

LAPORAN TUGAS BESAR 2

IF2211 STRATEGI ALGORITMA

Pemanfaatan Algoritma IDS dan BFS dalam Permainan WikiRace



Disusun oleh:

Matthew Vladimir Hutabarat 13522093

Marvin Scifo Y. Hutahean 13522110

Berto Richardo Togatorop 13522118

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
1.1 Penjelasan Wikirace.....	3
BAB II.....	4
2.1 Penjelajahan Graf.....	4
2.2 Algoritma BFS.....	4
2.3 Algoritma IDS.....	5
BAB III.....	6
3.1 Langkah-Langkah Pemecahan Masalah.....	6
3.2 Proses pemetaan masalah menjadi elemen-elemen algoritma IDS dan BFS.....	8
3.3 Fitur fungsional dan arsitektur aplikasi web yang dibangun.....	11
3.4 Contoh Ilustrasi Kasus.....	14
BAB IV.....	18
4.1 Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).....	18
4.2 Penjelasan tata cara penggunaan program.....	22
4.3 Hasil pengujian.....	23
4.4 Analisis hasil pengujian.....	30
BAB V.....	32
5.1 Kesimpulan.....	32
5.2 Saran.....	32
5.3 Refleksi.....	32
DAFTAR PUSTAKA.....	34
LAMPIRAN.....	34
Link Repository.....	34

BAB I

Deskripsi Tugas

1.1 Penjelasan Wikirace

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.

Dengan menggunakan berbagai cara, beberapa pengembang membuat sebuah program yang menerima masukkan berupa 2 judul Wikipedia berbeda yang bertugas sebagai judul awal dan judul akhir. Dari masukan tersebut, program diharapkan bisa mendapatkan solusi jalur terpendek dari judul awal ke judul akhir. Beginilah contoh program untuk mencari jalur terpendek tersebut.



Gambar 1.1.1 - Contoh Cara Kerja Wikirace

BAB II

Landasan Teori

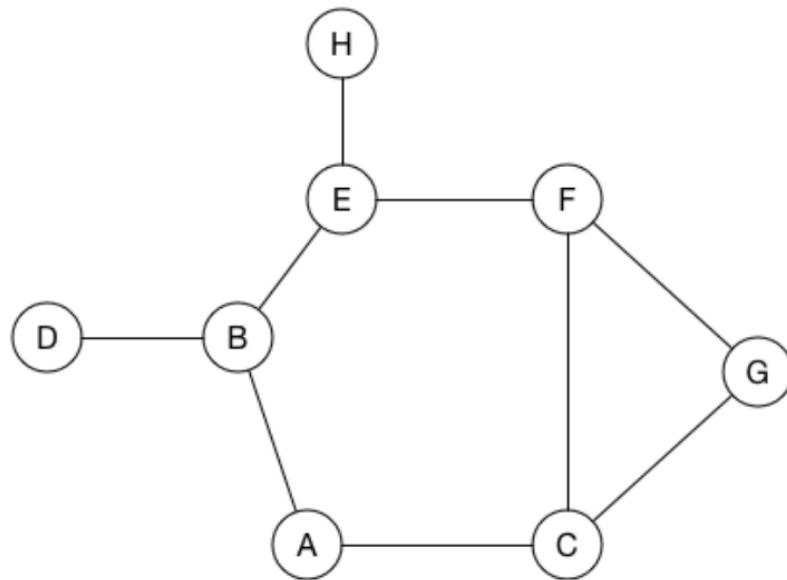
2.1 Penjelajahan Graf

Penjelajahan graf adalah suatu algoritma yang melakukan penelusuran berbagai node yang terhubung dengan garis. Dalam algoritma ini, terdapat starting point dan pada graf tersebut, dari starting point tersebut akan melakukan penelusuran terhadap node-node yang ada dengan cara-cara tertentu. Tujuan besar dari penjelajahan graf adalah mencari solusi yang terdapat di graf tersebut. Ada beberapa jenis algoritma yang bisa digunakan untuk melakukan penjelajahan graf. Pada permasalahan Wikirace, algoritma yang digunakan ada 2 yaitu BFS (*Breadth First Search*) dan IDS (*Iterative Deepening Search*).

2.2 Algoritma BFS

Algoritma BFS adalah skema pencarian secara melebar. Pada BFS, traversal dimulai dari sebuah node awal yang anggap saja disebut sebagai starting point. Algoritma BFS pada umumnya adalah sebagai berikut.

1. Kunjungi node dari starting point
2. Kunjungi semua node yang bertetangga dengan node starting point terlebih dahulu
3. Kunjungi node yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya



Gambar 2.2.1 - Contoh Graf 1

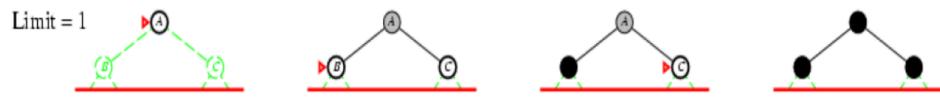
Melihat dari gambar ini, jika kita menggunakan algoritma BFS, jika dimulai dari node A dan memprioritaskan huruf terkecil, urutannya akan menjadi seperti ini -> A,B,C,D,E,F,G,H

2.3 Algoritma IDS

Algoritma IDS (*Iterative Deepening Search*) adalah skema penelusuran graf yang melakukan serangkaian DFS tetapi dengan sebuah komponen kedalaman yang bertugas untuk membatasi kedalaman pencarian. Contoh, jika IDS dilakukan dengan jumlah kedalaman 3, maka program hanya akan mencari solusi sampai kedalaman 3 saja dan lewat dari situ tidak akan dihiraukan.

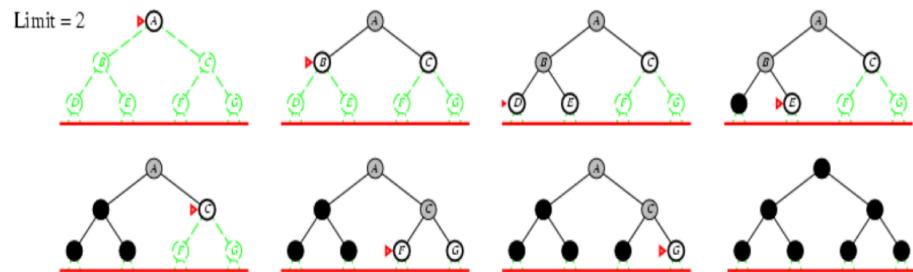
Beginilah contoh dari IDS berdasarkan iterasi kedalamannya.

1. Kedalaman 1



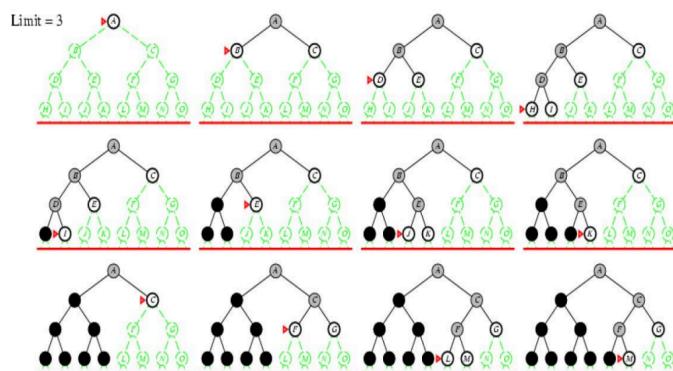
Gambar 2.3.1 - Graf dengan IDS kedalaman 1

2. Kedalaman 2



Gambar 2.3.2 - Graf dengan IDS kedalaman 2

3. Kedalaman 3



Gambar 2.3.3 - Graf dengan IDS kedalaman 3

BAB III

Analisis Pemecahan Masalah

3.1 Langkah-Langkah Pemecahan Masalah

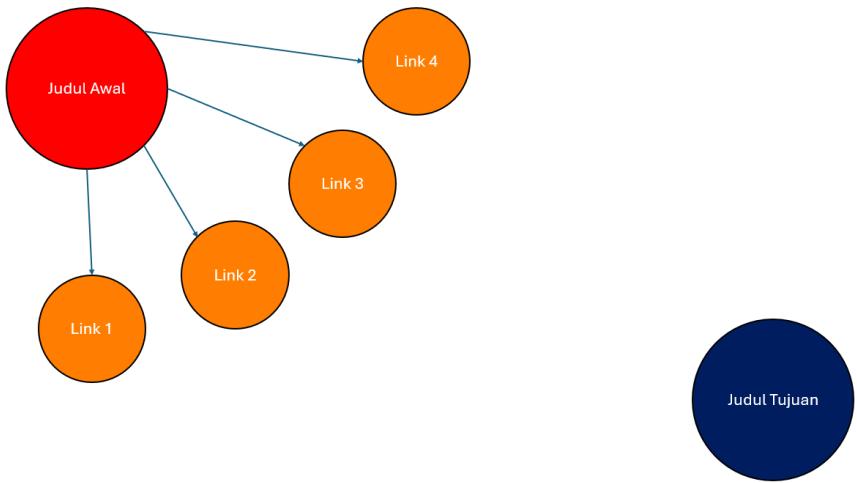
Permasalahan terkait pencarian solusi terdekat dari dua judul Wikipedia yang berbeda bisa diselesaikan dengan beberapa cara. Seperti yang dijelaskan sebelumnya, IDS dan BFS bisa menjadi algoritma yang cocok untuk menyelesaikan masalah tersebut. Namun, secara umum, skema pencarian bisa dijelaskan sebagai berikut.

1. Memasukkan input berupa dua judul wikipedia yang VALID
Solusi dari masalah ini dimulai dengan memasukkan dua judul Wikipedia. Pastikan input yang dimasukkan adalah valid dan benar-benar sebuah artikel di Wikipedia.
2. Lakukan Pencarian
Jika divisualisasikan, judul pertama akan menjadi starting point dan judul kedua akan menjadi finishing point. Bayangkan sebuah graf dengan hanya 2 simpul dan tanpa garis.



