

**LAPORAN TUGAS BESAR II**  
**IF2211 Strategi Algoritma**  
**Pemanfaatan Algoritma IDS dan BFS dalam Permainan WikiRace**



Tazkirah Amaliah	10023608
Mohammad Nugraha Eka Prawira	13522001
Muhamad Rifki Virziadeili Harisman	13522120

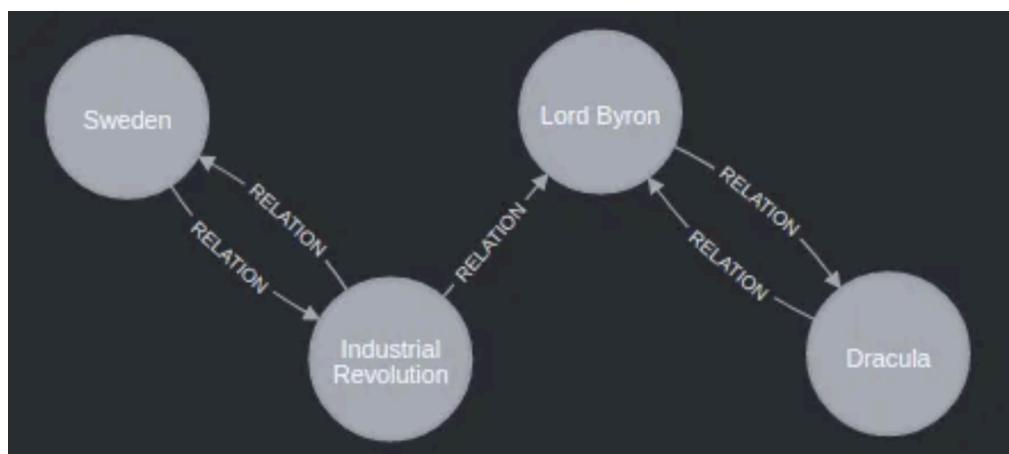
**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2024**

# Daftar Isi

<b>Daftar Isi.....</b>	<b>1</b>
<b>Bab 1: Deskripsi Tugas.....</b>	<b>2</b>
<b>Bab 2: Landasan Teori.....</b>	<b>3</b>
2.1. Penjelajahan Graf.....	3
2.2. Iterative Deepening Search (IDS).....	3
2.3. Breadth First Search (BFS).....	3
2.4. Cara Kerja Program.....	4
<b>Bab 3: Analisis Pemecahan Masalah.....</b>	<b>6</b>
3.1. Langkah-langkah pemecahan masalah.....	6
3.2. Proses pemetaan masalah menjadi elemen algoritma IDS dan BFS.....	7
3.3. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun.....	7
3.4. Ilustrasi Kasus.....	8
<b>Bab 4: Implementasi dan pengujian.....</b>	<b>10</b>
4.1. Spesifikasi teknis program.....	10
4.2. Tata cara penggunaan program.....	17
4.3. Hasil pengujian.....	21
4.3. Analisis hasil pengujian.....	23
<b>Bab 5: Kesimpulan dan saran.....</b>	<b>24</b>
5.1. Kesimpulan.....	24
5.2. Saran.....	24
<b>Lampiran.....</b>	<b>25</b>
Tautan Repository GitHub.....	25
Tautan Video.....	25
<b>Daftar Pustaka.....</b>	<b>26</b>

## 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

Pada tugas besar ini, Anda akan diminta untuk membuat website WikiRace atau Wiki Game yang dapat mengimplementasikan algoritma IDS dan BFS dalam penyelesaian permainan WikiRace. Program ini akan menerima masukan berupa jenis algoritma (IDS/BFS), judul artikel awal, dan judul artikel tujuan. Kemudian memberikan keluaran berupa jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel (dari artikel awal hingga artikel tujuan), dan waktu pencarian dalam ms. Serta program ini cukup mengeluarkan salah satu rute terpendek.

## Bab 2: Landasan Teori

### 2.1. Penjelajahan Graf

Penjelajahan graf merupakan proses untuk mengunjungi atau memeriksa setiap simpul dan tepi yang terhubung dalam suatu graf. Terdapat beberapa metode utama yang digunakan dalam penjelajahan graf, diantaranya adalah Breadth First Search (BFS) dan Iterative Deepening Search (IDS). Kedua metode ini memiliki kegunaan dan pengaplikasian yang berbeda tergantung pada situasi dan tujuan dari penjelajahan graf yang ingin dilakukan. Baik BFS maupun IDS memiliki kompleksitas waktu yang berbeda, yaitu  $O(V + E)$ , di mana V adalah jumlah simpul (vertices) dan E adalah jumlah tepi (edges) dalam graf, sedangkan IDS adalah  $O(b^d)$ , di mana b adalah faktor cabang rata-rata dan d adalah kedalaman solusi terdalam.

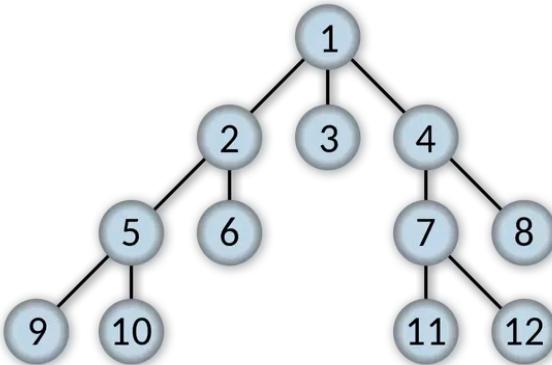
### 2.2. Iterative Deepening Search (IDS)

IDS (Iterative Deepening Search) merupakan gabungan dari strategi Depth First Search (DFS) dengan strategi Breadth First Search (BFS). IDS mulai dengan melakukan DFS pada graf dengan kedalaman terbatas, biasanya dimulai dengan kedalaman 0. Jika solusi tidak ditemukan pada kedalaman tersebut, maka algoritma akan mengingatkan kedalaman dan melakukan DFS kembali. Kemudian proses ini akan terus berlanjut hingga solusi ditemukan, lalu memastikan bahwa kedalaman pencarian ditingkatkan secara bertahap sehingga keseluruhan kompleksitas waktu adalah  $O(b^d)$ , di mana b adalah faktor cabang rata-rata dan d adalah kedalaman solusi terdalam.

IDS memiliki kompleksitas waktu yang sama dengan DFS karena hanya menyimpan informasi tentang jalur pencarian saat ini, tidak seperti BFS yang harus menyimpan semua simpul pada setiap level. IDS menggunakan memori lebih efisien karena hanya perlu menyimpan informasi tentang jalur pencarian saat ini, sehingga cocok untuk pencarian pada graf yang sangat besar. IDS menjamin pencarian solusi optimal ketika digunakan dalam pencarian graf yang memungkinkan untuk mencapai solusi. Hal ini karena IDS secara bertahap meningkatkan kedalaman pencarian hingga menemukan solusi optimal pada kedalaman yang paling dangkal.

