

LAPORAN TUGAS BESAR II
IF2211 STRATEGI ALGORITMA
“Pemanfaatan Algoritma IDS dan BFS
dalam Permainan WikiRace”



Dosen:

Ir. Rila Mandala, M. Eng, Ph. D.
Monterico Adrian, S. T., M. T.

Kelompok X:

13522139 Attara Majesta Ayub
13522145 Farrel Natha Saskoro
13522155 Axel Santadi Warih

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II TAHUN 2023/2024

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa kami ucapkan atas keberhasilan dalam menyelesaikan Tugas Besar I IF2211 Strategi Algoritma yang berjudul “Pemanfaatan Algoritma Greedy dalam Pembuatan Bot Permainan Diamonds” ini.

Laporan ini merupakan dokumentasi dan penjelasan atas program dan algoritma yang telah kami implementasikan. Kami berusaha untuk merancang dan mengimplementasikan strategi *greedy* ini sedemikian rupa sehingga bot dapat mengambil keputusan yang optimal dalam permainan ini.

Pengembangan bot dan penyelesaian tugas besar ini tidak lepas dari bimbingan dan arahan yang diberikan oleh dosen pengampu mata kuliah Strategi Algoritma. Kami juga ingin mengucapkan terima kasih kepada rekan-rekan kami yang telah memberikan masukan, saran, dan inspirasi selama proses pengembangan bot ini. Selain itu, kami juga mengapresiasi dukungan yang diberikan oleh asisten dan semua pihak yang telah membantu, baik secara langsung maupun tidak langsung, dalam penyelesaian tugas besar ini. Kami berharap bahwa *output* dari tugas besar ini dapat memberikan kontribusi bagi pengembangan ilmu pengetahuan, khususnya dalam bidang algoritma dan pemrograman.

Kami menyadari bahwa tugas besar ini masih jauh dari sempurna. Oleh karena itu, kami sangat terbuka untuk menerima kritik dan saran yang konstruktif demi perbaikan di masa yang akan datang. Akhir kata, semoga tugas besar ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Sumedang, 30 Februari 2024,

Kelompok 44.

DAFTAR ISI

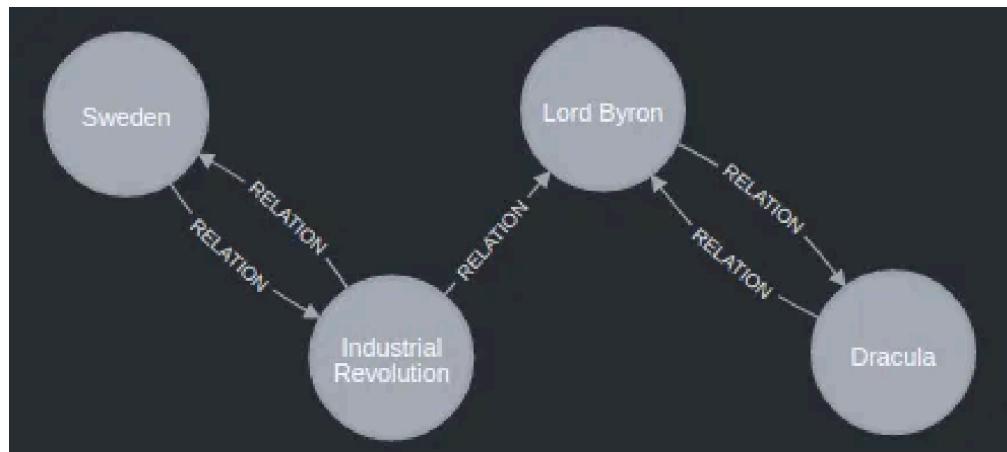
KATA PENGANTAR.....	1
DAFTAR ISI.....	2
BAB I	
DESKRIPSI TUGAS.....	4
BAB II	
LANDASAN TEORI.....	7
2.1 Dasar Teori Algoritma Breadth-First-Search (BFS).....	7
2.2 Dasar Teori Algoritma Iterative-Deepening Search (IDS).....	11
2.3 Penjelasan Singkat Aplikasi Web.....	12
BAB III	
ANALISIS PEMECAHAN MASALAH.....	14
3.1 Langkah-langkah Pemecahan Masalah.....	14
3.2 Proses Pemetaan Masalah Menjadi Elemen-elemen Algoritma IDS dan BFS..	14
3.2.1 IDS :.....	14
3.2.2 BFS :.....	15
3.3 Fitur Fungsional dan Arsitektur Aplikasi Web.....	15
3.4 Contoh Ilustrasi Kasus.....	17
BAB IV	
IMPLEMENTASI DAN PENGUJIAN.....	19
4.1 Spesifikasi Teknis Program.....	19
4.1.1 Struktur Data.....	19
4.1.2 Fungsi dan Prosedur yang Dibangun.....	19
4.2 Penjelasan Tata Cara Penggunaan Program.....	20
4.3 Hasil Pengujian.....	21
4.4 Analisis Hasil Pengujian.....	25
BAB V	
PENUTUP.....	26
5.1 Kesimpulan.....	26
5.2 Saran.....	27
5.3 Refleksi terhadap Tugas Besar.....	27
DAFTAR PUSTAKA.....	29
LAMPIRAN.....	30
6.1 GitHub Repository (Latest Release).....	30

6.2 Video.....	30
----------------	----

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. Ilustrasi Graf WikiRace

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

Cara Kerja Permainan WikiRace:

- 1) Pertama, setiap pemain (bot) akan ditempatkan pada board secara random. Masing-masing bot akan mempunyai home base, serta memiliki score dan inventory awal bernilai nol.
- 2) Setiap bot diberikan waktu untuk bergerak, waktu yang diberikan semua sama untuk setiap pemain.
- 3) Objektif utama bot adalah mengambil diamond-diamond yang ada di peta sebanyak-banyaknya. Seperti yang sudah disebutkan di atas, diamond

yang berwarna merah memiliki 2 poin dan diamond yang berwarna biru memiliki 1 poin.

- 4) Setiap bot juga memiliki sebuah inventory, dimana inventory berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini sewaktu-waktu bisa penuh, maka dari itu bot harus segera kembali ke home base.
- 5) Apabila bot menuju ke posisi home base, score bot akan bertambah senilai diamond yang tersimpan pada inventory dan inventory bot akan menjadi kosong kembali.
- 6) Usahakan agar bot anda tidak bertemu dengan bot lawan. Jika bot A menimpa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle).
- 7) Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut.
- 8) Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

Mekanisme Teknis Permainan WikiRace:

Permainan ini merupakan permainan berbasis *web*, sehingga setiap aksi yang dilakukan – mulai dari mendaftarkan bot hingga menjalankan aksi bot – akan memerlukan HTTP *request* terhadap API *endpoint* tertentu yang disediakan oleh *backend*. Berikut adalah urutan *requests* yang terjadi dari awal mula permainan.

