

Tugas Besar 2 IF2211 Strategi Algoritma

Semester II tahun 2023/2024

Pemanfaatan Algoritma IDS dan BFS dalam Permainan WikiRace



Disusun Oleh:

Ariel Herfrison 13522002

Panji Sri Kuncara Wisma 13522028

Muhammad Dava Fathurrahman 13522114

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2024

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1	
DESKRIPSI TUGAS.....	3
BAB 2	
LANDASAN TEORI.....	4
A. Iterative Deepening Search (IDS).....	4
B. Breadth-First Search (BFS).....	4
BAB 3	
ANALISIS PEMECAHAN MASALAH.....	6
A. Langkah Pemecahan Masalah.....	6
B. Algoritma IDS.....	6
C. Algoritma BFS.....	8
D. Arsitektur dan Fitur Program.....	9
BAB 4	
IMPLEMENTASI DAN PENGUJIAN.....	11
A. Spesifikasi Teknis Program.....	11
A.1. Algoritma Titik Masuk Program.....	11
A.2. Algoritma IDS.....	11
A.3. Algoritma BFS.....	13
B. Penjelasan Tata Cara Penggunaan Program.....	14
C. Hasil Pengujian.....	15
D. Analisis Hasil Pengujian.....	26
BAB 5	
KESIMPULAN DAN SARAN.....	27
A. Kesimpulan.....	27
B. Saran.....	27
C. Refleksi.....	27
LAMPIRAN.....	28
DAFTAR PUSTAKA.....	29

BAB 1

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*jxmEbVn2FFWybZsIicJCWQ.png)

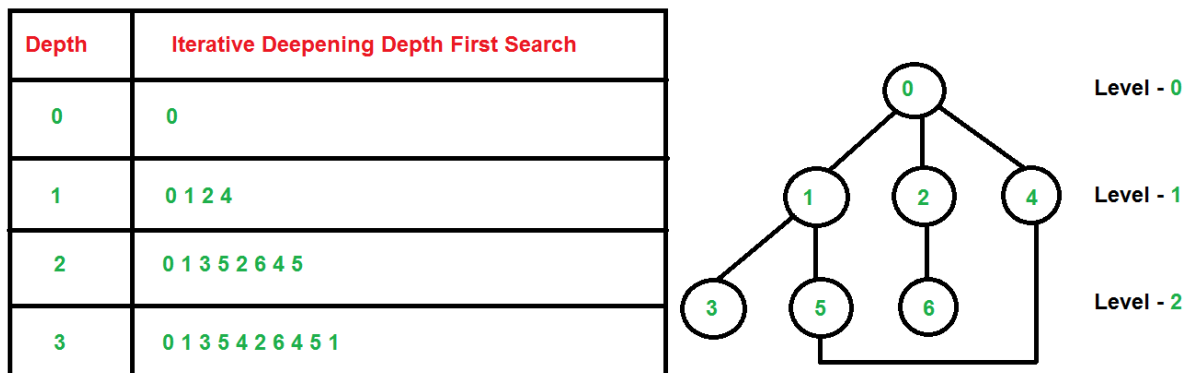
Tugas besar kali ini adalah untuk membuat program berbasis web dalam bahasa Go yang mengimplementasikan algoritma IDS dan BFS untuk menyelesaikan permainan WikiRace. Program menerima masukan berupa jenis algoritma, judul artikel awal, dan judul artikel tujuan. Program memberikan keluaran berupa jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel (dari artikel awal hingga artikel tujuan), dan waktu pencarian (dalam ms). Program cukup mengeluarkan salah satu rute terpendek saja dengan waktu kurang dari 5 menit untuk setiap permainan.

BAB 2

LANDASAN TEORI

A. Iterative Deepening Search (IDS)

Iterative Deepening Search (IDS) merupakan sebuah algoritma pencarian ruang keadaan yang memadukan karakteristik Depth-First Search (DFS) dan Breadth-First Search (BFS). Ia bekerja dengan cara berulang kali menjalankan pencarian berbasis DFS namun dengan batasan kedalaman yang terus ditingkatkan secara iteratif. Pada iterasi awal, kedalaman pencarian dibatasi pada nilai minimum, kemudian pada iterasi selanjutnya, batas kedalaman tersebut dinaikkan bertahap. Proses ini terus berulang hingga solusi ditemukan atau seluruh ruang pencarian telah dieksplorasi



Gambar 1 Ilustrasi IDS

Sumber : <https://geeksforgeeks.org>

B. Breadth-First Search (BFS)

Algoritma Breadth-First Search (BFS) merupakan teknik pencarian yang sistematis untuk menemukan simpul target dalam graf. Prinsip dasar BFS adalah mengunjungi simpul yang bertetangga dengan simpul awal terlebih dahulu, kemudian dilanjutkan dengan mengunjungi simpul tetangga dari simpul-simpul tersebut secara level demi level. BFS memanfaatkan struktur data Queue (Antrian) untuk menyimpan simpul yang akan dikunjungi.