### 2.3. Breadth First Search (BFS)

Breadth First Search (BFS) adalah algoritma penjelajahan graf yang mengunjungi simpul secara melebar, mulai dari simpul awal, kemudian menelusuri semua simpul tetangga sebelum melanjutkan ke simpul-simpul yang lebih jauh. Algoritma ini menggunakan antrian (queue) untuk menyimpan simpul yang akan dikunjungi selanjutnya. BFS berguna dalam mencari jalur terpendek antara dua simpul dalam graf, karena secara sistematis menjelajahi graf dalam urutan yang lebih terstruktur. Kompleksitas waktu BFS adalah  $O(V + E)$ , di mana V adalah jumlah simpul dan E adalah jumlah tepi dalam graf.



*Gambar 2. Contoh BFS pada pohon  
(sumber: [https://en.wikipedia.org/wiki/Breadth-first\\_search](https://en.wikipedia.org/wiki/Breadth-first_search))*

BFS cenderung membutuhkan lebih banyak memori karena perlu menyimpan semua simpul pada setiap level dalam antrian. Pada BFS, setiap simpul dan setiap tepi akan dikunjungi tepat satu kali. Oleh karena itu, kompleksitas waktu BFS tergantung pada jumlah total simpul dan tepi dalam graf.

## 2.4. Cara Kerja Program

Aplikasi web WikiRace atau Wiki Game yang dibangun adalah sebuah platform interaktif yang memungkinkan pengguna untuk bermain permainan WikiRace menggunakan algoritma Iterative Deepening Search (IDS) atau Breadth First Search (BFS). Pengguna akan diminta untuk memasukkan atau memilih jenis algoritma yang ingin digunakan (IDS atau BFS), judul artikel awal, dan judul artikel tujuan.

Setelah pengguna memberikan masukan, aplikasi akan memproses informasi tersebut dengan menggunakan algoritma yang dipilih untuk menemukan jalur terpendek antara artikel awal dan artikel tujuan di Wikipedia. Selama pencarian, aplikasi akan melacak jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel dari artikel awal hingga artikel tujuan, serta waktu pencarian artikel.

Setelah menemukan jalur terpendek, aplikasi akan menampilkan rincian pencarian, termasuk jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian dalam milisekom (ms). Pengguna juga akan melihat salah satu rute terpendek yang ditemukan oleh algoritma tersebut. Kemudian setelah dilakukan pencarian, maka menampilkan juga informasi-informasi terkait artikel awal dan artikel tujuan.

Secara keseluruhan, program ini menyediakan pengalaman bermain WikiRace yang interaktif dan informatif, sambil memberikan wawasan tentang bagaimana algoritma IDS dan BFS digunakan dalam penyelesaian permainan WikiRace.

## Bab 3: Analisis Pemecahan Masalah

### 3.1. Langkah-langkah pemecahan masalah

Dalam program WikiRace yang menggunakan pendekatan IDS dan BFS, proses pemetaan pencarian jalur artikel Wikipedia dapat melibatkan identifikasi langkah-langkah yang

dapat dimodelkan menggunakan kedua algoritma tersebut. Dalam hal ini, IDS dan BFS menjadi pendekataan yang berguna untuk memetakan persoalan ini ke dalam serangkaian langkah-langkah yang berfokus pada menemukan jalur terpendek antara artikel awal dan artikel tujuan dengan memperhatikan jumlah artikel yang diperiksa dan dilalui. Dengan mengimplementasikan kedua algoritma tersebut, aplikasi dapat mencapai tujuan pencarian dengan efisien dan memastikan ketersediaan rute terpendek bagi pengguna.

Berikut langkah-langkah pemecahan masalah terkait deskripsi program WikiRace yang telah diberikan:

1. Memproses masukan pengguna: program harus menerima inputan dari pengguna berupa jenis algoritma yang ingin digunakan (IDS atau BFS), judul artikel awal, dan judul artikel tujuan.
2. Mengambil data dari wikipedia: setelah menerima masukan pengguna, aplikasi perlu mengambil data artikel dari Wikipedia.
3. Implementasi algoritma pencarian: berdasarkan pilihan antara IDS atau BFS, aplikasi harus menerapkan algoritma pencarian yang sesuai untuk menemukan jalur terpendek antara artikel awal dan artikel tujuan. Jika memilih IDS, program mengimplementasikan algoritma dengan meningkatkan kedalaman pencarian secara secara bertahap hingga solusi ditemukan. Jika memilih BFS, program mengimplementasikan algoritma BFS untuk melakukan penjelajahan graf dalam pencarian jalur terpendek.
4. Melacak statistik pencarian: selama proses pencarian, program harus melacak statistik seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, dan rute penjelajahan artikel dari artikel awal hingga artikel tujuan.
5. Menampilkan hasil pencarian: setelah menemukan jalur terpendek, aplikasi harus menampilkan rincian pencarian kepada pengguna, termasuk statistik pencarian seperti jumlah artikel yang diperiksa, jumlah artikel yang dilalui, rute penjelajahan artikel, dan waktu pencarian dalam milisekon (ms).

### 3.2. Proses pemetaan masalah menjadi elemen algoritma IDS dan BFS

Proses pemetaan masalah menjadi elemen algoritma IDS (Iterative Deepening Search) dan BFS (Breadth First Search) berdasarkan langkah pemecahan masalah program WikiRace dilakukan sebagai berikut:

1. Identifikasi persoalan: mencari jalur terpendek antara dua artikel Wikipedia yang ditentukan, dimulai dari artikel awal dan berakhir di artikel tujuan.
2. Modeling persoalan: merupakan graf di mana setiap artikel Wikipedia merupakan simpul, dan tautan antara artikel merupakan tepi atau edge. Persoalan dapat dimodelkan sebagai pencarian jalur dari simpul artikel awal ke simpul artikel tujuan.
3. Pemilihan algoritma: berdasarkan deskripsi program, IDS dan BFS dipilih sebagai algoritma pencarian yang cocok karena keduanya dapat digunakan untuk mencari jalur terpendek dalam graf.
4. Implementasi IDS: memerlukan iterasi pencarian dengan meningkatkan kedalaman pencarian secara bertahap hingga solusi ditemukan. Implementasi IDS dalam program akan melibatkan iterasi yang berulang dengan meningkatkan kedalaman pencarian pada setiap iterasi hingga artikel tujuan ditemukan.
5. Implementasi BFS: melakukan penjelajahan secara melebar, yang berarti akan menelusuri semua simpul tetangga pada setiap level sebelum melanjutkan ke level berikutnya. Implementasi BFS dalam program akan melibatkan penelusuran graf secara melebar dari artikel awal ke artikel tujuan.
6. Pemetaan langkah-langkah algoritma: mengatur antrian atau tumpukan pencarian, melakukan penelusuran graf, dan memperbaharui status pencarian.
7. Pencarian jalur terpendek: program akan mencari jalur terpendek dari artikel awal ke artikel tujuan. Setiap langkah dalam pencarian akan mempertimbangkan jumlah artikel yang diperiksa, dilalui, serta rute penjelajahan artikel dari awal hingga tujuan.