Gambar 3.1.1 - Graf Awal

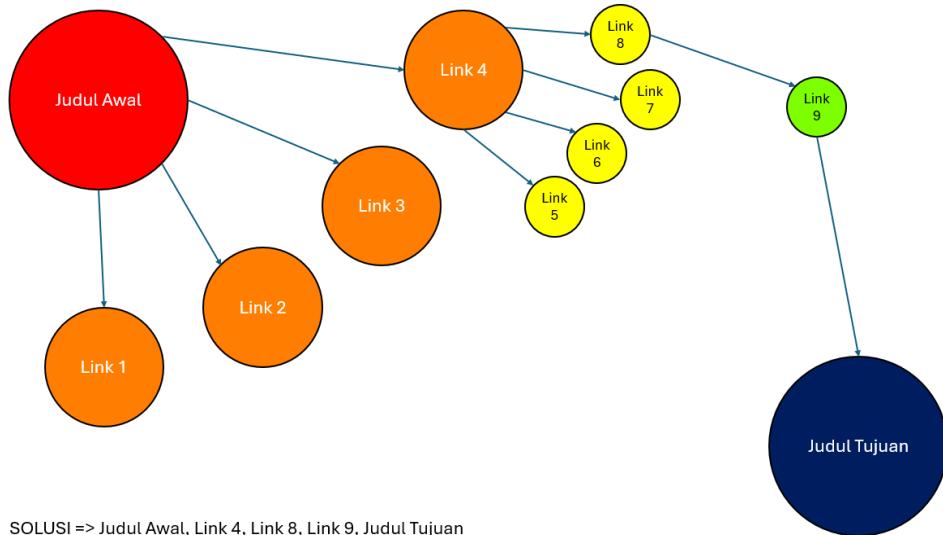
3. Menghubungkan link yang ada di artikel Wikipedia
Pada artikel Wikipedia, terdapat beberapa teks yang dijadikan sebuah tempat untuk meletakkan tautan ke halaman artikel lainnya. Dalam bahasa HTML, tautan seperti ini biasa digambarkan sebagai `Text`. Ada beberapa alat yang bisa digunakan untuk langsung mendapatkan semua tautan yang ada di artikel Wikipedia tersebut. Perlu diketahui bahwa link Wikipedia yang akan dihubungkan dengan judul awal hanyalah link Wikipedia yang tidak punya namespace ("Wikipedia:", "Category:", "Special:", dll). Daftar link yang didapat bisa divisualisasikan ke dalam sebuah graf dengan daftar link tadi dihubungkan dengan artikel yang mempunyai daftar link tersebut.



Gambar 3.1.2 - Gambar Graf dengan tautan tambahan

4. Cek jika tautan awal dan tautan akhir

Solusi akan didapat jika saat melakukan penelusuran tautan, tautan tersebut adalah tautan yang sama dengan tautan tujuan. Jika sudah ditemukan, pencarian akan dihentikan. Jika tidak, pencarian akan dilakukan sampai selesai (tergantung algoritma yang digunakan)



Gambar 3.1.3 - Contoh Hasil Akhir Pencarian

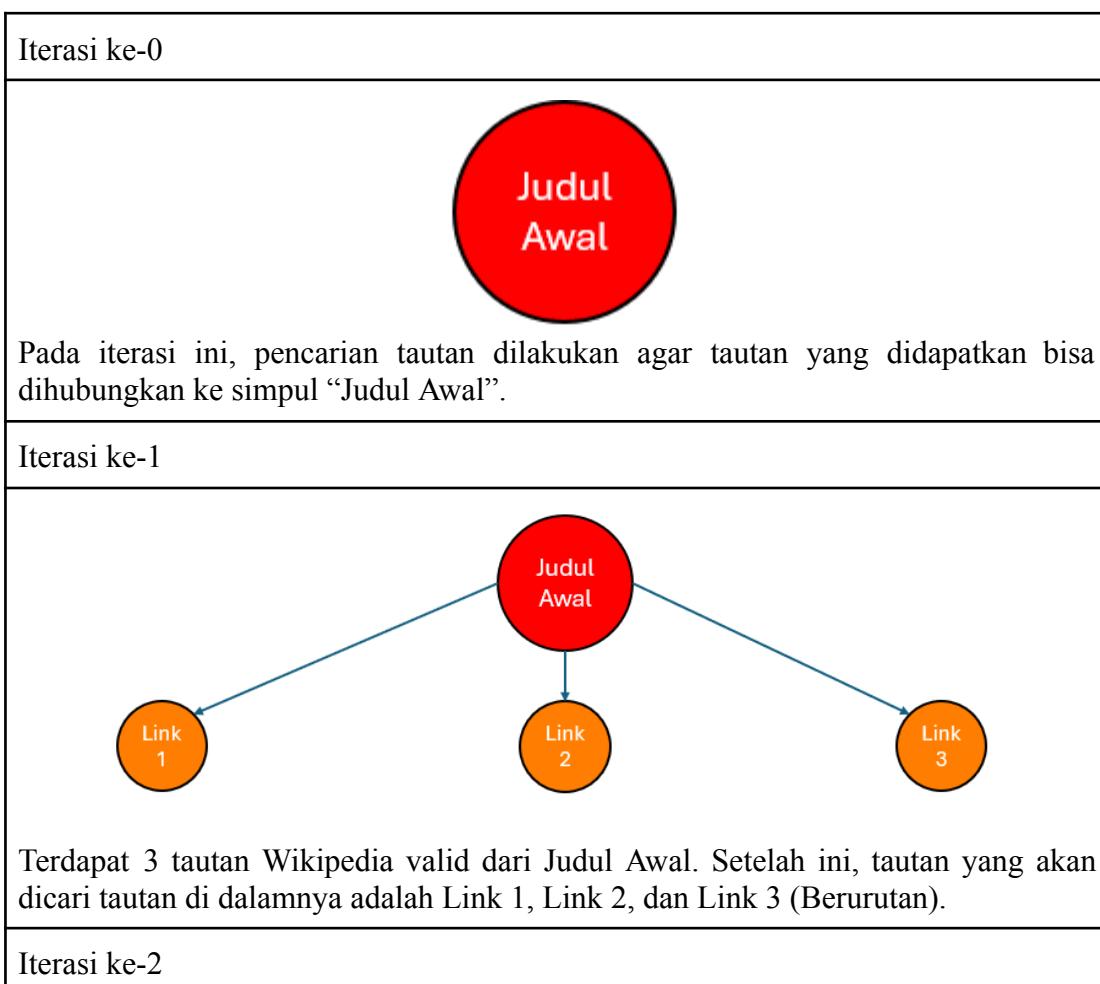
5. Pemunculan solusi

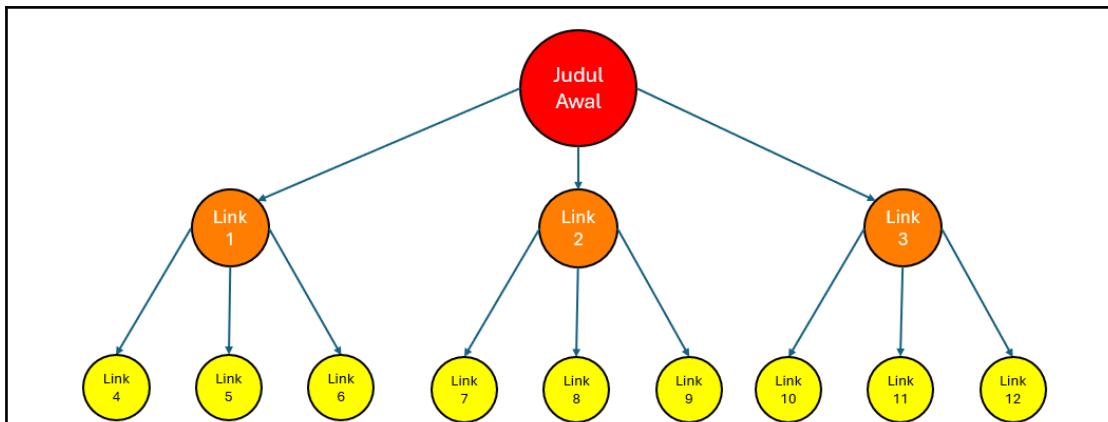
Beberapa program menyediakan layanan untuk hanya memunculkan satu solusi saja dan ada beberapa program yang bisa memunculkan banyak solusi. Dan dari solusi tersebut, bisa dimunculkan dalam bentuk list (daftar nama) atau bentuk graf (Banyak solusi sangat disarankan menggunakan graf)

3.2 Proses pemetaan masalah menjadi elemen-elemen algoritma IDS dan BFS

1. BFS (*Breadth First Search*)

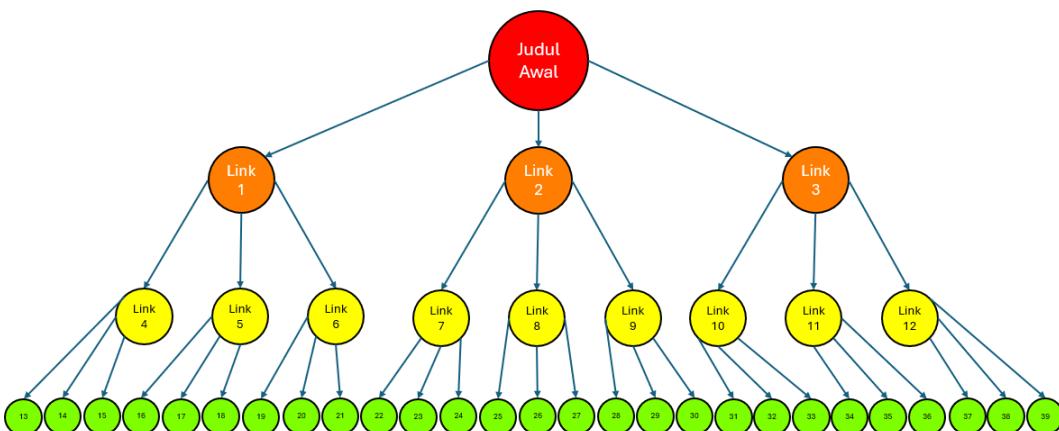
Seperti yang dijelaskan sebelumnya, judul awal dan judul tujuan akan dijadikan simpul dan daftar tautan Wikipedia valid yang ada di artikel judul awal adalah anak dari simpul judul awal yang dihubungkan dengan garis. Untuk BFS, judul tujuan sebagai simpul tidak perlu divisualisasikan. Misal solusinya memerlukan 3 kali klik. Beginilah visualisasi untuk Iterasi BFS tersebut. Pada pencarian ini, tautan yang membawa pengguna kembali ke link sebelumnya tidak akan dijadikan tautan yang valid untuk dilakukan pencarian.