- 1) Program bot akan mengecek apakah bot sudah terdaftar atau belum, dengan mengirimkan POST *request* terhadap *endpoint* **/api/bots/recover** dengan body berisi email dan *password* bot. Jika

bot sudah terdaftar, maka *backend* akan memberikan *response code* 200 dengan *body* berisi id dari bot tersebut. Jika tidak, *backend* akan memberikan *response code* 404.

- 2) Jika bot belum terdaftar, maka program bot akan mengirimkan POST request terhadap endpoint `/api/bots` dengan *body* berisi *email*, *name*, *password*, dan *team*. Jika berhasil, maka *backend* akan memberikan *response code* 200 dengan *body* berisi id dari bot tersebut.
- 3) Ketika id bot sudah diketahui, bot dapat bergabung ke board dengan mengirimkan POST request terhadap endpoint `/api/bots/{id}/join` dengan *body* berisi *board* id yang diinginkan (*preferredBoardId*). Apabila bot berhasil bergabung, maka *backend* akan memberikan *response code* 200 dengan *body* berisi informasi dari *board*.
- 4) Program bot akan mengkalkulasikan *move* selanjutnya secara berkala berdasarkan kondisi *board* yang diketahui, dan mengirimkan POST request terhadap endpoint `/api/bots/{id}/move` dengan *body* berisi *direction* yang akan ditempuh selanjutnya (“NORTH”, “SOUTH”, “EAST”, atau “WEST”). Apabila berhasil, maka *backend* akan memberikan *response code* 200 dengan *body* berisi kondisi *board* setelah *move* tersebut. Langkah ini dilakukan terus-menerus hingga waktu bot habis. Jika waktu bot habis, bot secara otomatis akan dikeluarkan dari *board*.
- 5) Program *frontend* secara periodik juga akan mengirimkan GET request terhadap endpoint `/api/boards/{id}` untuk mendapatkan kondisi *board* terbaru, sehingga tampilan *board* pada *frontend* akan selalu ter-update.

BAB II

LANDASAN TEORI

2.1 Dasar Teori Algoritma Breadth-First-Search (BFS)

Materi:

Algoritma Breadth-First Search (BFS) atau algoritma pencarian melebar merupakan salah satu algoritma untuk melakukan traversal pada graf. Algoritma ini menelusuri dari sebuah simpul akar pada graf lalu mengunjungi semua tetangga dari simpul tersebut. Algoritma BFS memerlukan sebuah antrian (queue) untuk menyimpan simpul yang telah dikunjungi. Berikut ini adalah langkah-langkah traversal graf dengan memanfaatkan algoritma BFS.

1. Masukan simpul akar ke dalam antrian.
2. Keluarkan sebuah simpul dari antrian.
3. Kunjungi semua tetangga dari simpul ini yang belum pernah dikunjungi sebelumnya. Masukan juga simpul-simpul tetangga ini ke dalam antrian.
4. Jika antrian telah kosong maka penelusuran telah selesai. Jika antrian masih ada, ulangi langkah nomor 2.

Berikut ini adalah pseudocode penelusuran graf dengan BFS.

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian BFS.
Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpul yang dikunjungi dicetak ke layar
}

Deklarasi
w : integer
```

```

q : antrian

procedure BuatAntrian(input/output q : antrian)
{ membuat antrian kosong, kepala(q) diisi 0 }

procedure MasukAntrian(input/output q:antrian,
input v:integer)
{ memasukkan v ke dalam antrian q pada posisi
belakang }

procedure HapusAntrian(input/output q:antrian,
output v:integer)
{ menghapus v dari kepala antrian q }

function AntrianKosong(input q:antrian) boolean
{ true jika antrian q kosong, false jika
sebaliknya }

```

Algoritma

```

BuatAntrian(q) { buat antrian kosong }
write(v) { cetak simpul awal yang dikunjungi }
dikunjungi[v] <- true { simpul v telah dikunjungi,
tandai dengan true}
MasukAntrian(q,v) { masukkan simpul awal
kunjungan ke dalam antrian}
{ kunjungi semua simpul graf selama antrian belum
kosong }

while not AntrianKosong(q) do
HapusAntrian(q,v) { simpul v telah dikunjungi,
hapus dari antrian }
for tiap simpul w yang bertetangga dengan simpul v do
if not dikunjungi[w] then
write(w) {cetak simpul yang dikunjungi}
MasukAntrian(q,w)
Dikunjungi[w] <- true

```