Dengan demikian, proses pemetaan masalah menjadi elemen algoritma IDS dan BFS memungkinkan program WikiRace untuk mencari jalur terpendek antara dua artikel Wikipedia dengan efisien dan efektif sesuai dengan deskripsi program yang diberikan.

### 3.3. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

#### A. Fitur Fungsional:

1. Pilihan algoritma pencarian: program dapat memilih jenis algoritma yang ingin digunakan untuk pencarian jalur, yaitu IDS atau BFS.
2. Input artikel awal dan tujuan: program dapat memasukkan judul artikel Wikipedia awal dan tujuan untuk memulai permainan.
3. Pencarian jalur terpendek: program akan mencari jalur terpendek antara artikel awal dan tujuan menggunakan algoritma yang dipilih.
4. Statistik pencarian: menampilkan statistik pencarian, seperti jumlah artikel yang diperiksa, dilalui, rute penjelajahan artikel, dan waktu pencarian.
5. Tampilan rute terpendek: dapat melihat salah satu rute terpendek yang ditemukan oleh algoritma.

#### B. Arsitektur Aplikasi Web:

1. Frontend (User Interface):

- Menggunakan teknologi web seperti HTML, CSS, dan JavaScript untuk membuat antarmuka pengguna interaktif.
- Menampilkan formulir input untuk jenis algoritma, judul artikel awal, dan judul artikel tujuan.
- Menampilkan hasil pencarian dan statistik dalam tampilan yang mudah dimengerti oleh pengguna.

2. Backend (Logika Aplikasi):

- Menggunakan bahasa pemrograman seperti GoLang untuk logika aplikasi.
- Memproses input dari pengguna dan menerapkan algoritma IDS dan BFS untuk mencari jalur terpendek.
- Mengambil data artikel dari Wikipedia menggunakan API Wikipedia atau dengan melakukan scraping.
- Menghitung statistik pencarian seperti jumlah artikel yang diperiksa, dilalui, dan waktu pencarian.
- Menyediakan API endpoint untuk komunikasi antara frontend dan backend.

3. Database:

- Menggunakan database untuk menyimpan data pengguna, riwayat pencarian, atau cache hasil pencarian.

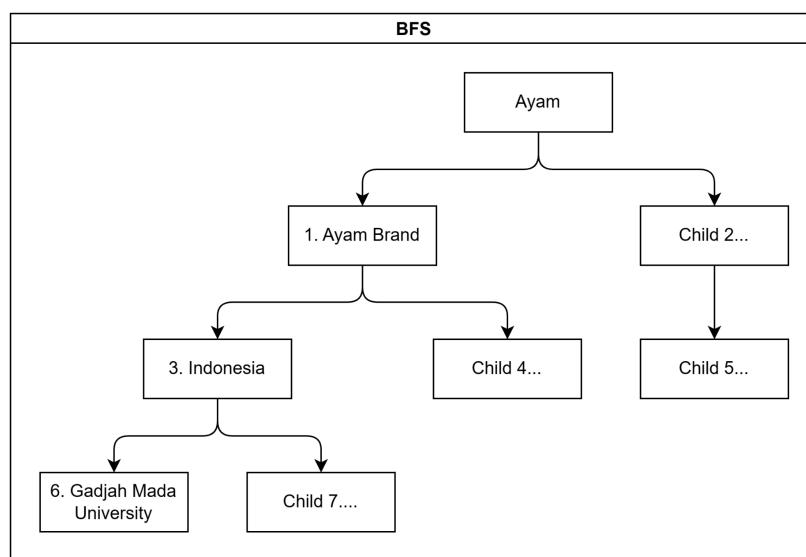
4. Integrasi dengan API Wikipedia:

- Menggunakan API Wikipedia untuk mengambil informasi artikel dan tautan antar artikel.
- Melakukan permintaan HTTP untuk mengakses data Wikipedia.

### 3.4. Ilustrasi Kasus

Ilustrasi pencarian dengan Artikel awal berjudul “Ayam” dengan artikel tujuan “Gadjah Mada University”

#### 3.4.1 Metode Pencarian BFS

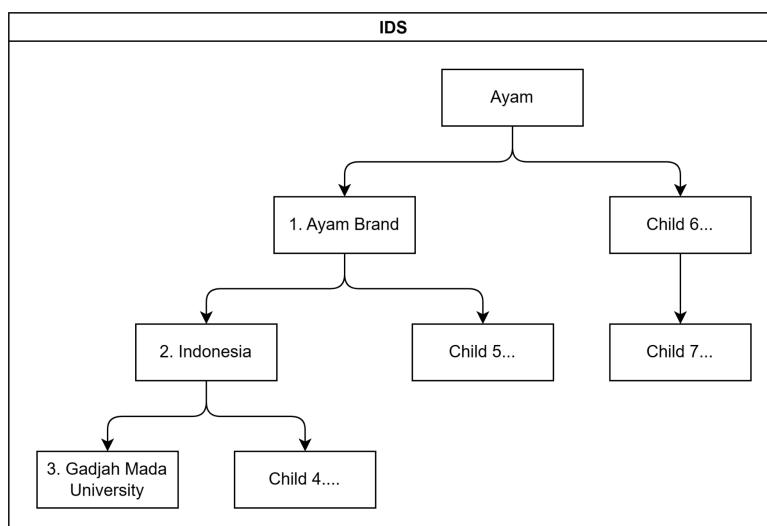


Pencarian akan dimulai dengan pengguna memasukan input artikel awal dan tujuan serta memilih algoritma yang diinginkan yaitu BFS. Setelah pengguna meng click “GO!”, maka pencarian akan dimulai dan pengguna akan diminta menunggu hingga hasilnya keluar. Oleh karena ditemukannya jalur menuju target artikel, maka program akan mengeluarkan hasil berupa jalur dari artikel awal menuju artikel tujuan, jumlah artikel yang di check, panjang rute dari artikel awal menuju artikel akhir serta waktu yang dibutuhkan untuk menemukan jalur tersebut.