Setiap tautan yang dikunjungi saat Iterasi ke-1 masing-masing berisi 3 tautan juga. Mulai dari kiri, setiap tautan akan dikunjungi lagi untuk mencari daftar tautan yang ada.

Iterasi ke-3



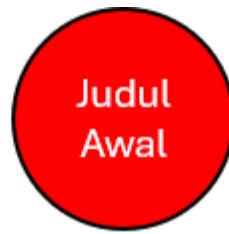
Misalkan solusi ada di lapis ke-3. Setiap kali tautan dari lapis ke-2 memberikan informasi terkait daftar tautan yang ada, tautan tersebut akan dibandingkan kesamaannya dengan masukan judul tujuan. Tautan yang diperiksa selalu mulai dari kiri dan jika sudah habis, tautan yang sudah dikunjungi tadi akan memberikan informasi tentang daftar tautan yang dipunyai oleh artikel dan proses ini dilakukan sampai tautan yang dikunjungi sama dengan tautan solusi.

Misal judul tujuan mempunyai tautan 34, maka solusi yang didapat adalah:
 Judul Awal -> Link 3 -> Link 11 -> Link 34

2. IDS (*Iterative Deepening Search*)

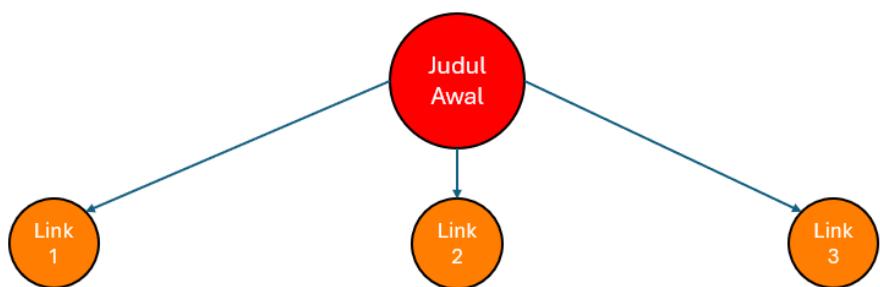
Sama seperti BFS, IDS juga memerlukan simpul dan garis untuk melakukan pencarian. Namun, skema pencarinya berbeda. IDS pencarinya sama dengan DFS tetapi dilakukan secara berulang dengan kedalaman yang berbeda-beda secara terurut mengecil. Contohnya pada iterasi ke-0, IDS melakukan DFS hanya pada simpul awal dan pada iterasi ke-4, IDS melakukan DFS hanya sampai kedalaman 4 saja. Proses ini dilakukan sampai solusi ditemukan. Misalkan IDS nya dengan kedalaman 2.

Iterasi ke-0



Pada iterasi ini, pencarian tautan dilakukan agar tautan yang didapatkan bisa dihubungkan ke simpul "Judul Awal".

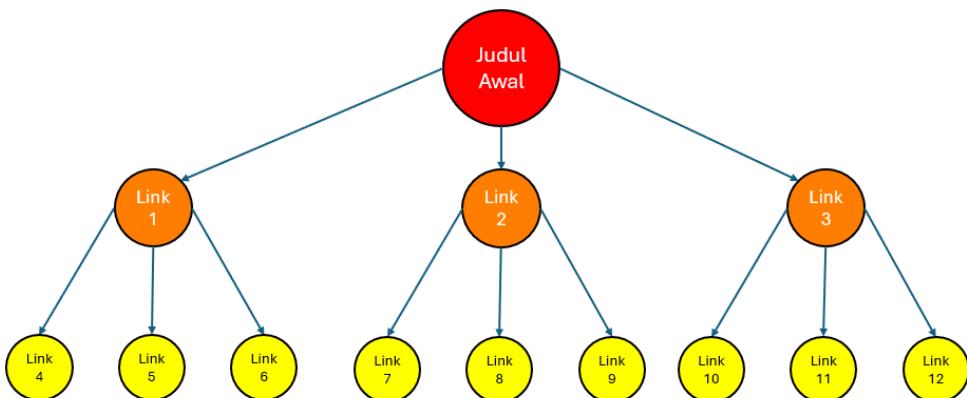
Iterasi ke-1



Terdapat 3 tautan Wikipedia valid dari Judul Awal. IDS dengan kedalaman 1 artinya pencarian dilakukan dengan pola seperti ini:

- Judul Awal -> Link 1
- Judul Awal -> Link 2
- Judul Awal -> Link 3

Iterasi ke-2

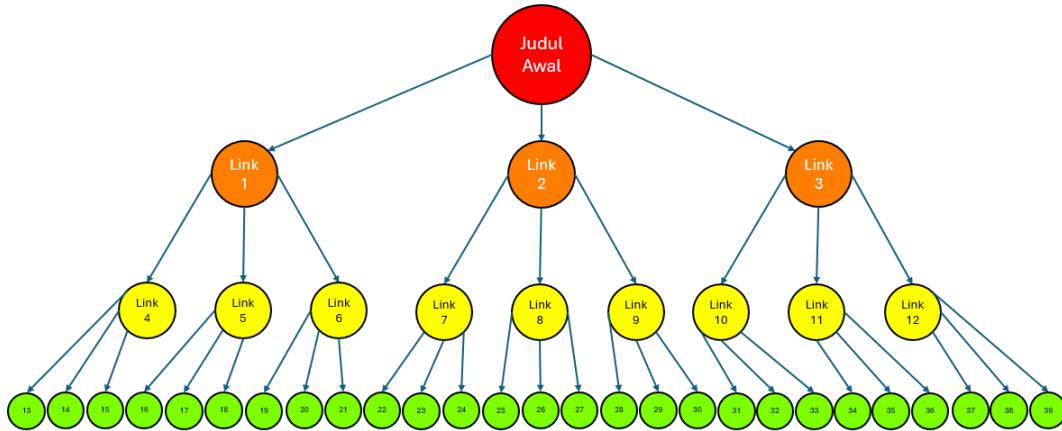


Setiap tautan yang dikunjungi saat Iterasi ke-1 masing-masing berisi 3 tautan juga. Pada situasi ini, pada iterasi IDS yang ke-2, DFS dilakukan hanya sampai kedalaman 2 saja. Polanya kira-kira akan seperti ini

- Judul Awal -> Link 1 -> Link 4
- Judul Awal -> Link 1 -> Link 5
- Judul Awal -> Link 1 -> Link 6
- Judul Awal -> Link 2 -> Link 7

Dan seterusnya...

Iterasi ke-3



Misalkan solusi ada di lapis ke-3. Setiap kali tautan dari lapis ke-2 memberikan informasi terkait daftar tautan yang ada, tautan tersebut akan dibandingkan kesamaannya dengan masukan judul tujuan. Tautan diperiksa dengan melakukan traversal ke simpul yang paling kiri lalu perlahan ke kanan dengan melakukan backtrack. Polanya kira-kira seperti ini

Judul Awal -> Link 1 -> Link 4 -> Link 13

Judul Awal -> Link 1 -> Link 4 -> Link 14

.

.

Judul Awal -> Link 3 -> Link 12 -> Link 39

Misal judul tujuan mempunyai tautan 34, maka solusi yang didapat adalah:

Judul Awal -> Link 3 -> Link 11 -> Link 34

3.3 Fitur fungsional dan arsitektur aplikasi web yang dibangun

Aplikasi web yang dibangun memiliki 2 komponen yaitu Frontend dan Backend. Frontend yang kami gunakan adalah React.js sedangkan Backend yang kami gunakan adalah bahasa Go. Web yang dibangun hanyalah memiliki satu halaman. Beginilah gambar web yang dibangun.



Gambar 3.3.1 - Halaman Utama Web

Bagian warna biru hanyalah sebuah kata-kata *quotes* buatan yang sebenarnya tidak dibuat oleh Sun Tzu (Hiasan). Pada bagian yang berwarna merah, terdapat komponen yang bertugas sebagai input dari web yang siap dimasukkan ke Backend.

1. BFS and IDS toggle

Komponen ini bertugas untuk memilih opsi algoritma yang akan digunakan untuk melakukan penelusuran tautan Wikipedia.

2. With or Without Namespaces

Komponen ini bertugas untuk memilih apakah akan menggunakan kata kunci seperti “Wikipedia:”, “Portal:”, “Special:”, dll. Sebenarnya pada Wikirace sebenarnya ini tidak digunakan tetapi untuk memberi kebebasan pada pengguna, opsi ini dibuat.