```
endif  
endfor  
endwhile  
{ AntrianKosong (q) }
```

Dalam pengembangan solusi komputasional untuk berbagai permasalahan, algoritma *breadth-first-search* (BFS) merupakan salah satu pendekatan pemrograman yang pragmatis dan cukup efisien. Algoritma *greedy* merupakan sebuah pendekatan dalam pemecahan masalah yang dilakukan secara bertahap. Dalam setiap tahapannya, algoritma ini memilih opsi terbaik yang tersedia saat itu, berpegang pada prinsip “*take what you can get now!*” tanpa mempertimbangkan dampak jangka panjang dari keputusan tersebut. Algoritma ini beroperasi dengan harapan bahwa serangkaian pilihan optimum lokal, yaitu pilihan terbaik yang dapat dibuat di setiap langkah tanpa memikirkan masa depan, akan secara keseluruhan mengarah pada solusi optimum global, atau dengan kata lain, solusi terbaik untuk seluruh persoalan. Pendekatan ini berlandaskan pada premis bahwa pemilihan solusi yang optimal pada setiap tahapan atau sub-problema secara individual akan mengkumulasikan hasil akhir yang merupakan solusi terbaik untuk masalah secara keseluruhan. Karakteristik ini membuat algoritma *greedy* menjadi pilihan yang atraktif untuk menyelesaikan beragam masalah optimisasi, di mana efisiensi dan kecepatan dalam mencapai solusi menjadi prioritas utama.

Algoritma *greedy* memiliki dua karakteristik utama yang mendefinisikan pendekatannya dalam pemecahan masalah, yaitu sifat *myopic* dan *irrevocable*. Sifat *myopic*, atau pandangan jangka pendek, merupakan aspek penting dari algoritma ini, di mana keputusan diambil berdasarkan informasi dan situasi yang tersedia pada saat itu tanpa mempertimbangkan

dampak jangka panjang dari keputusan tersebut. Di sisi lain, karakteristik *irrevocable* menekankan bahwa setiap keputusan yang telah diambil oleh algoritma tidak dapat dibatalkan atau diubah. Sekali pilihan dibuat, proses tidak dapat kembali untuk merevisi atau memilih alternatif lain. Kedua sifat ini, meskipun dapat membatasi fleksibilitas algoritma, juga merupakan alasan efisiensi dan kesederhanaan dalam pemecahan berbagai masalah optimisasi dengan algoritma *greedy*.

Kelebihan utama dari algoritma *greedy* terletak pada kesederhanaannya dalam implementasi dan efisiensi waktu komputasi yang luar biasa. Pendekatan ini mampu memberikan solusi cepat dan efektif untuk berbagai jenis masalah optimasi berkat kemampuannya dalam membuat serangkaian keputusan lokal yang optimal. Dengan kelebihan ini, algoritma *greedy* sering menjadi pilihan utama ketika dibutuhkan solusi yang dapat dihasilkan dalam waktu singkat, terutama untuk masalah yang sifatnya dapat didekati dengan mempertimbangkan pilihan yang terbaik pada setiap langkah tanpa harus memproses ulang seluruh data. Namun, pendekatan *greedy* juga memiliki keterbatasan yang signifikan, terutama ketika dihadapkan pada masalah yang memerlukan pertimbangan komprehensif atas konsekuensi dari setiap keputusan di masa depan. Sifat *myopic* dari algoritma *greedy* yang hanya fokus pada pemecahan masalah secara segera tanpa mempertimbangkan dampak jangka panjang, dapat mengakibatkan kegagalan dalam mencapai solusi yang benar-benar optimal.

