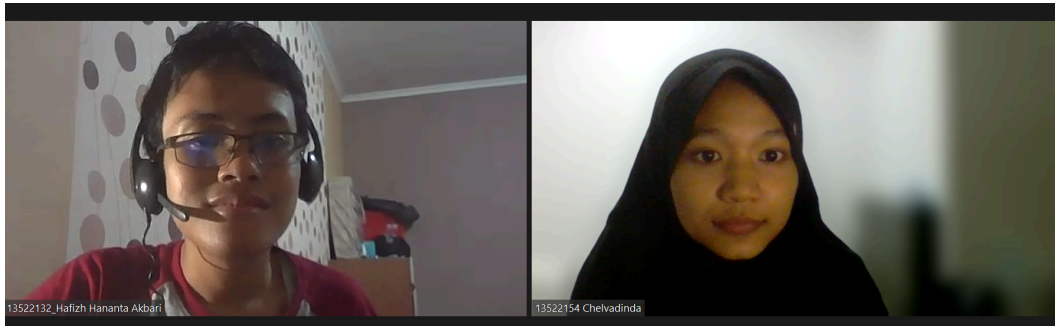


## **TUGAS BESAR 2 IF2211 STRATEGI ALGORITMA**

### **Pemanfaatan Algoritma IDS dan BFS Dalam Permainan WikiRace**



**Kelompok semogaKelar**

**Anggota:**

**Hafizh Hananta Akbari**

**(13522132)**

**Chelvadinda**

**(13522154)**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2023/2024**

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I DESKRIPSI TUGAS</b>	<b>3</b>
1.1 DESKRIPSI TUGAS	3
1.2 SPESIFIKASI TUGAS	3
<b>BAB 2 LANDASAN TEORI</b>	<b>4</b>
2.1 DASAR TEORI	4
2.1.1 GRAF	4
2.1.2 BFS	5
2.1.3 IDS	6
2.2 APLIKASI WEB	6
<b>BAB 3 ANALISIS PEMECAHAN MASALAH</b>	<b>8</b>
3.1 LANGKAH PEMECAHAN MASALAH	8
3.2 PEMETAAN MASALAH	8
3.3 FITUR FUNGSIONALITAS DAN ARSITEKTUR WEB	11
3.4 ILUSTRASI KASUS	12
<b>BAB 4 IMPLEMENTASI DAN PENGUJIAN</b>	<b>14</b>
4.1 SPESIFIKASI TEKNIS PROGRAM	14
4.2 TATA CARA PENGGUNAAN PROGRAM	19
4.3 HASIL PENGUJIAN	20
4.3.1 IDS	20
4.3.2 BFS	21
4.4 ANALISIS	21
4.4.1 IDS	21
4.4.2 BFS	22
<b>BAB 5 PENUTUP</b>	<b>24</b>
5.1 KESIMPULAN	24
5.2 SARAN	24
5.3 REFLEKSI	24
<b>LAMPIRAN</b>	<b>26</b>
<b>DAFTAR PUSTAKA</b>	<b>27</b>

# **BAB I**

## **DESKRIPSI TUGAS**

### **1.1 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.

### **1.2 Spesifikasi Tugas**

- Buatlah program 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 (cukup satu rute saja, tidak perlu seluruh rute kecuali mengerjakan bonus).
- Program berbasis web, sehingga perlu dibuat front-end dan back-end (tidak perlu di-deploy).
- Program wajib dapat mencari rute terpendek kurang dari 5 menit untuk setiap permainan.
- Program harus mengandung komentar yang jelas serta mudah dibaca.

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1 Dasar Teori**

##### **2.1.1 Graf**

Graf adalah representasi visual untuk menunjukkan hubungan antara objek-objek diskrit. Graf memiliki dua jenis komponen yaitu titik yang disebut sebagai simpul dan garis yang disebut sebagai sisi. Kedua komponen tersebut memiliki kegunaannya masing-masing dengan titik pada graf merepresentasikan objek diskrit yang ditinjau dan garis pada graf merepresentasikan hubungan antara dua objek diskrit yang berbeda. Suatu graf bisa digunakan dalam berbagai situasi karena graf mampu menggambarkan hubungan antara berbagai objek diskrit dengan jelas sehingga mampu digunakan di berbagai permasalahan seperti masalah-masalah komputasi.

Graf memiliki beberapa jenis yang berbeda. Jenis-jenis graf yang berbeda digunakan untuk merepresentasikan objek-objek diskrit yang berbeda serta hubungannya yang bisa digambarkan secara lebih sederhana atau lebih kompleks tergantung dengan permasalahannya. Pada umumnya, graf bisa dibagi menjadi dua yaitu graf sederhana dan graf tidak sederhana. Pembagian ini dilakukan dengan melihat ada atau tidak adanya gelang atau *loop* dan sisi ganda pada suatu graf. Suatu graf digolong sebagai graf sederhana apabila tidak mengandung gelang dan sisi ganda. Sebaliknya, suatu graf digolong sebagai graf tidak sederhana apabila mengandung salah satu atau kedua faktor pembeda tersebut.

Dalam perihal proses pencarian, suatu graf bisa dibagi menjadi dua yaitu graf statis dan graf dinamis. Graf statis adalah graf yang sudah terbentuk sebelum dilakukannya proses pencarian sementara graf dinamis adalah graf yang dibentuk ketika melakukan proses pencarian. Perbedaan dari kedua graf ini adalah bagaimana proses pembentukan graf berjalan dimana graf bisa dibentuk sebelum melakukan proses pencarian atau ketika melakukan proses pencarian.