3. Starting Link

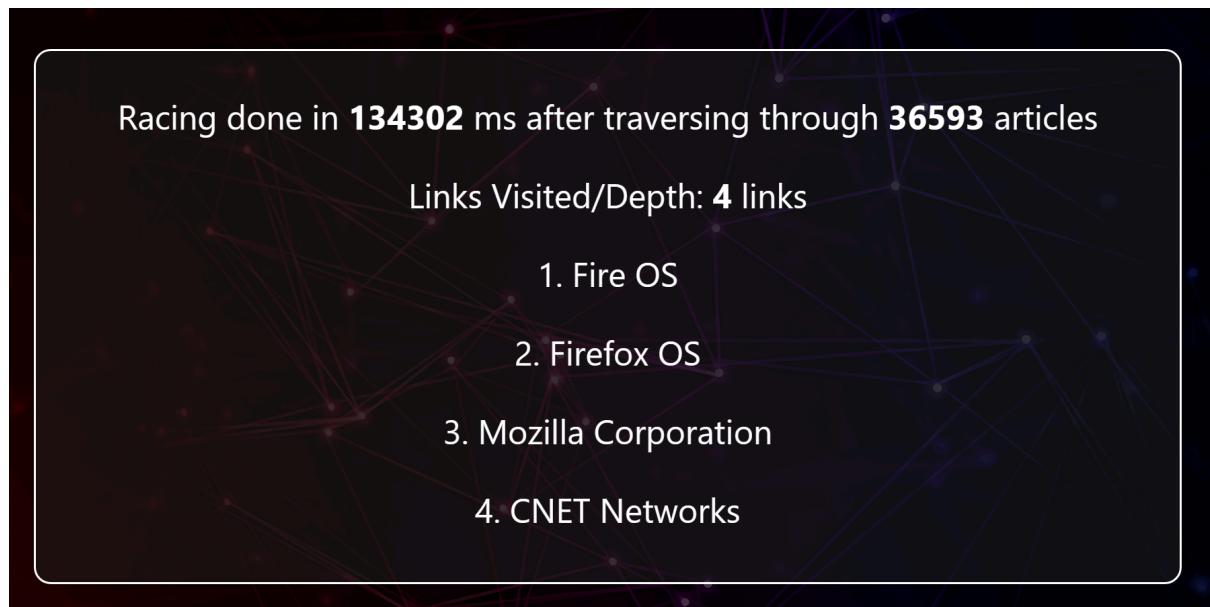
Komponen ini bertugas untuk memberikan pengguna tempat untuk memasukkan judul artikel Wikipedia yang akan dijadikan judul awal atau starting point dari penelusuran artikel Wikipedia.

4. Final Link

Komponen ini bertugas untuk memberikan pengguna tempat untuk memasukkan judul artikel Wikipedia yang akan dijadikan judul tujuan atau finish line dari penelusuran artikel Wikipedia.

5. Start Racing Button

Komponen ini bertugas untuk melaksanakan program yang ada di backend dan membawa semua informasi yang ada kepada sebuah server sehingga dari backend bisa diambil informasi itu agar bisa diproses oleh Backend.



Gambar 3.3.2 - Hasil Solusi terdekat dari Fire OS ke CNET Networks

6. Result

Komponen ini bertugas untuk memberikan output hasil penelusuran tautan Wikipedia baik secara BFS maupun secara IDS. Pada komponen ini, terdapat beberapa informasi yang didapat dari Backend.

a. Execution Time

Bagian ini adalah waktu yang diperlukan bagi program untuk menyelesaikan seluruh rangkaian program BFS atau IDS yang dilakukan di backend. Waktu diberikan dalam millisecond (ms).

b. Traversed Article Amount

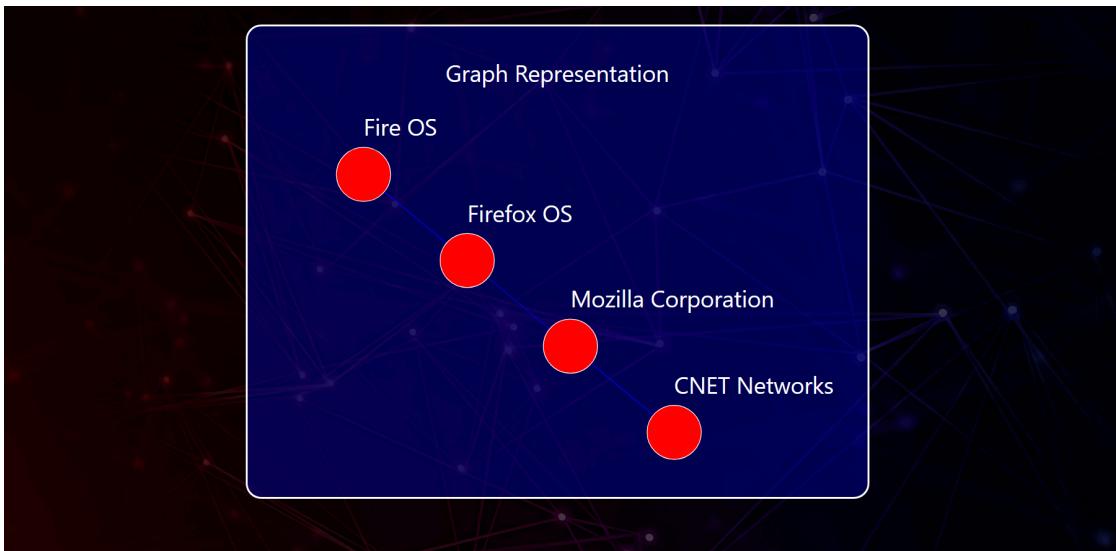
Bagian ini adalah jumlah dari artikel yang dilihat oleh program setelah melakukan scraping. Semakin lama program berjalan, biasanya artikel yang dilihat juga semakin banyak.

c. Link Depth

Bagian ini adalah kedalaman graf jika divisualisasikan. Semakin banyak kedalaman, biasanya artikel yang dilihat juga banyak sehingga program berjalan juga lebih lama.

d. Solutions

Bagian ini adalah jalan (path) yang ditempuh dari judul awal ke judul tujuan. Tulisan judul yang ada di komponen bisa diklik agar pengguna bisa pergi ke artikel Wikipedia tersebut.



Gambar 3.3.3 - Representasi Graf dari Solusi

7. Representasi Graf

Komponen ini bertugas untuk memberi representasi graf dari solusi yang ditempuh dari judul awal ke judul tujuan. Sama seperti graf biasanya, representasi ini memiliki node (simpul) dan link (garis/penghubung). Selain itu, graf ini bisa ditarik (graf) dan terdapat tulisan judul yang ditempuh oleh program.

3.4 Contoh Ilustrasi Kasus

Ini adalah ilustrasi kasus dari penggunaan web. (BFS without Namespace)

1. Masuk ke web

Untuk pengguna-pengguna biasa (bukan pengembang), biasanya akan memasukkan nama website ke aplikasi web seperti Microsoft Edge, Google Chrome, Firefox, dll. Namun untuk para developer yang belum deploy situs web-nya, biasanya akan menggunakan perintah berikut ini.

```
D:/Folder1/Folder2/Tubes2_Wikibowo/src> npm run start
```

```
D:/Folder1/Folder2/Tubes2_Wikibowo/src/backend> go run .
```

Setelah perintah ini dijalankan, pengguna akan langsung dibawa ke situs web yang diinginkan dan dengan menjalankan program backend, program sudah siap untuk mendeteksi masukkan.

2. Memilih opsi algoritma

Pada bagian web tersebut, sudah ada opsi untuk memilih algoritma BFS dan IDS. Memilih BFS artinya komponen toggle akan ada disebelah kiri dengan warna merah dan memilih IDS artinya komponen toggle akan ada disebelah kanan dengan warna biru.



Gambar 3.4.1 - Pengguna Memilih BFS



Gambar 3.4.2 - Pengguna Memilih IDS

3. Memilih opsi namespace (Misal pengguna menggunakan Without Namespace) Sama seperti opsi BFS dan IDS. Pengguna juga bisa memilih antara opsi With Namespace atau Without Namespace. Jika mengikuti aturan Wikirace, Namespace tidak boleh dipakai jadi lebih sering akan dipakai tanpa namespace. Misal pengguna menggunakan Without Namespace, tombol toggle Namespace akan menjadi warna merah.



Gambar 3.4.3 - Pengguna Memilih Without Namespaces



Gambar 3.4.4 - Pengguna Memilih With Namespaces

4. Memasukkan judul awal Pengguna bisa memasukkan judul artikel Wikipedia dalam bentuk teks. Ini akan menjadi starting point dari penelusuran Wikirace saat informasi ini dibawa ke backend.



Gambar 3.4.5 - Pengguna Memasukkan Judul Awal “Fire OS”

5. Memasukkan judul tujuan Pengguna bisa memasukkan judul artikel Wikipedia dalam bentuk teks. Ini akan menjadi finish line dari penelusuran Wikirace saat informasi ini dibawa ke backend.



Gambar 3.4.6 - Pengguna Memasukkan Judul Tujuan “CNET Networks”

6. Menekan tombol “Start Racing”

Dengan menekan tombol “Start Racing”, program akan memulai untuk mengambil informasi di Backend dan melakukan Scraping, BFS atau IDS, dan pembuatan list dari solusi.



Gambar 3.4.7 - Tombol “Start Racing”

7. Melihat hasil dalam bentuk tulisan

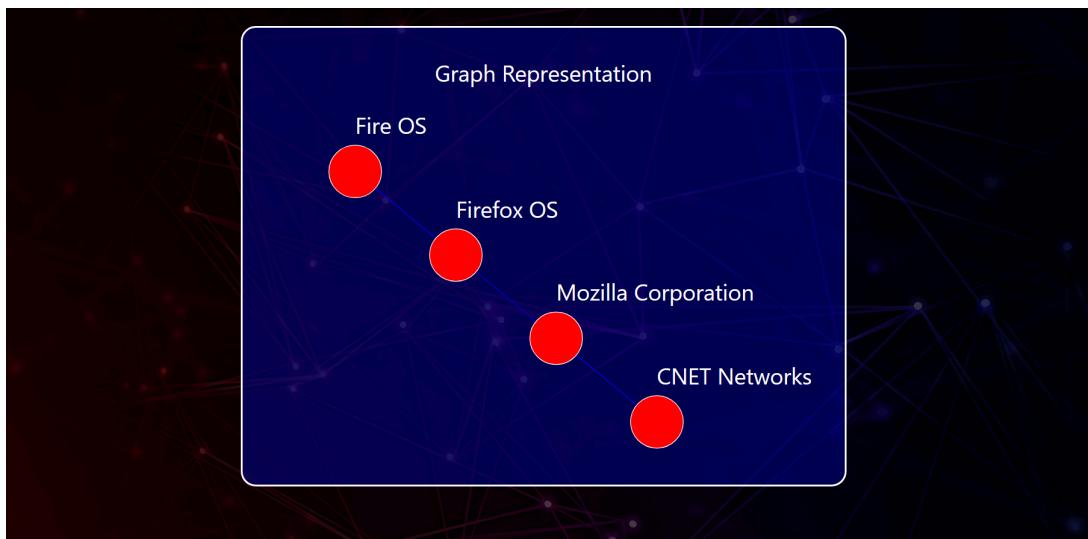
Hasil dengan bentuk tulisan akan muncul setelah pengguna menunggu setelah beberapa saat.



Gambar 3.4.8 - Hasil dalam bentuk tulisan

8. Melihat hasil dalam bentuk graf

Selain tulisan, pengguna juga bisa melihat hasil dalam bentuk graf. Beginilah hasil dengan menggunakan graf.



BAB IV

Eksperimen

4.1 Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun)

1. Struktur data

Pada program ini, penentuan struktur data akan dibagi menjadi dua yaitu struktur data front-end dan struktur data back-end.

a. Frontend

i. linkValue : {startLink : string, endLink : string}

Struktur data ini berguna untuk menyimpan informasi judul awal dan judul tujuan yang akan ditransfer ke Backend.

ii. isOn : boolean

Struktur data ini berguna untuk menyimpan informasi opsi BFS atau IDS yang akan ditransfer ke Backend.

iii. isName : boolean

Struktur data ini berguna untuk menyimpan informasi opsi Namespace yang akan ditransfer ke Backend.

iv. open : boolean

Struktur data ini berguna untuk membantu pemunculan komponen ketika diinginkan

v. resultResponse : {exec : int, len : int, urls : string[], resultLink : string[]}

Struktur data ini berguna untuk mengambil informasi backend yang berupa waktu eksekusi, panjang artikel, daftar url solusi, dan daftar judul artikel solusi ke frontend.

b. Backend

i. LinkInfo : Menyimpan info data yang diambil dari frontend

1. LinkValue : string
2. FinValue : string
3. IsOn : bool
4. IsName : bool

ii. Engine : Menyimpan info yang dibutuhkan oleh program agar berjalan

1. Start : string
2. End : string
3. Depth : int
4. Namespace : bool

5. Cache : map[string]bool

2. Fungsi

1. Fungsi membuat engine

```
function createEngine(emptyEngine, start, end, depth, namespace)
    // Update the properties of the empty engine object
    emptyEngine.Start = start
    emptyEngine.End = end
    emptyEngine.Depth = depth
    emptyEngine.Namespace = namespace
    emptyEngine.Cache = createEmptyMap() // Initialize an empty map
    return emptyEngine
```

2. Fungsi membuat elemen

```
function findElement(data, end)
    for i from length(data) - 1 to 0 step -1
        if data[i][length(data[i]) - 1] equals end
            return i
    return -1
```

3. Fungsi mengubah judul artikel menjadi url Wikipedia

```
function TurnToWikipedia(title)
    tokens = splitString(title, " ")
    temp = ""
    for i from 0 to length(tokens) - 1
        temp = temp + tokens[i]
        if i less than length(tokens) - 1
            temp = temp + "-"
    temp = "https://en.wikipedia.org/wiki/" + temp
    if contains(temp, "-")
        temp = replace(temp, "-", "%E2%80%93")
    return temp
```

4. Fungsi mengubah url Wikipedia menjadi judul artikel

```
function TurnToTitle(url)
    url = substring(url, 30, length(url) - 0)
    temp = ""
    tokens = splitString(url, "_")
    for i from 0 to length(tokens) - 1
        temp = temp + tokens[i]
        if i less than length(tokens) - 1
            temp = temp + " "
    if contains(temp, "%E2%80%93")
        temp = replace(temp, "%E2%80%93", "-")
    print(temp)
    return temp
```

3. Prosedur

1. Prosedur Scraping BFS

```
procedure Scrape(lenc[], eng, ch, wg, sem):
    try
        links = []
        c = NewCollector()

        // Set up filters
        c.DisallowedURLFilters(
            "Category:",
            "Wikipedia:",
            "Special:",
            "File:",
            "Help:",
            "Talk:",
            "Portal:",
            "Template:",
            "Template_talk:",
            "Main_Page"
        )
        c.URLFilters("en.wikipedia.org/wiki")

        // Set User-Agent header
        c.OnRequest(function (r)
            r.Headers.Set("User-Agent", "Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0
Safari/537.36")
        )

        c.OnResponse(function (r)
            // No action required for response

            c.OnHTML("a[href]", function (e)
                linkURL = e.Attr("href")

                if not contains(linkURL, "https://en.wikipedia.org"):
                    linkURL = "https://en.wikipedia.org" + linkURL

                // Check if the link satisfies namespace conditions
                if not eng.Namespace or (
                    eng.Namespace and not contains(linkURL, "Category:")
                    and not contains(linkURL, "Wikipedia:")
                    and not contains(linkURL, "Special:")
                    and not contains(linkURL, "File:")
                    and not contains(linkURL, "Help:")
                    and not contains(linkURL, "Talk:")
                    and not contains(linkURL, "Portal:")
                    and not contains(linkURL, "Template:")
                    and not contains(linkURL, "Template_talk:")
                    and not contains(linkURL, "Main_Page"))
                )
                    linkToAppend = copyOf(lenc)
                    if (not eng.Cache[linkURL] or eng.End == linkURL) and
                    eng.Start != linkURL
                        eng.Cache[linkURL] = true
                        linkToAppend.append(linkURL)
                    else
                        return

                    if length(linkToAppend) == eng.Depth + 1
                        links.append(linkToAppend)
```

```

c.Visit(lenc[length(lenc) - 1])

// Signal completion using semaphores and update wait group
sem.acquire()
wg.Add(1)
ch <- links
sem.release()
wg.Done()

// Handle panics
except Panic as p
    print("Recovered from panic in Scrape:", p)

```

2. BFS

```

procedure BFS(i, start, end, listScrape, eng, ch, wg, sem):
    defer wg.Done()

    var tempScrape[][]

    if ((i >= 0 and length(listScrape[i]) == eng.Depth + 1) or i == -1)
        eng.Depth++
        print("Layer ", eng.Depth)

    if (i == -1) // listScrape is empty
        chScrape = makeChannel()
        go Scrape([start], eng, chScrape, wg, sem)
        tempScrape = <-chScrape
        append(tempScrape, listScrape)

    else
        chScrape = makeChannel()
        go Scrape(listScrape[i], eng, chScrape, wg, sem)
        tempScrape = <-chScrape
        append(tempScrape, listScrape)

    idxTemp = findElement(tempScrape, end)

    if (idxTemp != -1)
        wg.Add(1)
        print("length : ", length(listScrape))
        ch <- tempScrape[idxTemp]
        return

    sem.acquire()
    defer sem.release()
    wg.Add(1)
    concur BFS(i + 1, start, end, listScrape, eng, ch, wg, sem)

```

3. Scraping versi IDS

```

function getLinks(lenc):
    // Initialize an empty list to store links
    links = []

```

```

// Initialize a new web scraper
scraper = createWebScraper()

// Define disallowed URL patterns
disallowedPatterns = [
    "Category:",
    "Wikipedia:",
    "Special:",
    "File:",
    "Help:",
    "Talk:",
    "Portal:",
    "Template:",
    "Template_talk:",
    "Main_Page"
]

// Define URL filter pattern
urlFilterPattern = "en.wikipedia.org/wiki"

// Configure scraper settings
configureScraper(scraper, disallowedPatterns, urlFilterPattern)

// Set user agent for the scraper
setUserAgent(scraper)

// When a request is made
onRequest(scraper):
    // Set user agent header

// When a response is received
onResponse(scraper):
    // Do nothing with the response

// When an anchor tag is encountered in HTML
onAnchorTag(scraper, link):
    // Extract the href attribute from the anchor tag
    href = extractHref(link)

    // If the href does not contain the full Wikipedia URL,
    // prepend it
    if not containsFullWikiURL(href):
        href = prependWikiURL(href)

    // If the href contains the Wikipedia article pattern
    if containsWikiArticlePattern(href):
        // Add the link to the list of links
        addLink(links, href)

// Visit the provided URL with the scraper
scrapeURL(scraper, lenc)

// Return the list of extracted links
return links

```

4. DLS

```
function DLS(graph, source, target, limit):
```

```

// If the depth limit is reached and the source equals the target
if limit == 0 and source == target:
    return [source], true
// If the depth limit is reached but the source is not the target
else if limit == 0 and source != target:
    return null, false

// If the source node doesn't exist in the graph
if source not in graph:
    // Get the children (links) of the source node
    children = getLinks(source)
    // Add the source node to the graph with its children
    graph.addNode(source, children)

// For each child of the source node
for each child in graph[source].children:
    // If the child node is the target
    if child == target:
        return [source, target], true

    // If the child node is not the source
    if child != source:
        // Perform Depth-Limited Search recursively with the
        child node
        path, found = DLS(graph, child, target, limit - 1)
        // If the target is found in the child's subtree
        if found:
            // Prepend the source node to the path and return
            return [source] + path, true

// If the target is not found within the depth limit
return null, false

```

5. IDS

```

function IDS(graph, source, target):
    depth = 0
    path = []
    found = false

    // Iterate until the target is found or the entire graph is
    searched
    while true:
        print("Depth:", depth)
        // Perform Depth-Limited Search with current depth limit
        path, found = DLS(graph, source, target, depth)
        // If the target is found within the current depth limit
        if found:
            return path, true
        // Increment the depth limit for the next iteration
        depth++

```

4.2 Penjelasan tata cara penggunaan program

1. Jalankan Frontend

Pergi ke direktori source code lalu jalankan perintah ini

```
D:/> cd Folder1/Folder2/Tubes/src  
D:/Folder1/Folder2/Tubes/src> npm run start
```

Ini akan membawa pengguna ke laman website yang diinginkan.

2. Jalankan Backend

Pergi ke direktori source code backend lalu jalankan perintah ini

```
D:/> cd Folder1/Folder2/Tubes/src/backend  
D:/Folder1/Folder2/Tubes/src/backend> go run .
```

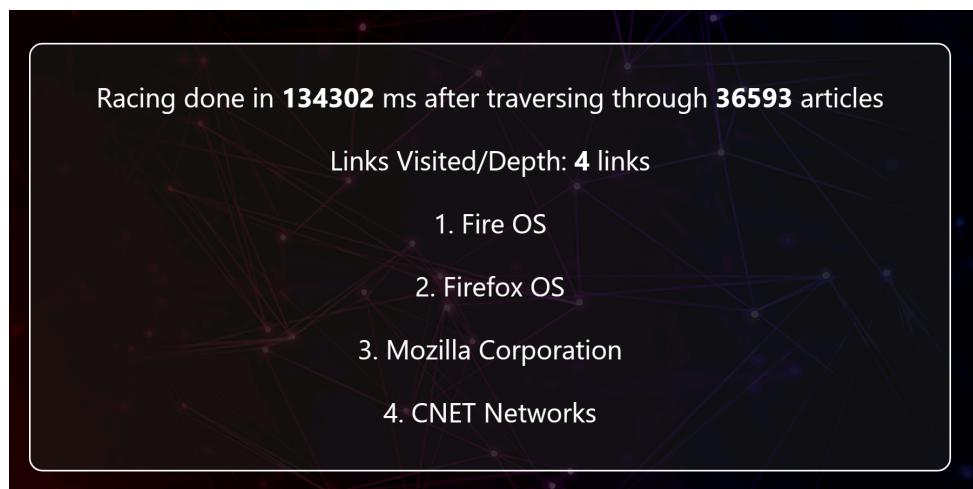
Ketika ada tulisan “Server is listening on Port 8000...” dan perintah untuk “Allow Network” tekan “OK” agar backend siap untuk mengambil informasi dari Frontend.

3. Input

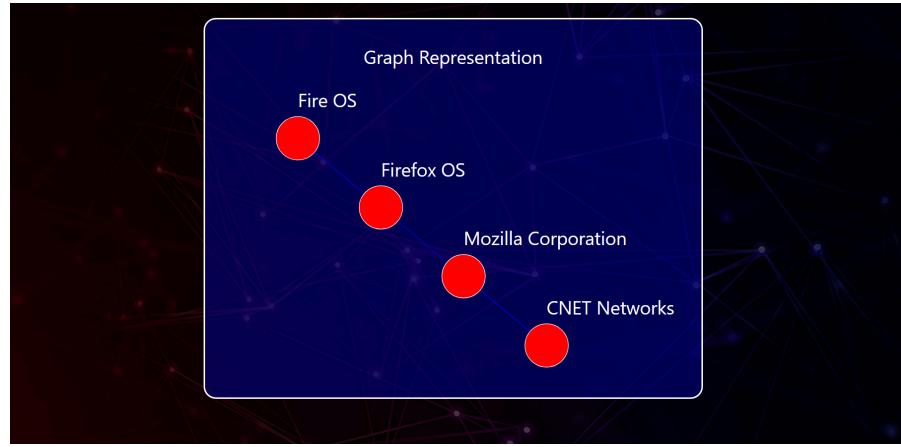
Masukkan semua informasi yang dibutuhkan dan pilihlah opsi algoritma dan opsi namespace. Tekan tombol “Start Racing” dan tunggu beberapa saat untuk mendapatkan hasil.

4. Pemunculan Hasil

Hasil akan muncul beserta informasi waktu eksekusi dan tautan yang dikunjungi. Tidak hanya hasil, graf juga akan masuk.



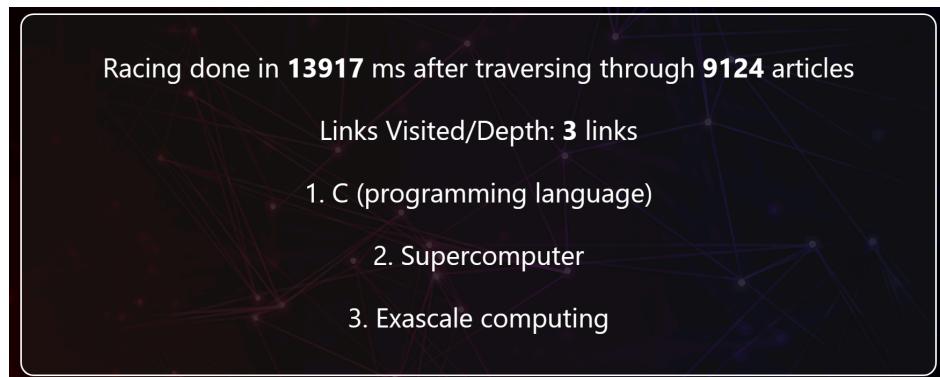
Gambar 4.2.1 - Hasil List Tautan



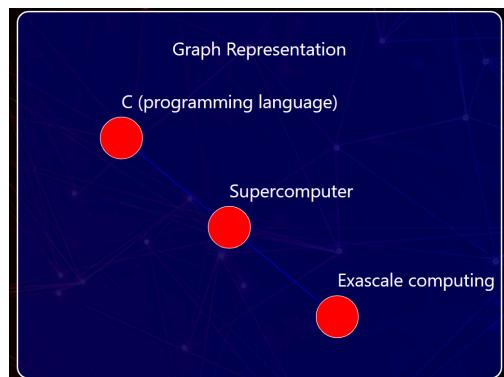
Gambar 4.2.2 - Representasi Graf

4.3 Hasil pengujian

1. C (Programming Language) -> Exascale computing
 - a. BFS

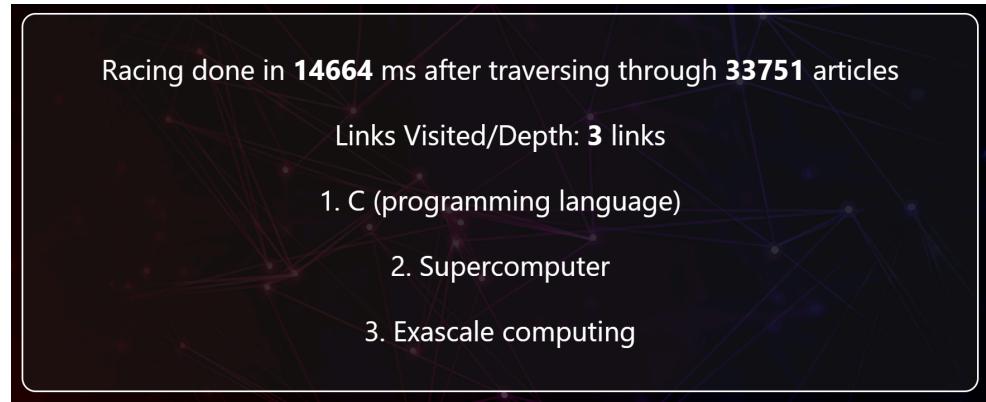


Gambar 4.3.1 - Hasil Daftar Tautan C (Programming language) -> Exascale computing

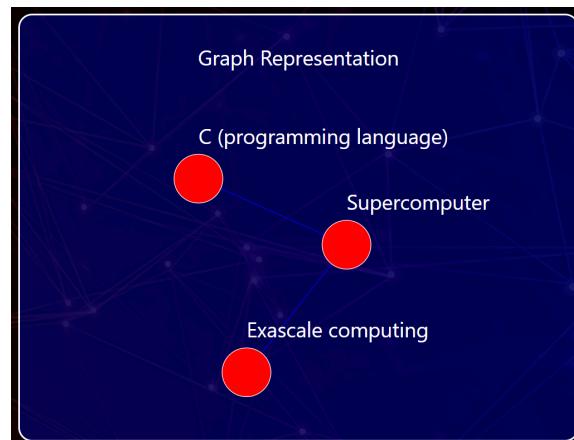


Gambar 4.3.2 - Representasi Graf Solusi C (Programming Language)-> Exascale computing

b. IDS

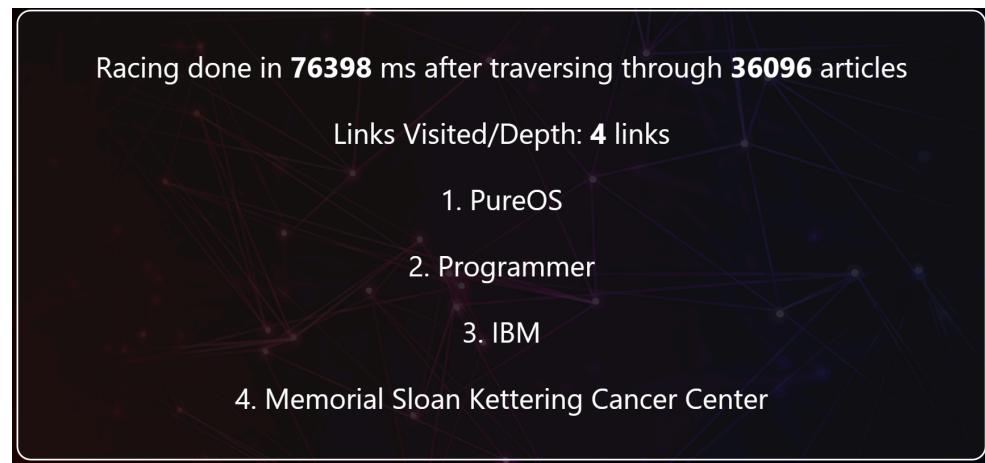


Gambar 4.3.3 - Hasil Daftar Tautan C (Programming language) -> Exascale computing

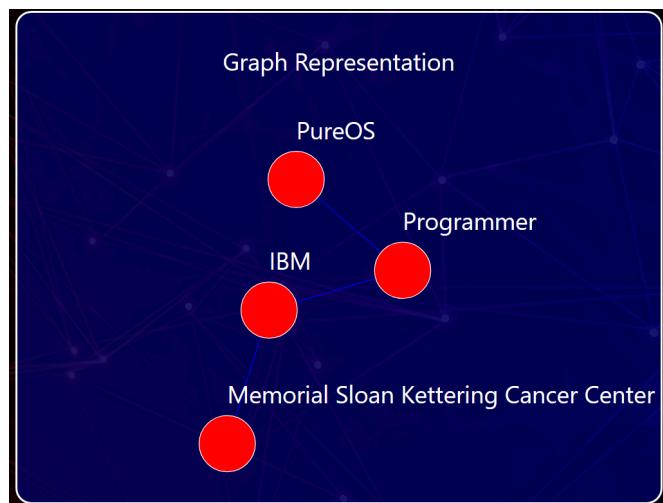


Gambar 4.3.4 - Representasi Graf Solusi C (Programming Language)-> Exascale computing

2. PureOS -> Memorial Sloan Kettering Cancer Center
 - a. BFS

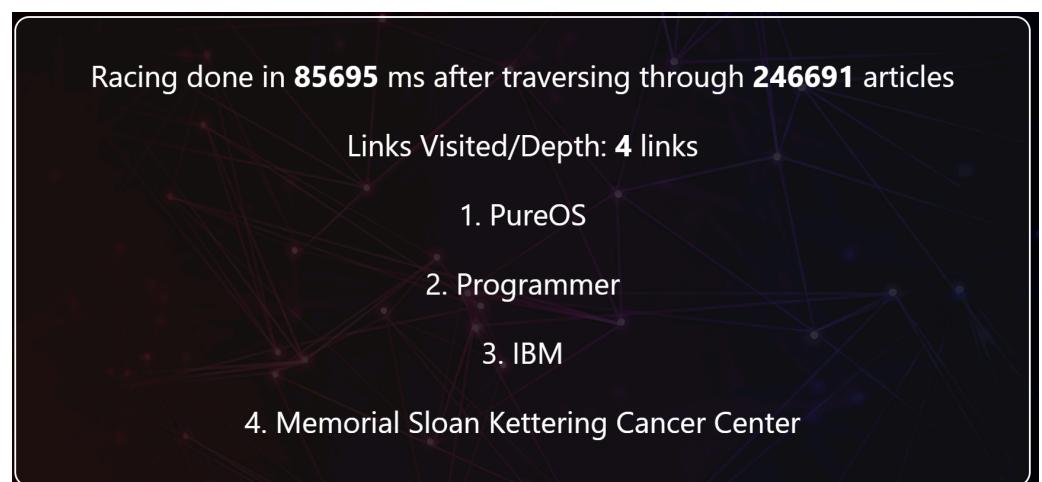


Gambar 4.3.5 - Hasil Daftar Tautan PureOS -> Memorial Sloan Kettering Cancer Center

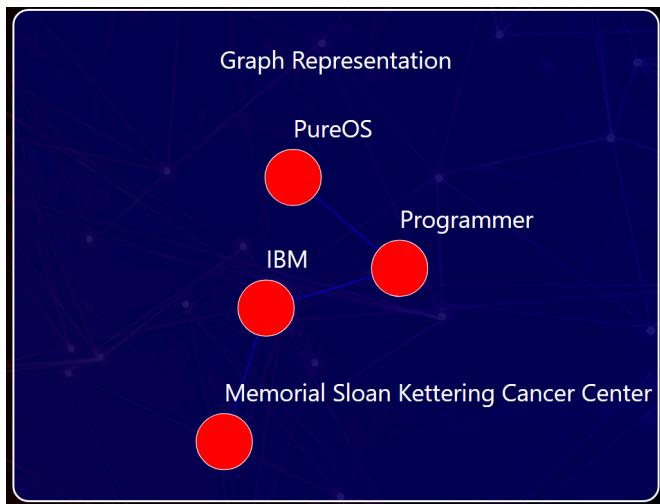


Gambar 4.3.6 - Representasi Graf Solusi PureOS -> Memorial Sloan Kettering Cancer Center

b. IDS



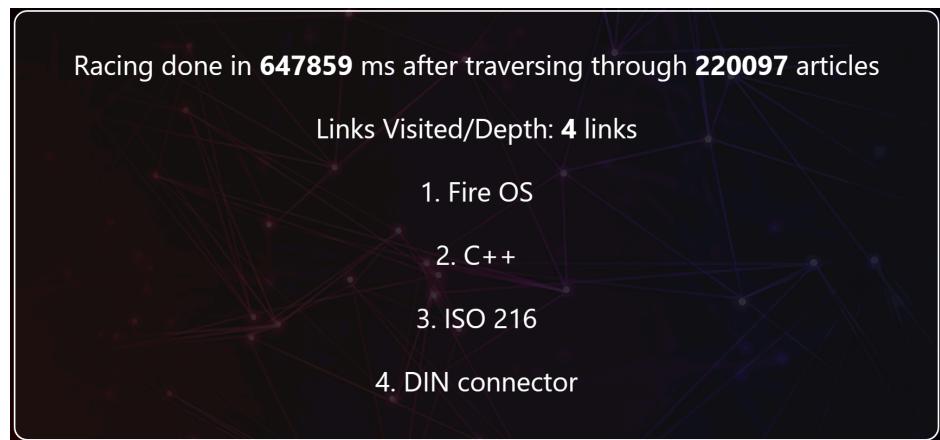
Gambar 4.3.7 - Hasil Daftar Tautan PureOS -> Memorial Sloan Kettering Cancer Center



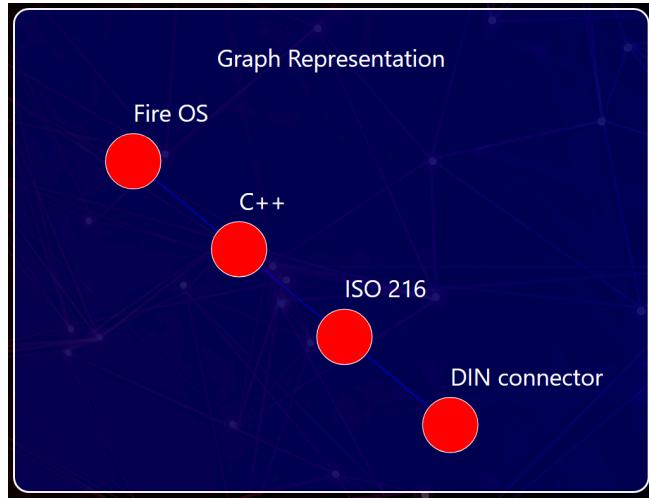
Gambar 4.3.8 - Representasi Graf Solusi PureOS -> Memorial Sloan Kettering Cancer Center

3. Fire OS -> DIN Connector

- a. BFS

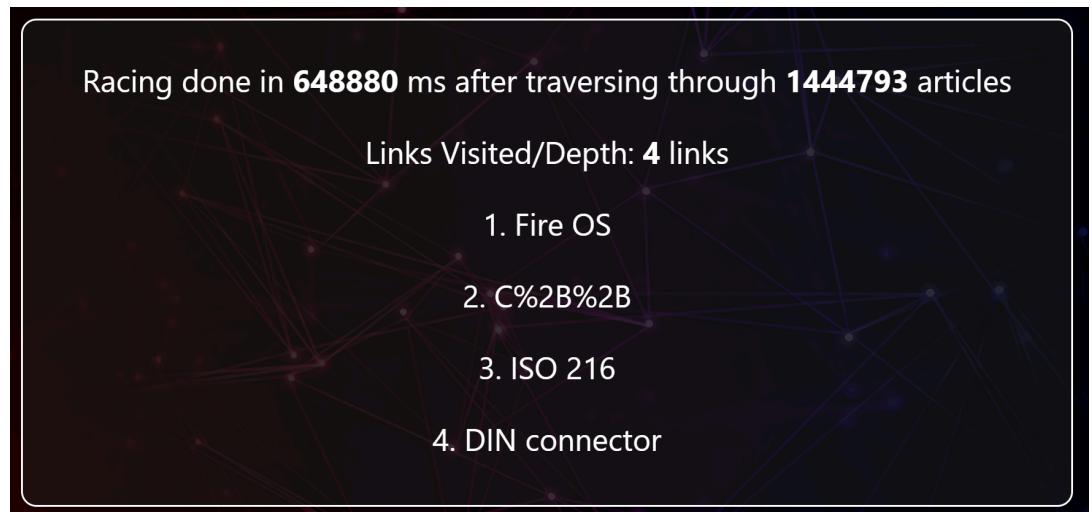


Gambar 4.3.9 - Hasil Daftar Tautan Fire OS -> DIN connector

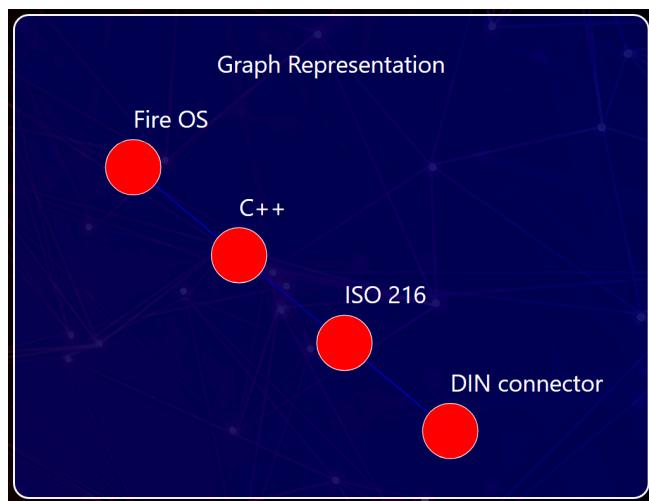


Gambar 4.3.10 - Representasi Graf Fire OS -> DIN connector

b. IDS



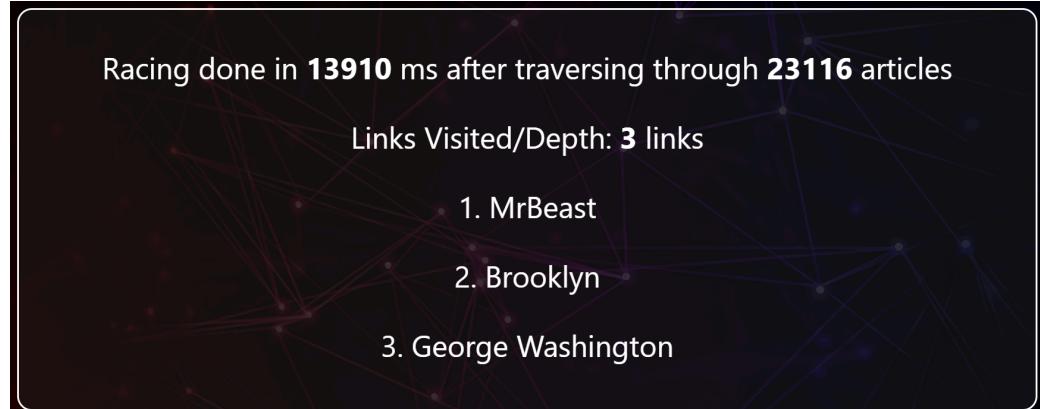
Gambar 4.3.11 - Hasil Daftar Tautan Fire OS -> DIN connector