Pada setiap page program akan menambahkan link wiki yang ada pada page tersebut ke dalam sebuah antrian dan akan memprosesnya sesuai antrian tersebut hingga ditemukan artikel tujuannya. Rute pencarian BFS sebagai berikut :

Ayam → Ayam Brand→ Child 2→ Indonesia→ Child 4→ Child 5→ Gadjah Mada University→ Child 7

### 3.4.2 Metode Pencarian IDS



Pencarian akan dimulai dengan pengguna memasukan input artikel awal dan tujuan serta memilih algoritma yang diinginkan yaitu IDS. Setelah pengguna meng click “GO!”, maka pencarian akan dimulai dan pengguna akan diminta menunggu hingga hasilnya keluar. Oleh karena ditemukannya jalur menuju target artikel, maka program akan mengeluarkan hasil berupa jalur dari artikel awal menuju artikel tujuan, jumlah artikel yang di check, panjang rute dari artikel awal menuju artikel akhir serta waktu yang dibutuhkan untuk menemukan jalur tersebut.

Pada setiap page program program akan bergerak sesuai kedalaman yang ditentukan lalu bergerak balik pada page sebelumnya untuk mencari hingga kedalaman yang tersebut dan berulang terus hingga ditemukan target artikel. Rute pencarian IDS sebagai berikut :

Ayam → Ayam Brand→ Indonesia→ Gadjah Mada University→ Child 4 → Child 5 → Child 6→Child 7

## Bab 4: Implementasi dan pengujian

### 4.1. Spesifikasi teknis program

#### A. Struktur data

1. Queue: slice yang digunakan untuk menyimpan URL artikel yang akan diperiksa selama proses BFS.

```
Var queue []string
```

2. Exist: map yang digunakan untuk memeriksa apakah suatu URL artikel sudah ada di dalam antrian atau tidak.

```
Exist := make(map[string]bool)
```

3. Visited: map yang digunakan untuk menyimpan URL artikel yang sudah diperiksa agar tidak diperiksa lagi.

```
Visited := make(map[string]bool)
```

4. Paths: map yang digunakan untuk menyimpan jalur terpendek antara artikel awal dan artikel tujuan.

```
Paths := make(map[string]string)
```

5. ch: channel yang digunakan untuk mengirim sinyal bahwa jalur terpendek telah ditemukan.

```
Ch := make(chan bool, 10000)
```

6. wg: WaitGroup yang digunakan untuk mengatur berjalannya goroutine secara koncurrent.

```
Var wg sync.WaitGroup
```

#### B. Fungsi dan Prosedur

1. Func scrapping: fungsi untuk melakukan web scraping dan mendapatkan tautan artikel terkait

```

func scrapping(currentURL string)([]string){
    queue := []string{}
    exist := make(map[string]bool)

    doc, err := goquery.NewDocument(currentURL)
    if err != nil {
        log.Fatal(err)
    }
    if isRedirect(doc) {
        return queue
    }
    doc.Find("div#bodyContent").Each(func(i int, s *goquery.Selection) {
        s.Find("a[href]").Each(func(i int, s *goquery.Selection) {
            link, _ := s.Attr("href")
            if strings.HasPrefix(link, "/wiki/") && !hasPrefix(unwantedWikiPrefixes[:],link) &&
            !strings.Contains(link,":") {
                fullURL := "https://en.wikipedia.org" + link

                if !exist[fullURL] {
                    queue = append(queue, fullURL)
                    exist[fullURL] = true
                }
            }
        })
    })
    return queue
}

```

2. Func BFS: fungsi utama untuk menjalankan algoritma BFS dalam pencarian jalur terpendek.

```

func BFS(startURL, targetURL string) ([]string, int,int,string) {
    startTime := time.Now()
    visited := make(map[string]bool)
    queue := make(chan string,100000000)
    paths := make(map[string]string)
    articleChecked :=0
    var mu sync.RWMutex
    var wg sync.WaitGroup
    ch := make(chan bool, max)
    go func() {
        queue <- startURL
    }()
    for currentURL := range queue {

```

```

articleChecked++
if time.Since(startTime) > (5 * time.Minute) {

    return nil,articleChecked,0,time.Since(startTime).String()
}
here:
mu.Lock()
if currentURL == targetURL {

    path := []string{targetURL}
    currentURL = targetURL
    for currentURL != startURL {
        currentURL = paths[currentURL]
        path = append([]string{currentURL}, path...)
    }

    return path,articleChecked, len(path),time.Since(startTime).String()
}
mu.Unlock()

if visited[currentURL] {
    continue
}

mu.Lock()
visited[currentURL] = true
mu.Unlock()
ch <- true
wg.Add(1)

copyURL := currentURL

go func(url string, targetURL string, paths *map[string]string) {
    defer wg.Done()
    defer func() { <-ch }()
    neighbors := scrapping(url)
    for _, neighbor := range neighbors {
        mu.Lock()
        if neighbor == targetURL {
            if _, exist := (*paths)[neighbor]; !exist {
                (*paths)[neighbor] = url
            }
            queue <- neighbor
        }
    }
}

```

```

        copyURL = neighbor
        mu.Unlock()
        return
    }
    mu.Unlock()

    if !visited[neighbor] {
        mu.Lock()
        if _, exist := (*paths)[neighbor]; !exist {
            (*paths)[neighbor] = url
        }
        mu.Unlock()
        queue <- neighbor
    }
}
}

}(copyURL, targetURL, &paths)
wg.Wait()
mu.Lock()
if copyURL == targetURL{
    currentURL = copyURL
    mu.Unlock()
    goto here
}
mu.Unlock()
}

wg.Wait()
close(queue)
return nil, 0,0,time.Since(startTime).String()
}

```

3. Func DFSConcurrentMT: fungsi rekursif untuk menjalankan algoritma DFS tanpa metode concurrent.

```

func DFSConcurrentMT(currentURL, targetURL string, depth int, visited map[string]bool, paths
map[string]string, ch chan bool, wg *sync.WaitGroup) {
    defer wg.Done()
    if currentURL == targetURL {
        mu.Lock()
        fmt.Println("panjang ch", len(ch))
        fmt.Println()
        fmt.Println()
        fmt.Println(currentURL)
    }
}

```

```

        if(len(ch)<=0){
            ch <- true
        }
        mu.Unlock()
        return
    }

    if depth == 0 {
        return
    }

    mu.Lock()
    visited[currentURL] = true
    mu.Unlock()

    resp, err := http.Get(currentURL)
    if err != nil {
        fmt.Printf("Error fetching URL %s: %v\n", currentURL, err)
        return
    }
    defer resp.Body.Close()

    if resp.StatusCode != 200 {
        fmt.Printf("Unexpected status code %d for URL %s\n", resp.StatusCode, currentURL)
        return
    }

    doc, err := goquery.NewDocumentFromReader(resp.Body)
    if err != nil {
        fmt.Printf("Error parsing document from URL %s: %v\n", currentURL, err)
        return
    }

    doc.Find("div#bodyContent").Each(func(i int, s *goquery.Selection) {
        s.Find("a").Each(func(i int, s *goquery.Selection) {
            link, _ := s.Attr("href")
            if link != "" && link[0] == '/' && len(link) > 1 && link[1] != '#' {
                fullURL := "https://en.wikipedia.org" + link
                mu.Lock()
                if !visited[fullURL] {
                    paths[fullURL] = currentURL
                    wg.Add(1)
                    go DFSConcurrentMT(fullURL, targetURL, depth-1, visited, paths, ch, wg)
                }
            }
        })
    })
}