BFS didasarkan pada konsep graf yang terdiri dari kumpulan simpul (vertex) dan sisi (edge) yang menghubungkan antar simpul. Setiap simpul dapat memiliki keadaan (state)

tertentu yang bisa jadi merupakan tujuan pencarian. Algoritma BFS mengeksplorasi graf secara level demi level dengan menambahkan simpul tetangga yang belum dikunjungi ke dalam Queue. Simpul yang diambil dari Queue kemudian ditandai sebagai telah dikunjungi dan keadaan (state) dari simpul tersebut diperiksa. Jika simpul tersebut merupakan target yang dicari, maka proses pencarian selesai. Jika tidak, seluruh simpul tetangga yang belum dikunjungi ditambahkan ke dalam Queue. Proses ini diulang hingga simpul target ditemukan atau seluruh graf telah dijelajahi.

BAB 3

ANALISIS PEMECAHAN MASALAH

A. Langkah Pemecahan Masalah

Berikut adalah langkah umum pemecahan WikiRace:

1. Website dijalankan melalui docker
2. Website menerima input title artikel awal dan artikel tujuan dari Form Entry yang disediakan
3. Program memanfaatkan Wiki API untuk melakukan autocomplete pencarian Wiki Article bersama URLnya
4. Website menerima input algoritma pencarian (IDS/BFS) dari Form Entry yang disediakan
5. Program menjalankan algoritma IDS atau BFS untuk mendapatkan jalur dari artikel awal menuju artikel tujuan serta jumlah artikel yang diperiksa
6. Website menampilkan jalur yang didapatkan, beserta jumlah artikel yang dilalui, jumlah artikel yang diperiksa, dan waktu yang diperlukan untuk menemukan solusi

B. Algoritma IDS

Dalam kasus web scraping dengan menggunakan algoritma IDS, setiap laman wikipedia dianggap sebagai *node*. Setiap halaman wikipedia atau *node* bisa memiliki banyak *child node* berupa URL yang terhubung dengan halaman wikipedia yang lain. Pencatatan rute dilakukan dengan menggunakan array yang dioperasikan sesuai dengan aturan penelusuran algoritma IDS.

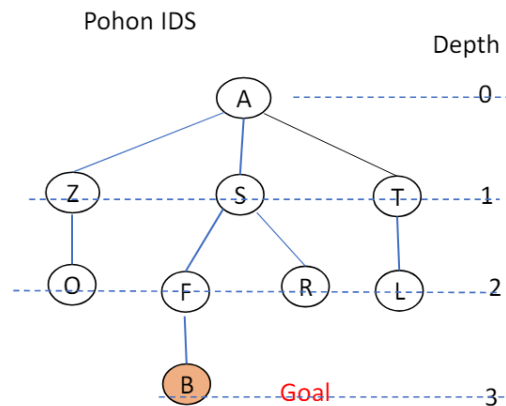
Berikut adalah langkah umum algoritma IDS yang diimplementasikan:

1. Menerima URL awal dan URL tujuan
2. Menganggap URL awal, URL tujuan, dan kumpulan URL yang akan dikunjungi sebagai node
3. Dilakukan iterasi untuk setiap kedalaman maksimal
4. Pada iterasi pertama, ketika kedalaman maksimal sama dengan nol, dilakukan pengecekan apakah URL awal sudah sama dengan URL tujuan, jika sama maka penelusuran berhenti
5. Pada iterasi kedua, kedalaman maksimal bertambah satu, dilakukan scrapping untuk mendapatkan semua URL pada laman tersebut.
6. Proses dilanjutkan dengan Iterasi setiap URL pada laman yang ditemukan dan melakukan rekursif dengan elemen iterasi sekarang sebagai URL. Jika URL awal dan URL tujuan sudah sama maka penelusuran dihentikan dan tidak perlu

dilanjutkan ke iterasi berikutnya dengan kedalaman yang berbeda. Jika belum sama, dilakukan pengecekan apakah kedalaman sekarang sudah lebih dari atau sama dengan dengan kedalaman maksimal. Apabila sudah, rekursif tidak akan dilakukan, kemudian berlanjut ke iterasi berikutnya dengan kedalaman maksimal yang ditambah dengan satu. Apabila kedalaman sekarang belum lebih dari atau sama dengan kedalaman maksimal maka proses rekursif akan dilanjutkan

7. Selama penelusuran jalur yang sudah dikunjungi disimpan pada suatu array, apabila jalur tersebut tidak mengarah pada URL tujuan, maka akan di *pop* atau dihapus dari array. Jalur akhir yang tetap berada pada array adalah jalur yang mengarah pada URL tujuan.

