

## **TUGAS BESAR 2**

### **STRATEGI ALGORITMA**

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



**Disusun Oleh:**

**Karunia Syukur Baeha (10023478)**

**Ahmad Hasan Albana (13522041)**

**Steven Tjhia (13522103)**

**Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

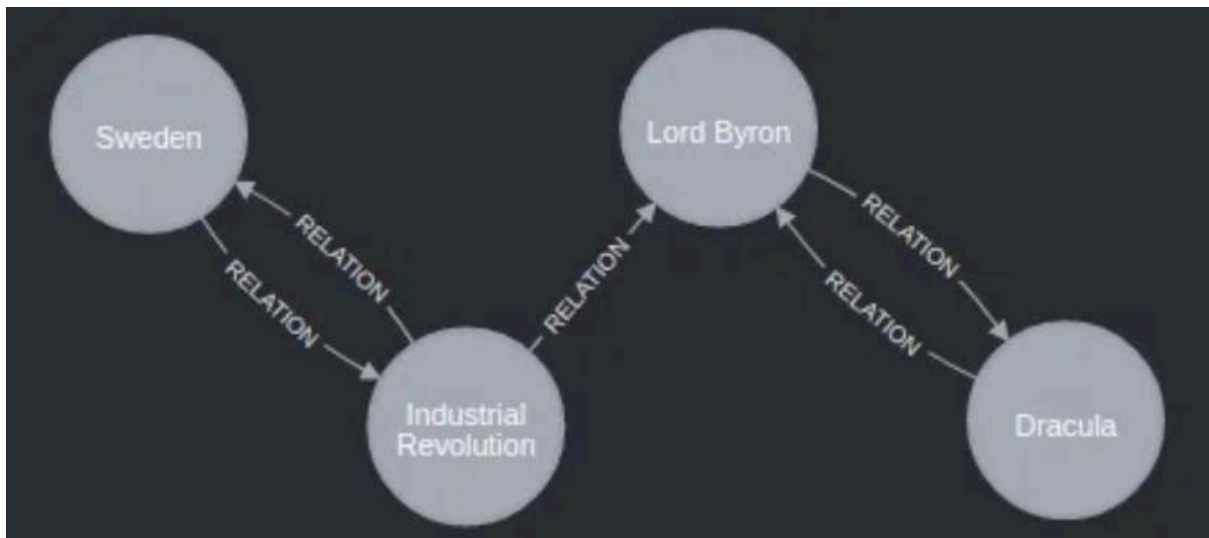
## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB 1</b>	
<b>Deskripsi Tugas.....</b>	<b>3</b>
Gambar Ilustrasi Graf WikiRace.....	3
<b>BAB 2</b>	
<b>Landasan Teori.....</b>	<b>4</b>
2.1 Graf.....	4
2.2 Iterative Deepening Search.....	4
2.3 Breadth First Search.....	4
2.4 Website Development.....	5
<b>BAB 3</b>	
<b>Analisis Pemecahan Masalah.....</b>	<b>6</b>
3.1 Web Scraping.....	6
3.2 Iterative Deepening Search.....	6
3.2 Breadth First Search.....	7
<b>BAB 4</b>	
<b>Implementasi dan Pengujian.....</b>	<b>8</b>
4.1 Spesifikasi Teknis Program.....	8
4.2 Tata Cara Penggunaan Program.....	11
4.3 Hasil Pengujian.....	13
4.4 Analisis Hasil Pengujian.....	19
<b>BAB 5</b>	
<b>Kesimpulan dan Saran.....</b>	<b>20</b>
5.1 Kesimpulan.....	20
5.2 Saran.....	20
5.3 Refleksi.....	20
<b>Lampiran.....</b>	<b>21</b>
<b>Daftar Pustaka.....</b>	<b>22</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 Ilustrasi Graf WikiRace

(Sumber: [https://miro.medium.com/v2/resize:fit:1400/1\\*jxmEbVn2FFWybZsIicJCWQ.png](https://miro.medium.com/v2/resize:fit:1400/1*jxmEbVn2FFWybZsIicJCWQ.png))

## **BAB 2**

### **Landasan Teori**

#### **2.1 Graf**

Graf adalah suatu struktur yang terdiri dari beberapa objek dan hubungan antar pasangan objek-objek tersebut. Secara sederhana, sebuah graf merupakan himpunan dari objek-objek yang dinamakan titik, simpul, atau sudut dihubungkan oleh penghubung yang dinamakan garis atau sisi. Dalam graf yang memenuhi syarat, di mana biasanya tidak berarah, sebuah garis dari titik A ke titik B dianggap sama dengan garis dari titik B ke titik A. Dalam graf berarah, garis tersebut memiliki arah. Pada dasarnya, sebuah graf digambarkan dengan bentuk diagram sebagai himpunan dari titik-titik (simpul) yang dihubungkan dengan sisi. Penjelajahan graf (graph traversal) adalah salah satu konsep dasar dalam teori graf yang melibatkan pengunjungan atau penelusuran setiap simpul (node) dalam sebuah graf. Tujuan dari penjelajahan graf adalah untuk mengunjungi setiap simpul tepat satu kali, sehingga setiap simpul dan tepinya (edge) dapat diproses atau dimanipulasi sesuai dengan kebutuhan.

#### **2.2 Iterative Deepening Search**

Algoritma Iterative-Deepening Search (IDS) merupakan algoritma yang menggabungkan keunggulan dari algoritma BFS dan DFS. Algoritma IDS (Iterative Deepening Search) adalah sebuah metode pencarian yang digunakan dalam teori graf dan kecerdasan buatan untuk menemukan jalur atau solusi di dalam struktur data berbentuk pohon atau graf. Algoritma IDS bekerja seperti DFS dengan melakukan penelusuran secara mendalam terlebih dahulu tetapi dibatasi oleh nilai kedalaman. Ketika sudah mencapai simpul terdalam, maka penelusuran akan diruntut balik. Tetapi jika tidak ditemukan solusinya, maka akan dilakukan penambahan kedalaman. Pencarian akan berhenti ketika sebuah solusi ditemukan. Algoritma ini digunakan dalam berbagai aplikasi seperti kecerdasan buatan, teori graf, permainan (game), dan masalah optimasi. Keunggulannya dalam menangani pencarian dengan ruang pencarian yang kompleks menjadikannya pilihan yang populer dalam konteks di mana kedalaman solusi tidak diketahui sebelumnya atau ketika memori terbatas.

#### **2.3 Breadth First Search**

Algoritma Breadth-First Search (BFS) atau algoritma pencarian melebar merupakan salah satu algoritma untuk melakukan traversal pada graf. Algoritma ini menelusuri dari sebuah simpul akar pada graf lalu mengunjungi semua tetangga dari simpul tersebut. Algoritma BFS (Breadth-First Search) memiliki beberapa kelebihan yang membuatnya

menjadi pilihan yang baik dalam beberapa kasus. Pertama, BFS tidak akan menemui jalan buntu (dead-end) dalam graf yang tidak berbobot. Ini berarti algoritma akan menjelajahi semua node yang terhubung pada setiap level sebelum beralih ke level berikutnya, sehingga memastikan bahwa semua kemungkinan jalur telah dieksplorasi sebelum memutuskan solusi. Selain itu, jika terdapat satu atau lebih solusi dalam graf, BFS akan menemukan solusi minimum atau jalur terpendek dari simpul awal ke simpul tujuan. Algoritma ini cocok untuk mencari jalur terpendek dalam graf yang tidak berbobot, di mana semua langkah memiliki bobot yang sama. Namun, BFS juga memiliki beberapa kekurangan yang perlu dipertimbangkan. Pertama, implementasi BFS membutuhkan alokasi memori yang signifikan, terutama untuk menyimpan semua node dalam satu pohon eksplorasi. Ini dapat menjadi masalah pada graf yang besar atau dalam aplikasi dengan memori terbatas. Selain itu, waktu yang dibutuhkan oleh BFS juga dapat menjadi lambat, terutama jika graf memiliki kedalaman yang besar. Algoritma harus menguji setiap level secara berurutan hingga solusi ditemukan pada level yang lebih dalam, yang dapat menghasilkan kinerja yang kurang efisien dalam beberapa kasus.