Perbedaan tersebut cukup relevan terhadap pengimplementasiannya dengan kedua graf yang memiliki kelebihan dan kekurangannya. Graf statis memiliki kelebihan yaitu pemrosesan data yang lebih efektif serta analisis yang berjalan dengan stabil, namun memiliki kekurangan berupa ketidakmampuan graf untuk menunjukkan perubahan sehingga tidak fleksibel. Sebaliknya, graf dinamis memiliki kelebihan yaitu fleksibilitasnya dalam menangani perubahan antar objek serta hubungannya, namun memiliki kekurangan berupa pemrosesan yang cenderung lebih kompleks serta analisis yang bisa fluktuatif.

### 2.1.2 BFS

BFS yang merupakan singkatan dari *Breadth First Search* adalah salah satu jenis algoritma pencarian. Algoritma BFS bekerja dengan menelusuri simpul yang bertetangga terlebih dahulu sebelum meninjau anak simpul. Hal ini membuat algoritma BFS melakukan pencarian secara melebar, bukan secara mendalam. Untuk struktur data pada BFS menyangkut matriks ketetanggaan yang memiliki ukuran  $n \times n$ , antrian untuk menyimpan simpul yang sudah dikunjungi, serta tabel boolean untuk melihat apakah suatu simpul sudah dikunjungi atau belum.

BFS atau *Breadth First Search* bisa menggunakan kedua jenis graf yaitu graf statis dan graf dinamis. Penggunaan graf statis pada BFS bisa dilakukan dengan memiliki permasalahan yang tidak akan mengalami berbagai perubahan dalam datanya. Kemudian, algoritma BFS meninjau simpul-simpul secara melebar hingga mencapai solusi yang diinginkan. Sebaliknya, graf dinamis pada BFS mampu menangani perubahan dalam datanya sehingga permasalahan tidak harus merupakan permasalahan yang bersifat statis. Algoritma BFS pada graf dinamis meninjau simpul-simpul secara melebar dengan bertahap namun dengan meninjau keadaan seluruh permasalahan tersebut untuk mengakomodasi perubahan yang mungkin terjadi.

### 2.1.3 IDS

IDS atau *Iterative Deepening Search* adalah suatu jenis penerapan dari DFS. DFS sendiri yang merupakan singkatan dari Depth First Search adalah satu jenis algoritma pencarian yang menelusuri anak simpul terlebih dahulu sebelum meninjau simpul tetangga. Hal ini berarti cara kerja DFS yaitu dengan melakukan pencarian secara mendalam, bukan secara melebar. Secara struktur, DFS memiliki struktur yang sama dengan BFS. Satu-satunya perbedaan antara DFS dan BFS adalah dengan konsep pencariannya dimana DFS menelusuri simpul-simpul secara mendalam dibandingkan dengan BFS yang menelusuri simpul-simpul secara melebar sehingga memiliki kegunaan serta pengimplementasian algoritma yang sedikit berbeda.

IDS atau Iterative Deepening Search menggunakan konsep DFS dan menggunakannya secara iteratif. Hal ini dilakukan dengan melakukan pengaturan kedalaman graf yang ditinjau. Graf ditelusuri menggunakan DFS namun dengan kedalaman yang terbatas. Nilai kedalaman bisa ditingkatkan ketika belum menemukan solusi yang diinginkan dan akan berhenti ketika solusi sudah ditemukan. Karena penelusurannya yang berdasarkan kedalaman yang berbeda-beda, IDS bisa terlihat mirip dengan BFS meskipun menggunakan konsep serta algoritma yang berbeda.

## 2.2 Aplikasi Web

Aplikasi web yang dibangun adalah aplikasi yang berbasis HTML dengan menggunakan bahasa pemrograman Go sebagai back-end dari aplikasi web ini. Aplikasi ini menerima tiga inputan yaitu dua judul artikel untuk suatu halaman di Wikipedia dan satu inputan lainnya berupa jenis algoritma yang digunakan. Judul artikel yang pertama digunakan sebagai titik awal pencarian sementara judul artikel kedua digunakan sebagai solusi yang harus dicapai. Terdapat pula suatu tombol untuk memulai pencarian.

Setelah mendapat ketiga inputan tersebut, tombol untuk memulai pencarian mampu digunakan untuk memulai pencarian artikel tujuan. Aplikasi menelusuri jalan yang tercepat untuk mencapai artikel yang kedua dengan artikel pertama sebagai titik

awal. Setelah melakukan pencarian dengan waktu dibawah 5 menit, web menunjukkan tampilan baru berupa satu graf yang menunjukkan jalur yang tercepat dari artikel pertama ke artikel tujuan. Web juga menunjukkan waktu yang digunakan untuk melakukan pencarian.

## **BAB 3**

### **ANALISIS PEMECAHAN MASALAH**

#### **3.1 Langkah Pemecahan Masalah**

Pada tugas besar ini, pemecahan masalah dilakukan dengan beberapa tahap. Yang pertama adalah melakukan web scraping terhadap halaman wikipedia. Web scraping pada suatu wiki racer dilakukan untuk mengambil nilai-nilai yang dicari dari suatu halaman wikipedia. Hal ini dilakukan dengan menggunakan alat yang bernama gocolly. Gocolly adalah suatu alat pada bahasa Go yang memiliki berbagai fungsi yang berbeda termasuk mengambil nilai-nilai tertentu di situs yang sedang dikunjungi. Mengambil nilai hyperlink pada halaman wikipedia dilakukan dengan menggunakan Gocolly dengan situs yang berupa input dari pengguna.