Berikut adalah ilustrasi kasus:



Gambar 2 Ilustrasi Implementasi Algoritma IDS

Sumber: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/stima23-24.html>

Asumsikan huruf A, Z, S, T, O, F, R, L, dan B adalah URL menuju suatu halaman wikipedia. Implementasi algoritma IDS kami akan melakukan iterasi mulai dari kedalaman maksimal 0 dan terus mengalami *increment* hingga solusi ditemukan atau waktu lebih dari 5 menit. Berikut adalah jalur penelusuran untuk setiap iterasi

1. Iterasi pertama (kedalaman 0) : A
2. Iterasi kedua (kedalaman 1) : A-Z-S-T
3. Iterasi ketiga (kedalaman 2) : A - Z - O - S - F - R - T - L
4. Iterasi keempat (kedalaman 3) : A - Z - O - S - F - B (solusi ditemukan, penelusuran dihentikan)

C. Algoritma BFS

Secara umum, elemen BFS terdiri atas matriks ketetanggaan yang menyimpan simpul-simpul dan tetangganya, queue yang menyimpan simpul yang akan dikunjungi, dan tabel boolean yang menyatakan apakah suatu simpul telah dikunjungi. Dalam kasus Web Scraping, matriks ketetanggaan tidak diperlukan dan pengunjungan tetangga dilakukan melalui Web Crawling. Tabel boolean dapat digantikan dengan sebuah array “visited” yang menyimpan setiap URL yang telah dikunjungi. Dalam implementasi kami, queue dan array visited digabungkan lebih lanjut menjadi satu array “list” untuk kenyamanan. Pencatatan rute dilakukan menggunakan struktur data *tree*. Pengambilan rute solusi dapat diperoleh dengan mencari *parent* dari *node* solusi sampai mencapai *root node*. Jumlah artikel yang diperiksa dapat diperoleh dengan menggunakan sebuah *counter* yang di-increment setiap kali suatu artikel dikunjungi.

Perlu dicatat juga bahwa walaupun secara standar pencarian BFS selesai ketika tujuan dikunjungi (dikeluarkan dari queue), pencarian dapat dipercepat apabila pencarian diselesaikan ketika tujuan ditemukan (dimasukkan ke dalam queue). Untuk mempercepat algoritma, algoritma kami pun menyelesaikan pencarian ketika artikel tujuan ditemukan dan bukan ketika artikel tujuan dikunjungi.

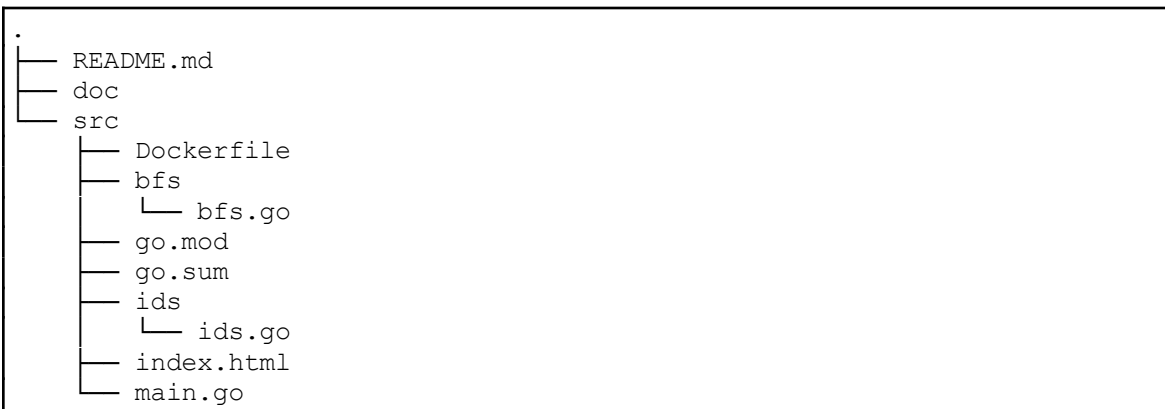
Berikut adalah langkah umum algoritma BFS yang diimplementasikan:

1. Algoritma menerima masukan berupa URL awal dan URL tujuan
2. Buat sebuah array “list” yang dapat di-iterasi menggunakan sebuah iterator “i”, beserta *counter* untuk mencatat jumlah artikel yang diperiksa. “List” merupakan gabungan dari queue berisi URL yang akan dikunjungi dan array visited berisi URL yang telah dikunjungi.
3. Periksa URL yang berada di depan queue (list[i]) dan increment *counter*. Jika URL adalah URL tujuan, pencarian selesai
4. Jika bukan, cari URL lain dalam URL tersebut yang menuju ke sebuah wiki page.
5. Jika ditemukan URL yang merupakan URL tujuan, pencarian selesain.
6. Jika tidak, masukkan semua URL tersebut ke dalam list. Tambahkan juga simpul-simpul pada tree dengan isi simpul tersebut berupa URL lain yang ditemukan dan *parent* dari simpul tersebut berupa URL yang sedang diperiksa sekarang.
7. Increment iterator “i” dan ulangi langkah 3-6 sampai pencarian selesai atau list sudah ditelusuri sepenuhnya.
8. Cari rute menuju URL tujuan dengan menelusuri tree. Jika solusi tidak ditemukan, rute kosong.

