS, seharusnya algoritma BFS memiliki waktu eksekusi yang lebih cepat daripada IDS karena memiliki alasan sebagai berikut.

- 1) Algoritma BFS menjelajahi semua simpul pada kedalaman yang sama terlebih dahulu sebelum bergerak ke kedalaman berikutnya sehingga memungkinkan BFS untuk menemukan solusi tanpa perlu kembali dan mengulang penelusuran pada kedalaman yang sama (hal yang terjadi pada algoritma IDS).
- 2) Algoritma BFS membutuhkan lebih banyak memori karena menyimpan semua simpul pada satu level dalam antrian, penggunaan memori ini bisa lebih efisien dari segi akses *cache* dibandingkan dengan IDS yang sering mengubah konteks kedalamannya (misalnya, melalui *backtracking*) sehingga mengakibatkan cache misses yang lebih sering dan mengurangi kecepatan eksekusi. Walau demikian, dalam konteks tugas ini di mana yang dicari adalah banyak solusi optimal, maka efisiensi memori IDS tidak menjadi keuntungan.
- 3) Algoritma BFS akan menemukan jalur terpendek (dalam hal jumlah tepi) ke target pada saat pertama kali mencapai kedalaman di mana solusi berada.

5.2. Saran

Penulis menyarankan agar spesifikasi WikiRace diperjelas lebih lanjut, seperti larangan penggunaan *Wikipedia API*, namun penggunaan *database* yang diperbolehkan, serta metode *double-ended* yang dilarang dilakukan secara *forward-backward* (berdasarkan referensi *medium* yang diberikan), melainkan harus

dilakukan secara *forward-forward*. Selain itu, konsep *caching* yang diperbolehkan, padahal teknik ini dapat mengurangi karakteristik khas atau konsep murni dari algoritma IDS dan BFS.

Sebab pada dasarnya, karakteristik murni IDS adalah kemampuannya untuk mendalam dan melakukan *backtracking* secara efisien sedangkan *Caching* mengurangi *backtracking* ini, yang bisa dianggap mengurangi esensi dari IDS. Selain itu, karakteristik BFS adalah eksplorasi sistematis dari semua simpul pada kedalaman tertentu sebelum bergerak ke kedalaman berikutnya. *Caching* mempercepat proses ini tetapi juga mengurangi jumlah operasi yang dibutuhkan untuk eksplorasi penuh, potensial mengurangi keperluan untuk eksplorasi menyeluruh di setiap level.

5.3. Refleksi

Dari pengonsepan algoritma, banyak sekali alternatif solusi untuk membuat program WikiRace dengan IDS dan BFS. Hal ini terkadang membuat kebingungan dan terdapat beberapa anomali, misalnya penggunaan *multithreading* yang dapat saja membuat kedua algoritma memiliki sifat yang hampir sama sehingga menyebabkan kebingungan. Oleh karena itu, penulis perlu eksplorasi lebih dalam tentang *libraries* yang dapat mendukung pembuatan WikiRace serta konsep IDS dan BFS yang telah dipelajari.

DAFTAR PUSTAKA

1. Baweja, Parul. “Racing through Wikipedia with Python” (online).
https://medium.com/@parulbaweja8/how-i-built-wikiracer-f493993fbd_d, diakses pada 17 April 2024).
2. Munir, Rinaldi. “Breadth/Depth First Search (BFS/DFS) Bagian 1” (online).
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/BFS-DFS-2021-Bag1-2024.pdf>, diakses pada 17 April 2024).
3. Munir, Rinaldi. “Breadth/Depth First Search (BFS/DFS) Bagian 2” (online).
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf>, diakses pada 17 April 2024).
4. Peppoloni, Lorenzo. “How to improve your GO code with empty structs” (online).
<https://medium.com/@l.peppoloni/how-to-improve-your-go-code-with-empty-structs-3bd0c66bc531>, diakses pada 17 April 2024).
5. Ward, Peder. “Beating Wikirace by Finding the Optimal Path with Neo4j and Dijkstra’s algorithm” (online).
<https://towardsdatascience.com/beating-wikirace-by-finding-the-optimal-path-with-neo4j-and-dijkstras-algorithm-1e11193c55bb>, diakses pada 17 April 2024).
6. Wenger, Jacob. “Six Degrees of Wikipedia” (online).
<https://github.com/jwngr/sdow>, diakses pada 17 April 2024).
7. Wu, Jerome. “Set in Go, map[]bool and map[]struct{} performance comparison” (online).
<https://itnext.io/set-in-go-map-bool-and-map-struct-performance-comparison-5315b4b107b>, diakses pada 17 April 2024).
8. “Next.js Documentation” (online).
<https://blog.miguelgrinberg.com/post/how-to-create-a-react--flask-project>, diakses pada 17 April 2024).
9. “Tailwindcss Documentation” (online).
<https://tailwindcss.com/docs/installation>, diakses pada 17 April 2024).

10. “Shadcn UI Documentation” (online).
[\(https://ui.shadcn.com/docs](https://ui.shadcn.com/docs), diakses pada 17 April 2024).

LAMPIRAN

Repository

Link Repository dari Tugas Besar 02 IF2211 Strategi Algoritma kelompok 41 “GoGoPowerRangers” adalah sebagai berikut.

https://github.com/fairuzald/Tubes2_GoGoPowerRangers.git

Youtube

Link video Youtube dari Tugas Besar 02 IF2211 Strategi Algoritma kelompok 41 “GoGoPowerRangers” adalah sebagai berikut.

https://youtu.be/uPR6T9T14_I?si=0v_EHA9FKE50B3f0