Selain web scraping, langkah berikutnya pada pemecahan masalah adalah dengan memetakan permasalahan untuk setiap algoritma yang digunakan yaitu BFS dan IDS. Pemetaan masalah ini bertujuan untuk mengkategorikan nilai-nilai yang diproses sesuai dengan komponen nilai masing-masing pada algoritma BFS dan IDS sehingga mampu diproses dengan BFS atau IDS dengan baik.

#### **3.2 Pemetaan Masalah**

##### **3.2.1 IDS**

IDS adalah algoritma pencarian dengan meninjau node-node secara mendalam yang dibatasi oleh program dengan setiap iterasi menambahkan kedalamannya hingga ditemukan hal yang dicari. Karena itu, IDS memiliki beberapa elemen yang digunakan dalam algoritmanya. Berikut ini adalah elemen-elemen IDS serta pemetaan data-data yang digunakan pada wiki racer ini.

1. Rekursi:

Elemen ini lebih merujuk ke metode dalam algoritma yang dibuat. Pada algoritma IDS yang dibuat, rekursi dilakukan untuk meningkatkan tingkat kedalaman pada setiap iterasinya. Hal ini selaras dengan definisi IDS (Iterative Deepening Search) yaitu pengecekan secara mendalam



sesuai dengan jumlah iterasi yang menambah tingkat kedalamannya pada setiap iterasinya.

2. Simpul:

Elemen ini lebih merujuk ke unsur-unsur pada konsep graf yang digunakan pada IDS. Unsur-unsur berupa simpul awal yaitu simpul pertama yang ditinjau oleh algoritma IDS, simpul tujuan yaitu simpul yang dituju dari simpul awal, simpul transisi yaitu simpul-simpul yang tidak termasuk pada simpul awal dan simpul tujuan, serta tepi yang merepresentasikan hubungan antara dua simpul berbeda. Elemen-elemen tersebut digunakan pada Wiki Racer ini dimana simpul awal berupa Start Article, simpul akhir berupa Target Article, simpul transisi berupa semua URL yang didapat dari hasil scrape, dan tepi yang berupa semua link yang merujuk ke halaman Wikipedia yang di-scrape oleh program.

3. Kedalaman:

Pada algoritma IDS, elemen kedalaman digunakan sebagai suatu batasan dalam berjalannya proses scraping. Secara definisi, algoritma IDS melakukan pencarian secara mendalam dengan kedalaman yang terus bertambah seiring dengan bertambahnya jumlah iterasi. Kedalaman pada definisi merupakan suatu batasan dimana pencarian yang dilakukan dengan algoritma IDS dilakukan hingga kedalaman yang telah terdefinisi sehingga menjadi suatu batasan pada algoritma IDS ketika melakukan pencariannya. Batasan ini didefinisikan sebagai depth pada program dengan kedalaman maksimum didefinisikan sebagai maxDepth agar program tidak berjalan dengan terlalu lama.

4. Array berupa Visited:

Pada algoritma IDS yang dibuat, terdapat elemen berupa array yang didefinisikan sebagai visited. Array ini memiliki fungsi untuk melihat suatu simpul dan apakah simpul tersebut sudah dikunjungi oleh

algoritma IDS atau belum. Setelah mengunjungi suatu simpul yang berupa halaman Wikipedia, array ini mengubah status halaman tersebut agar program tidak mengunjungi halaman yang sama ketika program sedang berjalan.

### **3.2.2 BFS**

BFS adalah algoritma pencarian dengan meninjau node-node secara meluas. Ini berarti node-node yang ditinjau adalah semua node yang berkaitan dengan node awal yaitu halaman wikipedia yang digunakan untuk memulai permainan sebelum meninjau node-node yang lebih dalam. Karakteristik ini memungkinkan BFS untuk memiliki beberapa elemen yang digunakan dalam algoritmanya. Berikut ini adalah elemen-elemen BFS serta pemetaan data-data yang digunakan pada wiki racer ini.

1. Queue:

Elemen ini lebih merujuk ke metode dalam algoritma yang dibuat. Pada algoritma BFS yang dibuat, queue digunakan untuk menyimpan simpul berikutnya yang akan ditinjau. Simpul yang ditinjau dimasukkan ke queue dan dikeluarkan setelah berhasil ditinjau oleh program.

2. Simpul:

Elemen ini lebih merujuk ke unsur-unsur pada konsep graf yang digunakan pada BFS. Unsur-unsur berupa simpul awal yaitu simpul pertama yang ditinjau oleh algoritma BFS, simpul tujuan yaitu simpul yang dituju dari simpul awal, simpul transisi yaitu simpul-simpul yang tidak termasuk pada simpul awal dan simpul tujuan, serta tepi yang merepresentasikan hubungan antara dua simpul berbeda. Elemen-elemen tersebut digunakan pada Wiki Racer ini dimana simpul awal berupa Start Article, simpul akhir berupa Target Article, simpul transisi berupa semua URL yang didapat dari hasil scrape, dan tepi yang berupa semua link yang merujuk ke halaman Wikipedia yang di-scrape oleh program.

### 3. Array berupa Visited:

Pada algoritma BFS yang dibuat, terdapat elemen berupa array yang didefinisikan sebagai visited. Array ini memiliki fungsi untuk melihat suatu simpul dan apakah simpul tersebut sudah dikunjungi oleh algoritma BFS atau belum. Setelah mengunjungi suatu simpul yang berupa halaman Wikipedia, array ini mengubah status halaman tersebut agar program tidak mengunjungi halaman yang sama ketika program sedang berjalan.