9. Kembalikan rute yang diperoleh beserta *counter*.

D. Arsitektur dan Fitur Program

Program merupakan aplikasi web yang menggabungkan frontend dan backend di dalam folder 'src/' dan dibangun menggunakan bahasa Go, HTML, dan JavaScript. Frontend bertanggung jawab atas tampilan dan interaksi pengguna, dengan menggunakan HTML untuk struktur halaman dan JavaScript untuk logika interaksi. Pengiriman permintaan antara frontend dan backend ditangani oleh pustaka HTMX, yang memungkinkan aplikasi untuk berkomunikasi dengan server secara dinamis tanpa perlu memuat ulang seluruh halaman. Di sisi backend, bahasa Go digunakan untuk mengelola logika bisnis, menangani permintaan dari frontend, dan menyediakan data kepada pengguna. Program ini menawarkan interaktivitas dinamis, pemisahan yang jelas antara frontend dan backend untuk kemudahan pengembangan dan pemeliharaan, serta responsivitas yang tinggi terhadap tindakan pengguna. Berikut adalah struktur direktori program yang dibuat.



1. Direktori Induk (root):
 - i. README.md: fail yang berisi dokumentasi atau panduan singkat tentang proyek.
 - ii. doc: direktori yang berisi laporan terkait dokumentasi proyek.
 - iii. src: direktori yang berisi kode sumber program.
2. Direktori doc:
 - i. Laporan.docx: file laporan
3. Direktori src:
 - i. Dockerfile: fail yang digunakan untuk membangun Docker image dari aplikasi. Dockerfile memastikan program dapat dijalankan secara konsisten di berbagai lingkungan komputasi.

- ii. bfs Directory: Direktori ini berisi fail bfs.go yang berisi implementasi algoritma BFS.
- iii. ids Directory: Direktori ini berisi fail ids.go yang berisi implementasi algoritma IDS.
- iv. go.mod: fail berisi daftar dependensi proyek dan versi-versinya
- v. go.sum: fail yang berisi checksum dari dependensi yang digunakan di go.mod.
- vi. index.html: fail HTML menyajikan halaman utama dari program.
- vii. main.go: fail utama yang memulai server web dan berfungsi sebagai titik masuk untuk program yang dibangun.

Program yang dibuat memiliki fitur sebagai berikut,

1. Autosuggestion: menampilkan rekomendasi judul beserta tautan artikel saat pengguna mengetik masukan.
2. Pencarian path menggunakan algoritma IDS
3. Pencarian jalur menggunakan algoritma BFS
4. Graph result: menampilkan jalur dalam bentuk graf berarah

BAB 4

IMPLEMENTASI DAN PENGUJIAN

A. Spesifikasi Teknis Program

A.1. Algoritma Titik Masuk Program

Algoritma titik masuk merupakan fungsi `main()` pada fail `main.go`. bertanggung jawab atas inisialisasi server web dan penanganan permintaan untuk tiga endpoint API, yaitu `/`, `/search`, dan `/race`. Endpoint root (`/`): Ketika pengguna mengakses URL utama program, server akan menyajikan halaman `index.html`. Ini adalah halaman beranda atau halaman utama dari aplikasi web. Endpoint `/search`: Endpoint ini bertanggung jawab untuk melayani fitur autosuggestion ketika pengguna memasukkan judul artikel awal dan akhir. Ketika pengguna mengirimkan permintaan pencarian, server akan menanggapi dengan memberikan saran atau rekomendasi berdasarkan judul artikel yang dimasukkan pengguna. Endpoint `/race`: Endpoint ini melayani pencarian artikel berdasarkan algoritma yang dipilih oleh pengguna. Selain memberikan hasil pencarian, endpoint ini juga memberikan informasi lain seperti nilai waktu, jumlah artikel yang diperiksa, jumlah artikel yang dilalui, dan rute penjelajahan artikel dalam bentuk graf.

Semua endpoint tersebut terhubung dengan frontend dan backend menggunakan pustaka `htmx`. Hal ini memungkinkan komunikasi antara frontend (HTML dan JavaScript) dan backend (Go) untuk berjalan dengan lancar dan responsif tanpa memuat ulang halaman secara keseluruhan. Dengan menggunakan `htmx`, interaksi antara pengguna dan aplikasi web menjadi lebih dinamis dan responsif, serta meningkatkan pengalaman pengguna.

A.2. Algoritma IDS

IDS terdiri atas fungsi berikut:

1. `func doesNotContainAnyPrefix(s string) bool`: fungsi utilitas yang berfungsi memeriksa apakah sebuah URL “s” mengandung prefix namespace
2. `func getWikipediaTitleFromURL(url string) string`: fungsi utilitas yang berfungsi memperoleh title dari URL “url”
3. `func Main(startURL, goalURL string) (*graph.Graph[string, string], int)`: fungsi ini menerima URL awal, URL tujuan dan mengembalikan alamat graf dari jalur hasil penelusuran dan beserta jumlah artikel yang diperiksa
4. `func IDS(startURL, goalURL string, currentDepth int, maxDepth int, visited []string, cek *int, periksa *map[string]bool, visited_all []string)`: prosedur ini menerima URL awal, URL tujuan, kedalaman sekarang,

kedalaman maksimum, jalur yang dikunjungi, variabel pengecekan, map untuk menghitung banyak artikel yang diperiksa, dan artikel yang diperiksa untuk kedalaman maksimum tertentu. Prosedur ini berjalan secara rekursif dengan `currentDepth` yang mengalami *increment* menuju kasus basis. Kasus basis adalah ketika `currentDepth` lebih besar atau sama dengan `maxDepth` dan ketika `startURL` dan `goalURL` bernilai sama.