```

```

        mu.Unlock()
    }
})
}
}

```

4. Func IDSCurrentMT: fungsi utama untuk menjalankan algoritma IDS tanpa metode concurrent.

```

func IDSCurrentMT(startURL, targetURL string) {
    depth := 1
    visited := make(map[string]bool)
    paths := make(map[string]string)
    found := false
    var wg sync.WaitGroup

    for !found {
        fmt.Println(depth)
        ch := make(chan bool, 10000)
        wg.Add(1)
        go DFSConcurrentMT(startURL, targetURL, depth, visited, paths, ch, &wg)
        if len(ch) > 0 {
            found = <-ch
            fmt.Println("woy anjing")
        }
        fmt.Println(found)
        if found {
            // Path found
            path := []string{targetURL}
            for targetURL != startURL {
                targetURL = paths[targetURL]
                path = append([]string{targetURL}, path...)
            }
            fmt.Println("Shortest path:")
            for _, p := range path {
                fmt.Println(p)
            }
            break
        }
        wg.Wait()
        fmt.Println("woy anjing")
        depth++
        if depth >= 100 {
    }
}

```

```

        fmt.Println("Tidak ditemukan")
        break
    }
}
}

```

5. Func FetchSuggestions: fungsi untuk mengambil saran pencarian dari Wikipedia berdasarkan input pengguna.

```

func FetchSuggestions(input string) ([]string, map[string]string, error) {
    url :=
        fmt.Sprintf("https://en.wikipedia.org/w/api.php?action=opensearch&limit=10&format=json&se
        arch=%s&origin=*", input)
    paths := make(map[string]string)
    resp, err := http.Get(url)
    if err != nil {
        return nil, nil, err
    }
    defer resp.Body.Close()

    var data []interface{}
    if err := json.NewDecoder(resp.Body).Decode(&data); err != nil {
        return nil, nil, err
    }

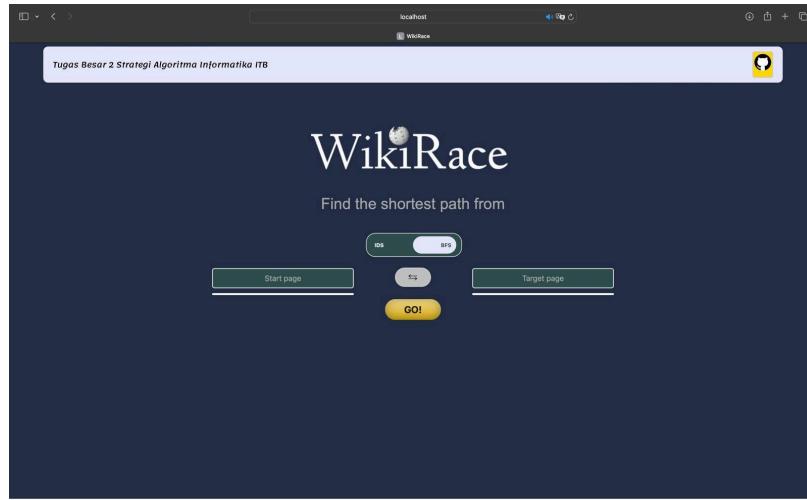
    suggestions := make([]string, 0)
    if len(data) > 1 {
        for _, suggestion := range data[1].([]interface{}) {
            suggestions = append(suggestions, suggestion.(string))
            formattedSuggestionArticle := strings.ReplaceAll(suggestion.(string), " ", "_")
            fullSuggestArticleURL := fmt.Sprintf("https://en.wikipedia.org/wiki/%s",
                formattedSuggestionArticle)
            paths[suggestion.(string)] = fullSuggestArticleURL
        }
    }

    return suggestions, paths, nil
}

```

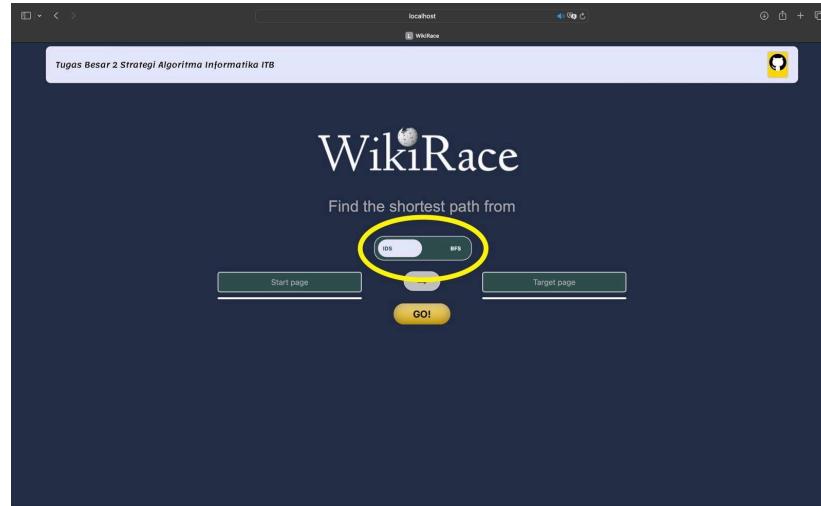
## 4.2.Tata cara penggunaan program

1. Jalankan program web Tubes 2 Stima yang ada pada <http://localhost:8080/> kemudian akan muncul tampilan awal “WikiRace” seperti ini.



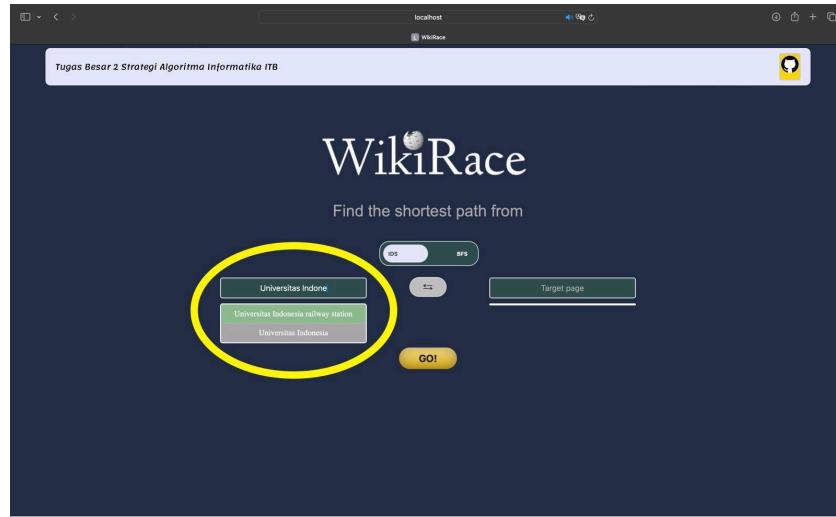
Gambar 3. Tampilan awal WikiRace

2. Memilih terlebih dahulu algoritma yang ingin digunakan pada pencarian antara algoritma Iterative Deepening Search (IDS) dan Breadth First Search (BFS).