## 3.3 Fitur Fungsionalitas dan Arsitektur Web

### Fungsionalitas

- Front-end:
  - Menyediakan formulir atau input untuk pengguna memasukkan jenis algoritma, judul artikel awal, dan judul artikel tujuan.
  - Menampilkan hasil pencarian yang diberikan oleh back-end kepada pengguna.
- Back-end:
  - Memvalidasi input pengguna.
  - Menginisiasi algoritma BFS atau IDS sesuai permintaan pengguna.
  - Melakukan pencarian rute terpendek dengan algoritma yang dipilih.
  - Menghitung jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian.

### Arsitektur

- Front-end:
  - Menggunakan teknologi web seperti HTML, CSS, dan JavaScript.
  - Bertugas untuk menampilkan antarmuka pengguna (UI) kepada pengguna.
  - Mengirim permintaan (request) ke back-end untuk memulai pencarian rute.
- Back-end:

- Dibangun menggunakan bahasa pemrograman Go.
- Memiliki dua modul: algoritma BFS dan IDS.
- Menerima permintaan dari front-end untuk memulai pencarian rute.
- Menggunakan algoritma IDS atau BFS (sesuai permintaan pengguna) untuk menemukan rute terpendek.
- Memberikan hasil pencarian (jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, waktu pencarian) kepada front-end.

### 3.4 Ilustrasi Kasus

Proses pencarian rute dalam permainan WikiRace dimulai ketika pengguna memasukkan jenis algoritma, judul artikel awal, dan judul artikel tujuan melalui antarmuka pengguna. Setelah menerima input tersebut, front-end mengirim permintaan pencarian ke back-end menggunakan teknologi web. Pada tahap ini, back-end menerima permintaan tersebut dan mulai memprosesnya.

Setelah menerima permintaan, back-end melakukan validasi terhadap input pengguna, memastikan bahwa judul artikel awal dan tujuan valid serta jenis algoritma yang dipilih tersedia. Setelah validasi selesai, back-end memulai proses pencarian rute.

Untuk memulai pencarian rute, back-end menggunakan algoritma yang dipilih, baik itu BFS atau IDS. Pada contoh ini, mari kita asumsikan bahwa algoritma BFS dipilih. Algoritma BFS memulai penjelajahan dari artikel awal yang diberikan oleh pengguna. Ini memeriksa artikel terkait dengan artikel awal secara bertahap, tanpa memperdulikan kedalaman pencarian. Setiap artikel yang diperiksa dicatat, termasuk rute penjelajahan yang diambil.

Selama proses penjelajahan, algoritma BFS terus mencari artikel tujuan yang telah ditentukan. Begitu artikel tujuan ditemukan, pencarian dihentikan. Jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian dicatat selama proses pencarian.

Setelah menemukan rute terpendek, back-end mengirimkan hasil pencarian ke front-end. Front-end kemudian menampilkan hasil tersebut kepada pengguna melalui antarmuka pengguna. Hasil pencarian mencakup informasi seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian. Dengan demikian, proses pencarian rute dari awal hingga akhir mencakup validasi input, penjelajahan artikel menggunakan algoritma yang dipilih, pencatatan statistik, dan penyampaian hasil kepada pengguna melalui antarmuka pengguna.

## BAB 4

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1 Spesifikasi Teknis Program

- Fungsi IDS

Fungsi utama yang melakukan pencarian dari sumber ke tujuan menggunakan algoritma IDS. Fungsi ini menerima dua parameter, yaitu URL sumber dan URL tujuan, kemudian mengembalikan objek Result. Fungsi ini juga menggunakan rekursi di setiap kedalamannya untuk memastikan pencarian yang terus berjalan hingga kedalaman tertentu.

```
func IDS(sourceURL, targetURL string) Result {  
  
    source := &scraper.Page{  
  
        URL: sourceURL,  
  
        Depth: 0,  
  
    }  
  
    var result Result  
  
    startTime := time.Now()  
  
    // Penerapan IDS  
  
    for depth := 0; depth <= maxDepth; depth++ {  
  
        fmt.Printf("Depth: %d\n", depth)  
  
        result = DFS(source, targetURL, depth)  
  
        if len(result.Route) > 0 {  
  
            result.SearchTime = time.Since(startTime)  
  
            return result  
  
        }  
  
    }  
  
    result.SearchTime = time.Since(startTime)
```

```
        return result
    }
}
```

- Fungsi DFS

Fungsi ini merupakan implementasi algoritma Depth-First Search (DFS) yang digunakan dalam kerangka IDS. Fungsi ini menerima tiga parameter: page (halaman saat ini yang sedang diproses), targetURL (URL halaman tujuan yang ingin dicapai), dan depth (kedalaman pencarian yang tersisa). Fungsi ini mengembalikan objek Result yang berisi hasil pencarian.

```
func DFS(page *scraper.Page, targetURL string, depth int) Result {
    var result Result

    // Kondisi target URL berhasil ditemukan
    if page.URL == targetURL {
        fmt.Println("Target found:", page.URL)
        result.Route = append(result.Route, getPageTitle(page.Name))
        return result
    }

    if depth <= 0 {
        return result
    }

    // Proses scraping
    scraper.PerformScrape(page)
    result.ArticlesChecked++

    for _, link := range page.Links {
        childURL := *link
        childPage := &scraper.Page{
            Name:  getPageTitle(strings.TrimPrefix(childURL,
                "https://en.wikipedia.org/wiki/")),
            URL:   childURL,
            Previous: page,
            Depth:  page.Depth + 1,
        }

        childResult := DFS(childPage, targetURL, depth-1)
        result.ArticlesChecked += childResult.ArticlesChecked

        if len(childResult.Route) > 0 {
            result.Route = append(result.Route, getPageTitle(page.Name))
            result.Route = append(result.Route, childResult.Route...)
            return result
        }
    }
}
```

```

    }

    return result
}