## 2.4 Website Development

Dalam proyek ini, kami menggunakan React.js dan Tailwind CSS untuk membangun antarmuka pengguna yang menarik dan fungsional. Dengan React.js, kami dapat mengorganisir komponen secara efisien, sementara Tailwind CSS memungkinkan desain tampilan yang responsif dan modern dengan mudah. Salah satu fitur utama proyek ini adalah penggunaan library React Graph Vis, yang memungkinkan kami menampilkan visualisasi hasil pencarian pada pencarian algoritma IDS dan BFS.

Untuk bagian backend digunakan go standard library yaitu *net/http* untuk menerima request dan memberi respons kepada frontend.

## BAB 3

### Analisis Pemecahan Masalah

#### 3.1 Web Scraping

Web Scraping dilakukan dengan menggunakan package *gocolly* yang telah ada sebelumnya. Package tersebut memungkinkan untuk mendapatkan elemen html pada laman yang dikunjungi berdasarkan *class*, *id*, ataupun jenis *tag*.

Beberapa *constraint* yang digunakan pada kedua algoritma adalah elemen dengan *class* “mw-page-title-main” dan memiliki *parent* dengan id “firstHeading” yang teks dari elemen tersebut merupakan judul dari artikel wikipedia tersebut. Lalu, diambil juga seluruh tag *<a>* pada website tersebut yang tidak memiliki *class* berupa “mw-file-description” yang menandakan bahwa tag tersebut hanya merupakan deskripsi dari suatu file atau gambar sehingga tidak perlu diperiksa. *Constraint* lain yang digunakan adalah tag *<link>* dengan atribut ‘rel’ bernilai ‘canonical’ untuk mendapatkan link sebenarnya yang dapat berasal dari link redirect, sehingga dapat diketahui apakah link redirect tersebut sebenarnya telah dikunjungi atau belum.

Pada program *web-scraping* ini, digunakan goroutine yang memungkinkan program dapat menjalankan beberapa *thread* sekaligus sehingga program dapat selesai lebih cepat. Selain itu, digunakan konsep mutex sehingga tidak ada konflik penggunaan global variabel antar *thread* dan juga penggunaan channel untuk membatasi jumlah *thread* yang terbentuk sehingga request ke laman wikipedia tidak terlalu banyak yang akan mengakibatkan laman tersebut memblokir jaringan.

#### 3.2 Iterative Deepening Search

Pada masalah pencarian solusi WikiRace atau Wiki Game menggunakan algoritma Iterative Deepening Search, setiap laman Wikipedia yang dikunjungi dianggap sebagai *node* pada graf dengan *edge* pada graf menunjukkan bahwa *node* atau laman yang ditunjuk dapat diakses melalui *node* atau laman asal.

Algoritma IDS ini dimulai dengan menggunakan depth bernilai 1. Jika solusi tidak ditemukan, maka akan dilakukan ulang Algoritma IDS dengan depth naik sebanyak 1, dengan hasil Algoritma IDS sebelumnya disimpan untuk mengetahui apakah *node* tersebut telah dikunjungi sebelumnya sehingga tidak perlu dikunjungi ulang untuk meningkatkan efisiensi waktu.

### 3.2 Breadth First Search

Algoritma BFS dapat diterapkan pada permainan Wikirace dengan menganalogikan node pada BFS sebagai link wikipedia. Pertama-tama akan dimulai pada link wikipedia awal, kemudian akan dilakukan scraping pada link tersebut. Scraping disini dianalogikan sebagai pembangkitan node selanjutnya. Setelah itu, link-link hasil scraping akan dilakukan perbandingan dengan link target yang diinginkan. Jika ditemukan, maka akan stop pada breadth tersebut, jika tidak, maka akan dilanjutkan dengan melakukan scraping pada link-link tersebut. Proses tersebut akan terus dilakukan sampai didapatkan link yang sama seperti link target. Pada implementasinya, terdapat caching dimana link yang telah dicek akan direkam sehingga suatu saat jika link tersebut ditemukan lagi, maka tidak perlu dicek lagi.

## BAB 4

### Implementasi dan Pengujian

#### 4.1 Spesifikasi Teknis Program

##### 4.1.1 Struktur Data

###### a. Struktur Umum

1. GraphView {Nodes: []Node, Edges: []Edge}
2. Node { Id: string, TitleArticle: string, UrlArticle: string, Shape: string, Size: int, Color: string, Font: string}
3. Font { Color: string, Size: int }
4. Color { Border: string, Background: string }
5. Edge { From: string, To: string }

###### b. IDS

1. childParentBool : map[string]map[string]bool  
> menyimpan informasi apakah key pertama merupakan *child* dari key kedua.
2. depthNode : map[string]int  
> menyimpan informasi depth dari suatu *node*.
3. scrapedNode : map[string]bool  
> menyimpan informasi apakah node telah dikunjungi.
4. alrFound : bool  
> menyimpan informasi apakah solusi telah ditemukan.
5. urlToTitle : map[string]string  
> menyimpan informasi judul artikel dari suatu link wiki
6. solutionParentChildBool : map[string]map[string]bool  
> menyimpan kumpulan *parent* dan *child* yang menjadi solusi, key berupa Title.
7. insertedNodeToJSON : map[string]bool  
> menyimpan informasi apakah *node* telah dimasukkan ke GraphSolusi.
8. GraphSolusi : GraphView  
> menyimpan informasi *node* dan *edge* pada graf solusi.

###### c. BFS

1. insertedNodeToJSON : map[string]bool  
> menyimpan informasi apakah *node* telah dimasukkan ke GraphSolusi.
2. GraphSolusi : GraphView  
> menyimpan informasi *node* dan *edge* pada graf solusi.
3. checkedNode : map[string]bool  
> menyimpan informasi apakah *node* sudah pernah dicek.
4. TotalCheckedArticle : int  
> menyimpan informasi total artikel yang dicek.
5. totalScrapedArticle : int  
> menyimpan informasi total artikel yang berhasil discape.
6. totalTryToScrapeArticle : int



- > menyimpan informasi total artikel yang berusaha untuk discape.
- 7. totalErrorScrape : int
  - > menyimpan informasi total artikel yang error saat ingin discape.
- 8. depthOfNode : map[string]int
  - > menyimpan informasi kedalaman suatu *node*.
- 9. depthTarget : int
  - > menyimpan informasi depth dari link target ketika ditemukan.
- 10. urlToTitle : map[string]string
  - > menyimpan informasi judul artikel dari suatu link/*node*.
- 11. Child\_parent\_bool : map[string]map[string]bool
  - > menyimpan informasi relasi *child-parent* antar-*node*.
- 12. solutionParentChildBool : map[string]map[string]bool
  - > menyimpan informasi relasi *parent-child* yang telah dimasukkan ke solusi.
- 13. currentDepth : int
  - > menyimpan informasi kedalaman saat ini
- 14. root : string
  - > menyimpan informasi link canonical root.
- 15. rootTitle : string
  - > menyimpan informasi judul artikel root.
- 16. target : string
  - > menyimpan informasi link canonical target.
- 17. targetTitle : string
  - > menyimpan informasi judul artikel target.

d. JSON Request

```
type RequestInfo struct {
    Algorithm string `json:"algorithm"`
    StartPage string `json:"startPage"`
    TargetPage string `json:"targetPage"`
}
```

e. JSON Response

```
type ResponseInfo struct {
    Status string `json:"status"`
    Error_Message string `json:"err"`
    Graph s.GraphView `json:"graph"`
    ResultDepth int `json:"depth"`
    ArticleChecked int `json:"checked"`
    ExecutionTime float64 `json:"time"`
}
```

#### 4.1.2 Fungsi

- IDS

Nama Fungsi	Keterangan
isWiki(link string) → boolean	Mengembalikan true jika link merupakan link wikipedia.