Gambar 4. Tampilan awal WikiRace mode IDS

3. Menginput keyword dari artikel awal yang ingin dicari pada kolom search sebelah kiri.



Gambar 3. Tampilan awal WikiRace mode IDS

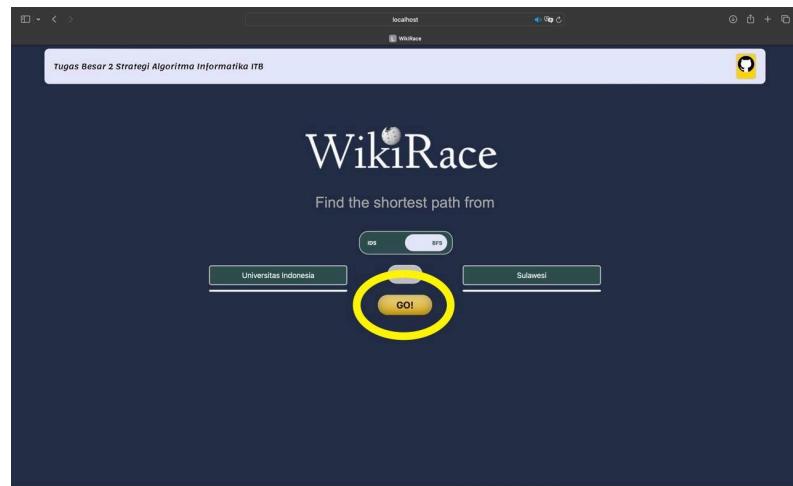
4. Menginput keyword dari artikel tujuan yang ingin dicari pada kolom search sebelah kanan.



5. Klik tanda switch untuk menukarkan keyword artikel awal dengan artikel tujuan.



6. Klik tombol “GO!” untuk memulai penelusuran berdasarkan artikel yang diinput.



7. Kemudian akan menampilkan informasi hasil dari penelusuran yaitu rute terpendek, jumlah artikel yang dicek, panjang rute, dan durasi pencarian.

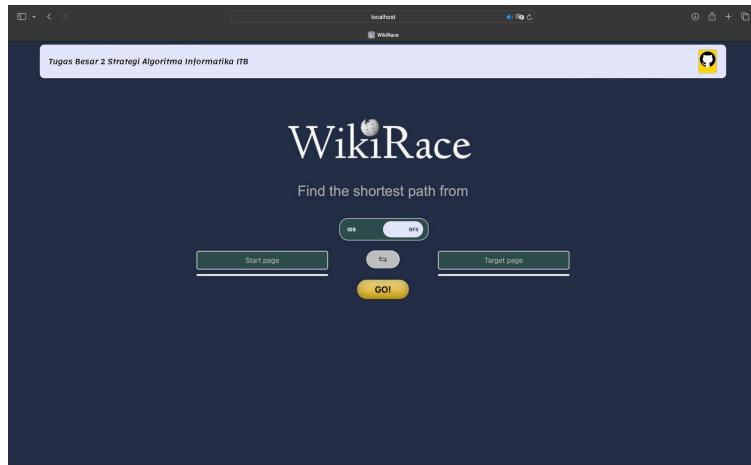
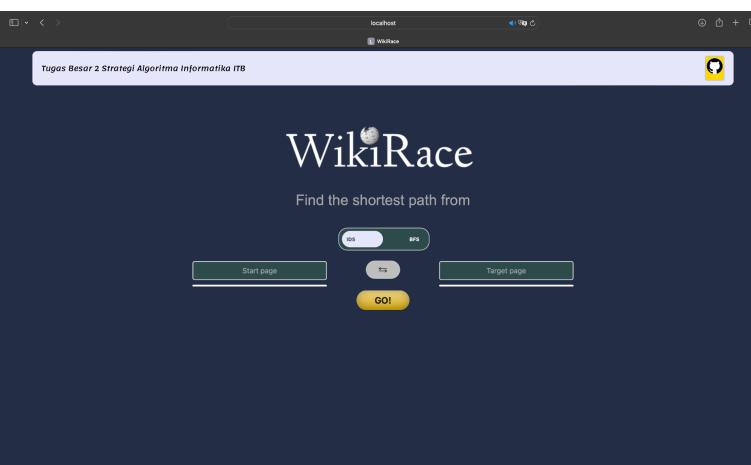


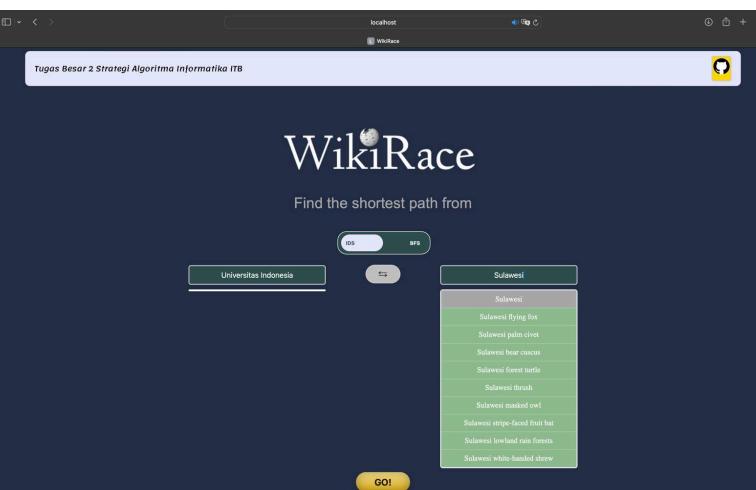
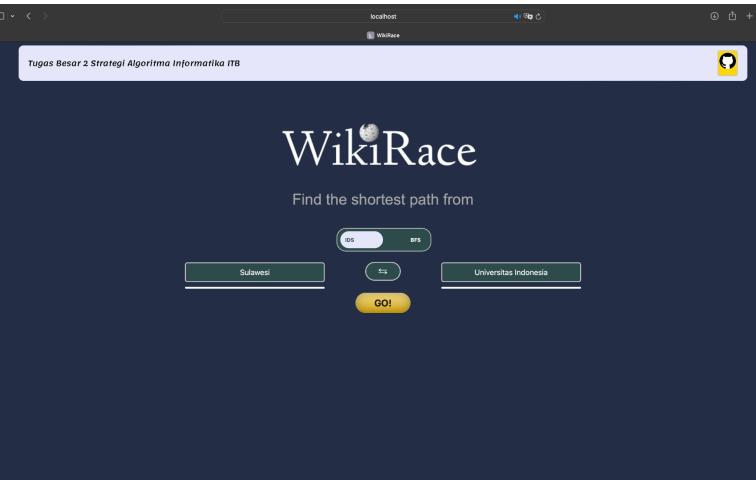
8. Untuk memulai kembali permainan, maka klik “Lagi yuk!” untuk diarahkan ke halaman awal.