```

- Fungsi BFS

Program ini adalah sebuah fungsi untuk melakukan pencarian jalur menggunakan algoritma Breadth-First Search (BFS) dari suatu URL awal ke URL target di dalam sebuah situs web Wikipedia. Fungsi ini menerima input berupa judul artikel awal, judul artikel target, dan tipe algoritma pencarian serta menggunakan algoritma BFS untuk mencari jalur dari startURL ke targetURL. Kemudian, mengembalikan Result yang berisi informasi hasil pencarian seperti jalur yang ditemukan, jumlah artikel yang diperiksa, dan waktu yang diperlukan. BFS menggunakan queue pada pengimplementasiannya untuk menentukan halaman-halaman yang perlu dikunjungi.

```

func BFS(startURL, targetURL string, algorithm string, start *scraper.Page) Result {
    var result Result
    startTime := time.Now()

    // Queue untuk BFS
    queue := []*scraper.Page{start}

    // Map visited untuk melihat apakah suatu halaman sudah dilalui
    visited := make(map[string]bool)
    visited[start.URL] = true

    targetFound := false

    // Looping untuk melakukan BFS
    for len(queue) > 0 && !targetFound {
        currentPage := queue[0]
        queue = queue[1:]

        // Kondisi untuk apabila targetURL sudah ditemukan
        if currentPage.URL == targetURL {
            result.Route = constructRoute(currentPage)
            result.SearchTime = time.Since(startTime)
            targetFound = true
            break
        }

        // Scrape di halaman saat ini
        scraper.PerformScrape(currentPage)
    }
}

```



```

        result.ArticlesChecked++

        for _, link := range currentPage.Links {
            childURL := *link
            if !visited[childURL] {
                childPage := &scraper.Page{
                    Name:
getPageTitle(strings.TrimPrefix(childURL, "https://en.wikipedia.org/wiki/")),
                    URL:  childURL,
                    Previous: currentPage,
                    Depth:  currentPage.Depth + 1,
                }

                if childURL == targetURL {
                    // Construct the route
                    result.Route =
append([]string{getPageTitle(start.Name)}, getPageTitle(strings.TrimPrefix(childURL,
"https://en.wikipedia.org/wiki/")))

                    result.SearchTime = time.Since(startTime)
                    targetFound = true
                    break
                }

                queue = append(queue, childPage)
                visited[childURL] = true
            }
        }
    }

    // Apabila tidak ada hasilnya
    if !targetFound {
        result.SearchTime = time.Since(startTime)
    }

    return result
}

```

- Fungsi constructRoute

Berfungsi untuk membangun jalur yang ditemukan dari halaman awal hingga halaman target.

```

func constructRoute(targetPage *scraper.Page) []string {
    var route []string
    current := targetPage

    for current != nil {
        pageTitle := current.Name
        pageTitle = getPageTitle(pageTitle)
        route = append(route, pageTitle)
    }
}

```

```

        current = current.Previous
    }

    for i, j := 0, len(route)-1; i < j; i, j = i+1, j-1 {
        route[i], route[j] = route[j], route[i]
    }

    return route
}

```

- Fungsi scrape

Berfungsi untuk melakukan scraping pada halaman saat ini untuk mendapatkan daftar tautan yang ada di dalamnya.

```

func scrape(page *Page) {
    // Buat collector dengan domain khusus en.wikipedia.org
    c := colly.NewCollector(
        colly.AllowedDomains("en.wikipedia.org"),
    )

    c.OnHTML("title", func(e *colly.HTMLElement) {
        // Mengambil judul page
        page.Name = e.Text
        fmt.Println("Title:", page.Name)
    })

    c.OnHTML("a[href]", func(e *colly.HTMLElement) {
        // Mengambil URL page
        href := e.Attr("href")
        if strings.HasPrefix(href, "/wiki/") && !strings.Contains(href, "Main_Page")
        && !strings.Contains(href, "Wikipedia:") && !strings.Contains(href, "Portal:") &&
        !strings.Contains(href, "Special:") && !strings.Contains(href, "Help:") &&
        !strings.Contains(href, "Talk:") && !strings.Contains(href, "Category:") &&
        !strings.Contains(href, "File:") && !strings.Contains(href, "Template:") &&
        !strings.Contains(href, "Template_talk") {
            url := "https://en.wikipedia.org" + href
            page.Links = append(page.Links, &url)
            fmt.Println("Link:", url)
        }
    })

    // Pergi ke URL page-nya biar bisa di-scrape
    fmt.Println("Visiting:", page.URL)
    c.Visit(page.URL)
}

```

## 4.2 Tata Cara Penggunaan Program

Berikut adalah tata cara penggunaan program WikiRace beserta fitur-fitur yang disediakan:

### Tata Cara Penggunaan

- Memasukkan Informasi:
  - Pengguna membuka halaman web program WikiRace melalui browser.
  - Pengguna memilih jenis algoritma (BFS atau IDS) yang akan digunakan untuk pencarian rute.
  - Pengguna memasukkan judul artikel awal dan judul artikel tujuan.
- Memulai Pencarian:
  - Setelah memasukkan semua informasi, pengguna menekan tombol "Cari Rute" atau sejenisnya.
  - Program mengirimkan permintaan pencarian ke server (back-end) untuk memulai proses pencarian.
- Menampilkan Hasil:
  - Setelah proses pencarian selesai, program menampilkan hasil pencarian kepada pengguna.
  - Pengguna dapat melihat informasi seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian.