Jika dibandingkan dengan algoritma pemecahan masalah lainnya, seperti algoritma *dynamic programming* dan *backtracking*, algoritma *greedy* menunjukkan keunggulan dalam kecepatan dan kesederhanaan. Namun, algoritma *dynamic programming* lebih unggul dalam menangani masalah yang memerlukan pertimbangan terhadap *sub-problem* dan mengoptimalkan solusi global, sementara *backtracking* memiliki fleksibilitas dalam menjelajahi

semua kemungkinan solusi hingga ditemukan yang optimal. Pilihan antara algoritma ini seringkali didasarkan pada sifat spesifik dari masalah yang dihadapi dan *trade-off* antara kecepatan implementasi dan keakuratan solusi yang diperoleh.

2.2 Dasar Teori Algoritma Iterative-Deepening Search (IDS)

Iterative-Deepening Search (IDS) merupakan variasi dari Depth-First Search (DFS) yang bertujuan untuk menemukan solusi dalam graf dengan menggunakan strategi pencarian secara berulang dengan kedalaman pencarian yang bertambah. Meskipun konsep dasar dari IDS mirip dengan DFS, IDS membatasi kedalaman pencarian pada setiap iterasi untuk mengatasi beberapa kelemahan DFS, seperti ketidakmampuan untuk menangani graf dengan kedalaman yang tak terbatas atau terlalu dalam.

Langkah-langkah utama dalam IDS adalah sebagai berikut:

1. Pencarian Bertahap: IDS melakukan pencarian dalam beberapa tahap atau iterasi. Pada setiap iterasi, pencarian dilakukan dengan membatasi kedalaman pencarian hingga batas tertentu.
2. Pencarian Depth-Limited: Pada setiap tahap iterasi, IDS menggunakan Depth-First Search (DFS) dengan pembatasan kedalaman pencarian. Ini berarti IDS akan menjelajahi simpul-simpul pada kedalaman tertentu sebelum bergerak ke kedalaman yang lebih dalam.
3. Penyelarasan Iterasi: IDS menyelaraskan hasil pencarian dari setiap iterasi untuk menentukan solusi terbaik. Jika solusi ditemukan pada iterasi sebelumnya, IDS akan mengembalikan solusi tersebut tanpa perlu melanjutkan pencarian ke iterasi berikutnya.

Keuntungan dari IDS adalah kemampuannya untuk menggabungkan keuntungan DFS dalam penggunaan memori yang efisien dengan kemampuan BFS dalam

menemukan solusi optimal. IDS cocok digunakan dalam kasus-kasus di mana kedalaman pencarian tidak diketahui sebelumnya atau ketika memori terbatas.

2.3 Penjelasan Singkat Aplikasi Web

Aplikasi web yang dibangun dalam proyek ini menggunakan algoritma BFS (Breadth-First Search) dan IDS (Iterative-Deepening Search) untuk mencari jalur terpendek antara dua halaman Wikipedia. Aplikasi ini memungkinkan pengguna untuk mencari jalur terpendek antara dua topik yang dipilih di Wikipedia dengan mengunjungi endpoint API yang disediakan.

Fitur Aplikasi:

1. Pencarian Jalur Terpendek: Pengguna dapat memasukkan dua topik yang berbeda dari Wikipedia sebagai titik awal dan titik akhir, dan aplikasi akan mencari jalur terpendek antara keduanya.
2. Visualisasi Jalur: Aplikasi memberikan respons dalam bentuk jalur yang disajikan dalam format JSON, yang mencakup daftar halaman Wikipedia yang membentuk jalur terpendek, serta waktu yang diperlukan untuk menemukan jalur tersebut.