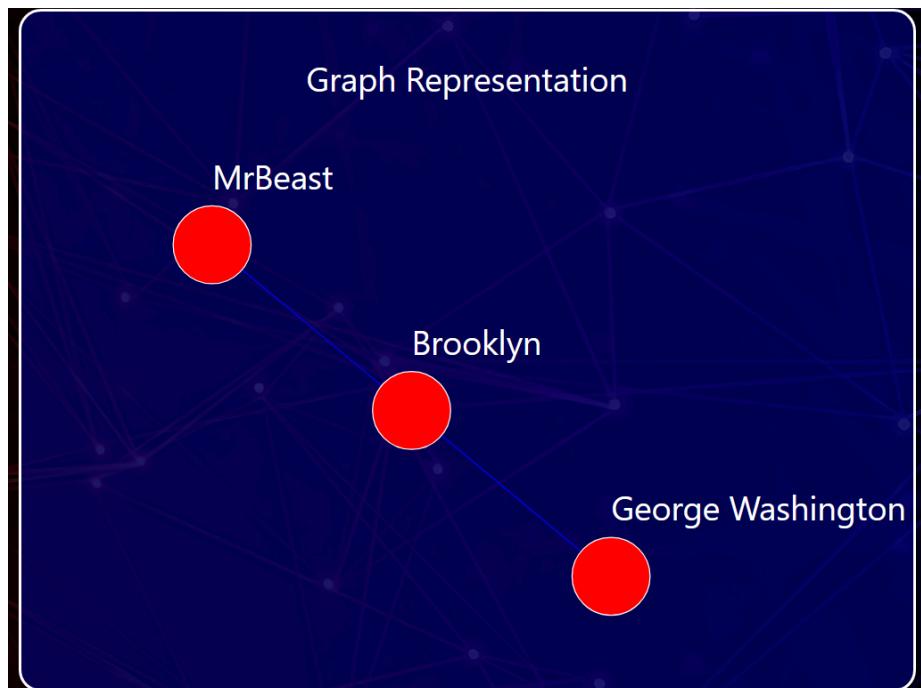
Gambar 4.3.12 - Representasi Graf Fire OS -> DIN connector

4. MrBeast -> George Washington

a. BFS

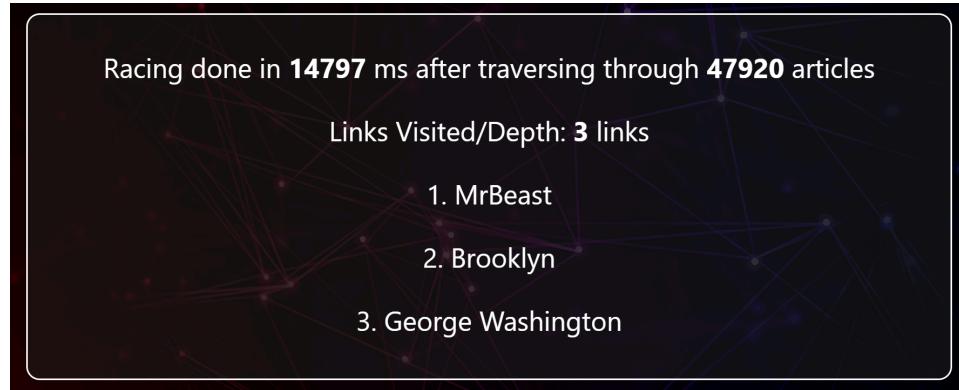


Gambar 4.3.13 - Representasi Graf MrBeast -> George Washington

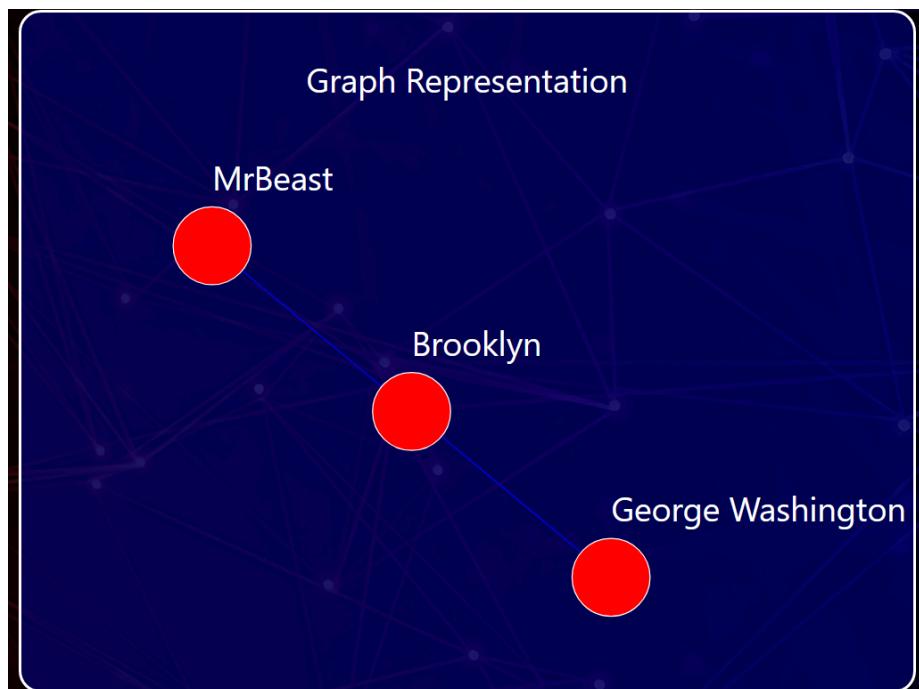


Gambar 4.3.14 - Representasi Graf MrBeast -> George Washington

b. IDS



Gambar 4.3.13 - Representasi Graf MrBeast -> George Washington



Gambar 4.3.14 - Representasi Graf MrBeast -> George Washington

4.4 Analisis hasil pengujian

Berdasarkan 4 kali tes masing masing menggunakan algoritma BFS dan IDS. Kami menyimpulkan bahwa menggunakan algoritma BFS dan IDS bisa menjadi cara yang efektif untuk menyelesaikan masalah pencarian solusi terpendek dari WikiRace. Namun berdasarkan hasil, BFS adalah cara yang relatif lebih cepat untuk mencari solusi terpendek dari permainan WikiRace.

BFS yang relatif lebih cepat mencari solusi permainan WikiRace disebabkan karena visualisasi graf yang relatif lebih panjang sehingga pencarian dengan memprioritaskan pencarian mendalam seperti IDS membutuhkan waktu yang relatif lebih lama. Namun, untuk beberapa kasus tetapi terdapat beberapa instansi IDS membutuhkan waktu eksekusi yang lebih

cepat. Ini bisa terjadi jika saat pencarian mendalam, solusinya langsung ditemukan (Jika divisualisasikan ke dalam graf, solusinya terdapat di sebelah kiri).

BAB V

Analisis

5.1 Kesimpulan

Tugas Besar 2 IF2211 Strategi Algoritma adalah tugas untuk mencari solusi terdekat dari sebuah permainan yang bernama WikiRace. Untuk menyelesaikan permasalahan ini, algoritma penelusuran graf seperti BFS dan IDS bisa menjadi algoritma yang cocok untuk diimplementasikan ke permasalahan tersebut. Aplikasi pencarian solusi WikiRace dikembangkan berbasis web. Frontend yang digunakan adalah React.js sedangkan Backend yang digunakan adalah Go. Aplikasi web menerima masukan berupa judul awal, judul tujuan, opsi BFS dan IDS, opsi Namespace, dan tombol mulai.

Dari semua yang kami kerjakan di Tugas Besar ini, kami bisa menyimpulkan bahwa algoritma BFS dan IDS bisa digunakan untuk menyelesaikan pencarian solusi WikiRace. BFS mencari tautan dengan menelusuri tautan per kedalaman sampai semua tautan di kedalaman tersebut sudah habis atau solusinya sudah ditemukan. IDS mencari tautan dengan melakukan DFS setiap kedalaman yang diinginkan.

5.2 Saran

Tugas Besar 2 IF2211 Strategi Algoritma Semester II menjadi pelajaran yang baik bagi seluruh anggota kelompok karena ini Tugas Besar pertama yang menggunakan bahasa Go dan Tugas Besar kedua (Dari semua tugas besar IF) yang wajib merancang web. Berdasarkan pengalaman anggota kelompok, kami menyarankan beberapa hal ini untuk para pembaca.

1. Pengerjaan bisa menggunakan real-time collaboration seperti Visual Studio Code Live Share Extension. Hal ini bisa membuat para pengerja bisa mengatur para pengerja lainnya dalam pengerjaan tugas
2. Pastikan mempunyai versi Go yang sama agar saat melakukan pulling dari Github, tidak ada conflict
3. Pastikan juga aplikasi bisa dijalankan di Windows dan di Linux, sehingga disarankan memasang WSL bagi yang tidak menggunakan Linux

5.3 Refleksi

Ada beberapa hal yang kami sarankan untuk pengerjaan Tugas Besar selanjutnya agar pengerjaan menjadi lebih optimal.

1. Mengatur waktu agar kelompok bisa lebih sering bertemu secara sinkron karena pertemuan secara sinkron hanya diadakan saat hari-H
2. Lebih sering membaca dokumentasi terkait Web Development agar lebih cepat dan mengerti terkait bagaimana cara merancang web sesuai spesifikasi

3. Menyempatkan waktu untuk mengerjakan tugas besar meskipun terdapat hari libur dan tugas besar lain untuk meringankan pekerjaan kelompok saat hari-hari penggerjaan tugas besar yang efektif.

DAFTAR PUSTAKA

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

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

LAMPIRAN

Link Repository:

[scifo04/Tubes2_Wikibowo \(github.com\)](https://github.com/scifo04/Tubes2_Wikibowo)