### Interface Program (Front-end)

- Terdiri dari halaman web yang dapat diakses melalui browser.
- Menampilkan formulir input untuk memasukkan jenis algoritma, judul artikel awal, dan judul artikel tujuan.
- Menampilkan hasil pencarian seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian.

### Server (Back-end)

- Berjalan sebagai server web menggunakan bahasa pemrograman Go.
- Menangani permintaan dari front-end dan melakukan proses pencarian rute.

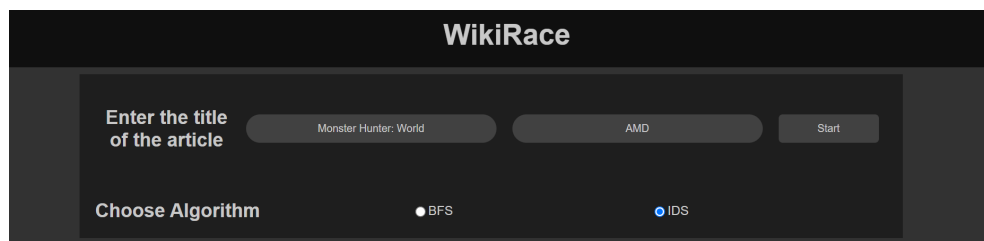
## Fitur-Fitur Program

- Pencarian Rute:
  - Pengguna dapat memilih jenis algoritma yang akan digunakan untuk mencari rute terpendek, yaitu BFS atau IDS.
  - Memasukkan judul artikel awal dan judul artikel tujuan yang ingin dicapai.
  - Setelah pengguna memasukkan semua informasi, program akan memulai pencarian rute.
- Tampilan Hasil:
  - Setelah pencarian selesai, program akan menampilkan hasil pencarian kepada pengguna.
  - Hasil mencakup informasi seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian.
  - Rute penjelajahan artikel ditampilkan dalam bentuk urutan artikel yang dilalui dari artikel awal hingga artikel tujuan.

## 4.3 Hasil Pengujian

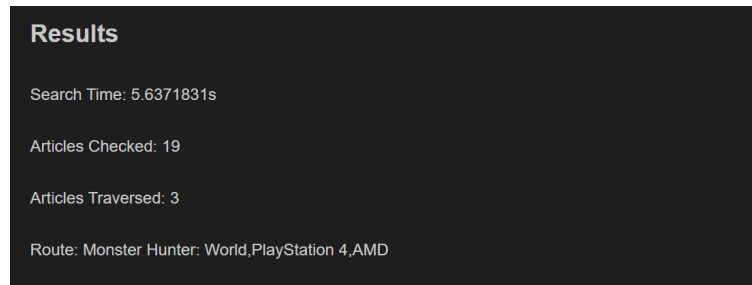
### 4.3.1 IDS

Input



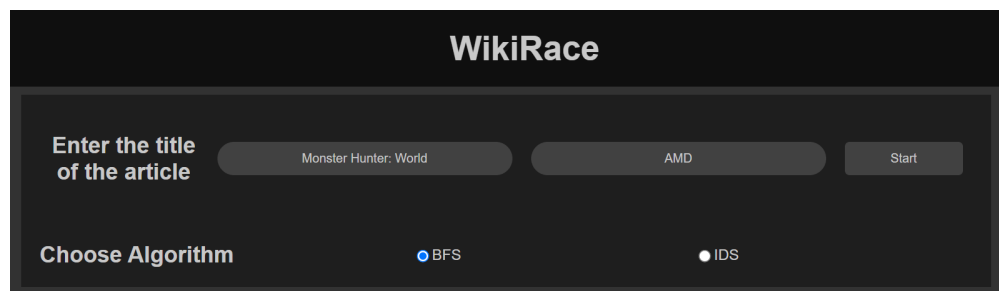
The screenshot shows the 'WikiRace' application interface. At the top, the title 'WikiRace' is displayed. Below it, there is a section for input. On the left, it says 'Enter the title of the article'. To the right of this text are two input fields: the first contains 'Monster Hunter: World' and the second contains 'AMD'. To the right of these fields is a 'Start' button. Below the input fields, there is a section labeled 'Choose Algorithm'. Under this label, there are two radio buttons: 'BFS' and 'IDS'. The 'IDS' radio button is selected, indicated by a blue dot.

Output

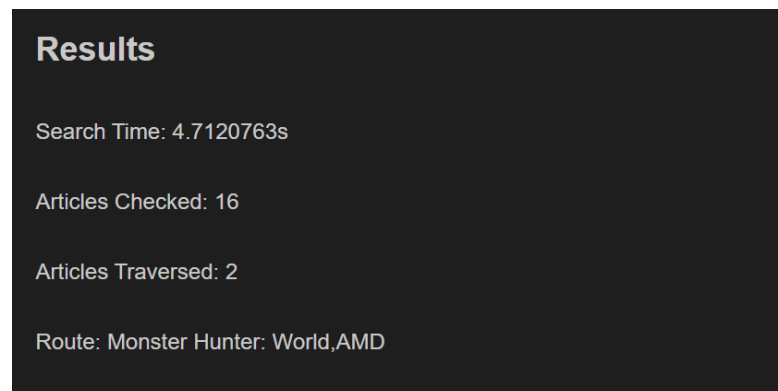


#### 4.3.2 BFS

Input



Output



### 4.4 Analisis

#### 4.4.1 IDS

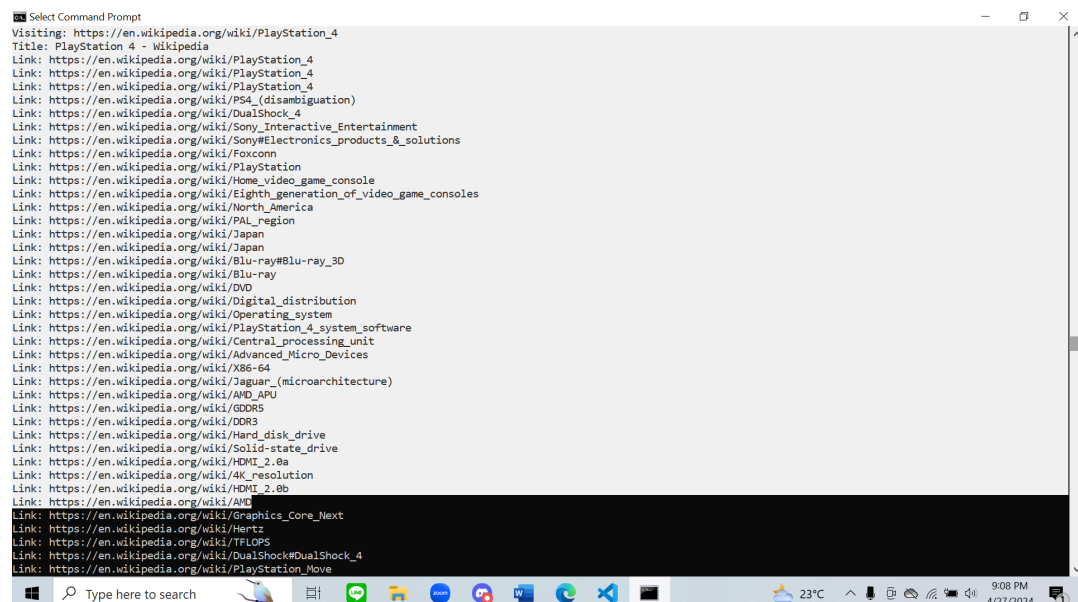
Pada program, terdapat input start article berupa “Monster Hunter: World” dan target article berupa “AMD”. Dengan menggunakan algoritma IDS, didapat bahwa waktu pencariannya adalah selama 5.6371831 detik, banyak artikel yang dicek adalah 19, banyak artikel yang dilalui adalah 3, dan rute yang ditempuh yang mulai dari Monster Hunter: World dan kemudian ke PlayStation 4 dan yang terakhir ke AMD. Hasil seperti ini berhasil didapat karena rute terdekat dari

halaman “Monster Hunter: World” ke halaman “AMD” melalui halaman “PlayStation 4”. Rute ini adalah rute yang dijadikan sebagai output dengan halaman yang dilalui merupakan panjang dari rute tersebut.

#### 4.4.2 BFS

Pada program, terdapat input start article berupa “Monster Hunter: World” dan target article berupa “AMD”. Dengan menggunakan algoritma BFS, didapat bahwa waktu pencariannya adalah selama 4.7 detik, banyak artikel yang dicek adalah 16, banyak artikel yang dilalui adalah 2, dan rute yang ditempuh yang mulai dari “Monster Hunter: World” dan kemudian ke “AMD”. Hasil seperti ini didapat karena kesalahan dari bagian algoritma yang digunakan untuk membuat rute. Kesalahan tersebut berupa ketidaklengkapan rute yang dibuat sehingga hanya menghasilkan start article dan target article. Hal tersebut dibuktikan oleh terminal yang menunjukkan aktivitas scraping pada halaman “PlayStation 4” yang mengandung URL menuju halaman “AMD”. Meskipun sudah melalui halaman “PlayStation 4” dari proses scraping untuk mengakses target article, program tidak mampu mengeluarkan output berupa rute serta panjang rute dengan benar dimana pada kasus ini, halaman “PlayStation 4” tidak berhasil dimasukkan sebagai rute.