Arsitektur Aplikasi:

- Aplikasi menggunakan bahasa pemrograman Go (Golang) untuk backend, dengan bantuan beberapa library, termasuk `net/http` untuk penanganan HTTP, `github.com/gocolly/colly` untuk ekstraksi data dari halaman web, dan `container/list` untuk implementasi struktur data antrian.
- Algoritma BFS dan IDS diimplementasikan untuk mencari jalur terpendek antara dua halaman Wikipedia.

- Aplikasi menyediakan endpoint API yang dapat diakses oleh klien untuk melakukan pencarian jalur terpendek.

Tujuan Aplikasi:

- Tujuan dari aplikasi ini adalah untuk memberikan solusi yang efisien dan mudah digunakan bagi pengguna yang ingin menemukan jalur terpendek antara dua topik di Wikipedia.
- Dengan algoritma pencarian yang efisien seperti BFS dan IDS, aplikasi ini dapat memberikan respons yang cepat terhadap permintaan pencarian yang kompleks.

Keunggulan Aplikasi:

- Aplikasi menggunakan pendekatan yang kuat dari sisi algoritma dengan menerapkan BFS dan IDS, yang dikenal efisien dalam menemukan jalur terpendek dalam graf.
- Arsitektur aplikasi yang sederhana dan bersih membuatnya mudah untuk dipahami dan dikembangkan lebih lanjut.
- Respons yang cepat dan akurat dari aplikasi memastikan pengalaman pengguna yang baik.

Dengan demikian, aplikasi web ini menyediakan alat yang efisien dan mudah digunakan bagi pengguna untuk menemukan jalur terpendek antara topik-topik yang berbeda di Wikipedia.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-langkah Pemecahan Masalah

1. Pemahaman Masalah: Langkah pertama adalah memahami masalah yang dihadapi, yaitu menemukan jalur terpendek antara dua halaman Wikipedia. Kami menganalisis masalah ini dengan mempelajari struktur dan pola tautan antar halaman Wikipedia.
2. Penyusunan Algoritma: Kami menyusun algoritma untuk menyelesaikan masalah ini. Kami mempertimbangkan penggunaan algoritma Iterative Deepening Search (IDS) dan Breadth-First Search (BFS) karena keduanya cocok untuk mencari jalur terpendek dalam graf yang tidak memiliki bobot.
3. Implementasi IDS: Kami mengimplementasikan algoritma Iterative Deepening Search (IDS) untuk menemukan jalur terpendek. IDS merupakan variasi dari Depth-First Search (DFS) yang membatasi kedalaman pencarian pada setiap iterasi.
4. Implementasi BFS: Selain IDS, kami juga mengimplementasikan algoritma Breadth-First Search (BFS). BFS digunakan untuk mencari jalur terpendek dengan mengeksplorasi semua simpul pada kedalaman yang sama sebelum melanjutkan ke kedalaman berikutnya.

3.2 Proses Pemetaan Masalah Menjadi Elemen-elemen Algoritma IDS dan BFS

3.2.1 IDS :

Langkah 1: Pencarian dimulai dari kedalaman 0.

Langkah 2: Jika jalur terpendek tidak ditemukan pada kedalaman tersebut, penelusuran dilakukan pada kedalaman 1.

Langkah 3: Proses berlanjut hingga jalur terpendek ditemukan atau batas kedalaman maksimum tercapai.

3.2.2 BFS :

Langkah 1: Pencarian dimulai dari simpul awal.

Langkah 2: Menjelajahi semua simpul yang terhubung dengan simpul saat ini pada kedalaman saat ini.

Langkah 3: Simpul-simpul baru yang ditemukan dimasukkan ke dalam antrian untuk dieksplorasi selanjutnya.

3.3 Fitur Fungsional dan Arsitektur Aplikasi Web

Aplikasi web yang kami bangun memiliki beberapa fitur fungsional utama, antara lain:

3.3.1 Navigation Bar



Memiliki fungsi untuk memudahkan user berpindah section dalam web.

3.3.2 Interactive About Us

The screenshot shows an "About Us" section with three cards. Each card contains a name, a small profile picture, and a student ID. The names are Attara Majesta Ayub, Farrel Natha Saskoro, and Axel Santadi Warih. The student IDs are 13522139, 13522145, and 13522155 respectively. The cards have rounded corners and are set against a light background.

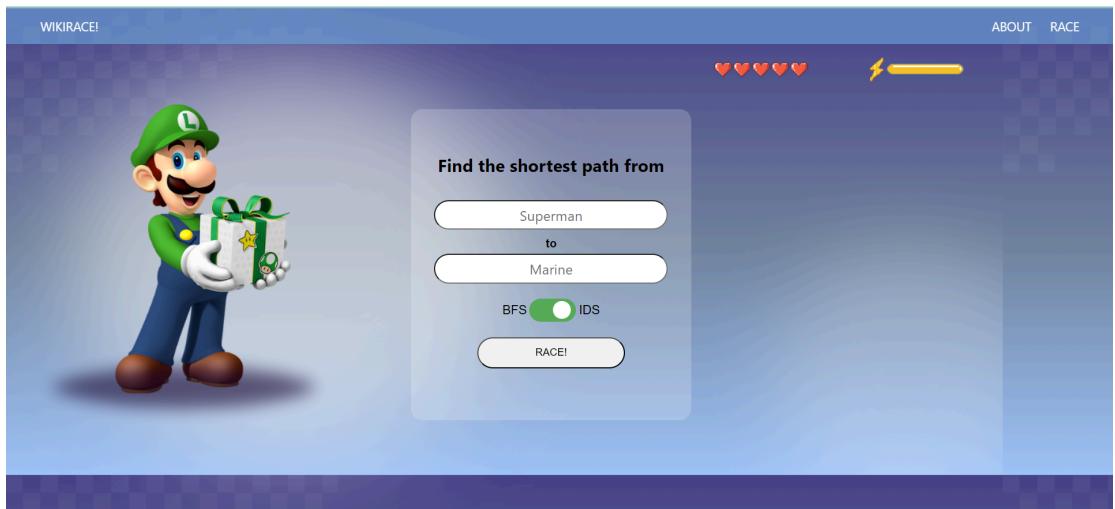
Name	ID
Attara Majesta Ayub	13522139
Farrel Natha Saskoro	13522145
Axel Santadi Warih	13522155

About Us

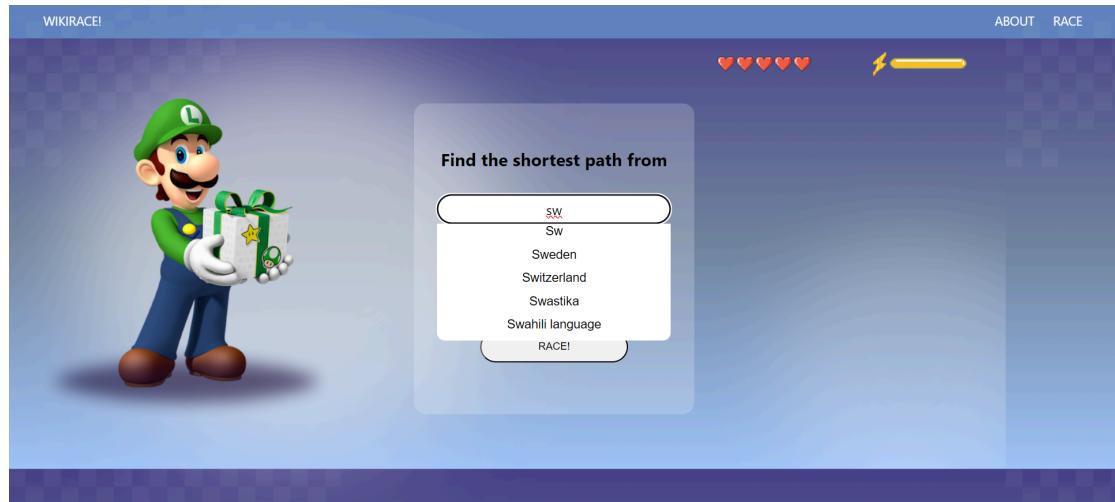


Dimana saat cursor hover maka akan terlihat deskripsi atau quotes dari developer.

3.3.3 Race



Fitur race untuk mencari shortest path dari wiki page ke wiki page. Fitur ini menerima input startpage, targetpage, dan button untuk memilih algoritma yang dipakai.

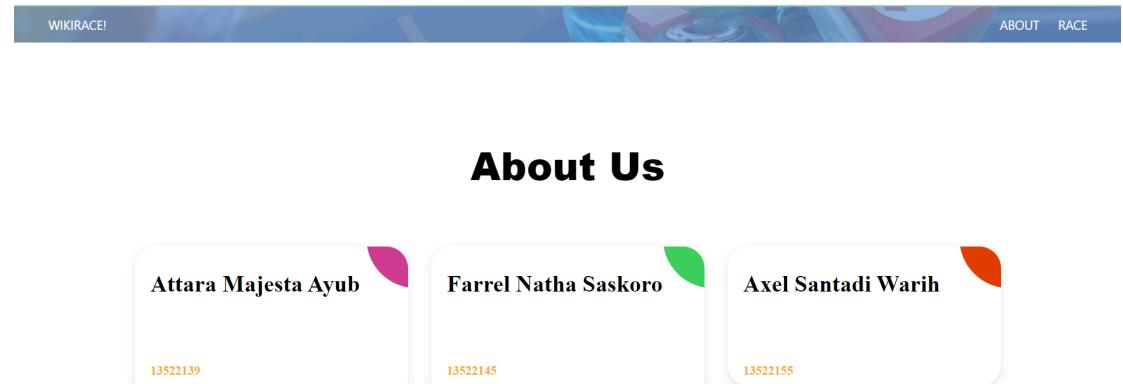


Dalam fitur race juga memastikan input dari user adalah sebuah wikipedia title yang valid.

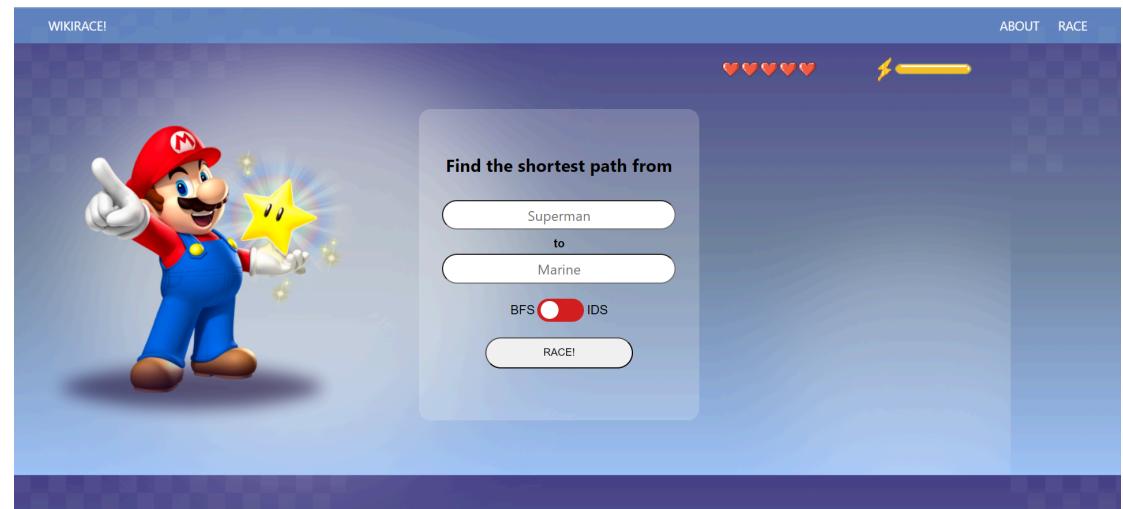
3.4 Contoh Ilustrasi Kasus



Gambar 3.4.1 Main Page



Gambar 3.4.2 Page About US



BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

4.1.1 Struktur Data

1. Link: Struktur data Link digunakan untuk merepresentasikan tautan antara halaman Wikipedia. Setiap Link memiliki atribut URL yang menyimpan URL halaman Wikipedia.
2. Request: Struktur data Request digunakan untuk menyimpan parameter permintaan dari pengguna, yaitu Start dan Target, yang merupakan halaman Wikipedia awal dan tujuan.

4.1.2 Fungsi dan Prosedur yang Dibangun

1. getLinks: Fungsi ini bertugas untuk mengambil tautan dari halaman Wikipedia menggunakan teknik web scraping dengan bantuan library Colly. Fungsi ini mengembalikan slice dari struktur data Link.
2. findShortestPath: Fungsi ini menggunakan algoritma Breadth-First Search (BFS) untuk menemukan jalur terpendek antara dua halaman Wikipedia. Fungsi ini mengembalikan jalur terpendek dalam bentuk slice dari struktur data Link beserta waktu yang diperlukan untuk menemukan jalur tersebut.
3. BFSHandler: Handler HTTP yang digunakan untuk menerima permintaan pencarian jalur terpendek dengan menggunakan BFS. Handler ini mengekstrak parameter start dan target dari permintaan HTTP dan memanggil fungsi findShortestPath.
4. IDS: Fungsi ini mengimplementasikan algoritma Iterative-Deepening Search (IDS) untuk mencari jalur terpendek antara dua halaman Wikipedia. Fungsi ini juga mengembalikan jalur terpendek beserta waktu yang diperlukan untuk menemukan jalur tersebut.

5. DLS: Fungsi rekursif yang digunakan oleh IDS untuk melakukan pencarian dengan kedalaman terbatas (Depth-Limited Search).
6. IDSHandler: Handler HTTP yang digunakan untuk menerima permintaan pencarian jalur terpendek dengan menggunakan IDS. Handler ini juga mengekstrak parameter start dan target dari permintaan HTTP dan memanggil fungsi IDS.

4.2 Penjelasan Tata Cara Penggunaan Program

4.1.1 Backend

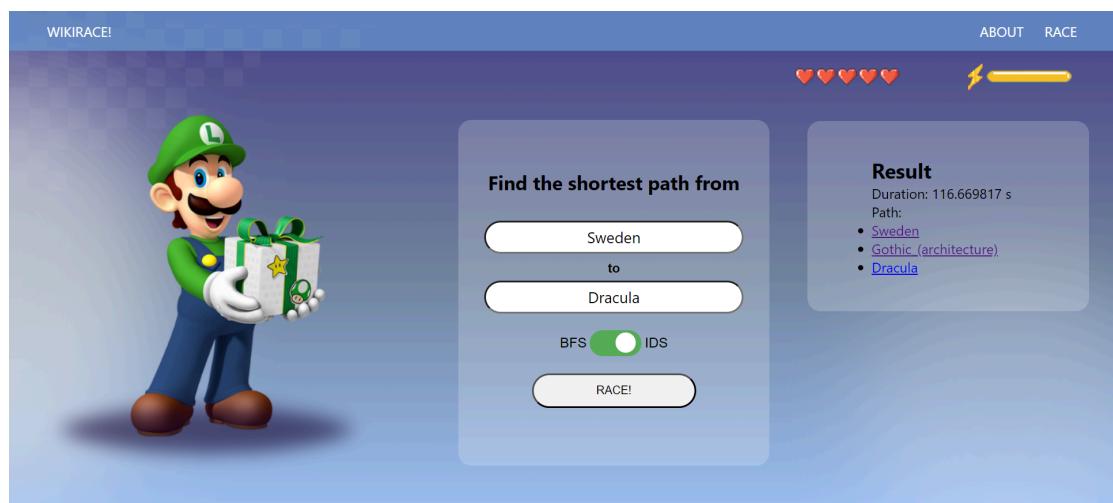
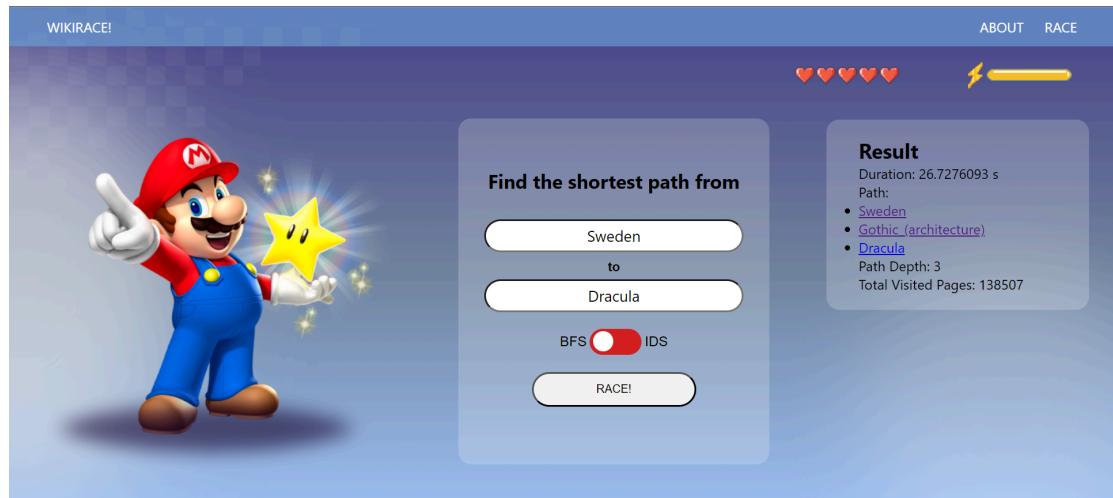
- Pastikan telah menginstall Go Lang pada perangkat kalian
- Pindah *directory* ke backend dengan menginput ke terminal “cd src/backend”
- Lalu input ke terminal “go run main.go” untuk menjalankan backend

4.1.2 Frontend

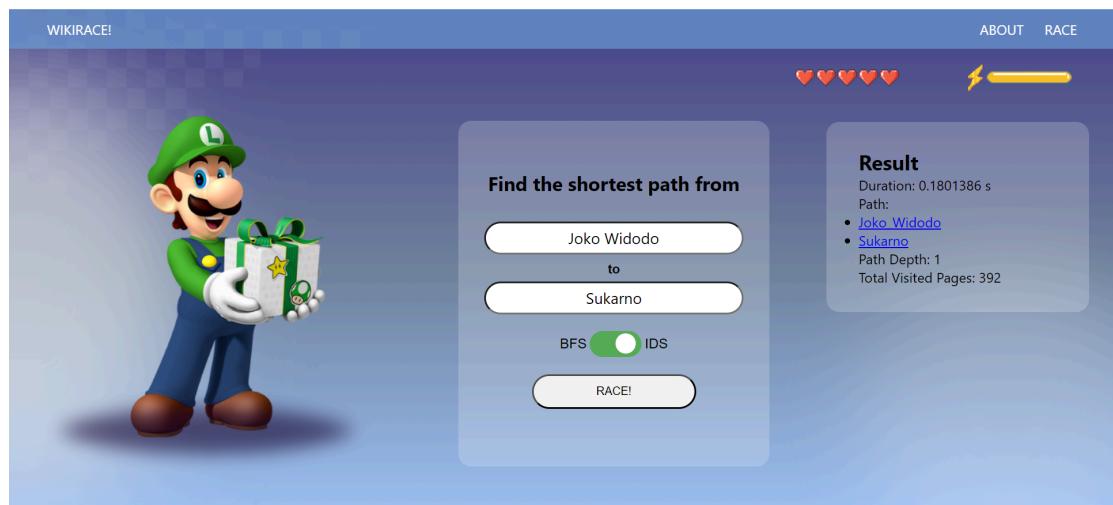
- Pastikan telah menginstall npm atau sejenisnya
- Pindah *directory* ke frontend dengan menginput ke terminal “cd src/frontend”
- Lalu input ke terminal “npm run dev” untuk menjalankan frontend
- Klik local host yang mucul pada terminal dengan menekan “ctrl” dan klik link local host dengan cursor.
- Klik Race di NavBar.
- Input judul wikipage awal dan akhir, sudah terdapat suggestion yang dapat di-klik.
- Pilih algoritma yang akan dipakai.
- Klik Race.
- Hasil akan ditampilkan di sebelah kanan.

4.3 Hasil Pengujian

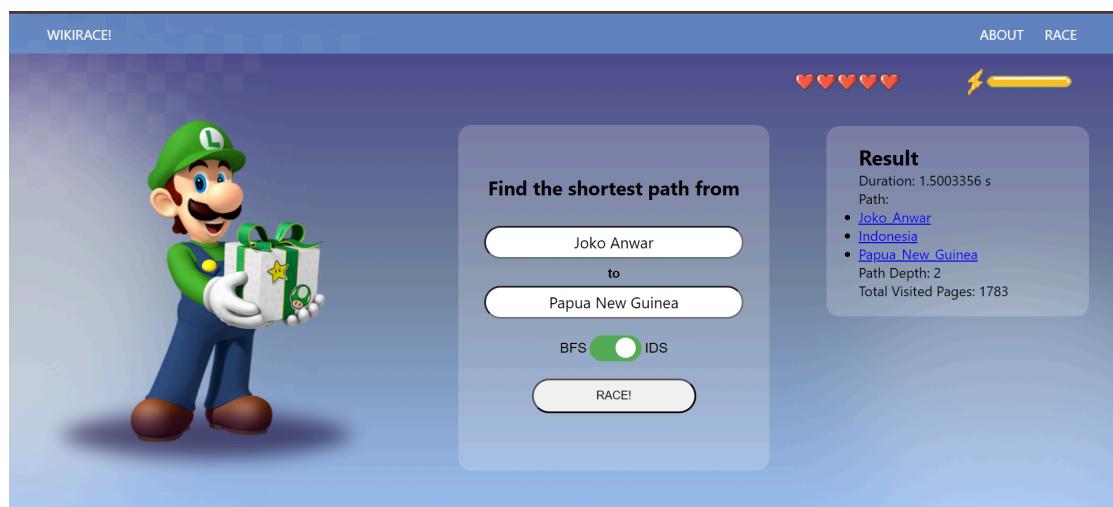
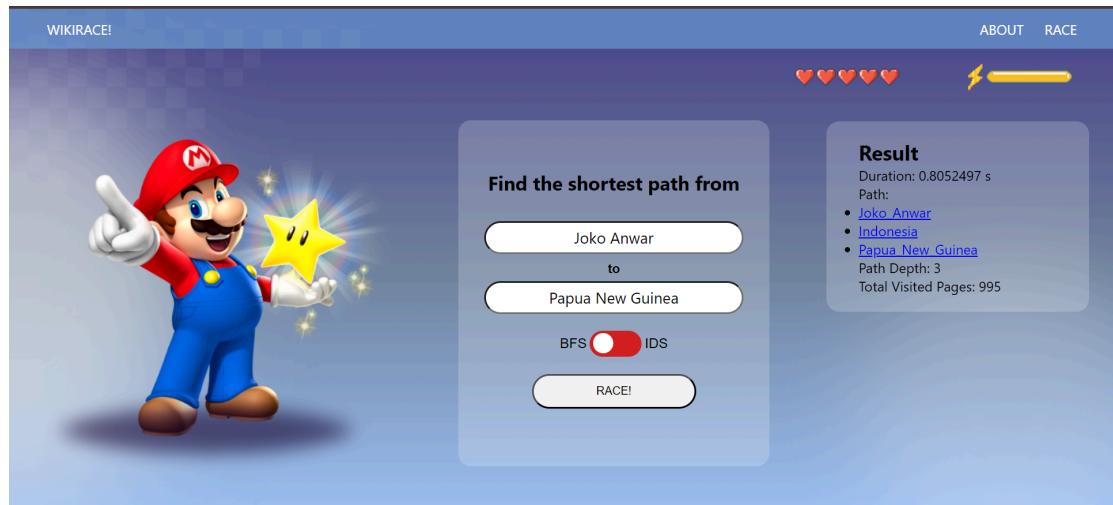
4.3.1. Testcase 1: “Sweden” ke “Dracula”



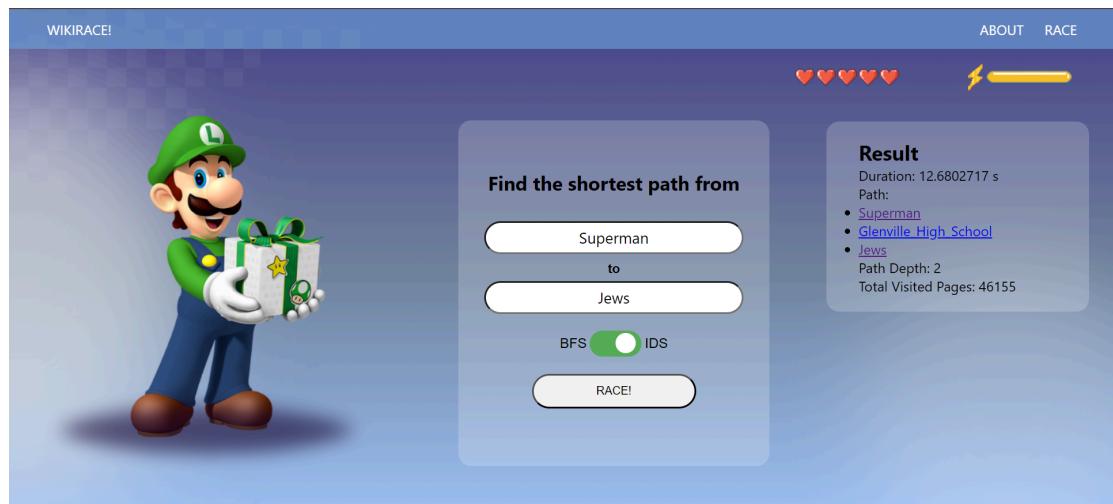
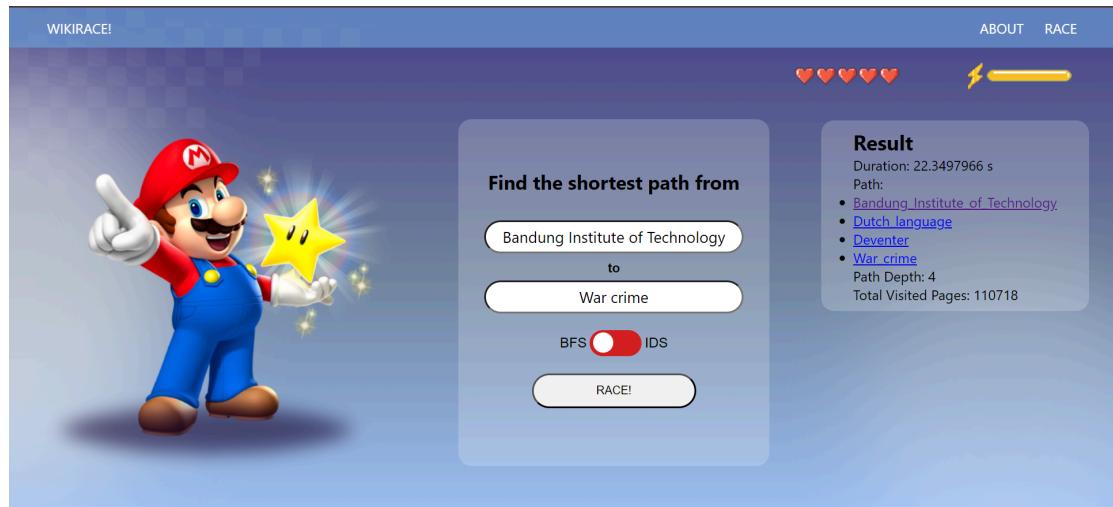
4.3.2. Testcase 2: “Joko Widodo” ke “Soekarno”



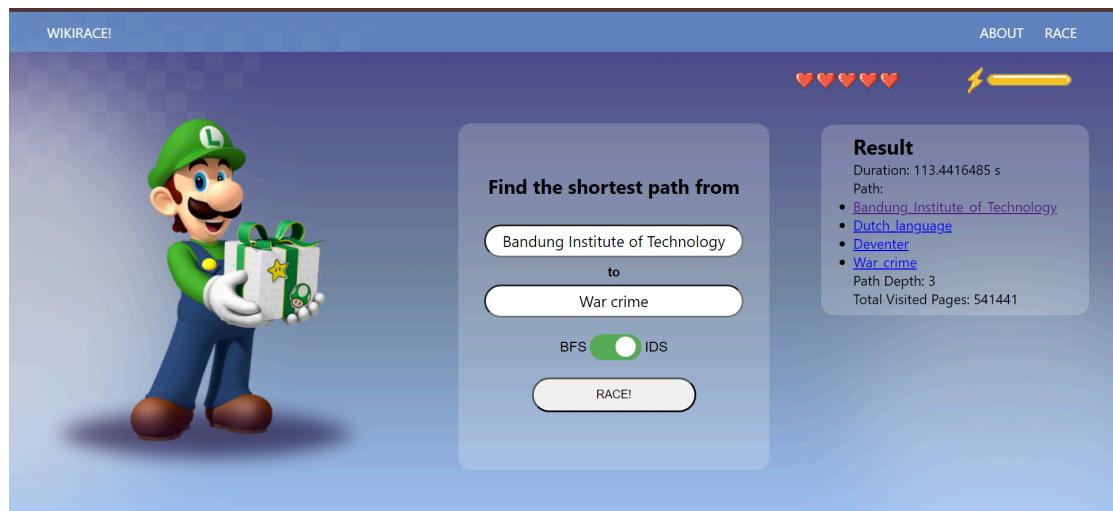
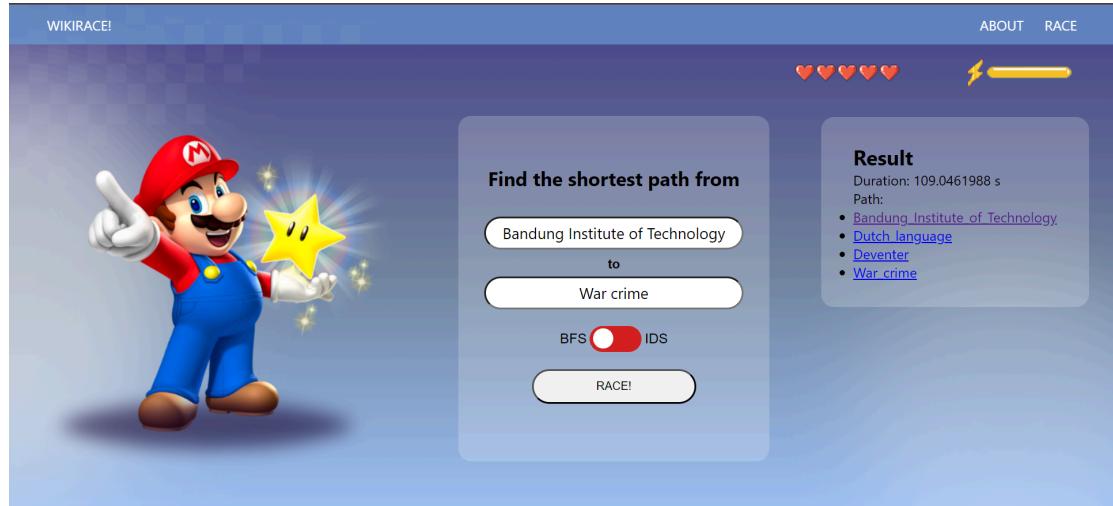
4.3.3. Testcase 3: “Joko Anwar” ke “Papua New Guinea”



4.3.4. Testcase 4: “Superman” ke “Jews”



4.3.5. Testcase 5: “Bandung Institute of Technology” ke “War crime”



4.4 Analisis Hasil Pengujian

Algoritma IDS dan BFS belum tentu menemukan jalur terdekat. Tetapi, pasti akan menemukan jalur pertama sesuai dengan urutan *scraping hyperlink*. Tidak ada konsistensi waktu yang lebih cepat antara kedua algoritma yang disajikan, namun secara teori seharusnya algoritma BFS lebih cepat karena dapat mengunjungi kedalaman yang lebih dalam dengan lebih cepat dibandingkan algoritma IDS.

BAB V

PENUTUP

5.1 Kesimpulan

Dalam tugas besar ini, kami berhasil mengembangkan sebuah solver untuk menemukan jalur terpendek antara dua halaman Wikipedia menggunakan algoritma Iterative Deepening Search (IDS) dan Breadth-First Search (BFS). Melalui langkah-langkah pemecahan masalah yang kami terapkan, kami berhasil menyusun algoritma dan mengimplementasikannya dalam bentuk aplikasi web.

Kami dapat menyimpulkan beberapa poin penting dari proyek ini:

Pemahaman Masalah: Kami memulai proyek ini dengan memahami masalah yang dihadapi, yaitu mencari jalur terpendek antara dua halaman Wikipedia. Melalui analisis masalah, kami dapat menentukan pendekatan yang tepat dalam memecahkan masalah tersebut.

Penggunaan Algoritma: Kami memilih untuk menggunakan algoritma Iterative Deepening Search (IDS) dan Breadth-First Search (BFS) karena keduanya cocok untuk mencari jalur terpendek dalam graf yang tidak memiliki bobot. Implementasi kedua algoritma ini memberikan fleksibilitas dalam menemukan solusi terbaik tergantung pada kasus penggunaan.

Arsitektur Aplikasi Web: Kami berhasil mengembangkan aplikasi web menggunakan bahasa pemrograman Go untuk bagian backend dan HTML/CSS/JavaScript untuk bagian frontend. Arsitektur aplikasi web kami didesain dengan baik sehingga pengguna dapat dengan mudah menggunakan solver yang telah kami buat.

Kemampuan Solver: Solver yang kami bangun dapat mengembalikan jalur terpendek antara dua halaman Wikipedia dengan cepat dan efisien.

Dengan menggunakan algoritma IDS dan BFS, solver kami dapat menangani berbagai kasus penggunaan dengan baik.

Dengan demikian, tugas besar ini tidak hanya memberikan pengalaman dalam pengembangan solver untuk masalah pencarian jalur terpendek, tetapi juga mengasah kemampuan dalam menerapkan konsep algoritma dan arsitektur perangkat lunak dalam pembangunan aplikasi web.

Kami berharap bahwa solver yang kami kembangkan dapat bermanfaat bagi pengguna dalam menemukan jalur terpendek antara halaman Wikipedia dan memberikan kontribusi positif dalam pengembangan aplikasi web yang lebih kompleks di masa depan.

5.2 Saran

Untuk pengembangan bot Permainan Diamonds lebih lanjut, kami menyarankan penelitian lebih dalam pada algoritma-algoritma lain yang dapat bekerja sinergis dengan algoritma greedy, seperti algoritma *dynamic programming* atau *backtracking*, untuk situasi di mana perhitungan jangka panjang menjadi penting. Kemudian, memperkaya basis data pengujian dengan skenario permainan yang lebih beragam dan kompleks akan membantu dalam meningkatkan *robustness* dan adaptabilitas bot. Integrasi teknologi kecerdasan buatan lanjutan, seperti *deep learning*, dapat dieksplorasi untuk meningkatkan kemampuan prediksi dan adaptasi bot terhadap strategi lawan yang dinamis dan tidak terduga.

5.3 Refleksi terhadap Tugas Besar

Tugas besar ini telah memberikan kami pelajaran tentang pentingnya aplikasi ilmu komputasi dalam menyelesaikan masalah yang nyata. Kami memahami bahwa tidak ada satu solusi yang sempurna. Oleh karena itu, eksplorasi dan inovasi terus-menerus diperlukan. Kerja sama tim, komunikasi

yang efektif, dan pengelolaan waktu yang baik telah menjadi kunci dalam menyelesaikan tugas besar ini. Kami juga belajar menghadapi kegagalan sebagai bagian dari proses pembelajaran dan menggunakan kegagalan tersebut sebagai langkah untuk maju lebih jauh.

DAFTAR PUSTAKA

GeeksForGeeks. (2024). *Greedy Algorithms*. Retrieved from GeeksForGeeks:

<https://www.geeksforgeeks.org/greedy-algorithms/>

Munir, R. (2024). *Algoritma Greedy*. Retrieved from Homepage Rinaldi

Munir:

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)

Sari, A. M. (2023). *Algoritma Greedy: Pengertian ,Jenis dan Contoh*

Program. Retrieved from Fakultas Ilmu Komputer dan Teknologi
Informasi UMSU:

<https://fiki.umsu.ac.id/algoritma-greedy-pengertian-jenis-dan-contoh-program/>

LAMPIRAN

6.1 GitHub Repository (Latest Release)

https://github.com/attaramajesta/Tubes2_AFaTar

6.2 Video

<https://youtu.be/PKvnOfV1NbQ>