Fungsi Main secara umum terdiri atas variabel variabel berikut:

1. `visited := []string{} (slice/dynamic array of string)`

Variabel ini akan berisi URL-URL jalur yang dikunjungi ketika melakukan penelusuran secara IDS.

2. `periksa := make(map[string]bool)var tempArrayURL []string`

Variabel ini adalah map yang memetakan URL-URL yang telah diperiksa, `true` jika sudah diperiksa, dan `false` jika belum diperiksa

3. `g : graph.Graph`
4. `func findPath(id uint) []string`

“g” merupakan graf akhir yang menggambarkan rute dari artikel awal menuju artikel tujuan.

Fungsi IDS secara umum terdiri atas variabel variabel berikut:

1. `var tempArrayURL []string`

Variabel ini digunakan untuk menyimpan URL-URL yang ada pada halaman website tertentu

2. `c : *collyCollector`

`collyCollector` adalah variabel utama yang digunakan untuk melakukan Web Scraping. `collyCollector` memiliki beberapa method utama sebagai berikut:

3. `Visit(URL string)`

`Visit` memulai pekerjaan `Collector` dengan `Request` ke URL yang ditentukan dalam parameter.

4. `OnHTML(string, colly.HTMLCallback)`

`OnHTML` dapat digunakan untuk menemukan setiap link yang berada dalam URL yang sedang dikunjungi. Link-link yang ditemukan dapat

dipastikan menuju suatu wiki page dengan diperiksa menggunakan suatu Regex. Setiap kali link ditemukan, link tersebut ditambahkan ke dalam variabel tempArrayURL.

A.3. Algoritma BFS

BFS terdiri atas fungsi berikut

1. func doesNotContainAnyPrefix(s string) bool: fungsi utilitas yang berfungsi memeriksa apakah sebuah URL “s” mengandung prefix namespace
2. func getWikipediaTitleFromURL(url string) string: fungsi utilitas yang berfungsi memperoleh title dari URL “url”
3. func Main(startURL, goalURL string) (*graph.Graph[string, string], int): fungsi utama BFS yang mencari rute dari startURL menuju goalURL beserta jumlah artikel yang diperiksa

Fungsi Main secara umum terdiri atas variabel-variabel dan fungsi-fungsi berikut:

1. list : []string (slice/dynamic array of string)

visit_id : uint

found : bool

“list” merupakan gabungan queue dan array visited yang menyimpan URL-URL yang telah dikunjungi dan akan diperiksa. List diiterasi menggunakan visit_id yang menandakan *front* dari queue. Pencarian selesai apabila visit_id melebihi panjang dari “list” yang ditandai dengan variabel found yang bernilai true.

2. t : tree.Tree

parent_id, current_id, final_id : uint

“t” adalah struktur data *tree*/pohon yang berfungsi mencatat *parent* dari URL. Pencarian rute solusi dilakukan dengan menelusuri “t” dari daun menuju akarnya. Pembuatan simpul pada “t” dibantu dengan parent_id yang berfungsi menandakan id dari *parent* dan current_id. Simpul yang dibuat memiliki data berupa URL dengan *parent* dari simpul tersebut ditandai dengan parent_id dan simpul tersebut ditandai dengan current_id. “final_id” berfungsi menandakan id simpul yang mengandung URL tujuan (apabila solusi ditemukan)

3. c : *collyCollector

collyCollector adalah variabel utama yang digunakan untuk melakukan Web Scraping. collyCollector memiliki beberapa method utama sebagai berikut:

a. Visit(URL string)

Visit memulai pekerjaan Collector dengan Request ke URL yang ditentukan dalam parameter.

b. OnResponse(f ResponseCallback)

OnResponse dijalankan ketika Collector menerima Response, yaitu, secara efektif, ketika URL dikunjungi. Jumlah artikel yang dikunjungi dapat dihitung dengan menambahkan *counter* setiap kali OnResponse dijalankan.

c. OnHTML(string, colly.HTMLCallback)

OnHTML dijalankan setelah OnResponse. OnHTML dapat digunakan untuk menemukan setiap link yang berada dalam URL yang sedang dikunjungi. Link-link yang ditemukan dapat dipastikan menuju suatu wiki page dengan diperiksa menggunakan suatu Regex. Setiap kali link ditemukan, link tersebut ditambahkan ke dalam variabel list dan ke dalam tree “t”.

4. c : *collyCollector

5. g : graph.Graph

```
func findPath(id uint) []string
```

“g” merupakan graf akhir yang menggambarkan rute dari artikel awal menuju artikel tujuan. Rute dapat dicari dari fungsi findPath yang meminta parameter id URL tujuan dan mengembalikan rute dari artikel awal menuju artikel tujuan.