- BFS

Nama Fungsi	Keterangan
isWiki(link string) → boolean	Mengembalikan true jika link merupakan link wikipedia.

#### 4.1.3 Prosedur

- IDS

Nama Prosedur	Keterangan
IDS(input string, target string, iteration int, wg *sync.WaitGroup)	Melakukan proses IDS.
MainIDS(input string, target string)	Melakukan inisiasi proses IDS.
insertToSolution(child string, parent string)	Memasukkan pasangan <i>child-parent</i> ke dalam solusi
insertToJSON(child string, parent string)	Memasukkan pasangan <i>child-parent</i> ke dalam format JSON.
GetSolutionAndConvertToJSON()	Memasukkan hasil akhir ke solusi dan mengubahnya ke dalam format JSON.
ResetData()	Mengembalikan nilai setiap variabel global menjadi nilai awal.

- BFS

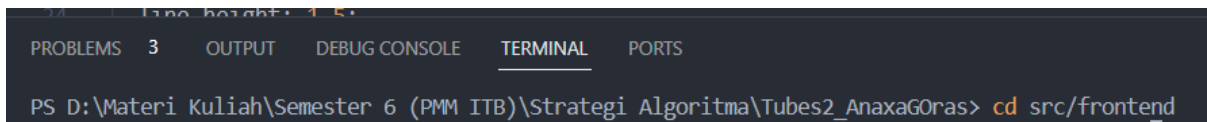
Nama Prosedur	Keterangan
BFS(start_page []string, target_page string)	Melakukan proses BFS
insertToSolution(child string, parent string)	Memasukkan pasangan <i>child-parent</i> ke dalam solusi
insertToJSON(child string, parent string)	Memasukkan pasangan <i>child-parent</i> ke dalam format JSON.

GetSolutionAndConvertToJson()	Memasukkan hasil akhir ke solusi dan mengubahnya ke dalam format JSON.
ResetData()	Mengembalikan nilai setiap variabel global menjadi nilai awal.

## 4.2 Tata Cara Penggunaan Program

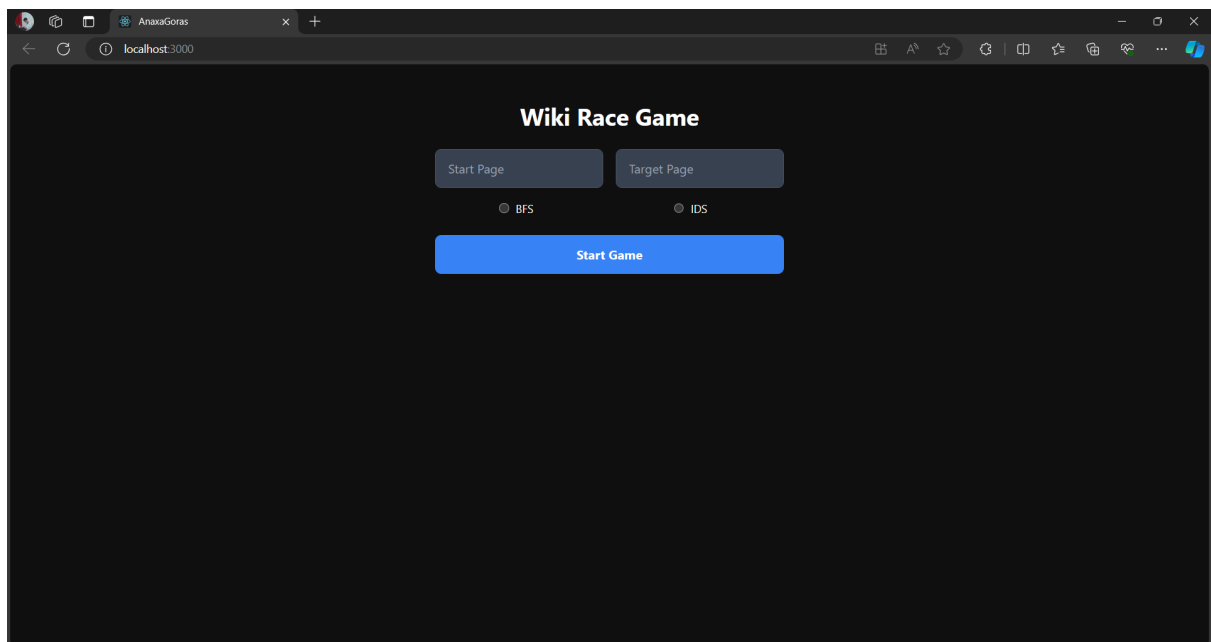
Tampilan dan penggunaan website:

- Buka terminal lalu masuk ke file frontend: “cd src/frontend”

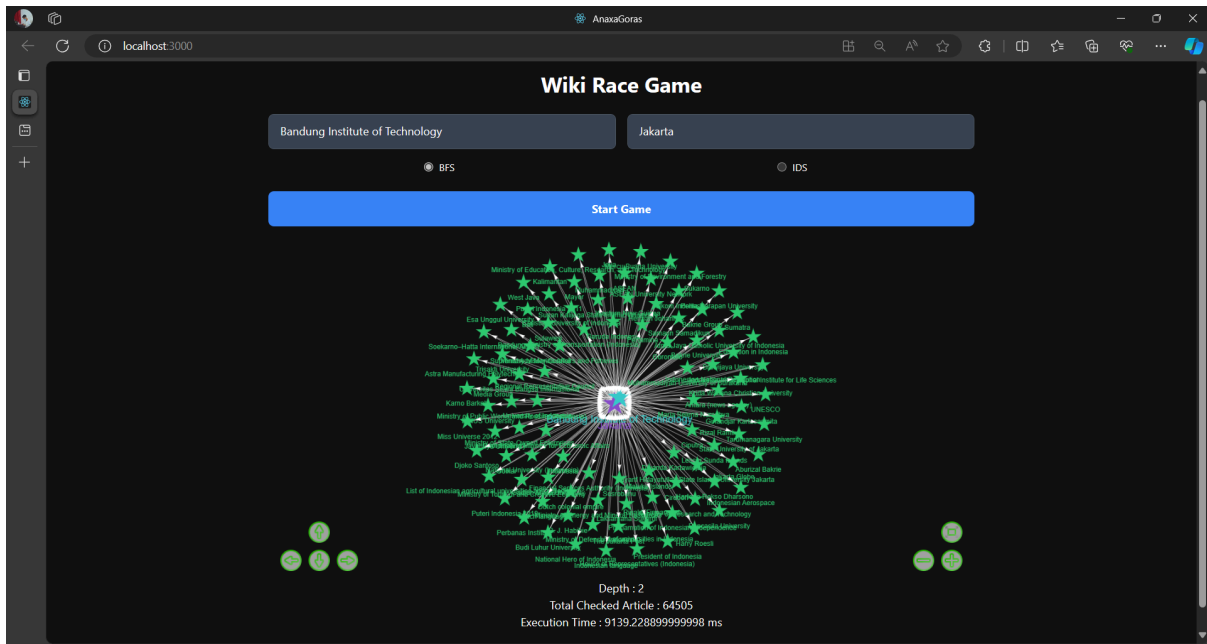


```
PS D:\Materi Kuliah\Semester 6 (PMM ITB)\Strategi Algoritma\Tubes2_AnaxaGOras> cd src/frontend
```