Program WikiRace menyediakan sejumlah fitur yang memungkinkan pengguna untuk mencari jalur terpendek antara dua artikel Wikipedia dengan mudah dan efisien. Pengguna dapat memilih jenis algoritma pencarian yang ingin digunakan, yaitu BFS atau IDS, dan memasukkan judul artikel awal serta tujuan melalui antarmuka program. Setelah proses pencarian selesai, program akan menampilkan rute terpendek antara artikel awal dan tujuan dalam bentuk daftar judul artikel beserta dengan rincian langkah-langkah yang dilakukan algoritma dalam menemukan jalur terpendek. Program juga mencatat durasi waktu yang diperlukan untuk menemukan jalur terpendek, yang ditampilkan dalam satuan milidetik. Dengan fitur-fitur ini, pengguna dapat dengan mudah menggunakan program WikiRace untuk memperoleh informasi dengan proses pencarian yang baik.

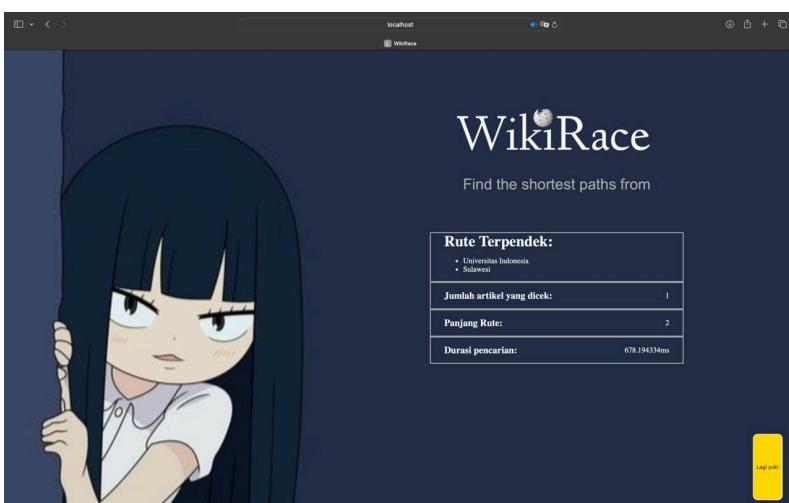
## 4.3. Hasil pengujian

### a. Percobaan 1

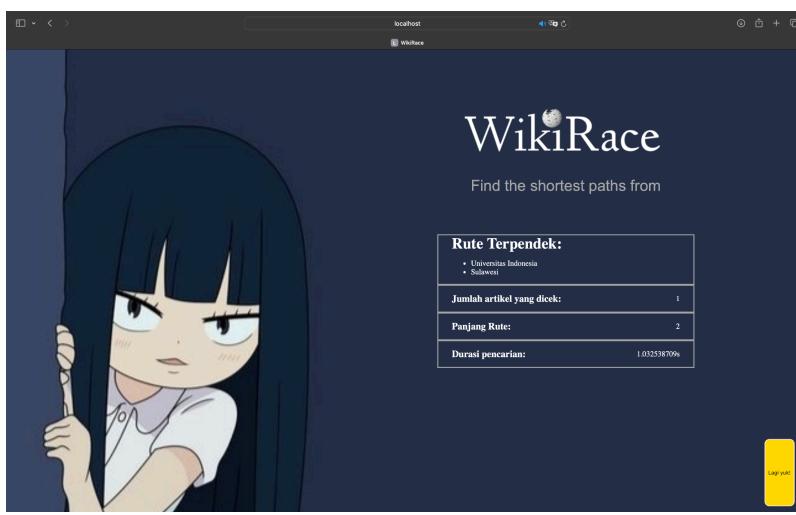
Interface Program	
Tampilan awal mode BFS	 A screenshot of a web browser window titled "Tugas Besar 2 Strategi Algoritma Informatika ITB". The main title is "WikiRace". Below it, the text "Find the shortest path from" is followed by two input fields: "Start page" and "Target page", each with a dropdown menu. Between them is a button labeled "GO!". Above the input fields is a radio button group with two options: "BFS" (selected) and "IDS".
Tampilan awal mode IDS	 A screenshot of a web browser window titled "Tugas Besar 2 Strategi Algoritma Informatika ITB". The main title is "WikiRace". Below it, the text "Find the shortest path from" is followed by two input fields: "Start page" and "Target page", each with a dropdown menu. Between them is a button labeled "GO!". Above the input fields is a radio button group with two options: "IDS" (selected) and "BFS".

Tampilan input artikel awal	
Tampilan input artikel tujuan	
Tampilan input ketika digunakan switch	

Tampilan hasil IDS program WikiRace



Tampilan hasil BFS program WikiRace



b. Percobaan 2

Interface Program

Algoritma  
BFS  
dengan  
hasil 3  
path

The screenshot shows the WikiRace application running on a Windows desktop. The title bar reads "Tugas Besar 2 Strategi Algoritma Informatika ITB". The main interface has a dark blue background with white text. At the top, it says "WikiRace" with a small globe icon. Below that, it asks "Find the shortest path from". There are two green rectangular boxes labeled "Ayam" and "Indonesia". Between them is a central area with a double-headed arrow icon and a yellow "GO!" button. Above the boxes are two buttons: "IDS" (disabled) and "BFS" (highlighted). The taskbar at the bottom shows various open applications like Spesifikasi, Tugas, and several browser tabs.

The screenshot shows the WikiRace application after the search has completed. The title bar now reads "localhost:8080/process". The main interface features a large anime-style illustration of a girl with dark hair and bangs looking thoughtful. To the right of the illustration is a summary table:

Rute Terpendek:	
• Ayam	
• Ayam Brand	
• Indonesia	

Below this table are three more rows of data:

Jumlah artikel yang dicek:	4
Panjang Rute:	3
Durasi pencarian:	8368ms

A yellow "Lagi yah!" button is located at the bottom right of the summary area. The taskbar at the bottom shows the same set of open applications as the previous screenshot.