B. Penjelasan Tata Cara Penggunaan Program

Struktur halaman aplikasi web berbentuk sebagai berikut,

Wiki Race

Title



Algorithm

☐ Iterative Deepening Search (IDS)

☐ Breadth First Search (BFS)

START

00 : 00

How To Use

1. Select the starting title from the autocomplete suggestions
2. Choose the goal title from the autocomplete suggestions
3. Pick the desired search algorithm
4. Initiate the search process by clicking the START button.
5. Await the display of the output

About Us



Ariel Herfrison

13522002



Panjri Sri Kuncara

13522028



M. Dava Fathurrahman

13522114


Tata cara penggunaan program adalah sebagai berikut,

1. Pilih judul awal dari saran otomatis.
Pilih judul awal dari daftar saran yang muncul saat mengetik di kotak Start Title.
2. Pilih judul tujuan dari saran otomatis.
Pilih judul yang ingin Anda capai dari daftar saran yang muncul saat mengetik di kotak Goal Title.
3. Pilih algoritma pencarian yang diinginkan.
Pilih algoritma pencarian yang diinginkan, IDS atau BFS.
4. Mulai proses pencarian dengan mengklik tombol MULAI.
Setelah memilih judul awal, judul tujuan, dan algoritma pencarian, klik tombol "START" untuk memulai proses pencarian.
5. Tunggu tampilan hasil.
Setelah proses pencarian dimulai, tunggu beberapa saat sampai hasilnya ditampilkan. Hal ini tergantung pada kompleksitas algoritma dan jumlah data yang diproses.

C. Hasil Pengujian

1.1 Faith → Person (2 artikel)

BFS:

 Nasi Goreng Magolang [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

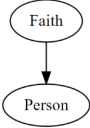
Faith → Person

Algorithm

☐ Iterative Deepening Search (IDS) ☒ Breadth First Search (BFS)

START

BFS Output




```
graph TD; Faith((Faith)) --> Person((Person));
```

Number of checked article: 1
Number of article in solution: 1
Time: 229 ms

00 : 00

IDS:

 Nasi Goreng Magolang [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

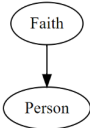
Faith → Person

Algorithm

☒ Iterative Deepening Search (IDS) ☐ Breadth First Search (BFS)

START

IDS Output



```
graph TD; Faith((Faith)) --> Person((Person));
```

Number of checked article: 1
Number of article in solution: 1
Time: 204 ms

00 : 00

1.2 Adolf Hitler → Germany (2 artikel)

BFS:

Wiki Race

Title

Adolf Hitler



Germany

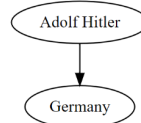
Algorithm

☐ Iterative Deepening Search (IDS)

☒ Breadth First Search (BFS)

START

BFS Output



Number of checked article: 1

Number of article in solution: 1

Time: 373 ms

00 : 00

IDS:

Wiki Race

Title

Adolf Hitler



Germany

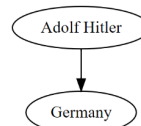
Algorithm

☒ Iterative Deepening Search (IDS)

☐ Breadth First Search (BFS)

START

IDS Output



Number of checked article: 1

Number of article in solution: 1

Time: 268 ms

00 : 00

2.1 Vector → Mathematics (3 artikel - cepat)

BFS:



Wiki Race

Title

Vector



Mathematics

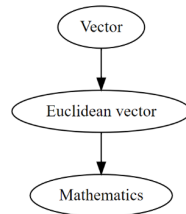
Algorithm

☐ Iterative Deepening Search (IDS)

☒ Breadth First Search (BFS)

START

BFS Output



Number of checked article: 2

Number of article in solution: 2

Time: 505 ms

00 : 00

IDS:



Wiki Race

Title

Vector



Mathematics

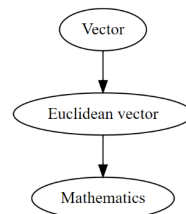
Algorithm

☒ Iterative Deepening Search (IDS)

☐ Breadth First Search (BFS)

START

IDS Output



Number of checked article: 2


Number of article in solution: 2

Time: 489 ms

00 : 00

2.2 Toilet → Bog (3 artikel)

BFS:

 Nasi Goreng Magolang [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

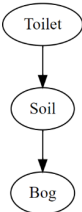
Toilet → Bog

Algorithm

☐ Iterative Deepening Search (IDS) ☒ Breadth First Search (BFS)

START

BFS Output




```
graph TD; Toilet([Toilet]) --> Soil([Soil]); Soil --> Bog([Bog]);
```

Number of checked article: 77
Number of article in solution: 2
Time: 6366 ms

00 : 06

IDS:

 Nasi Goreng Magolang [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

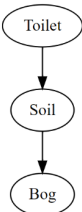
Toilet → Bog

Algorithm

☒ Iterative Deepening Search (IDS) ☐ Breadth First Search (BFS)

START

IDS Output




```
graph TD; Toilet([Toilet]) --> Soil([Soil]); Soil --> Bog([Bog]);
```

Number of checked article: 77
Number of article in solution: 2
Time: 3049 ms

00 : 03

3.1 Vector → Knowledge (4 artikel)

BFS:

 Nasi Goreng Magolang

[Home](#) [How To Use](#) [About](#)

Wiki Race

Title

Vector

→

Knowledge

Algorithm

☐ Iterative Deepening Search (IDS)

☒ Breadth First Search (BFS)

START

BFS Output

Vector

↓

Euclidean vector

↓

Mathematics

↓

Knowledge

Number of checked article: 84

Number of article in solution: 3

Time: 3081 ms

00 : 03

IDS:



Wiki Race

Title

Vector

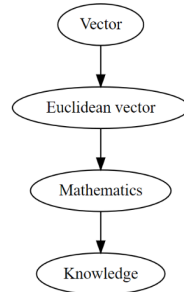


Knowledge

Algorithm

☒ Iterative Deepening Search (IDS)☐ Breadth First Search (BFS)**START**

IDS Output



Number of checked article: 84

Number of article in solution: 3

Time: 3011 ms

00 : 03

3.2 Gagaku → Brokeback Mountain (4 artikel)

BFS:



Wiki Race

Title

Gagaku

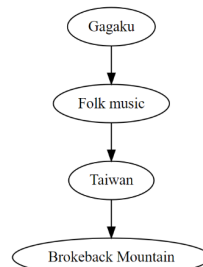


Brokeback Mountain

Algorithm

☐ Iterative Deepening Search (IDS)☒ Breadth First Search (BFS)**START**

BFS Output




Number of checked article: 2011

Number of article in solution: 3

Time: 187221 ms

02 : 48

IDS:

 [Nasi Goreng Magolang](#) [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

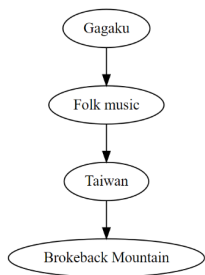
→

Algorithm

☒ Iterative Deepening Search (IDS) ☐ Breadth First Search (BFS)

[START](#)

IDS Output




```
graph TD; A([Gagaku]) --> B([Folk music]); B --> C([Taiwan]); C --> D([Brokeback Mountain]);
```

Number of checked article: 691
Number of article in solution: 3
Time: 27586 ms

00 : 27

3.3 Espada → Bleach (4 artikel)

BFS:

 [Nasi Goreng Magolang](#) [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

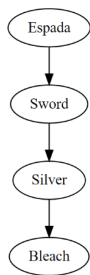
→

Algorithm

☐ Iterative Deepening Search (IDS) ☒ Breadth First Search (BFS)

[START](#)

BFS Output




```
graph TD; A([Espada]) --> B([Sword]); B --> C([Silver]); C --> D([Bleach]);
```

Number of checked article: 64
Number of article in solution: 3
Time: 3111 ms

00 : 03

IDS:

 [Nasi Goreng Magolang](#) [Home](#) [How To Use](#) [About](#)

Wiki Race

Title

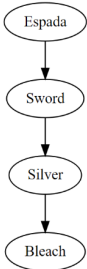
Espada → Bleach

Algorithm

☒ Iterative Deepening Search (IDS) ☐ Breadth First Search (BFS)

[START](#)

IDS Output



```
graph TD; Espada --> Sword; Sword --> Silver; Silver --> Bleach;
```

Number of checked article: 64
Number of article in solution: 3
Time: 2674 ms

00 : 02

4.1 Memphis Belle → Shetland Islands (5 artikel)

BFS:

Wiki Race

Title

Memphis Belle



Shetland Islands

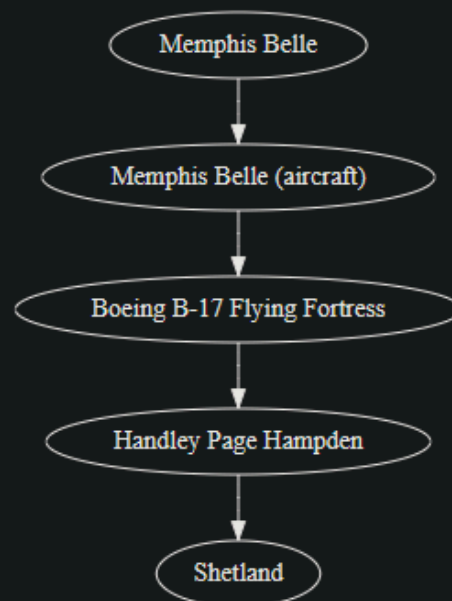
Algorithm

☐ Iterative Deepening Search (IDS)

☒ Breadth First Search (BFS)

START

BFS Output



Number of checked article: 902

Number of article in solution: 4

Time: 2 minutes 28.52695 seconds

01 : 43

IDS:

Nasi Goreng Magelang
Home
How To Use
About

Wiki Race

Title

Memphis Belle

→

Shetland Islands

Algorithm

☒ Iterative Deepening Search (IDS)
☐ Breadth First Search (BFS)

START

IDS Output

```

graph TD
    A([Memphis Belle]) --> B([Memphis Belle (aircraft)])
    B --> C([Boeing])
    C --> D([Helicopter])
    D --> E([Shetland])

```

Number of checked article: 689
Number of article in solution: 4
Time: 1 minutes 31.25739 seconds
01 : 18

5. Espada → Bleach (pengaruh koneksi internet yang berbeda kualitas)

Nasi Goreng Magelang
Home
How To Use
About

Wiki Race

Title

Espada

→

Bleach

Algorithm

☐ Iterative Deepening Search (IDS)
☒ Breadth First Search (BFS)

START

BFS Output

```

graph TD
    A([Espada]) --> B([Sword])
    B --> C([Silver])
    C --> D([Bleach])

```

Number of checked article: 64
Number of article in solution: 3
Time: 15524 ms
00 : 15

Wiki Race

Title

Espada



Bleach

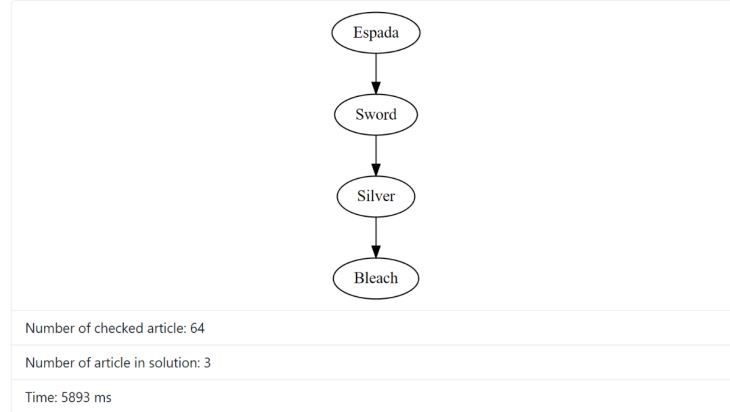
Algorithm

☒ Iterative Deepening Search (IDS)

☐ Breadth First Search (BFS)

START

IDS Output



00 : 05

D. Analisis Hasil Pengujian

Berdasarkan hasil ujicoba 1, 2, dan 3, terlihat bahwa waktu pencarian meningkat secara eksponensial. Meskipun perbedaan waktu antara 1 artikel dengan 2 artikel sangat kecil, perbedaan waktu antara 2 artikel dengan 3 artikel cukup meningkat dan perbedaan waktu antara 3 artikel dan 4 artikel sangat besar. Hal ini karena program terhalangi kebutuhan untuk mengunjungi semua artikel dalam suatu kedalaman sebelum dapat memulai pencarian pada kedalaman selanjutnya. Misalnya waktu pengunjungan adalah 1 detik, waktu pencarian adalah 0.1 detik, dan setiap artikel memiliki tautan menuju 10 artikel lainnya, maka apabila rute solusi terdiri atas 3 artikel, pada kasus terburuk, program hanya harus mengunjungi 10 artikel dan memeriksa 100 artikel sehingga memakan waktu $(10 \times 1) + (100 \times 0.1) = 20 \text{ detik}$. Namun, apabila rute solusi terdiri atas 4 artikel, maka pada kasus terburuk, program harus mengunjungi 100 artikel dan memeriksa 1000 artikel sehingga program memakan waktu $(100 \times 1) + (1000 \times 0.1) = 200 \text{ detik}$. Hal tersebut berlaku pada kasus BFS. Pada kasus IDS, waktu akan semakin lama karena algoritma harus mengulang dari awal untuk setiap iterasi dimana untuk setiap iterasi kedalaman maksimal yang dikunjungi mengalami *increment*.

Berdasarkan percobaan 8, diketahui jika koneksi dan kecepatan internet juga berpengaruh pada kecepatan pencarian. Terlihat bahwa pada penelusuran yang sama, bisa saja waktu yang dibutuhkan berbeda. Pada kondisi koneksi yang cepat dan bagus, kecepatan pencarian bisa lebih cepat dibandingkan ketika pada koneksi yang kurang bagus. Oleh karena itu, untuk mendapatkan kecepatan yang optimal, jaringan internet yang bagus juga harus menjadi pertimbangan.

BAB 5

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil uji coba, ditemukan bahwa algoritma IDS dan BFS memiliki kecepatan waktu yang cukup sama apabila kedalaman pencarian cukup kecil. Namun, untuk kedua algoritma tersebut, waktu pencarian meningkat secara eksponensial apabila panjang rute solusi bertambah. Terdapat juga beberapa faktor lain yang mempengaruhi kecepatan pencarian, di antaranya adalah kelancaran jaringan internet dan volume ruang pencarian (ukuran wiki page).

B. Saran

- Spesifikasi dan batasan dijelaskan dengan lebih rinci dan dengan detail yang lebih mendalam.
- Spesifikasi jangan terlalu restriktif dan informasi batasannya jangan terlalu umum.

C. Refleksi

- Pertimbangan mengenai *edge case* memang penting, tapi sebaiknya penyelesaian kasus-kasus normal harus jadi prioritas utama terlebih dahulu.

LAMPIRAN

Tautan repository:

https://github.com/mdavaf17/Tubes2_Nasi-Goreng-MaGolang

Tautan video:

https://youtu.be/_197y064nUM

DAFTAR PUSTAKA

Munir, R. (n.d.). Homepage Rinaldi Munir.
<https://informatika.stei.itb.ac.id/~rinaldi.munir/>