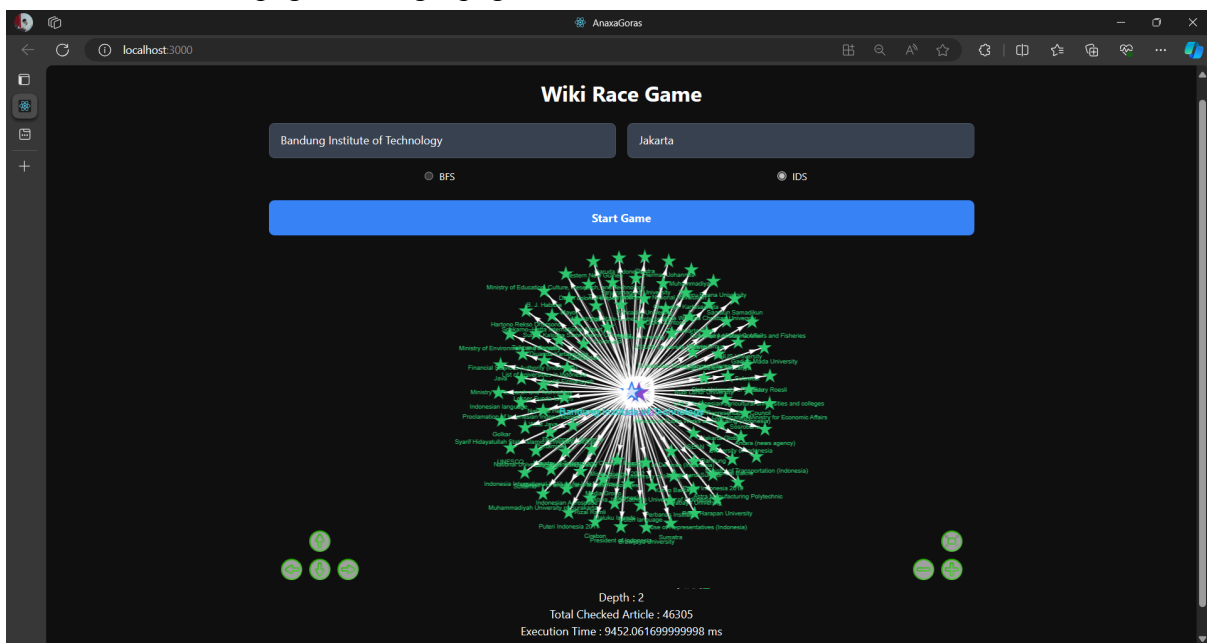
- Lakukan penginstalan: “npm install” lalu “npm start”. Tampilan website:



- Melakukan pencarian dengan algoritma BFS, dengan memasukkan kata pada start page dan target page lalu tekan start. Hasil:

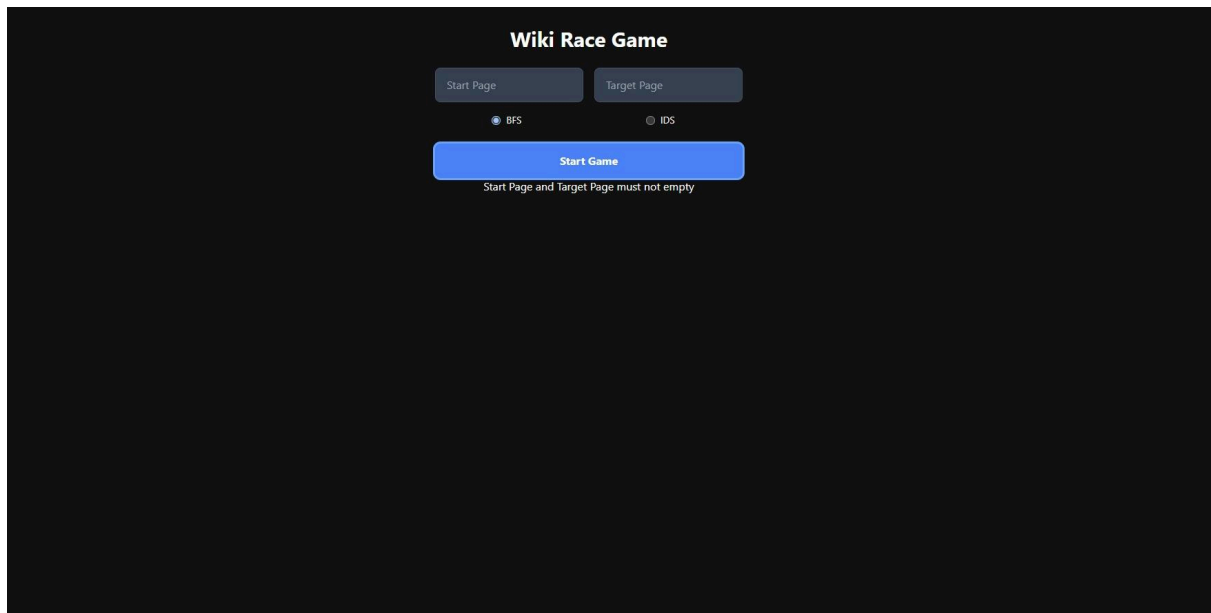


- Melakukan pencarian dengan algoritma IDS, dengan memasukkan kata pada start page dan target page lalu tekan start. Hasil:



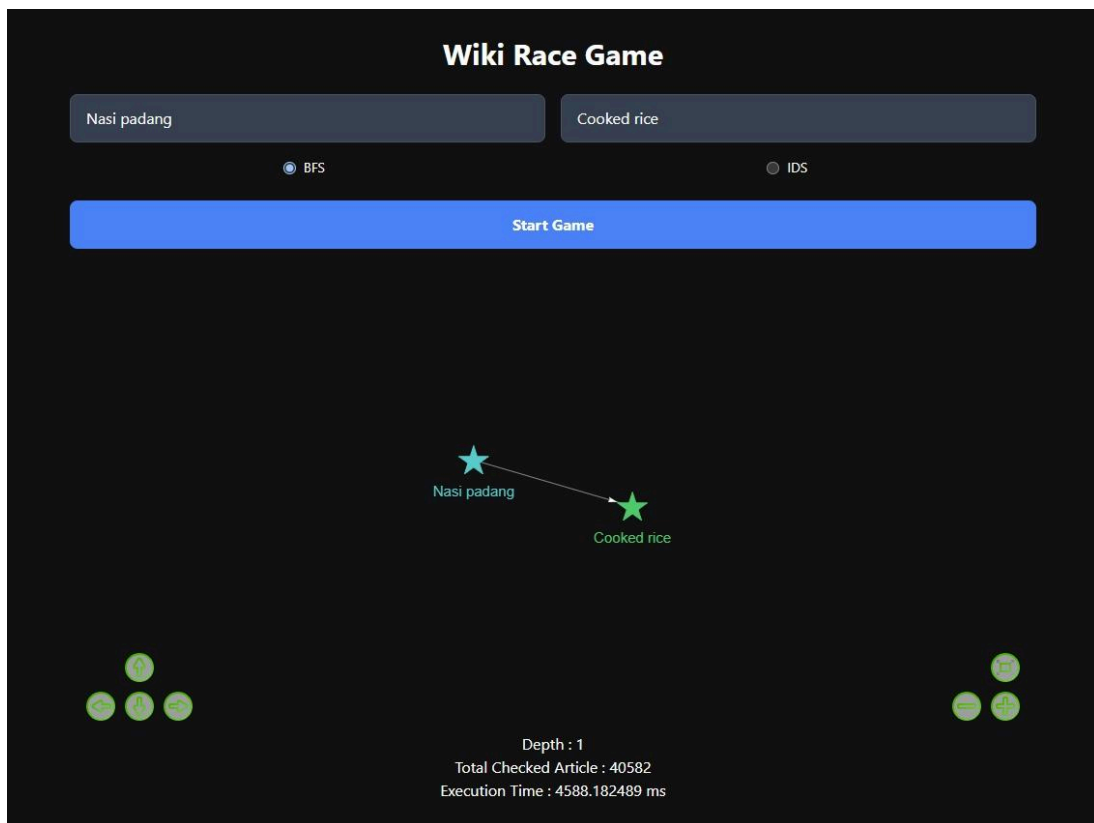
## 4.3 Hasil Pengujian

### 4.3.1 Invalid Input

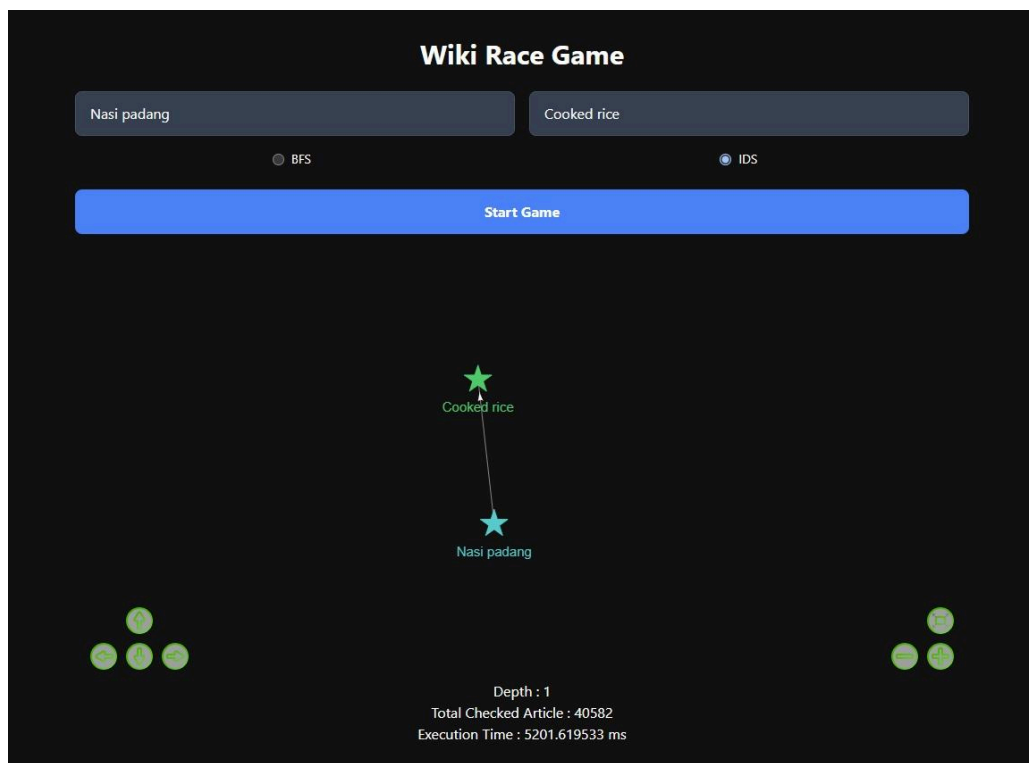


### 4.3.2 Redirected Link

- BFS

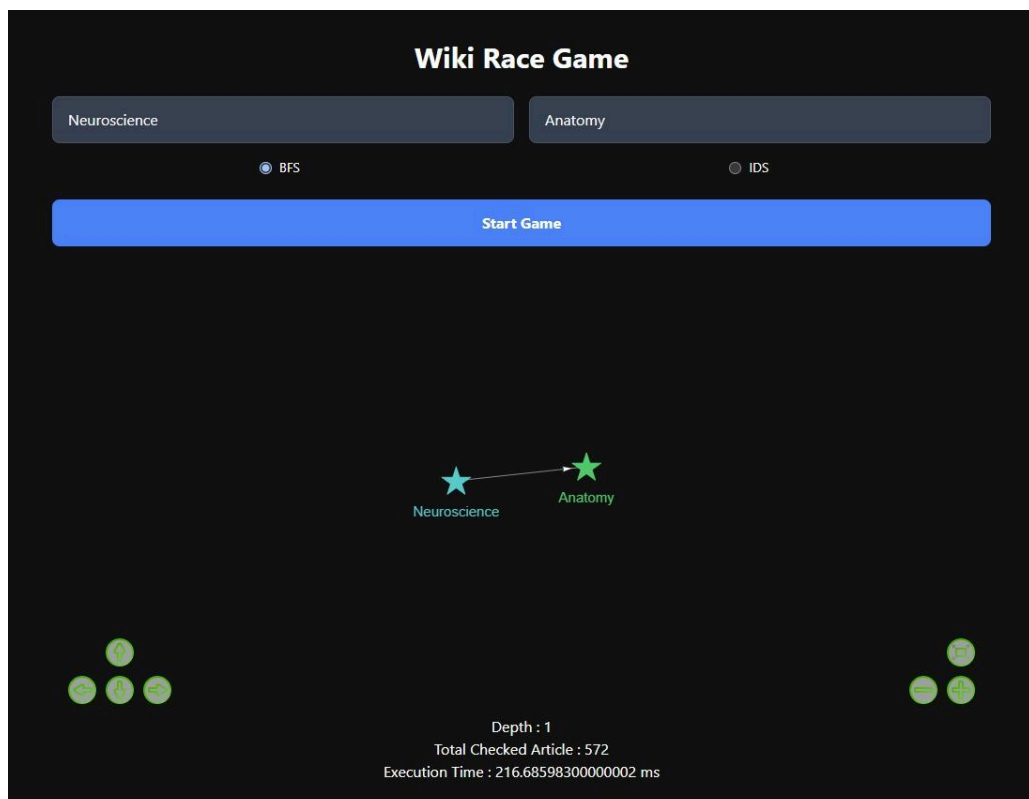


- **IDS**

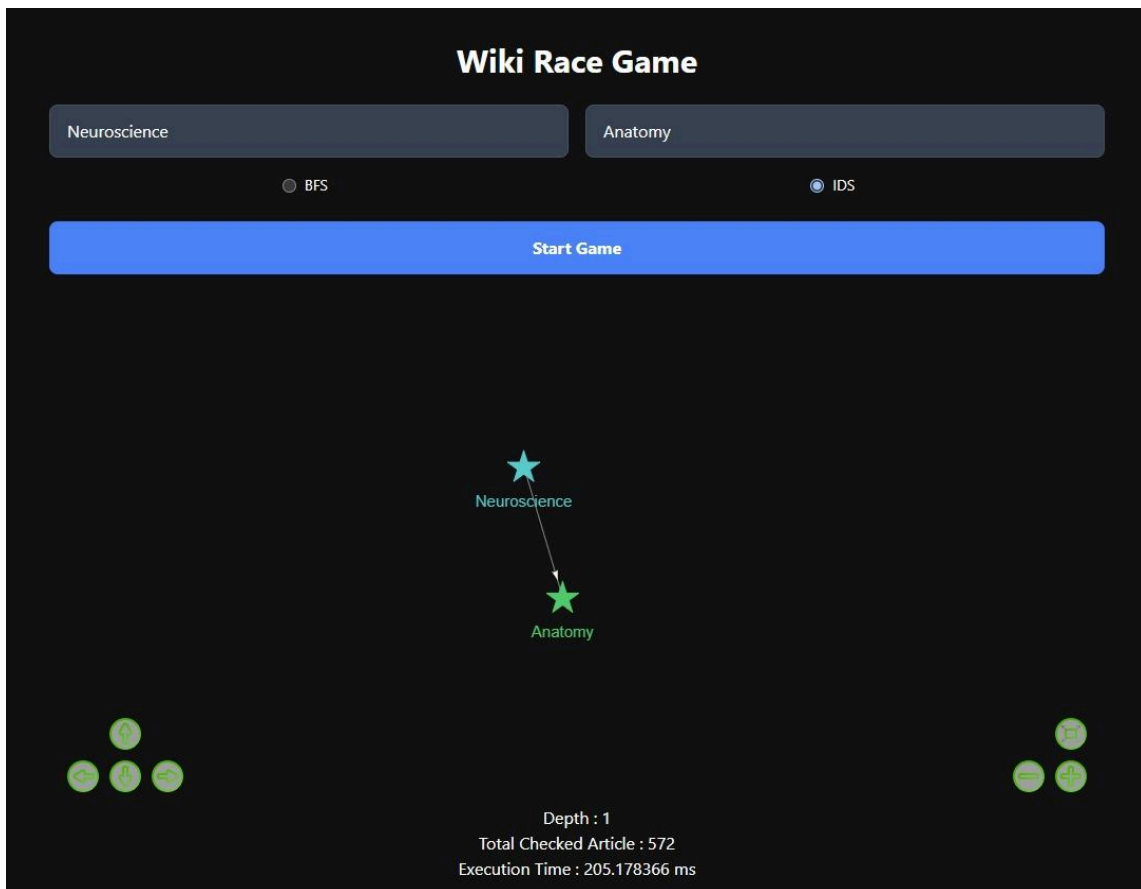


#### 4.3.3 Depth 1

- **BFS**

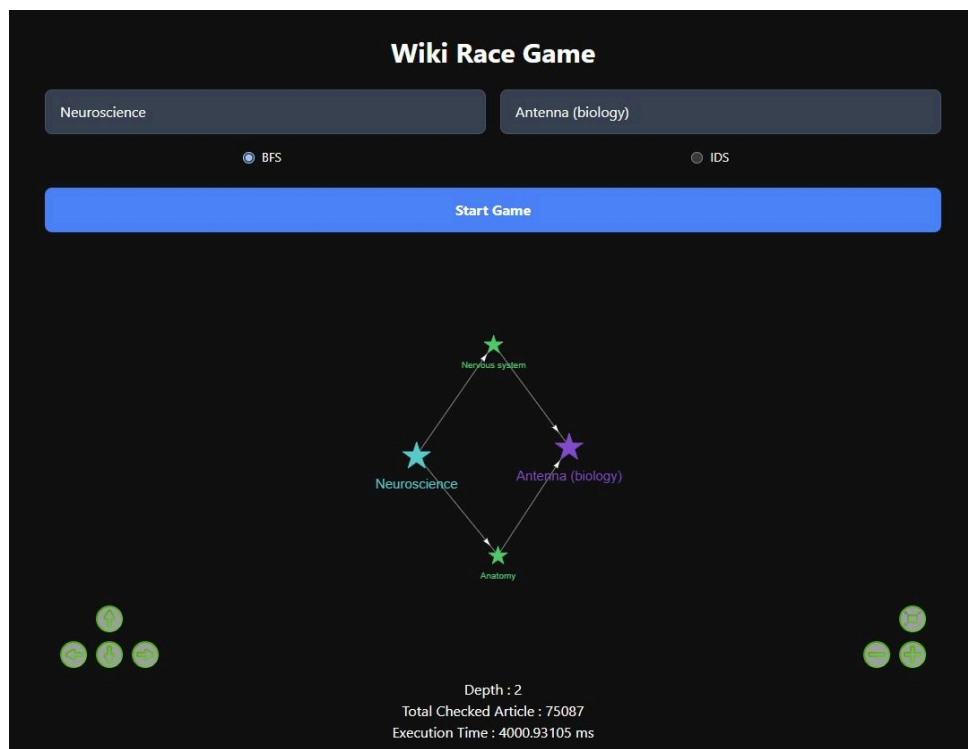


- **IDS**

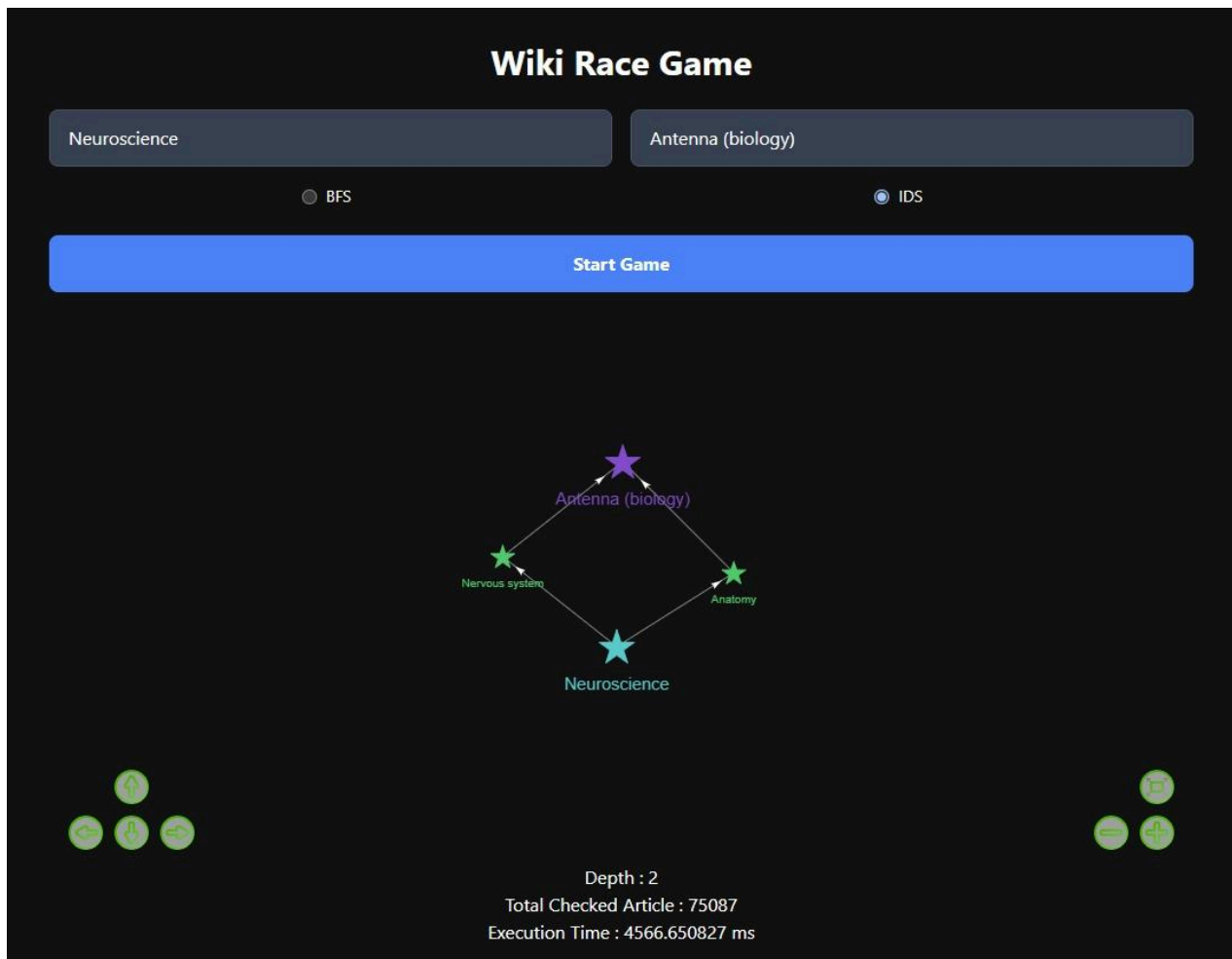


#### 4.3.4 Depth 2

- **BFS**



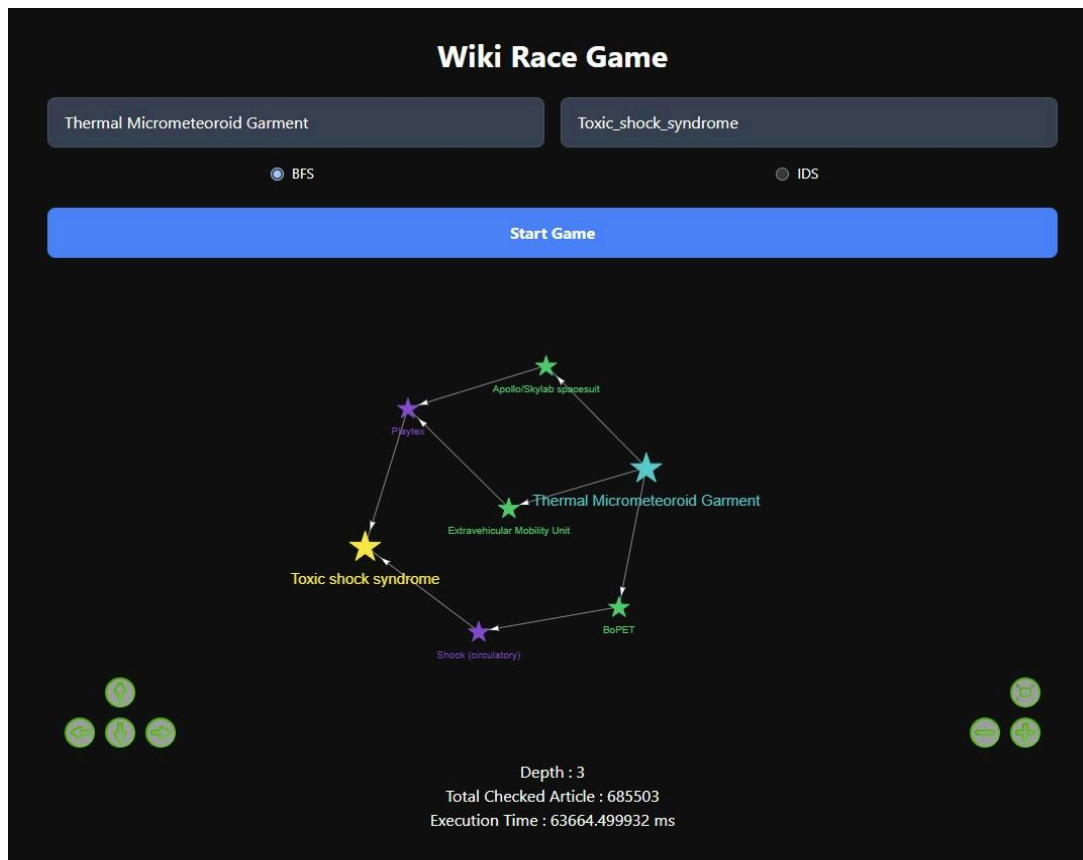
- **IDS**



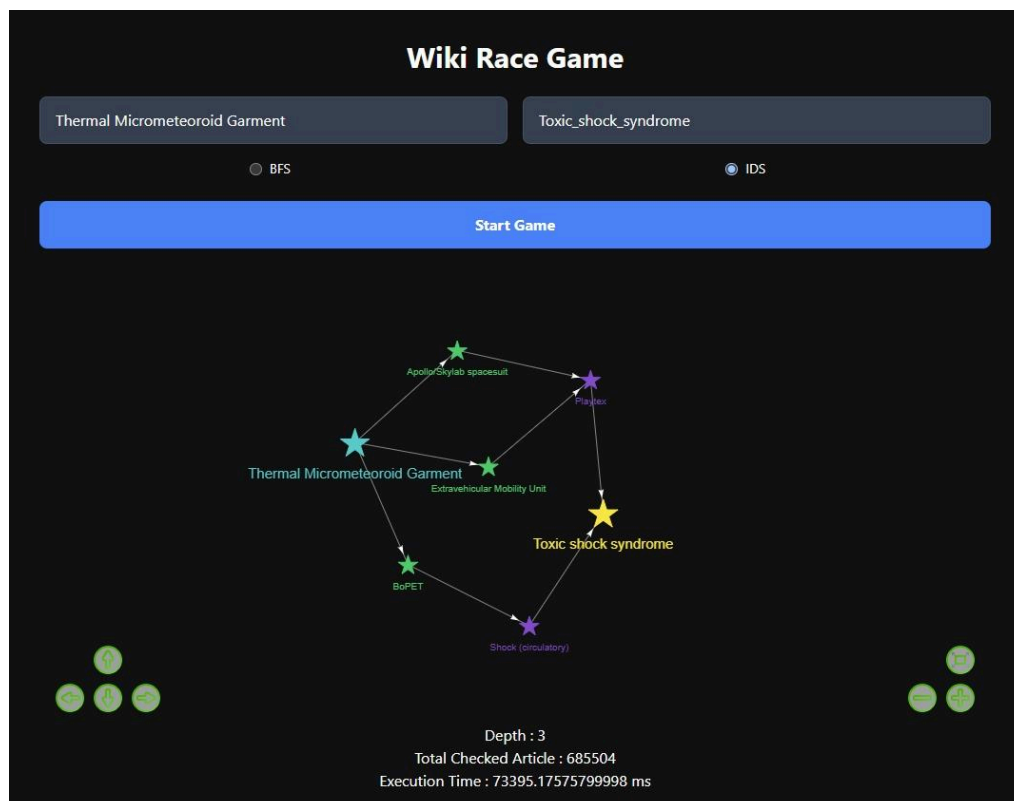


### 4.3.5 Depth 3

- BFS

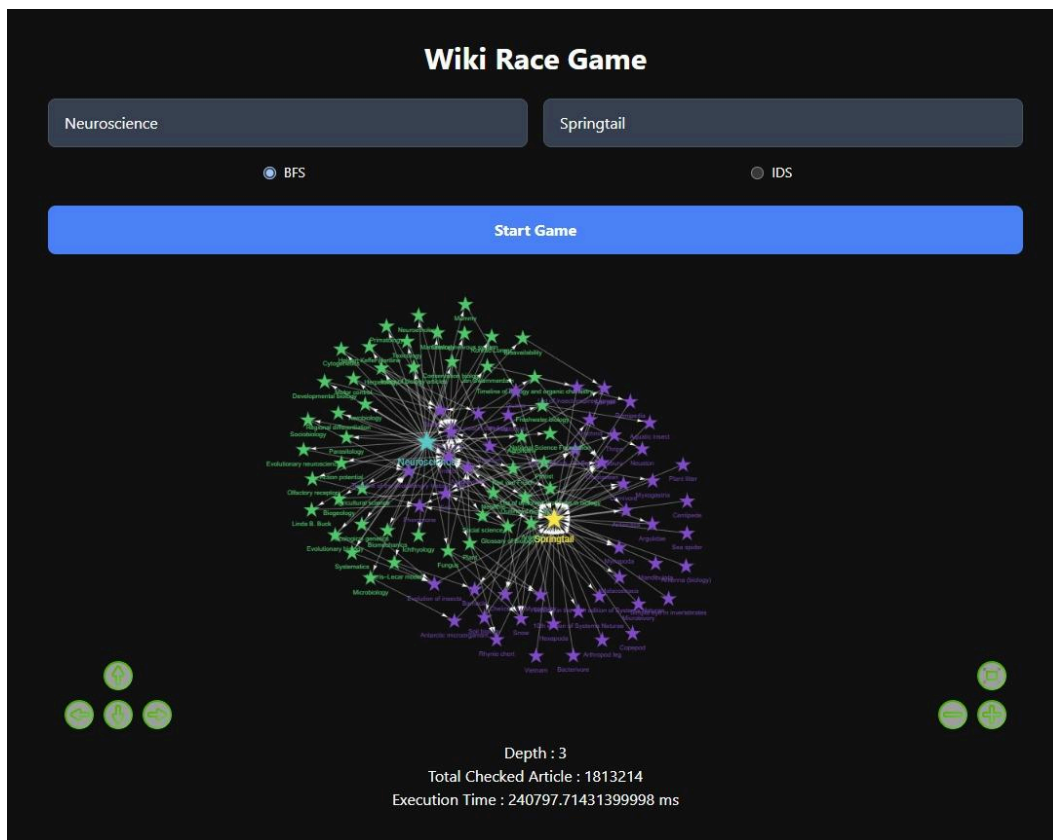


- IDS

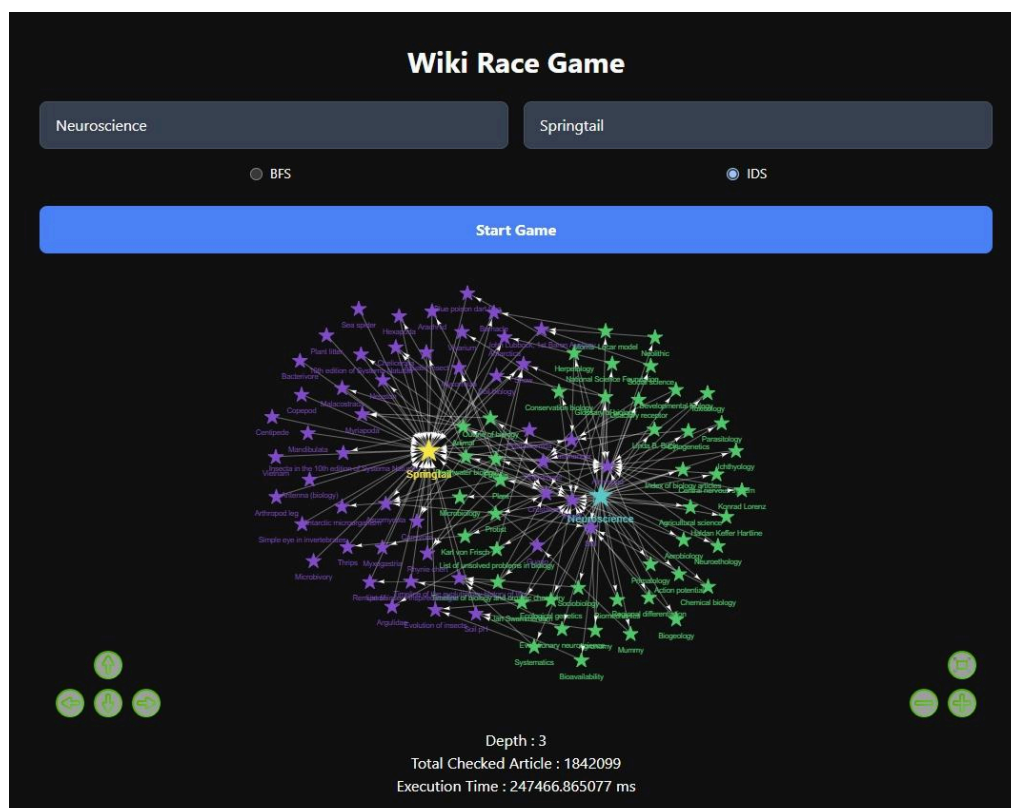


#### 4.3.6 Sadis(?)

- BFS



- IDS



#### 4.4 Analisis Hasil Pengujian

Berdasarkan pengujian yang telah dilakukan, kecepatan eksekusi program dipengaruhi oleh beberapa faktor. Faktor yang paling utama adalah algoritma karena dengan algoritma yang mangkus dan sangkil, waktu eksekusi yang diperlukan program menjadi lebih singkat. Faktor berikutnya yang tidak kalah penting adalah jaringan internet. Jaringan internet akan mempengaruhi kecepatan eksekusi program wikirace ini karena program ini melakukan web-scraping yang tentunya memerlukan internet. Selain itu juga dengan diimplementasikannya multithreading dengan goroutine, program dapat berjalan lebih cepat.

Kecepatan eksekusi program yang menggunakan BFS dan IDS tidak berbeda terlalu jauh karena BFS dan IDS memiliki kompleksitas waktu yang sama yaitu  $O(b^d)$  dengan  $b$  adalah jumlah cabang dan  $d$  adalah jumlah kedalaman. Pada kasus pencarian solusi banyak, algoritma IDS relatif sedikit lebih lambat dibandingkan dengan algoritma BFS. Hal tersebut dikarenakan pada algoritma IDS dilakukan penelusuran ulang ketika terjadi penambahan kedalaman.

## BAB 5

### Kesimpulan dan Saran

#### 5.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan analisis hasil pengujian adalah web-scraping dapat dilakukan dengan algoritma BFS maupun IDS. Pada kasus pencarian solusi banyak, algoritma IDS relatif lebih lambat dibandingkan dengan algoritma BFS karena algoritma IDS harus melakukan penelusuran ulang dari *root node* setiap terjadi penambahan kedalaman.

#### 5.2 Saran

- Meningkatkan tampilan website dan fungsionalitas agar dapat menampilkan hasil pencarian dalam bentuk single solution. Dengan demikian, pengguna dapat lebih mudah memahami alur pencarian tanpa harus melihat banyak solusi yang mungkin. Hal ini akan meningkatkan pengalaman pengguna dan membuat website ini lebih efisien dalam menyajikan informasi.
- Sebaiknya terdapat fitur *auto-complete* pada pengisian Start Page dan Target Page sehingga memudahkan pengguna.
- Perlu diperhatikan juga untuk UX-nya, misal seperti list path solusi yang ada karena jika solusi sangat banyak tidak memungkinkan untuk melihat langsung dari graf.

#### 5.3 Refleksi

- Hal yang kami dapat yaitu dalam mengimplementasikan library graph dan mempelajari dasar-dasar bahasa pemrograman Golang serta belajar tentang pengimplementasian algoritma IDS dan BFS dalam permainan wiki race agar project ini berjalan dengan baik.
- Hal yang kami dapatkan adalah ilmu mengenai dasar-dasar bahasa pemrograman Golang beserta cara melakukan webscraping dan goroutine untuk melakukan program secara multithread.

## Lampiran

**Link Repository:**

[https://github.com/Zechtro/Tubes2\\_AnaxaGOras.git](https://github.com/Zechtro/Tubes2_AnaxaGOras.git)

**Link Video:**

<https://youtu.be/M7I2T-VZUAW?si=Ib6hMffXB1cbUnFh>

## Daftar Pustaka

<https://github.com/PuerkitoBio/goquery>

<https://medium.com/@parulbaweja8/how-i-built-wikiracer-f493993fbdd>

<https://medium.com/@defytamara2610/bfs-breadth-first-search-pengertian-kekurangan-kelebihan-dan-contohnya-775a7d808fbc>