Berikut adalah bagian hasil scraping dari analisis kasus.



```
Select Command Prompt
Visiting: https://en.wikipedia.org/wiki/PlayStation_4
Title: PlayStation 4 - Wikipedia
Link: https://en.wikipedia.org/wiki/PlayStation_4
Link: https://en.wikipedia.org/wiki/PlayStation_4
Link: https://en.wikipedia.org/wiki/PlayStation_4
Link: https://en.wikipedia.org/wiki/PS4_(disambiguation)
Link: https://en.wikipedia.org/wiki/DualShock_4
Link: https://en.wikipedia.org/wiki/Sony_Interactive_Entertainment
Link: https://en.wikipedia.org/wiki/Sony#Electronics_products_&_solutions
Link: https://en.wikipedia.org/wiki/Foxconn
Link: https://en.wikipedia.org/wiki/PlayStation
Link: https://en.wikipedia.org/wiki/Home_video_game_console
Link: https://en.wikipedia.org/wiki/Eighth_generation_of_video_game_consoles
Link: https://en.wikipedia.org/wiki/North_America
Link: https://en.wikipedia.org/wiki/PAL_region
Link: https://en.wikipedia.org/wiki/Japan
Link: https://en.wikipedia.org/wiki/Japan
Link: https://en.wikipedia.org/wiki/Blu-ray#Blu-ray_3D
Link: https://en.wikipedia.org/wiki/Blu-ray
Link: https://en.wikipedia.org/wiki/DVD
Link: https://en.wikipedia.org/wiki/Digital_distribution
Link: https://en.wikipedia.org/wiki/Operating_system
Link: https://en.wikipedia.org/wiki/PlayStation_4_system_software
Link: https://en.wikipedia.org/wiki/Central_processing_unit
Link: https://en.wikipedia.org/wiki/Advanced_Micro_Devices
Link: https://en.wikipedia.org/wiki/X86-64
Link: https://en.wikipedia.org/wiki/Jaguar_(microarchitecture)
Link: https://en.wikipedia.org/wiki/AMD_APU
Link: https://en.wikipedia.org/wiki/GDDR5
Link: https://en.wikipedia.org/wiki/DDR3
Link: https://en.wikipedia.org/wiki/Hard_disk_drive
Link: https://en.wikipedia.org/wiki/Solid-state_drive
Link: https://en.wikipedia.org/wiki/HDMI_2.0a
Link: https://en.wikipedia.org/wiki/4K_resolution
Link: https://en.wikipedia.org/wiki/HDMI_2.0b
Link: https://en.wikipedia.org/wiki/AMD
Link: https://en.wikipedia.org/wiki/Graphics_Core_Next
Link: https://en.wikipedia.org/wiki/Hertz
Link: https://en.wikipedia.org/wiki/TFLOPS
Link: https://en.wikipedia.org/wiki/DualShockDualShock_4
Link: https://en.wikipedia.org/wiki/PlayStation_Move
```



## **BAB 5**

### **PENUTUP**

#### **5.1 Kesimpulan**

Tugas Besar 2 telah memberikan kesempatan untuk mengimplementasikan algoritma BFS dan IDS dalam bahasa pemrograman Go untuk menyelesaikan permainan WikiRace. Melalui tugas ini, saya mempelajari banyak hal tentang pemrograman web, pengelolaan permintaan antara front-end dan back-end, serta implementasi algoritma pencarian yang efisien.

#### **5.2 Saran**

Pada kesempatan mendatang, akan bermanfaat untuk memperdalam pemahaman tentang konsep-konsep dasar dalam pengembangan web dan juga cara menghubungkan back-end dan front-end. Melakukan lebih banyak uji coba dan pengujian untuk memastikan kestabilan dan kehandalan program, terutama dalam menangani skenario yang kompleks atau input yang tidak valid.

#### **5.3 Refleksi**

Kami mengalami kesulitan dalam implementasi algoritma BFS dan menghubungkan front-end dan back-end, terutama dalam pengelolaan permintaan antara keduanya serta menyinkronkan proses-proses yang terjadi di masing-masing sisi. Meskipun kami memiliki pemahaman tentang kedua komponen tersebut secara terpisah, namun mengintegrasikannya menjadi sebuah sistem yang berfungsi dengan baik ternyata lebih rumit dari yang kami perkirakan.

Kesulitan ini terutama terjadi dalam memahami konsep-konsep seperti pengiriman permintaan HTTP, penanganan respons, komunikasi asinkron antara client dan server, serta pengaplikasian algoritma-algoritma pencarian. Meskipun kami telah berusaha keras untuk mengatasi masalah tersebut, namun hasilnya tidak sesuai dengan harapan.

Meskipun menghadapi tantangan tersebut, kami melihatnya sebagai pembelajaran berharga. Pengalaman ini mengajarkan pentingnya kesabaran, ketekunan, dan



kemampuan untuk beradaptasi dengan tantangan dalam pengembangan perangkat lunak. Kami menyadari bahwa kegagalan adalah bagian alami dari proses pembelajaran, dan kami percaya bahwa dari kesalahan-kesalahan tersebutlah kami dapat belajar dan berkembang menjadi seorang pengembang yang lebih baik.

Bagi kami, kegagalan ini bukanlah akhir dari segalanya, melainkan awal dari perbaikan dan peningkatan kemampuan kami di masa depan. Kami berencana untuk terus belajar dan meningkatkan pemahaman tentang pengembangan web. Dengan semangat yang kuat dan sikap yang positif, kami yakin bahwa kami dapat mengatasi tantangan ini dan mencapai kesuksesan dalam pengembangan program-program yang lebih kompleks di masa mendatang.

## LAMPIRAN

**Link Repository:** [https://github.com/chelvadinda/Tubes2\\_semogaKelar](https://github.com/chelvadinda/Tubes2_semogaKelar)

**Link Video:** <https://youtu.be/BRhL3i2Z4lw?si=LS4uSnqTjsQbji2>

## DAFTAR PUSTAKA

*YouTube: Home*, 9 November 2017, <https://youtu.be/ReV4617j9gk?si=ZOwyAe0bLjHNjnI8>.

Accessed 18 April 2024.

*W3Schools Online Web Tutorials*, <https://www.w3schools.com/>. Accessed 23 April 2024.

Munir, Rinaldi. "Breadth/Depth First Search (BFS/DFS)." *Homepage Rinaldi Munir*,

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/BFS-DFS-2021-Bag1-2024.pdf>. Accessed 3 April 2024.

Munir, Rinaldi. "Breadth/Depth First Search (BFS/DFS)." *Homepage Rinaldi Munir*,

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf>. Accessed 3 April 2024.