Algoritma  
BFS  
dengan  
hasil 3  
path

The screenshot shows a dark-themed web application titled "WikiRace". At the top, there's a navigation bar with tabs like "Tugas", "Six De...", "Ayam", "Shelf", "Gajah", "Palap...", "Ayam", "ChatG", "Tubes", "Down", and a "+" button. Below the navigation is a header bar with "localhost:8080" and a yellow GitHub icon. The main content area has a title "Tugas Besar 2 Strategi Algoritma Informatika ITB" and a large "WikiRace" logo. It asks "Find the shortest path from" and provides two input fields: "Ayam" and "Indonesia", each with a green rounded rectangle below it. Between the fields is a central button with a double-headed arrow icon. To the right of the "Indonesia" field is a yellow "GO!" button. Above the input fields are two small buttons: "IDS" (highlighted) and "BFS". At the bottom of the page is a Windows taskbar with various pinned icons.

The screenshot shows the same dark-themed "WikiRace" application after a search. On the left is a large anime-style illustration of a girl with dark hair and bangs, looking slightly to the side with a neutral expression. On the right, the search results are displayed in a sidebar:

- Rute Terpendek:**
  - Ayam
  - Ayan Brand
  - Indonesia
- Jumlah artikel yang dicek:** 4
- Panjang Rute:** 3
- Durasi pencarian:** 8368ms

A yellow "Lagi yah!" button is located at the bottom right of the sidebar. The Windows taskbar at the bottom is visible.

c. Percobaan 3

Interface Program

Algoritma  
BFS  
dengan  
hasil 4  
path

The screenshot shows the WikiRace application running on a Windows desktop. The main window title is "Tugas Besar 2 Strategi Algoritma Informatika ITB". The interface has a dark blue background with white text. At the top, there are tabs for "IDS" and "BFS", with "BFS" currently selected. Below the tabs are two green rectangular boxes labeled "Ayam" and "Gajah Mada". Between them is a small button with a double-headed arrow symbol. A yellow "GO!" button is located below the arrows. The text "Find the shortest path from" is displayed above the boxes. The bottom of the window shows a progress bar with a yellow segment. The taskbar at the bottom of the screen displays various icons for Microsoft Office applications like Word, Excel, and PowerPoint, as well as other common desktop tools.

The screenshot shows the WikiRace application after the search has completed. The main window title is "localhost:8080/process". The interface features a large cartoon illustration of a girl with dark hair and bangs, looking slightly to the side with a neutral expression. To the right of the illustration, the text "WikiRace" is prominently displayed. Below the illustration, the text "Find the shortest paths from" is followed by a section titled "Rute Terpendek:" which contains a bulleted list of four items: "Ayam", "Ayan Brand", "Indonesia", and "Gajah Mada". Further down, there are three more sections with data: "Jumlah artikel yang dicek: 247", "Panjang Rute: 4", and "Durasi pencarian: 50107ms". A yellow "Lagi yuh!" button is located at the bottom right. The taskbar at the bottom of the screen is identical to the one in the previous screenshot.

Algoritma BFS dengan hasil 4 path

The screenshot shows two windows of the WikiRace application. The top window displays the search interface with two input fields: 'Ayam' and 'Gajah Mada'. A 'GO!' button is centered between them. Above the inputs, there are two buttons: 'IDS' (highlighted) and 'BFS'. Below the inputs, there is a small icon of a person running. The bottom window shows the results of the search, featuring a cartoon character on the left and a summary table on the right.

Rute Terpendek:
<ul style="list-style-type: none"> <li>Ayan</li> <li>Ayan Brand</li> <li>Indonesia</li> <li>Gajah Mada</li> </ul>

Jumlah artikel yang dicek: 98894

Panjang Rute: 4

Durasi pencarian: 45439ms

### 4.3. Analisis hasil pengujian

Pada algoritma BFS yang menggunakan multi Threading, program akan berhenti apa bila antrian yang terdapat dalam program sudah mencapai ketentuan yang sudah ditentukan. Program akan menjalankan go routine pada setiap page agar proses pencarian lebih cepat. Jika jalur tidak ditemukan maka program juga akan mengembalikan hasil bahwa jalur dari page awal menuju page target tidak ditemukan. Dari hasil analisis, dengan metode BFS program dapat menemukan jalur cukup cepat dengan batasan 4 artikel dalam jalur tersebut dan program juga dapat menyelesaikan hal tersebut kurang dari satu menit.

Pada algoritma IDS program menjalankan fungsi pencarinya secara loop dan rekursif untuk mengecheck setiap jawaban yang memungkinkan. Pada setiap page akan dilakukan rekursif sepanjang kedalaman yang ditentukan. Apabila kedalaman melewati yang sudah ditentukan maka program akan berhenti dan mengembalikan bahwa jalur tidak ditemukan. Dari hasil analisis metode ini dapat juga menemukan jalur paling banyak 4 dengan waktu kurang dari satu menit.

Pada setiap metode setelah pencarian dijalankan dan program mengembalikan hasilnya maka page website akan diarahkan ke pada page hasil proses dari pencarian tersebut.

## Bab 5: Kesimpulan dan saran

### 5.1. Kesimpulan

Kesimpulan yang dapat ditarik oleh kami adalah bahwa materi ini menguraikan deskripsi, pemecahan masalah, dan spesifikasi teknik dalam membangun program WikiRace yang memanfaatkan algoritma Iterative Deepening Search (IDS) dan Breadth First Search (BFS). WikiRace merupakan permainan interaktif yang memungkinkan pengguna mencari jalur terpendek antara dua artikel Wikipedia dengan memilih jalur optimal dalam graf artikel. Implementasi IDS dan BFS sebagai algoritma pencarian memungkinkan aplikasi mencapai tujuan pencarian dengan efisien dan efektif.

### 5.2. Saran

Saran yang dapat kami berikan untuk proses penggerjaan Tugas Besar IF2211 Strategi Algoritma 2023/2034 adalah:

1. Memperluas diskusi mengenai penanganan kasus-kasus dalam pengimplementasian algoritma IDS dan BFS, seperti penanganan siklus pada graf.
2. Lebih mendalami terkait teknologi web sehingga dapat menjalankan program WikiRace sebaik mungkin.
3. Melakukan lebih banyak eksplorasi terkait GoLang, algoritma IDS dan BFS.
4. Melengkapi *code program* dengan komentar agar pada saat pengembangan atau *debug* dapat memahami *code* dengan mudah.
5. Serta lebih teratur dalam menjadwalkan penggerjaan Tugas Besar

## Lampiran

Tautan *Repository GitHub*

[Repository Github](#)

Tautan Video

[Video Tubes Stima 2](#)

## Daftar Pustaka

Munir, Rinaldi & Maulidevi, Nur Ukfa. (2021). “Breadth/Depth First Search (BFS/DFS) Bagian 1”. Accessed April 20, 2024.  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/BFS-DFS-2021-Bag1-2024.pdf>

Munir, Rinaldi & Maulidevi, Nur Ukfa. (2021). “Breadth/Depth First Search (BFS/DFS) Bagian 2”. Accessed April 20, 2024.  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf>