

LAPORAN TUGAS BESAR 2

IF2211 Strategi Algoritma

Pemanfaatan Algoritma IDS dan BFS dalam Permainan WikiRace



Disusun oleh

Shafiq Irvansyah : 13522003

Ahmad Naufal Ramadan : 13522005

Yusuf Ardian Sandi : 13522015

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

Daftar Isi

Bab 1 Deskripsi Tugas.....	4
Bab 2 Landasan Teori.....	5
2.1. Penjelasan mengenai Website.....	5
2.2. Breadth First Search (BFS).....	6
2.3. Depth First Search (DFS).....	7
2.4. Iterative Deepening Search (IDS).....	8
Bab 3 Analisis Pemecahan Masalah.....	10
3.1. Langkah-Langkah Pemecahan Masalah.....	10
3.2. Proses Pemetaan Masalah Menjadi Elemen-Elemen Algoritma IDS dan BFS.....	11
3.3. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun.....	12
3.4. Contoh Ilustrasi Kasus.....	15
Bab 4 Implementasi dan Pengujian.....	18
4.1. Spesifikasi Teknis Program.....	18
a. Struktur Data.....	18
i. Struct SafeMap.....	18
ii. Struct SafeArray.....	18
iii. Struct Node.....	18
iv. Struct linkJson.....	18
b. Fungsi dan Prosedur.....	18
i. BFS.....	18
ii. IDS.....	23
4.2. Tata Cara Penggunaan Program.....	29
4.3. Hasil Pengujian.....	30
a. BFS (Breadth First Search).....	30
i. One Path.....	30
1. Depth 1.....	30
2. Depth 2.....	33
3. Depth 3.....	36
4. Depth 4.....	39
ii. More Than One Path.....	40
1. Depth 1.....	40
2. Depth 2.....	43
3. Depth 3.....	46
4. Depth 4.....	49
b. IDS (Iterative Deepening Search).....	51
i. One Path.....	51
1. Depth 1.....	51

2. Depth 2.....	54
3. Depth 3.....	57
4. Depth 4.....	60
ii. More Than One Path.....	61
1. Depth 1.....	61
2. Depth 2.....	64
3. Depth 3.....	67
4. Depth 4.....	70
4.4. Analisis Hasil Pengujian.....	72
Bab 5 Kesimpulan, Saran, dan Refleksi.....	73
5.1. Kesimpulan.....	73
5.2. Saran.....	73
5.3. Refleksi.....	73
Lampiran.....	73
Daftar Pustaka.....	75

Bab 1

Deskripsi Tugas

WikiRace atau Wiki Game adalah permainan yang melibatkan Wikipedia, sebuah ensiklopedia online gratis yang dikelola oleh berbagai sukarelawan di seluruh dunia, di mana pemain memulai dari sebuah artikel Wikipedia dan harus menelusuri artikel lain di Wikipedia (dengan mengklik tautan di dalam setiap artikel) untuk mencapai tujuan yang telah ditentukan. Artikel dalam waktu sesingkat-singkatnya atau dengan klik paling sedikit (artikel)

Bab 2 Landasan Teori

2.1. Penjelasan mengenai Website

Website ini bertujuan untuk mencari rute artikel terpendek dari satu artikel ke artikel lainnya. Setelah User memasukkan judul artikel awal dan judul artikel tujuan dan menge-klik tombol “Find!”, akan ditampilkan:

- Jumlah artikel yang diperiksa
- Jumlah artikel yang lolos
- Rute eksplorasi artikel
- Waktu pencarian, dan
- Visualisasi grafik eksplorasi rute

User dapat memilih metode pencarian dengan IDS (Iterative Deepening Search) atau BFS (Breadth First Search). IDS akan melakukan pencarian dengan meningkatkan nilai depth-cutoff menggunakan rangkaian DFS (Depth First Search) hingga ditemukan solusi. Dalam mencari suatu node dalam suatu graf, DFS akan melakukan pencarian dengan cara memperluas root child pertama dari pohon pencarian yang dipilih dan masuk lebih dalam lagi hingga node target ditemukan, atau hingga menemukan node yang tidak memiliki anak. Sementara itu, BFS akan memulai pencarian grafik dari node akar dan kemudian menjelajahi semua node tetangganya. Kemudian setiap node terdekat tersebut menelusuri node tetangga yang belum diperiksa, begitu seterusnya hingga node target ditemukan.

Selain itu, User juga dapat mengatur batasan dalam pencarian rute eksplorasi artikel sehingga rute terpendek yang ditampilkan tidak hanya satu, tetapi semua rute terpendek dari hasil pencarian juga akan ditampilkan.

Dalam mengembangkan *front-end* web ini, kami menggunakan library JavaScript yang bernama React. Salah satu keunggulan utama React adalah kemampuannya untuk membuat komponen yang dapat digunakan kembali. Komponen dalam React adalah bagian-bagian kecil dari antarmuka pengguna yang dapat dikembangkan, dipelihara, dan diuji secara terpisah. Ini memungkinkan kami untuk membangun situs web yang kompleks dengan membagi-bagi fungsi-fungsi ke dalam komponen-komponen yang lebih kecil, yang memudahkan pengelolaan kode dan pengembangan lebih lanjut.

Sementara itu, pada *back-end* kami menggunakan bahasa programming Go dengan *framework* Gin. Gin adalah sebuah framework web yang terkenal karena keunggulannya yang ringan dan cepat. Dengan ukuran yang kecil dan overhead yang minimal, Gin memungkinkan penggunaan sumber daya yang efisien pada server, memungkinkan

aplikasi untuk berjalan dengan lancar bahkan pada infrastruktur yang terbatas. Selain itu, Gin menggunakan routing yang sangat cepat dan efisien, sehingga memungkinkan aplikasi untuk menanggapi permintaan HTTP dengan cepat.

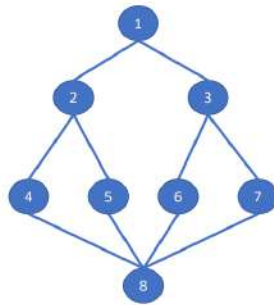
2.2. Breadth First Search (BFS)

Breadth-first search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder, yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang telah dikunjungi, demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada kedalaman d dikunjungi lebih dahulu sebelum simpul-simpul pada kedalaman $d+1$.

Algoritma ini menggunakan sebuah *queue*, misalkan q , untuk menyimpan simpul yang telah dikunjungi (*visited node*). Simpul-simpul tersebut akan digunakan sebagai acuan untuk mengunjungi simpul-simpul lain yang bertetangga dengan simpul-simpul tersebut. Untuk mengetahui apakah suatu simpul bertetangga atau tidak, diperlukan matriks ketetanggaan untuk mengetahuinya, misal simpul i dan j dikatakan bertetangga jika matriks ketetanggaan $M[i,j] = 1$, dan tidak bertetangga jika $M[i,j] = 0$. Tiap simpul yang telah dikunjungi hanya masuk ke dalam antrian sebanyak satu kali. BFS juga memerlukan tabel *boolean* untuk menyimpan simpul yang telah dikunjungi, misal simpul i dikatakan telah dikunjungi jika $A[i] = true$ dan dikatakan belum dikunjungi jika $A[i] = false$.

Langkah Kerja Algoritma BFS:

1. Masukkan simpul akar ke dalam antrian (q).
2. Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi.
3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian.
5. Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan.
6. Ulangi pencarian dari langkah kedua.



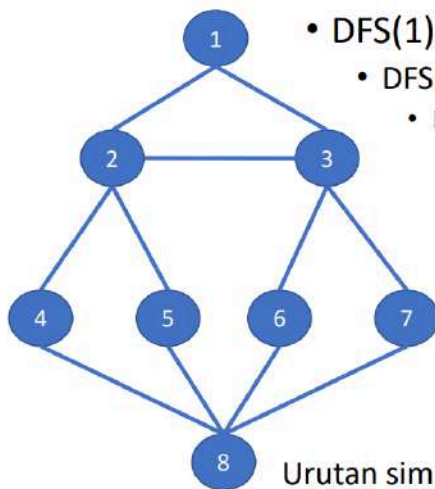
Iterasi	v	Q	dikunjungi							
			1	2	3	4	5	6	7	8
Inisialisasi	1	{1}	T	F	F	F	F	F	F	F
Iterasi 1	1	{2,3}	T	T	T	F	F	F	F	F
Iterasi 2	2	{3,4,5}	T	T	T	T	T	F	F	F
Iterasi 3	3	{4,5,6,7}	T	T	T	T	T	T	T	F
Iterasi 4	4	{5,6,7,8}	T	T	T	T	T	T	T	T
Iterasi 5	5	{6,7,8}	T	T	T	T	T	T	T	T
Iterasi 6	6	{7,8}	T	T	T	T	T	T	T	T
Iterasi 7	7	{8}	T	T	T	T	T	T	T	T
Iterasi 8	8	{}	T	T	T	T	T	T	T	T

Urutan simpul2 yang dikunjungi: 1, 2, 3, 4, 5, 6, 7, 8

Gambar 2.2.1 Contoh pencarian algoritma BFS

2.3. Depth First Search (DFS)

DFS merupakan metode pencarian yang dimulai dengan memperluas simpul pertama dari pohon pencarian yang dipilih, kemudian terus berlanjut ke dalam dan lebih dalam lagi hingga simpul tujuan ditemukan, atau sampai mencapai simpul yang tidak memiliki anak. Setelah itu, metode pencarian akan melakukan backtracking, yaitu kembali ke simpul yang belum selesai ditelusuri. Dalam implementasi non rekursif, semua simpul yang diperluas akan ditambahkan ke dalam tumpukan LIFO untuk penelusuran. DFS pada graf dengan n simpul dan m sisi membutuhkan waktu $O(n + m)$.



- DFS(1): v=1; dikunjungi[1]=true; DFS(2)
- DFS(2): v=2; dikunjungi[2]=true; DFS(3)
- DFS(3): v=3; dikunjungi[3]=true; DFS(6)
- DFS(6): v=6; dikunjungi[6]=true; DFS(8)
 - DFS(8): v=8; dikunjungi[8]=true; DFS(4)
 - DFS(4): v=4; dikunjungi[4]=true; DFS(8): DFS(5)
 - DFS(5): v=5; dikunjungi[5]=true; DFS(8): DFS(7)
 - DFS(7): v=7; dikunjungi[7]=true

Urutan simpul2 yang dikunjungi: 1, 2, 3, 6, 8, 4, 5, 7

Gambar 2.3.1 Contoh pencarian algoritma DFS

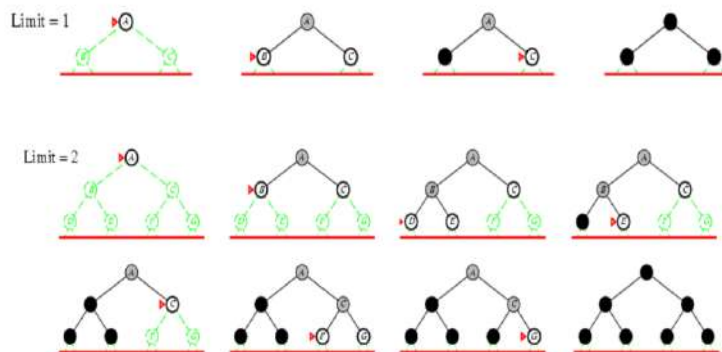
2.4. Iterative Deepening Search (IDS)

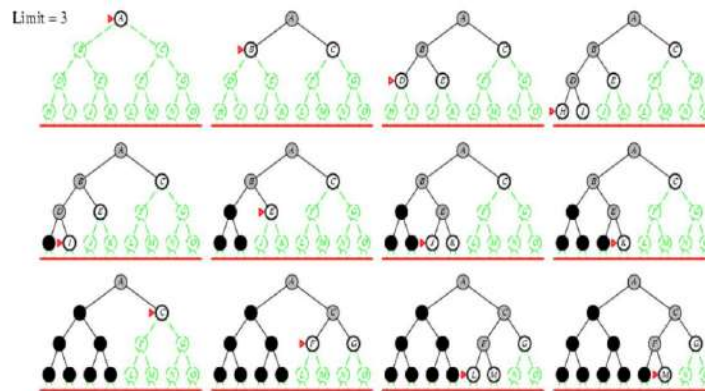
Iterative Deepening Search (IDS) adalah strategi pencarian graf yang menggabungkan keunggulan dari Depth-First Search (DFS) dan Breadth-First Search (BFS). Tujuannya adalah untuk mencari jalur terpendek antara simpul awal dan simpul tujuan dalam sebuah grafik.

IDS beroperasi dengan melakukan serangkaian penelusuran DFS dengan pembatasan kedalaman yang terus meningkat. Dimulai dengan pembatasan kedalaman 0, algoritma ini secara bertahap meningkatkan pembatasan tersebut hingga menemukan simpul tujuan atau menyelesaikan eksplorasi seluruh grafik. Proses ini diulang sampai mencapai simpul tujuan atau menyelesaikan eksplorasi seluruh grafik.

Langkah Kerja IDS:

1. Tetapkan batasan kedalaman awal menjadi 0.
2. Lakukan penelusuran DFS yang dimulai dari simpul awal, dengan membatasi kedalaman sesuai dengan batasan kedalaman saat ini.
3. Jika simpul tujuan ditemukan, hentikan penelusuran dan kembalikan jalur yang ditemukan.
4. Jika simpul tujuan tidak ditemukan dan masih ada simpul yang belum dikunjungi pada batasan kedalaman saat ini, kembali ke langkah 2.
5. Jika semua simpul pada batasan kedalaman saat ini telah dikunjungi dan simpul tujuan tidak ditemukan, tingkatkan batasan kedalaman sebesar 1 dan kembali ke langkah 2.
6. Ulangi langkah 2-5 sampai simpul tujuan ditemukan atau seluruh grafik telah dilalui.





Gambar 2.2 Contoh pencarian algoritma IDS depth 1-3

Bab 3

Analisis Pemecahan Masalah

3.1. Langkah-Langkah Pemecahan Masalah

Permasalahan yang ingin diselesaikan adalah bagaimana mendapatkan rute dari artikel awal hingga artikel tujuan dengan jarak yang sependek mungkin menggunakan algoritma BFS dan IDS.

Algoritma BFS

- Inisialisasi: Mulai dari simpul awal yang akan dijadikan titik awal pencarian. Tandai simpul awal tersebut sebagai sudah dikunjungi dan tambahkan ke dalam antrian (queue).
- Pencarian secara melebar: Lakukan iterasi secara melebar (breadth-first) dari simpul awal ke simpul-simpul tetangga yang belum dikunjungi. Dalam setiap iterasi, ambil simpul pertama dari antrian, lalu periksa semua tetangga yang belum dikunjungi. Tandai tetangga tersebut sebagai sudah dikunjungi dan tambahkan ke dalam antrian.
- Lanjutkan pencarian: Ulangi langkah kedua sampai semua simpul yang terhubung telah dikunjungi atau sampai simpul tujuan ditemukan.
- Penyimpanan jalur: Untuk pencarian jalur tertentu, simpan jalur yang dilalui dari simpul awal ke simpul tujuan. Ini dapat dilakukan dengan memelihara informasi tentang simpul-simpul yang saling terhubung selama pencarian.
- Penanganan simpul tujuan: Jika simpul tujuan ditemukan dan solusi yang diminta ialah tunggal, proses pencarian dihentikan dan jalur menuju simpul tujuan dapat dikembalikan. Namun, jika simpul tujuan ditemukan dan solusi yang diminta ialah banyak, proses pencarian dilanjutkan hingga satu layer (simpul yang kedalamannya sama) telah ditelusuri.
- Penanganan graf yang tidak terarah: Pada graf tidak terarah, pastikan untuk tidak mengunjungi simpul yang sama dua kali dan tidak melakukan perjalanan bolak-balik antara dua simpul.

Algoritma IDS

- Inisialisasi: Mulai dari kedalaman pencarian yang minimal (bernilai 0) dan tentukan batas kedalaman pencarian maksimal.
- Pencarian secara berulang: Lakukan pencarian menggunakan DFS (Depth-First Search) dengan batasan kedalaman yang bertambah secara iteratif dari 0 hingga menemukan solusi.
- Pencarian dengan kedalaman terbatas: Lakukan pencarian DFS pada setiap iterasi dengan batasan kedalaman tertentu. Cari jalur mulai dari simpul awal

dengan mengikuti cabang-cabang graf hingga mencapai kedalaman yang ditentukan.

- Penanganan simpul tujuan: Jika simpul tujuan ditemukan pada suatu iterasi dan hanya diminta solusi tunggal, proses pencarian dihentikan dan jalur menuju simpul tujuan dapat dikembalikan. Namun, jika simpul tujuan ditemukan dan diminta banyak solusi, proses pencarian dilanjutkan hingga batas kedalaman yang sama hingga tidak ada lagi simpul solusi yang ada pada kedalaman tersebut.
- Penambahan kedalaman: Jika solusi tidak ditemukan pada iterasi sebelumnya, tambahkan kedalaman pencarian dan lakukan pencarian kembali.
- Iterasi berulang: Ulangi langkah-langkah di atas dengan meningkatkan batasan kedalaman pencarian hingga solusi ditemukan atau mencapai batas maksimal.

3.2. Proses Pemetaan Masalah Menjadi Elemen-Elemen Algoritma IDS dan BFS

WikiRace adalah sebuah permainan di mana pemain mencoba untuk mencapai halaman Wikipedia tertentu hanya dengan menelusuri tautan di dalam artikel Wikipedia lainnya. Permasalahan WikiRace dapat dipetakan ke dalam masalah penelusuran graf, di mana setiap halaman Wikipedia dianggap sebagai simpul yang terhubung dengan halaman lain melalui tautan dalam artikel tersebut. Dalam tugas besar ini, penyelesaian WikiRace menggunakan algoritma Iterative Deepening Search dan Breadth First Search untuk menemukan jalur terpendek antara dua halaman Wikipedia yang diberikan.

Proses pembangkitan graf dalam permasalahan WikiRace melibatkan scraping untuk mengumpulkan informasi dari halaman Wikipedia dengan mengikuti seluruh tautan yang ada di setiap halaman. Pada tugas besar ini, limitasi halaman yang dapat dituju adalah halaman Wikipedia yang berupa artikel. Program akan memulai mengunjungi halaman dimulai dari halaman awal menuju halaman tujuan dengan mengekstrak semua tautan yang mengarah ke halaman-halaman lain dan mengunjungi halaman-halaman baru tersebut untuk mengumpulkan lebih banyak tautan. Proses ini berlanjut hingga program menemukan halaman tujuan atau ketika tidak menemukan solusi. Selama proses ini, program juga memperhatikan halaman-halaman yang telah dikunjungi sebelumnya untuk menghindari siklus yang tidak diinginkan dalam graf dan memastikan pencarian dilakukan dengan efisien.

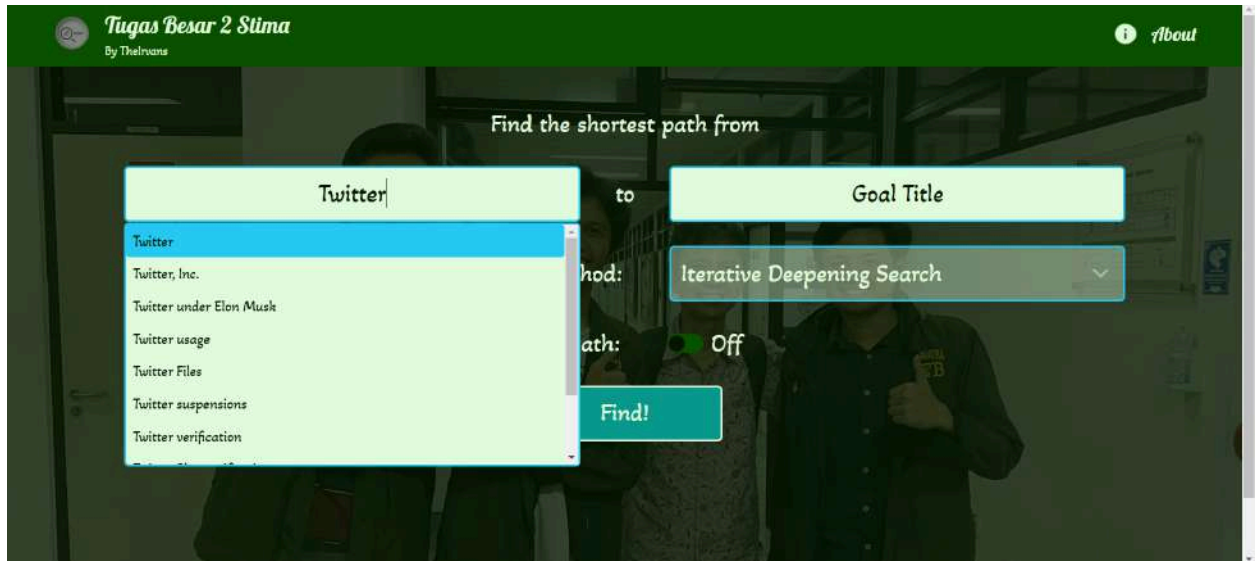
Proses pemetaan masalah WikiRace ke dalam algoritma Iterative Deepening Search melibatkan langkah-langkah sebagai berikut. Pertama, IDS melakukan pencarian jalur secara iteratif dengan membatasi kedalaman pencarian pada setiap iterasi. IDS akan memulai pencarian dari kedalaman 0, kemudian kedalaman ditingkatkan secara bertahap hingga mencapai solusi. Hal ini memungkinkan IDS untuk melakukan

pencarian secara sistematis dengan memastikan bahwa solusi yang ditemukan adalah solusi terpendek.

Proses pemetaan masalah WikiRace ke dalam algoritma Breadth First Search melibatkan langkah-langkah sebagai berikut. BFS melakukan pencarian secara melebar, artinya BFS akan mencoba mengeksplorasi semua simpul pada kedalaman yang sama sebelum melanjutkan ke kedalaman berikutnya. Dalam konteks WikiRace, BFS akan mengunjungi halaman-halaman yang terhubung langsung dengan halaman awal terlebih dahulu sebelum mengunjungi halaman-halaman yang lebih jauh. Dengan demikian, BFS dapat menemukan jalur terpendek antara dua halaman Wikipedia dengan lebih cepat dibandingkan dengan metode pencarian lain yang mungkin hanya mengeksplorasi jalur tertentu.

3.3. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

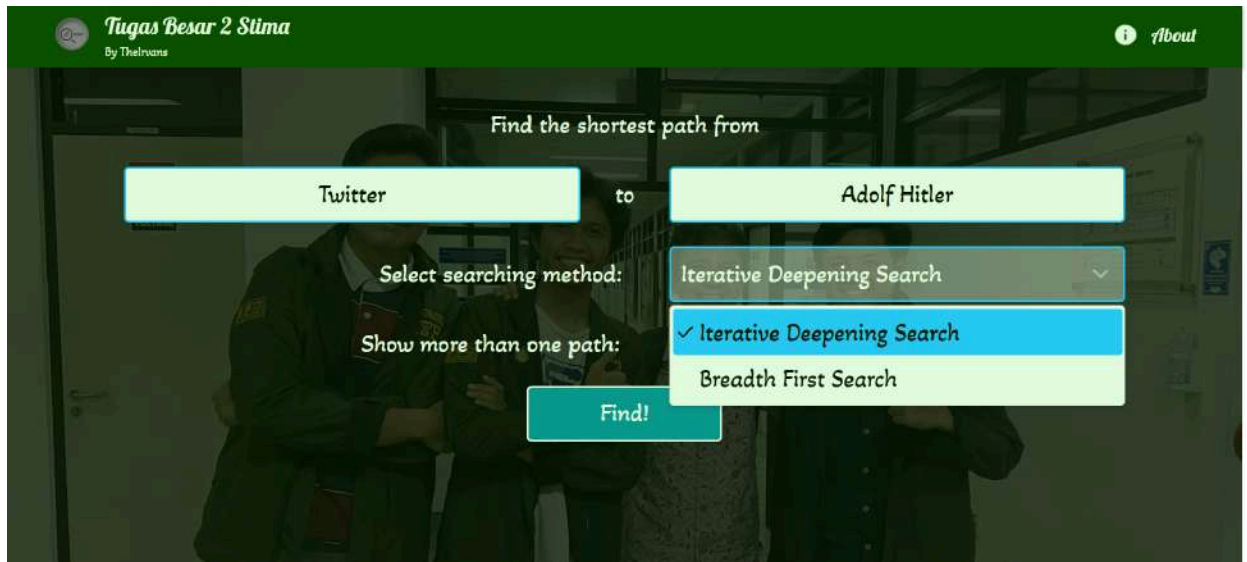
a. Fitur *Autocomplete*



Gambar 3.3.1 Fitur *Autocomplete*

Kami menyediakan fitur autocomplete yang menyesuaikan masukkan pada saat itu dan menampilkan daftar dari judul-judul artikel terkait. Hal ini dapat mempermudah pengguna untuk mempercepat proses *input* karena bisa secara otomatis dilengkapi oleh web.

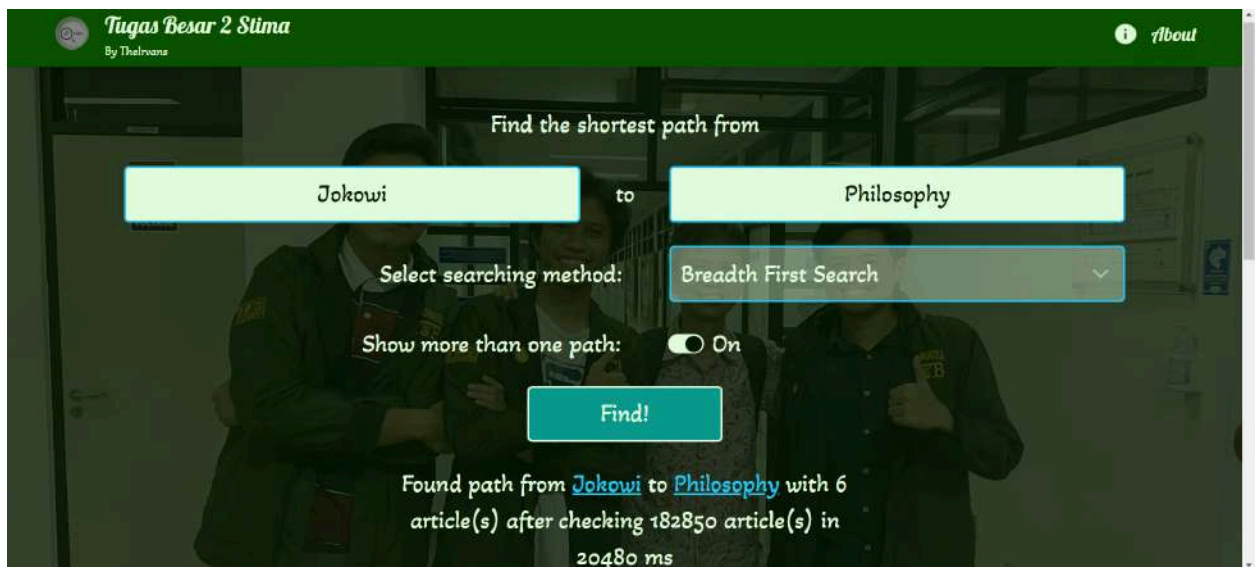
b. Fitur *Select (Dropdown)*



Gambar 3.3.2 Fitur *Select (Dropdown)*

Kami menyediakan fitur memilih algoritma dengan tampilan *dropdown*.

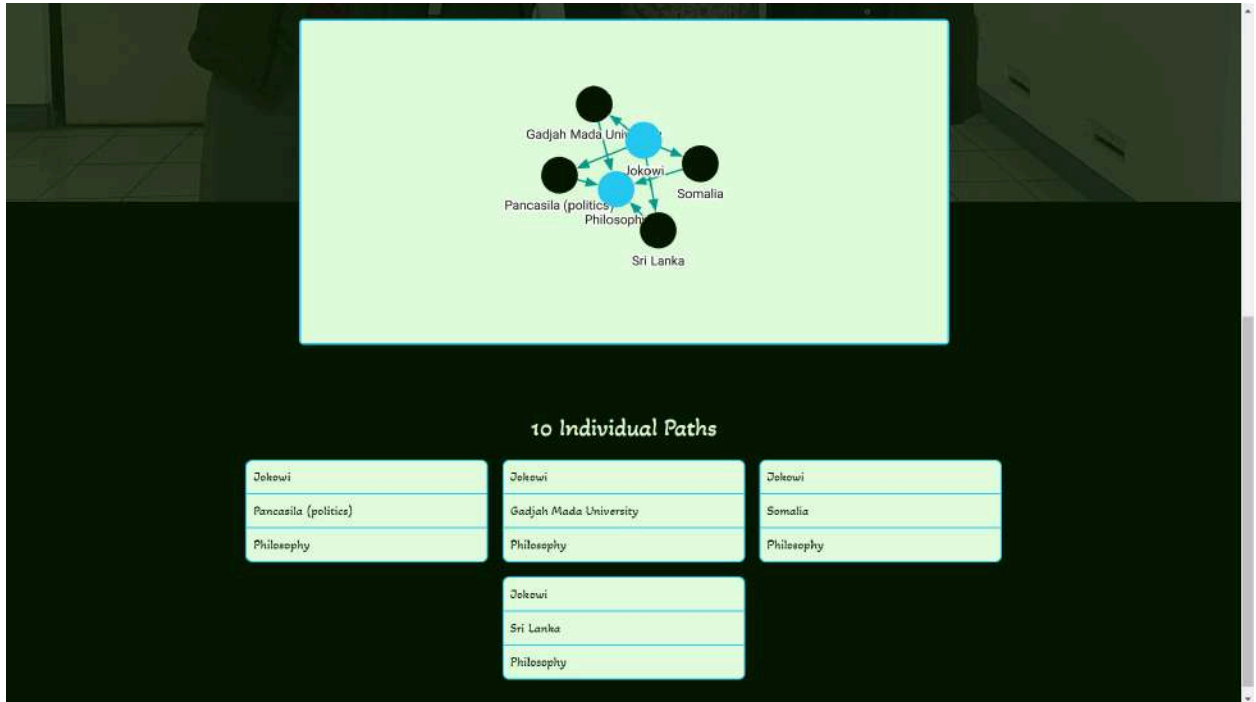
c. Fitur *Switch*



Gambar 3.3.3 Fitur *Switch* dan tampilan keluaran

Kami menyediakan fitur *switch* untuk memilih solusi yang ingin ditampilkan berupa solusi tunggal (*single path*) atau solusi banyak (*multi path*).

d. Visualisasi Keluaran *Paths*



Gambar 3.3.4 Visualisasi Graf

Selain menampilkan keluaran jumlah artikel dan waktu pencarian, kami menyediakan fitur visualisasi graf untuk memperlihatkan rute yang dilalui. Kami juga memperlihatkan rute secara individual.

e. Fitur *Button About Page* dan *Home Page*

Pada *navbar*, terdapat dua *button* utama di sebelah kanan dan kiri untuk berpindah antara *About* dan *Home Page*.

f. Fitur *back-end*

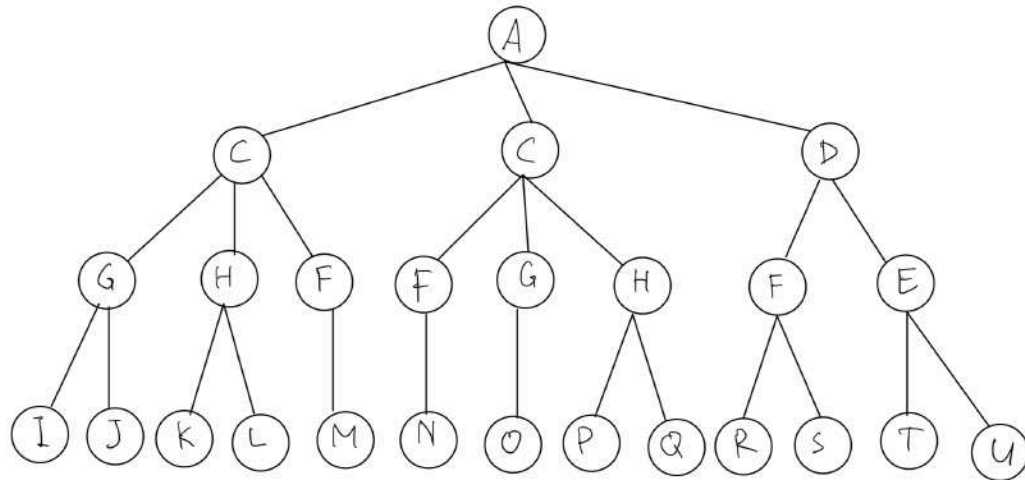
Pada bagian back-end, program kami membuat beberapa API endpoint yang dapat digunakan untuk melakukan pencarian jalur terpendek dari halaman awal ke halaman akhir. Seluruh endpoint yang kami buat menerima body berupa json yang berisi start dan end yang masing-masing bertipe string dan berupa url wikipedia. Penjelasan lebih lanjut mengenai API endpoint yang kami buat dapat dilihat pada tabel berikut.

Tabel 3.3.1 Dokumentasi API Endpoint yang Digunakan pada Program

Method	Endpoint	Deskripsi
POST	/multiple/bfs	Melakukan pencarian dengan metode BFS dan mengembalikan semua path yang mungkin.
POST	/single/bfs	Melakukan pencarian dengan metode BFS dan mengembalikan hanya satu path.
POST	/multiple/ids	Melakukan pencarian dengan metode IDS dan mengembalikan semua path yang mungkin.
POST	/single/ids	Melakukan pencarian dengan metode IDS dan mengembalikan hanya satu path.

3.4. Contoh Ilustrasi Kasus

Untuk memperjelas analisis pemecahan masalah, kami akan melakukan ilustrasi kasus. Misalkan ingin dicari *path* dari A ke F dengan *tree of wikipedia link* seperti gambar berikut:



Gambar 3.4.1 Ilustrasi contoh *tree of wikipedia link*

dimana masing masing simbol pada simpul mewakili suatu link artikel wikipedia. Proses pencarian BFS ditunjukkan pada tabel berikut:

Head	Queue	Visited Node	Solution Path
A	[C, C, D]	[]	[]
C	[C, D, G, H, F]	[A]	[]
C	[D, G, H, F]	[A, C]	[]
D	[G, H, F, F, E]	[A, C]	[]
G	[H, F, F, E, I, J]	[A, C, D]	[]
H	[F, F, E, I, J, K, L]	[A, C, D, G]	[]
F	[F, E, I, J, K, L, M]	[A, C, D, G, H]	[[A, C, F]]
F	[E, I, J, K, L, M]	[A, C, D, G, F]	[[A, C, F], [A, D, F]]
E	[I, J, K, L, M]	[A, C, D, G, F]	[[A, C, F], [A, D, F]]

Proses pencarian IDS ditunjukkan pada tabel berikut:

Head	Stack	Visited Node	Solution Path	Depth
A	[C, C, D]	[]	[]	1
C	[C, D]	[A]	[]	1
C	[D]	[A, C]	[]	1
D	[]	[A, C]	[]	1
A	[C, C, D]	[]	[]	2
C	[G, H, F, C, D]	[A]	[]	2
G	[H, F, C, D]	[A, C]	[]	2
H	[F, C, D]	[A, C, G]	[]	2
F	[C, D]	[A, C, G, H]	[[A, C, F]]	2
C	[D]	[A, C, G, H, F]	[[A, C, F]]	2
D	[F, E]	[A, C, G, H, F]	[[A, C, F]]	2
F	[E]	[A, C, G, H, F]	[[A, C, F] , [A, D, F]]	2
E	[]	[A, C, G, H, F]	[[A, C, F] , [A, D, F]]	2

Catatan: Jika diminta solusi tunggal, pencarian akan langsung terpotong setelah mendapatkan satu path saja.

Bab 4

Implementasi dan Pengujian

4.1. Spesifikasi Teknis Program

a. Struktur Data

i. Struct SafeMap

Struct SafeMap adalah struct wrapper untuk struktur data map yang dapat digunakan untuk keperluan Mutex. SafeMap memiliki atribut tambahan yaitu mu yang memiliki tipe sync.Mutex. SafeMap memiliki beberapa method, diantaranya Add, Get, dan Replace.

ii. Struct SafeArray

Struct SafeArray adalah struct wrapper untuk struktur data array yang dapat digunakan untuk keperluan Mutex. SafeArray memiliki atribut tambahan yaitu mu yang memiliki tipe sync.Mutex. SafeArray memiliki beberapa method, diantaranya Add, Get, dan Set.

iii. Struct Node

Struct Node adalah struktur data buatan untuk menyimpan data simpul pada graf yang akan dibangun. Node memiliki atribut Value berupa string, Children berupa map of Node, dan atribut tambahan mu yang memiliki tipe sync.Mutex agar Node dapat digunakan untuk keperluan Mutex. Node memiliki beberapa method, diantaranya NewNode, AddChild, DFS, dfsIterative, DFSSingle, dfsIterativeSingle.

iv. Struct linkJson

Struct linkJson adalah struktur data buatan yang menjelaskan tipe data yang dikirimkan dari request HTTP ke server. linkJson memiliki atribut Start dan End yang masing-masing bertipe string.

b. Fungsi dan Prosedur

i. BFS

```
function bfs(startURL: String, endURL: String,
baseURL: String) -> (solutions: Array of Array of
String, visitedCount: Integer)

{ Melakukan pencarian BFS dari startURL ke endURL }
```

Deklarasi

```
visitedURL: Map of Boolean  
queriedURL: Map of Boolean  
found: Boolean  
paths: Array of Array of String  
solutions: Array of Array of String  
depth: Integer
```

Inisialisasi

```
visitedURL ← InitializeMap()  
queriedURL ← InitializeMap()  
found ← false  
paths ← InitializeArray()  
paths.Add(InitializeArray(startURL))  
solutions ← InitializeArray()  
depth ← 1
```

Algoritma

```
while not found do  
    newPaths: Array of Array of String  
    pathsSize ← paths.Size()  
    CreateWaitGroup(pathsSize)
```

```

Print("Depth:", depth)

depth <- depth + 1

maxConcurrentRequests <- 250

semaphore
CreateSemaphore(maxConcurrentRequests)

i traversal [0..pathSize-1]

    semaphore.Acquire()

    GoRoutine(i, semaphore, {

        DeferRelease(semaphore)

        DeferDone()

        // Mendapatkan path

        p ← paths.Get(i)

        // Mendapatkan node terakhir

        node ← p[len(p) - 1]

        // Jika sudah ketemu, maka hentikan

        if queriedURL.Contains(node) then

            ->

```

```

doc <- makeRequest(node)

if doc != nil then

    duplicateURL <- InitializeMap()

    // Hanya mengecek pada bagian yang
    memiliki id bodyContent

    bodyContent <-
doc.Find("#bodyContent")

    bodyContent.Find("a").Each(func(_, s)
{

        link, _ <- s.Attr("href")

        matched, _ <-
regexp.MatchString("^/wiki/[^:]+$", link)

        // Jika link sesuai dan tidak
        duplikat

        if matched and not
duplicateURL[baseURL + link] then

            duplicateURL[baseURL +
link] = true

            // Jika link adalah tujuan

            if baseURL + link ==
endURL then

```

```

Print("Found!")

found ← true

solutions.Add(append(p,
baseURL + link))

endif

visitedURL.Get(baseURL + link) _' ok <-

if not ok then

visitedURL.Add(baseURL +
link, true)

// Jika belum ketemu, maka
tambahkan path baru ke newPaths

if not found then

newPath <- CopyArray(p)

newPath.Add(baseURL +
link)

newPaths.Add(newPath)

endif

endif

}))

// Tandai node sudah diquery
queriedURL.Add(node, true)

endif

```

```

        })

    endfor

    WaitGroup()

    // Set paths dengan newPaths
    paths ← newPaths

endwhile

-> solutions, visitedURL.Size()

```

ii. IDS

```

procedure DFS(endURL: String) -> (solutions: Array of
Array of String, visitedCount: Integer)

{ Melakukan pencarian DFS dari node saat ini ke
endURL }

Deklarasi

visited: Map of Boolean

paths: Array of Array of String

Inisialisasi

visited ← InitializeMap()

paths ← InitializeArray()

```

Algoritma

```
dfsIterative(self, endURL, visited, [], paths)

-> paths, visited.Size()
```

```
procedure dfsIterative(endURL: String, visited: Map
of Boolean, currentPath: Array of String, paths:
reference to Array of Array of String)
```

```
{ Melakukan pencarian DFS secara iteratif }
```

Deklarasi

```
stack: Stack of Node
```

```
pathStack: Stack of Array of String
```

Inisialisasi

```
stack ← InitializeStack()
```

```
pathStack ← InitializeStack()
```

```
stack.Push(n)
```

```
pathStack.Push(currentPath)
```

Algoritma

```
while not stack.isEmpty() do
```

```
    node ← stack.Pop()
```

```
    currentPath ← pathStack.Pop()
```



```

visited[node.Value] ← true

currentPath.Add(node.Value)

if node.Value == endURL then

    Print("Found endURL when traversing!")

    pathCopy ← CopyArray(currentPath)

    paths.Add(pathCopy)

allVisited ← true

for each child in node.Children do

    if not visited[child.Value] then

        allVisited ← false

        stack.Push(child)

        newPath ← CopyArray(currentPath)

        pathStack.Push(newPath)

if allVisited then

    visited[node.Value] ← false

```

```

function ids(startURL: String, endURL: String,
baseURL: String) -> (solutions: Array of Array of
String, visitedCount: Integer)

{ Melakukan pencarian IDS dari startURL ke endURL }

```

Deklarasi

```
queriedURL: Map of Boolean  
root: Node  
paths: Array of Array of String  
visitedCount: Integer  
currentLastNodes: Array of Node  
currentLastNodesMap: Map of Boolean  
depth: Integer
```

Inisialisasi

```
queriedURL ← InitializeMap()  
root ← NewNode(startURL)  
paths ← InitializeArray()  
visitedCount ← 0  
currentLastNodes ← InitializeArray(root)  
currentLastNodesMap ← InitializeMap()  
currentLastNodesMap.Add(startURL, true)  
depth ← 0
```

Algoritma

```
while len(paths) == 0 do  
    newLastNodes ← InitializeArray()  
    newLastNodesMap ← InitializeMap()
```

```

CreateWaitGroup(len(currentLastNodes))

maxConcurrentRequests <- 250

semaphore
CreateSemaphore(maxConcurrentRequests) <-

for each node in currentLastNodes do

    semaphore.Acquire()

    GoRoutine(node, semaphore, {

        DeferRelease(semaphore)

        DeferDone()

        if queriedURL.Contains(node.Value) then

            ->

            doc <- makeRequest(node.Value)

            if doc != nil then

                duplicateURL <- InitializeMap()

                bodyContent
doc.Find("#bodyContent") <-

```

```

        bodyContent.Find("a").Each(func(_, s)
{
    link, _ <- s.Attr("href")

    matched, _ <-
regexp.MatchString("^/wiki/[^:]+$", link)

    if matched and not
duplicateURL[baseURL + link] then

        duplicateURL[baseURL
link] = true

        if endURL == baseURL +
link then

            Print("Found endURL when
querying!")

            _, okNodeMap <-
currentLastNodesMap.Get(baseURL + link)

            _, okQuery <-
queriedURL.Get(baseURL + link)

            if not okNodeMap and not
okQuery then

                node.AddChild(baseURL
link)

newLastNodes.Add(node.Children[baseURL + link])

newLastNodesMap.Add(baseURL + link, true)

```

```

                                endif
                                endif

                                })

                                queriedURL.Add(node.Value, true)

                                endif

                                })

                                endfor

                                WaitGroup()

                                currentLastNodes ← newLastNodes

                                currentLastNodesMap.Replace(newLastNodesMap.data)

                                Print("Recurring")

                                paths, visitedCount ← root.DFS(endURL)

                                Print("Finished Recursing")

                                depth <- depth + 1

                                Print("Depth:", depth)

                                endwhile

                                -> paths, visitedCount

```

4.2. Tata Cara Penggunaan Program

Berikut prosedur penggunaan program:

- a. Lakukan prosedur yang ada pada README.md Github
- b. Buka link pada terminal untuk menuju website
- c. Masukkan judul artikel awal dan judul artikel tujuan
- d. Pilih algoritma yang digunakan
- e. Pilih *switch* pada mode “On” jika ingin ditampilkan solusi banyak
- f. Klik tombol “Find!”

Jika terdapat anomali dalam penggunaan program, silahkan bertanya pada kami.

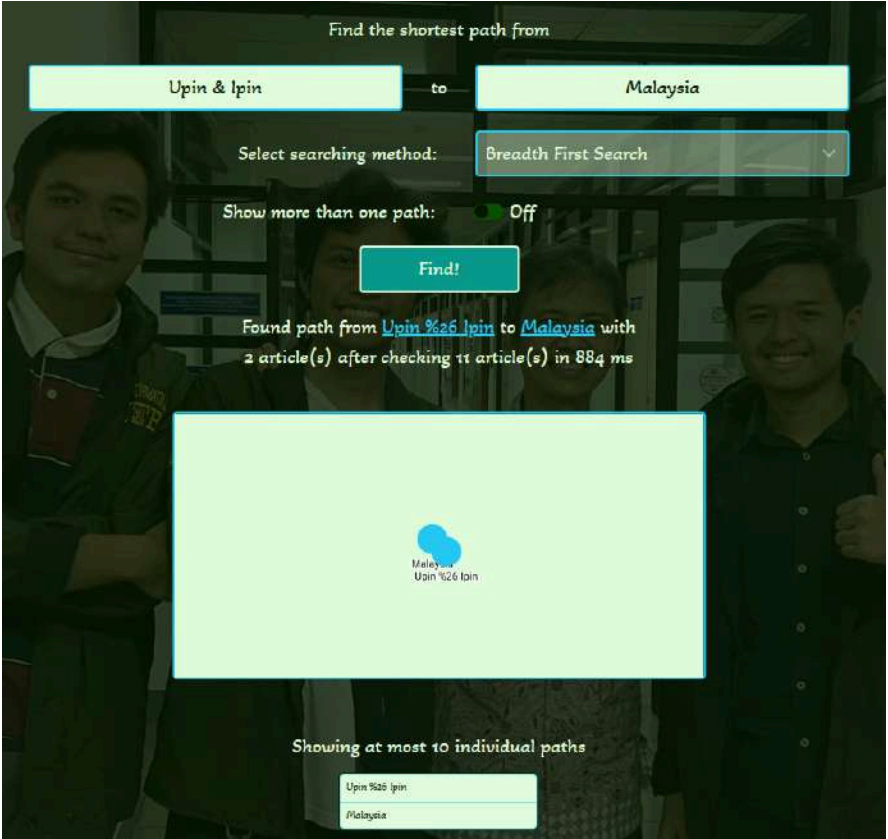
4.3. Hasil Pengujian

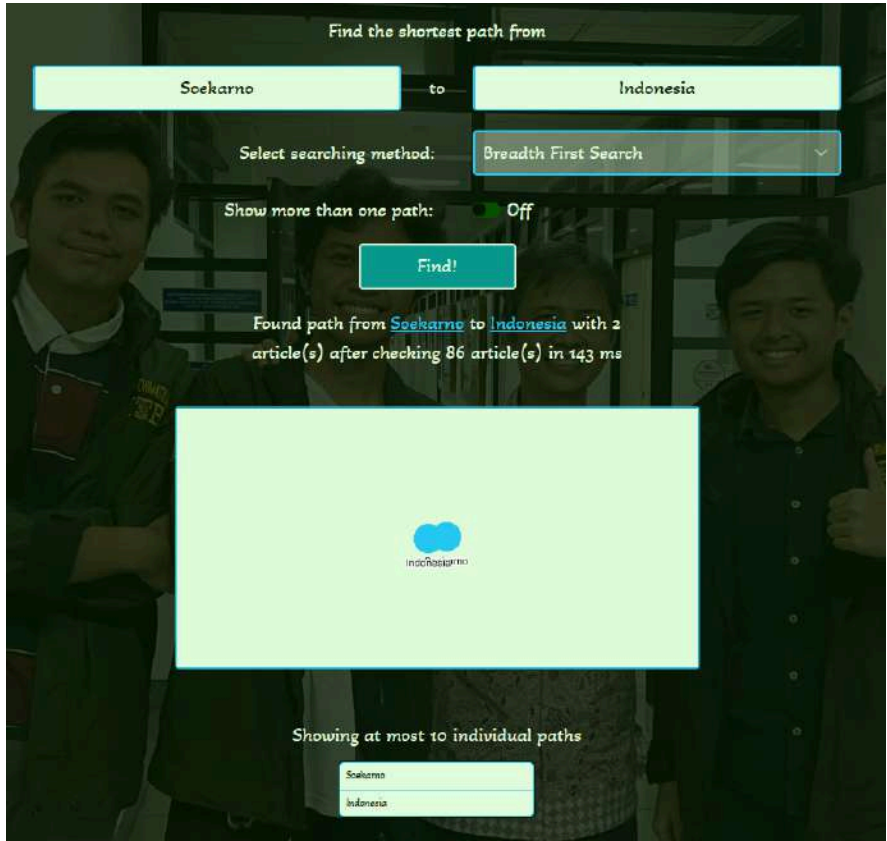
a. BFS (*Breadth First Search*)

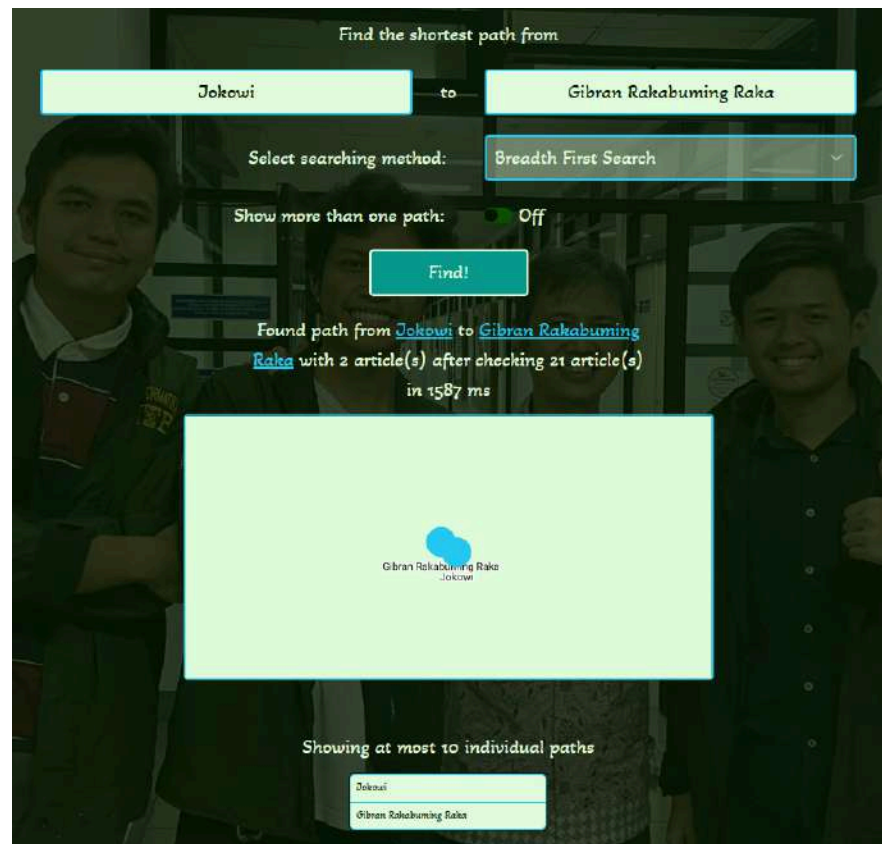
i. One Path

1. Depth 1

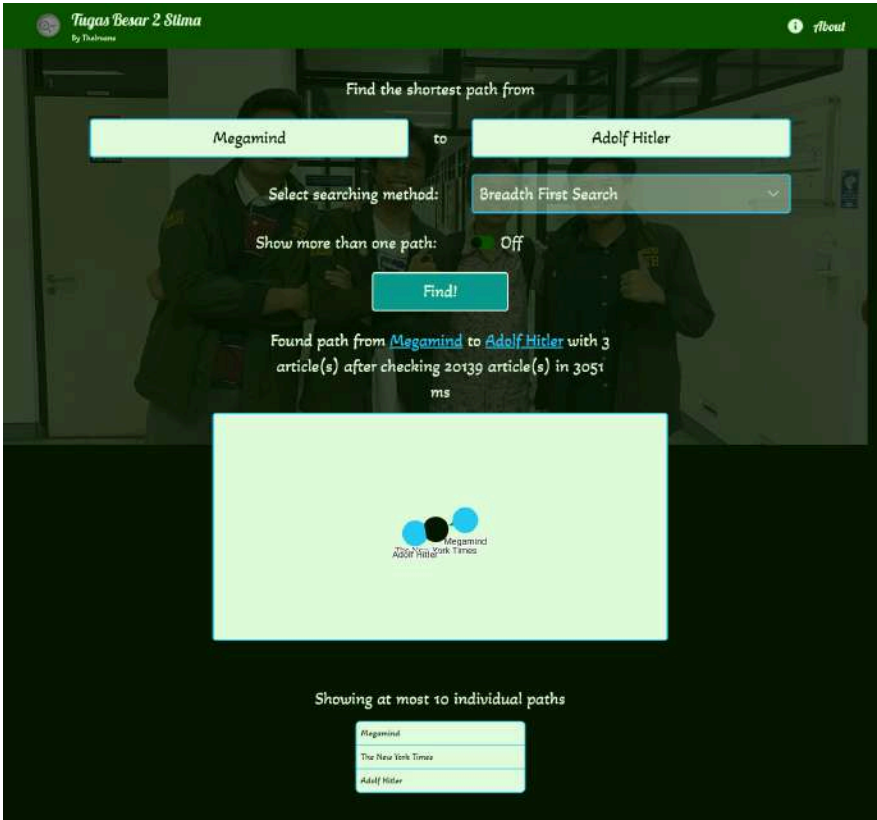
No	Path
1	Upin & Ipin -> Malaysia: 884 ms

	
2	Soekarno -> Indonesia: 143 ms

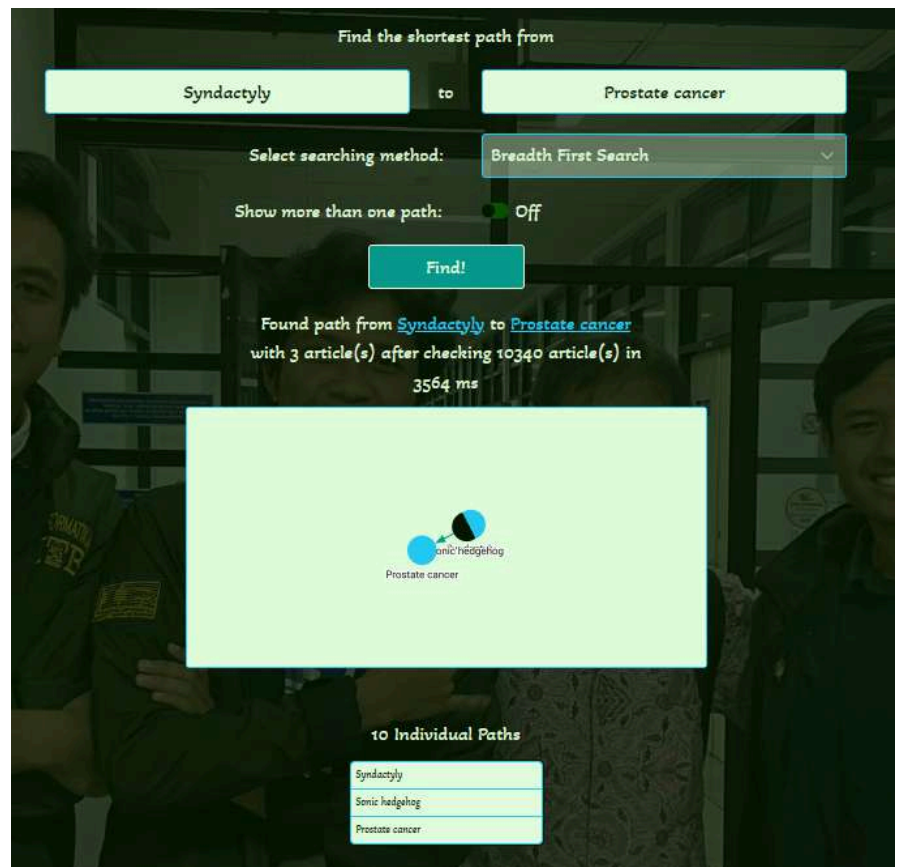
	
3	Jokowi -> Gibran Rakabuming Raka: 1567 ms



2. Depth 2

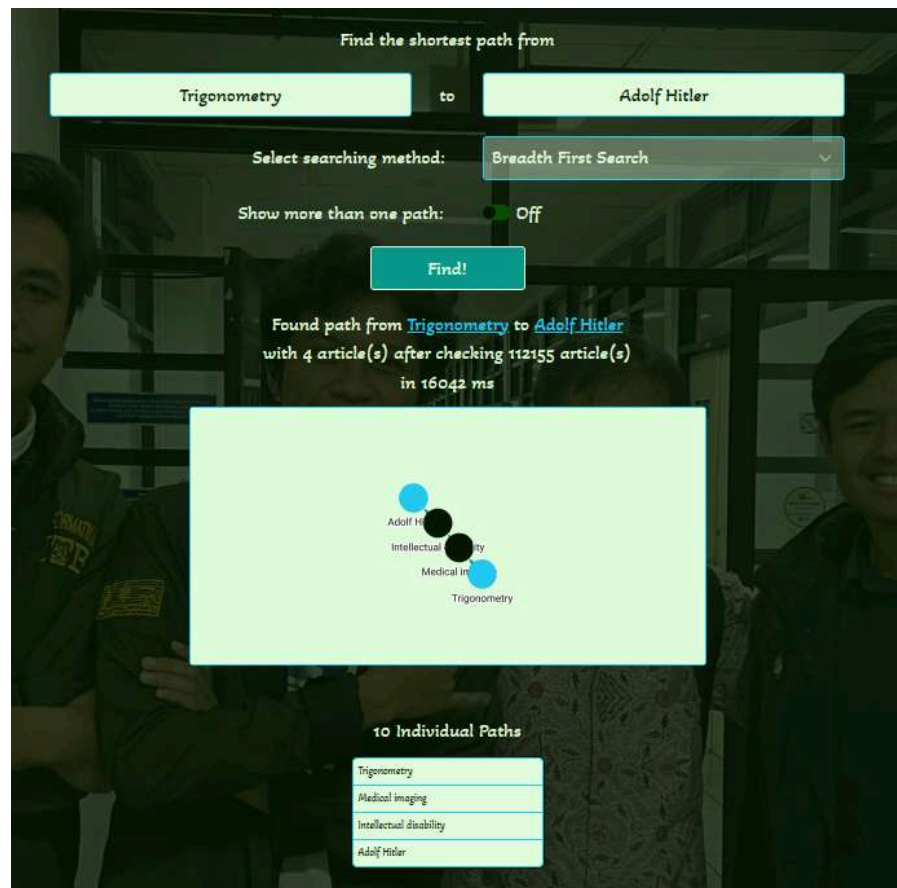
No.	Path
1	<p>Megamind -> Adolf Hitler: 3051 ms</p> 
2	<p>Femboy -> World War II: 5261 ms</p>

	<div><p>Find the shortest path from</p><div><div>Femboy</div>to<div>World War II</div></div><p>Select searching method: Breadth First Search</p><p>Show more than one path: <input type="checkbox"/> Off</p><p>Find!</p><p>Found path from Femboy to World War II with 3 article(s) after checking 34569 article(s) in 5261 ms</p><div><pre>graph LR; A((Femboy)) --- B((Wartime cross-dressers)); B --- C((World War II))</pre></div><p>10 Individual Paths</p><table><tr><td>Femboy</td></tr><tr><td>Wartime cross-dressers</td></tr><tr><td>World War II</td></tr></table></div>	Femboy	Wartime cross-dressers	World War II
Femboy				
Wartime cross-dressers				
World War II				
3	Syndactyly -> Prostate cancer: 3563 ms			



3. Depth 3

No.	Path
1	<div><div>SMA Negeri 8 Jakarta</div> to <div>Institut Teknologi Bandung</div><div>Select searching method: Breadth First Search</div><div>Show more than one path: Off</div><div>Find!</div><div>Found path from SMA Negeri 8 Jakarta to Institut Teknologi Bandung with 4 article(s) after checking 227377 article(s) in 19484 ms</div><div><div><div>SMA Negeri 8 Jakarta</div><div>Jakarta</div><div>Depok</div><div>Institut Teknologi Bandung</div></div></div><div>10 Individual Paths</div><div><div>SMA Negeri 8 Jakarta</div><div>Jakarta</div><div>Depok</div><div>Institut Teknologi Bandung</div></div></div>
2	Trigonometry -> Adolf Hitler: 16.042 ms



3

Fire -> Crotch: 352.818 ms

Find the shortest path from

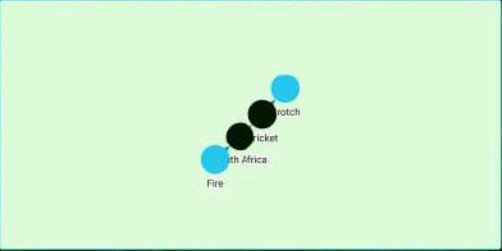
Fire to Crotch

Select searching method: Breadth First Search

Show more than one path: Off

Find!

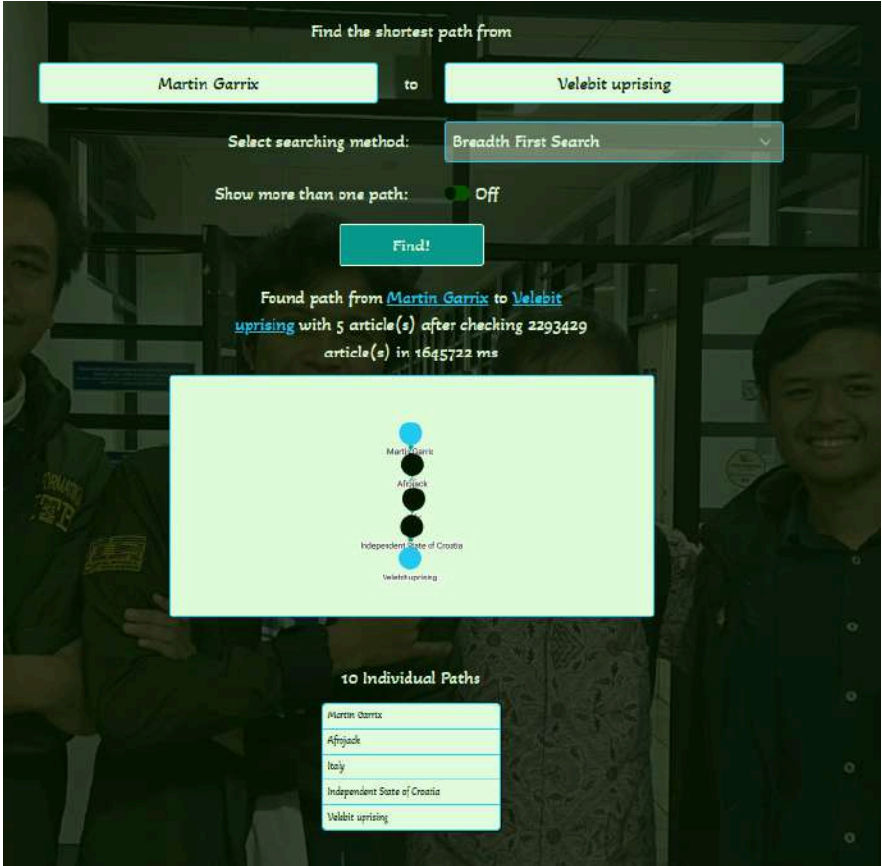
Found path from Fire to Crotch with 4 article(s)
after checking 1617096 article(s) in 352818 ms



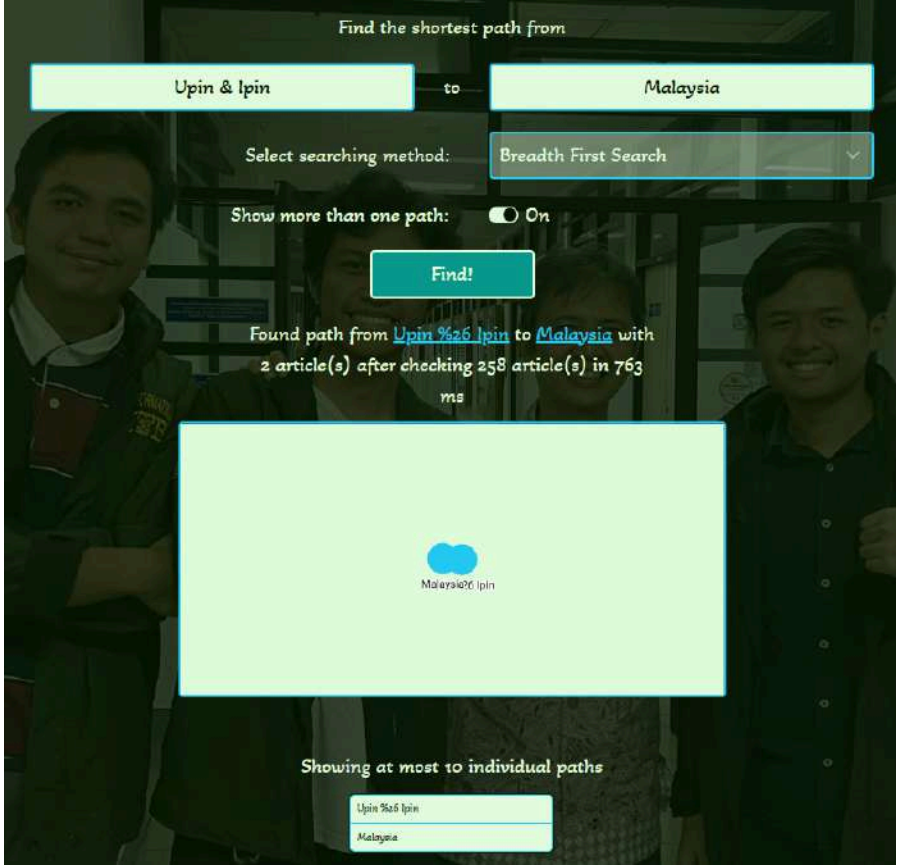
10 Individual Paths

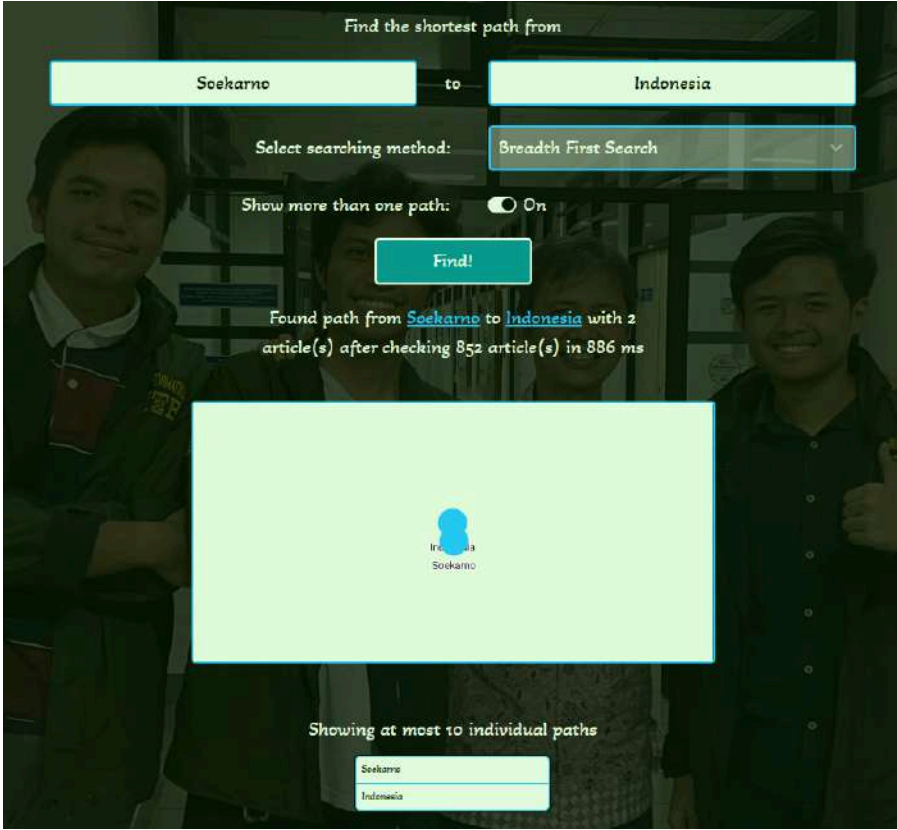
Fire
South Africa
Cricket
Crotch

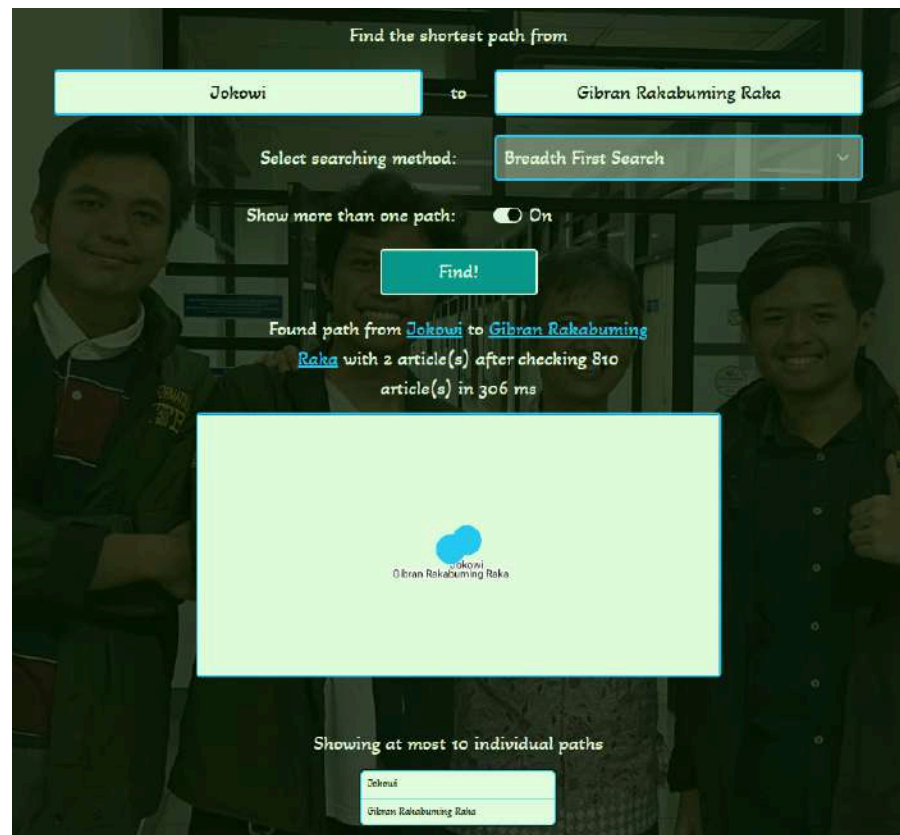
4. Depth 4

No.	Path
1	<p>Martin Garrix -> Velebit uprising: 1.645.722 ms</p>  <p>Find the shortest path from</p> <p>Martin Garrix to Velebit uprising</p> <p>Select searching method: Breadth First Search</p> <p>Show more than one path: Off</p> <p>Find!</p> <p>Found path from Martin Garrix to Velebit uprising with 5 article(s) after checking 229342 article(s) in 1645722 ms</p> <p>Visualized path:</p> <ul style="list-style-type: none">Martin GarrixAfrin JadeIndependent State of CroatiaVelebit uprising <p>10 Individual Paths</p> <ul style="list-style-type: none">Martin GarrixAfrin JadeItalyIndependent State of CroatiaVelebit uprising

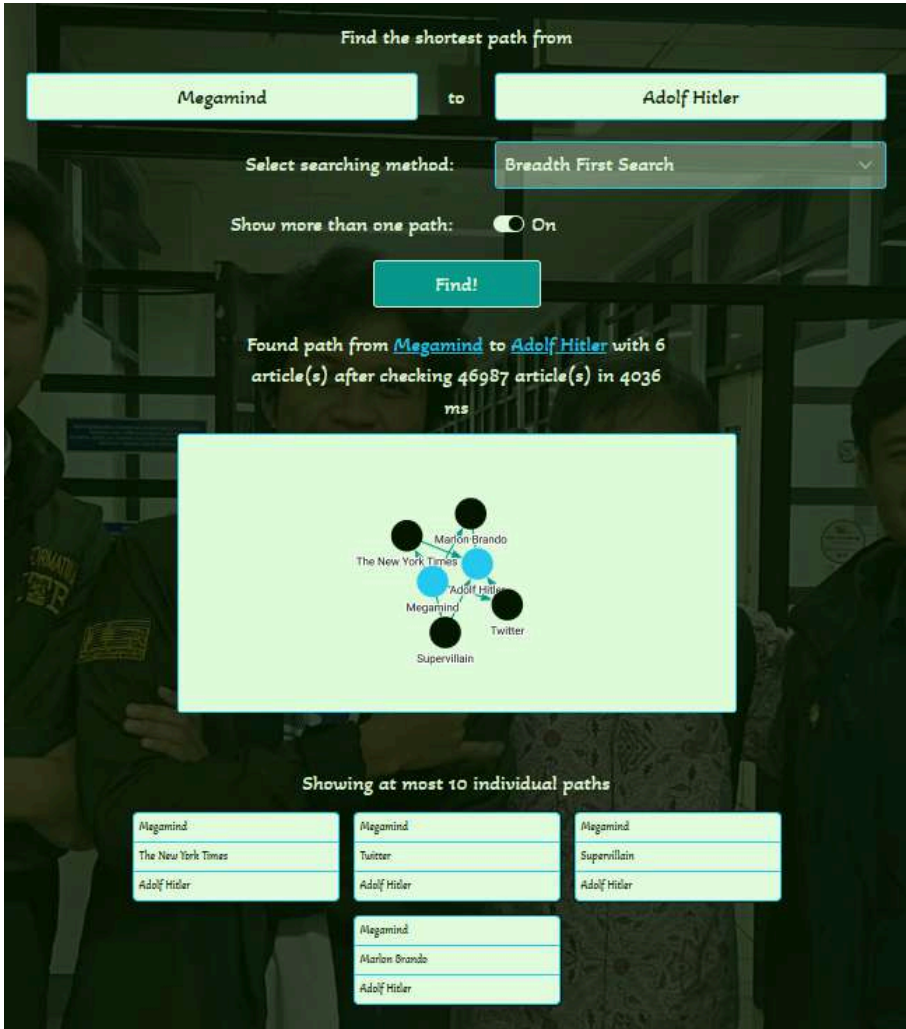
ii. More Than One Path
1. Depth 1

No.	Path
1	<p>Upin & Ipin -> Malaysia: 699 ms</p> 
2	Soekarno -> Indonesia: 886 ms

	
3	Jokowi -> Gibran Rakabuming Raka: 306 ms



2. Depth 2

No.	Path
1	<p>Megamind -> Adolf Hitler: 4.036 ms</p>  <p>Find the shortest path from</p> <p>Megamind to Adolf Hitler</p> <p>Select searching method: Breadth First Search</p> <p>Show more than one path: <input checked="" type="checkbox"/> On</p> <p>Find!</p> <p>Found path from <u>Megamind</u> to <u>Adolf Hitler</u> with 6 article(s) after checking 46987 article(s) in 4036 ms</p> <p>Graph showing connections between nodes: Megamind, The New York Times, Marlon Brando, Adolf Hitler, Megamind, Twitter, Supervillain.</p> <p>Showing at most 10 individual paths</p> <ul style="list-style-type: none"> Megamind -> The New York Times -> Adolf Hitler Megamind -> Twitter -> Adolf Hitler Megamind -> Supervillain -> Adolf Hitler Megamind -> Marlon Brando -> Adolf Hitler
2	<p>Femboy -> World War II: 4.891 ms</p>

Find the shortest path from

Femboy to **World War II**

Select searching method: **Breadth First Search**

Show more than one path: ☒ On

Find!

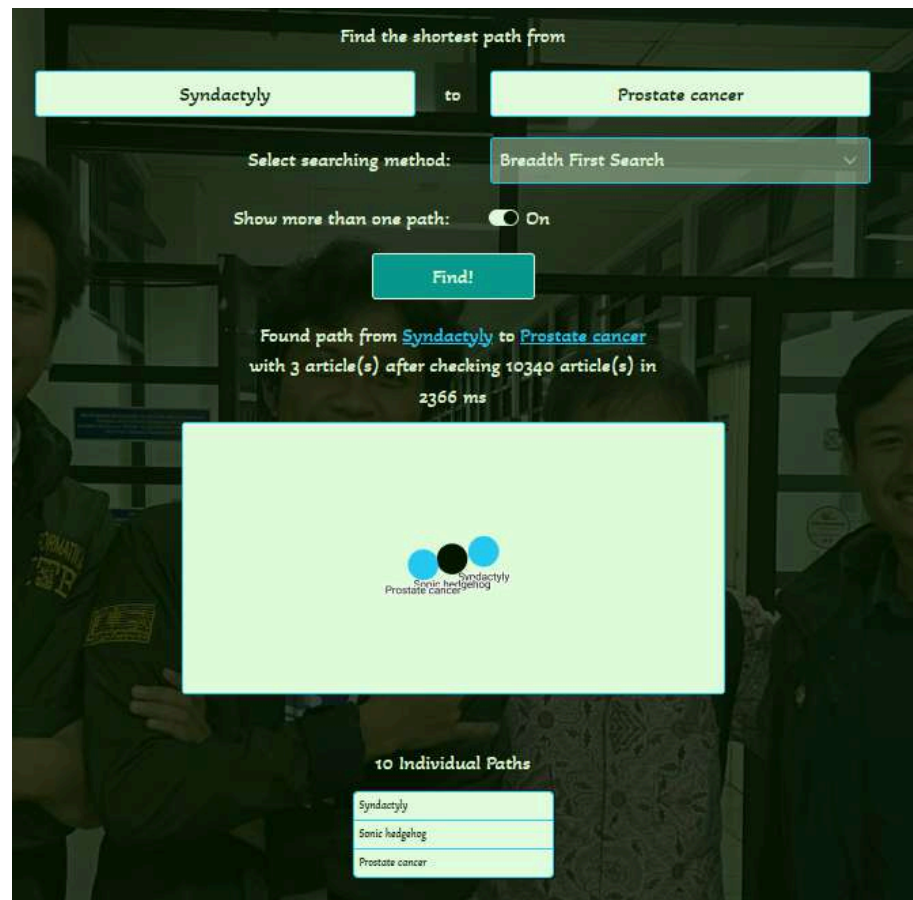
Found path from **Femboy** to **World War II** with 30 article(s) after checking 41812 article(s) in 4891 ms

10 Individual Paths

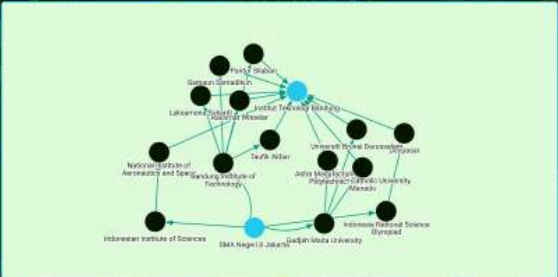
Femboy	Femboy	Femboy
Anti-war movement	Male gender	Skirt
World War II	World War II	World War II
Femboy	Femboy	Femboy
Femme	Dress	Kinsey scale
World War II	World War II	World War II
Femboy	Femboy	Femboy
Cross-dressing	Queer	Wartime cross-dressers
World War II	World War II	World War II
Femboy	Femboy	
Babla		
World War II		

3

Syndactyly -> Prostate Cancer: 2.366 ms



3. Depth 3

No.	Path																																																
1	<div><div>SMA Negeri 8 Jakarta</div>to<div>Institut Teknologi Bandung</div><div>Select searching method: Breadth First Search</div><div>Show more than one path: <input checked="" type="checkbox"/> On</div><div>Find!</div><div>Found path from SMA Negeri 8 Jakarta to Institut Teknologi Bandung with 16 article(s) after checking 1912701 article(s) in 401878 ms</div><div></div><div>10 Individual Paths</div><table><tr><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td></tr><tr><td>Indonesia National Science Olympiad</td><td>Gadjah Mada University</td><td>Gadjah Mada University</td></tr><tr><td>Denpasar</td><td>De La Salle Catholic University, Manado</td><td>Astra Manufacturing Polytechnic</td></tr><tr><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td></tr><tr><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td></tr><tr><td>Gadjah Mada University</td><td>Indonesian Institute of Sciences</td><td>Bandung Institute of Technology</td></tr><tr><td>Universiti Brunel Darussalam</td><td>National Institute of Aeronautics and Space</td><td>Rachmat Witoelar</td></tr><tr><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td></tr><tr><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td></tr><tr><td>Bandung Institute of Technology</td><td>Bandung Institute of Technology</td><td>Bandung Institute of Technology</td></tr><tr><td>Lukasmana Sukardi</td><td>Samaun Samadikun</td><td>Pantur Silaban</td></tr><tr><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td></tr><tr><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td><td>SMA Negeri 8 Jakarta</td></tr><tr><td>Bandung Institute of Technology</td><td>Bandung Institute of Technology</td><td>Bandung Institute of Technology</td></tr><tr><td>Taufik Akbar</td><td>Taufik Akbar</td><td>Taufik Akbar</td></tr><tr><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td><td>Institut Teknologi Bandung</td></tr></table></div>	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	Indonesia National Science Olympiad	Gadjah Mada University	Gadjah Mada University	Denpasar	De La Salle Catholic University, Manado	Astra Manufacturing Polytechnic	Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	Gadjah Mada University	Indonesian Institute of Sciences	Bandung Institute of Technology	Universiti Brunel Darussalam	National Institute of Aeronautics and Space	Rachmat Witoelar	Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	Bandung Institute of Technology	Bandung Institute of Technology	Bandung Institute of Technology	Lukasmana Sukardi	Samaun Samadikun	Pantur Silaban	Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	Bandung Institute of Technology	Bandung Institute of Technology	Bandung Institute of Technology	Taufik Akbar	Taufik Akbar	Taufik Akbar	Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung
SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta																																															
Indonesia National Science Olympiad	Gadjah Mada University	Gadjah Mada University																																															
Denpasar	De La Salle Catholic University, Manado	Astra Manufacturing Polytechnic																																															
Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung																																															
SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta																																															
Gadjah Mada University	Indonesian Institute of Sciences	Bandung Institute of Technology																																															
Universiti Brunel Darussalam	National Institute of Aeronautics and Space	Rachmat Witoelar																																															
Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung																																															
SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta																																															
Bandung Institute of Technology	Bandung Institute of Technology	Bandung Institute of Technology																																															
Lukasmana Sukardi	Samaun Samadikun	Pantur Silaban																																															
Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung																																															
SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta	SMA Negeri 8 Jakarta																																															
Bandung Institute of Technology	Bandung Institute of Technology	Bandung Institute of Technology																																															
Taufik Akbar	Taufik Akbar	Taufik Akbar																																															
Institut Teknologi Bandung	Institut Teknologi Bandung	Institut Teknologi Bandung																																															

2

Trigonometry -> Adolf Hitler: 328.348 ms

Trigonometry to Adolf Hitler

Select searching method: Breadth First Search

Show more than one path: ☒ On

Find!

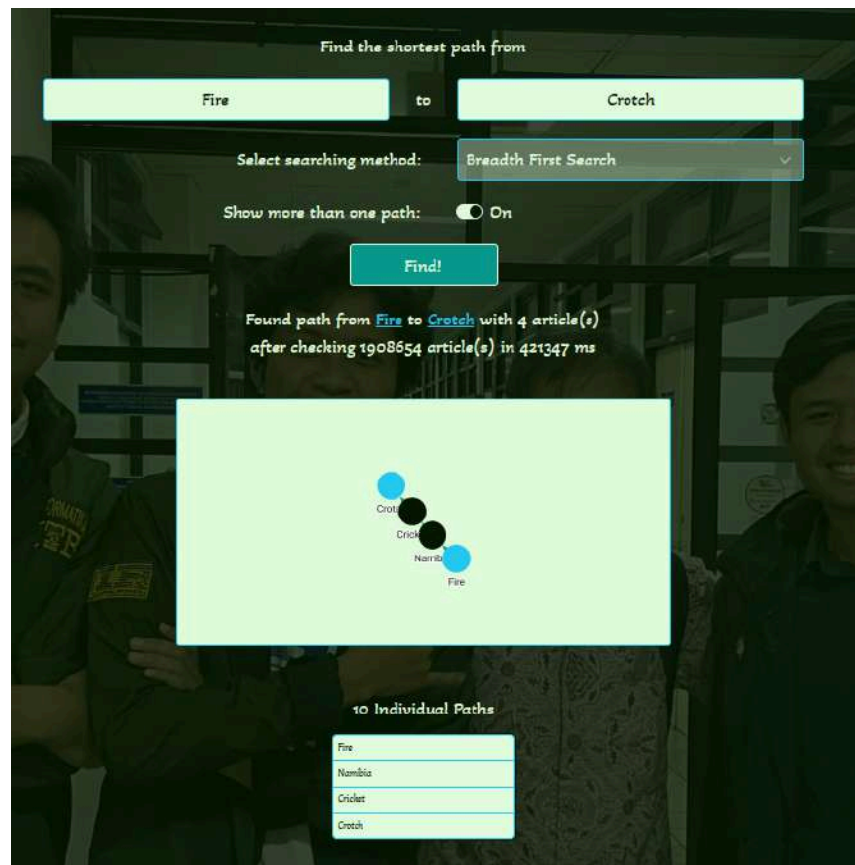
Found path from [Trigonometry](#) to [Adolf Hitler](#)
with 397 article(s) after checking 1412635
article(s) in 328348 ms

10 Individual Paths

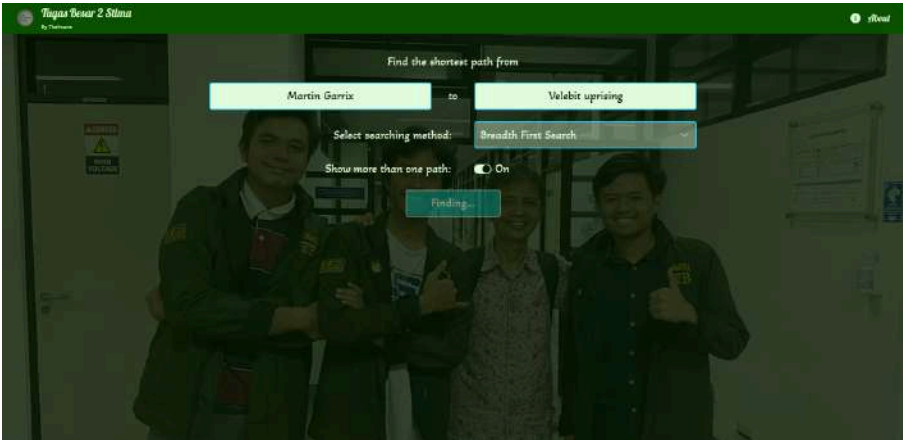
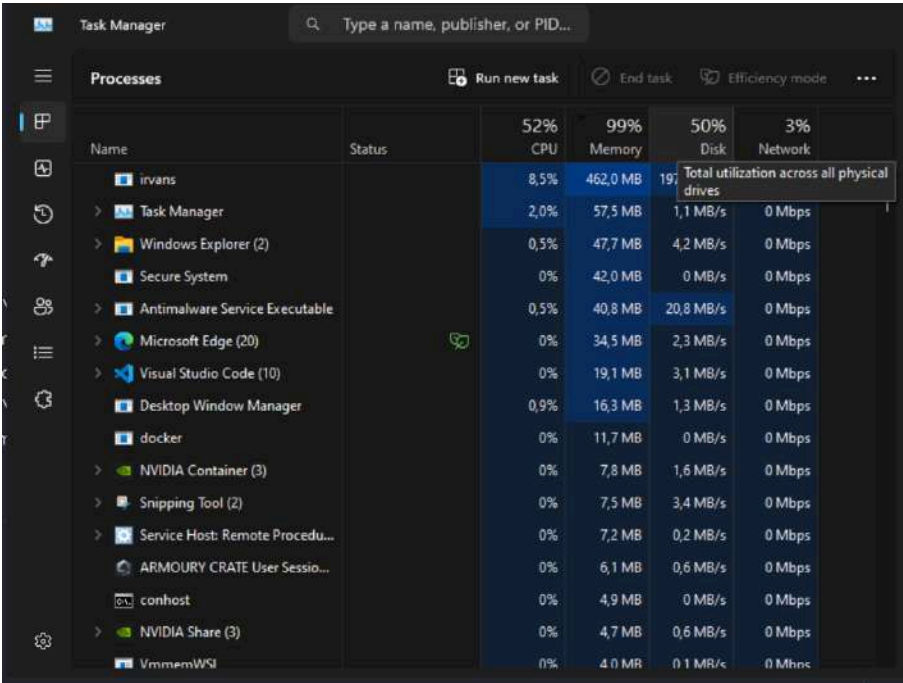
Trigonometry	Trigonometry	Trigonometry
History of geometry	History of geometry	Navigation
Alexander Grothendieck	Albert Einstein	Germany
Adolf Hitler	Adolf Hitler	Adolf Hitler
Trigonometry	Trigonometry	Trigonometry
Navigation	Navigation	Foundations of mathematics
Spain	World War II	Gottlob Frege
Adolf Hitler	Adolf Hitler	Adolf Hitler
Trigonometry	Trigonometry	Trigonometry
Foundations of mathematics	Foundations of mathematics	Foundations of mathematics
Gerhard Gentzen	List of paradoxes	Kurt Gödel
Adolf Hitler	Adolf Hitler	Adolf Hitler
	Trigonometry	
	Foundations of mathematics	
	Alan Turing	
	Adolf Hitler	

3

Fire -> Crotch: 421.347 ms



4. Depth 4

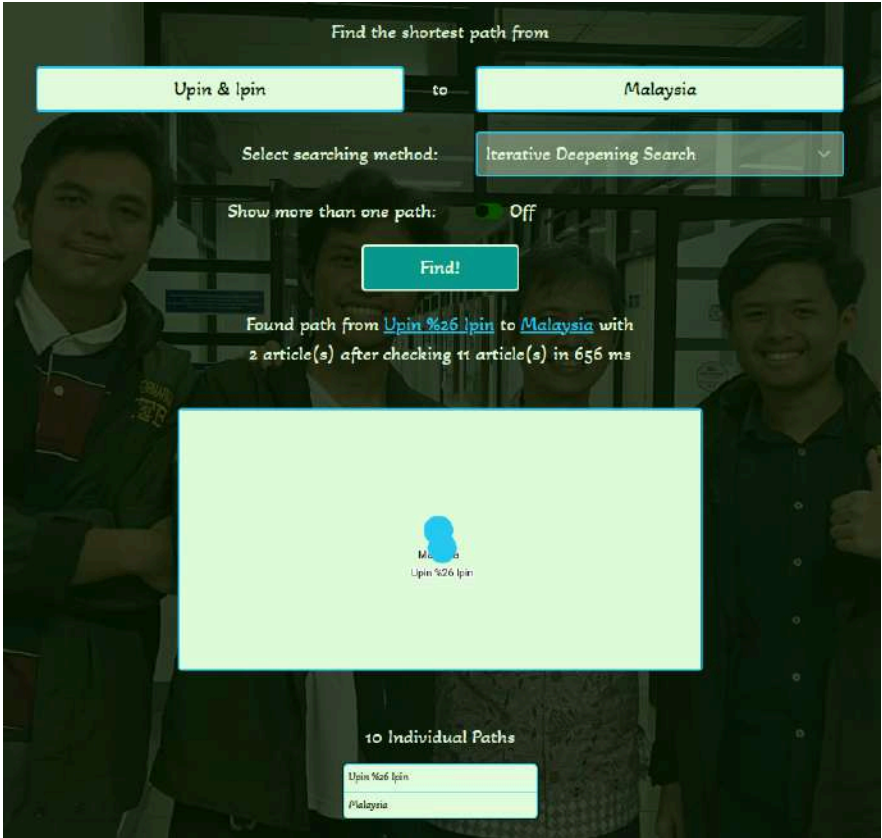
No.	Path
1	<div><div>Martin Garrix -> Velebit uprising: Null</div><div></div><div>Kapasitas memory penuh:</div><div></div><div>Program berhasil melakukan pencarian hingga depth 4:</div></div>

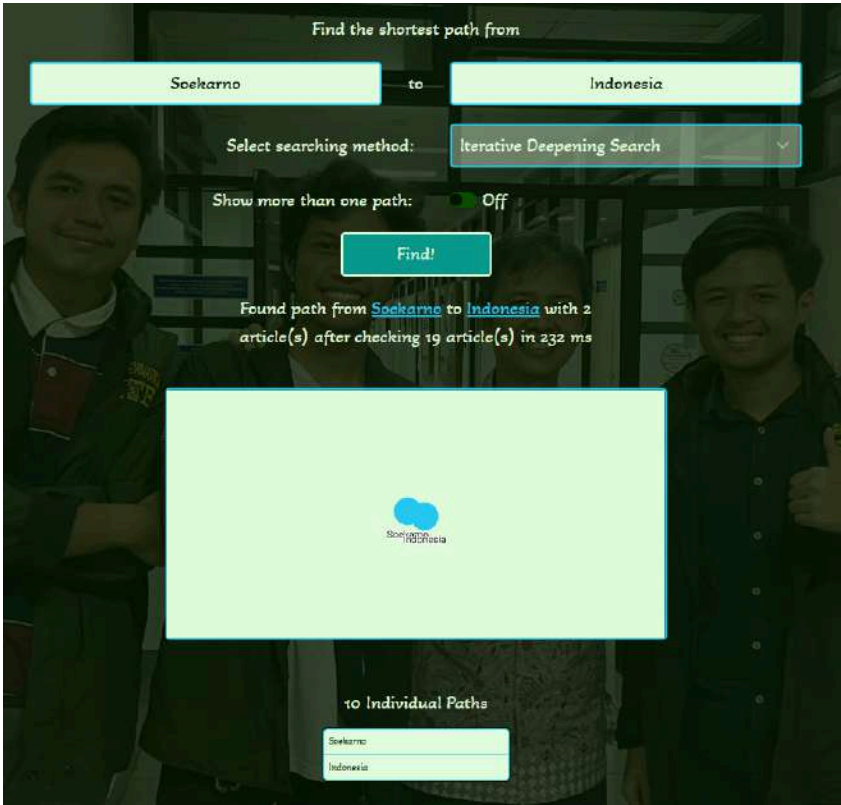
```
Found endURL when traversing!  
Finished Recursing  
Depth: 2  
[GIN] 2024/04/27 - 19:45:04 | 200 | 7.115881s | ::1 | POST | "/single/ids"  
Depth: 1  
Depth: 2  
Depth: 3  
Depth: 4
```

b. IDS (*Iterative Deepening Search*)

i. One Path

1. Depth 1

No.	Path
1	<p>Upin & Ipin -> Malaysia: 656 ms</p> 
2	<p>Soekarno -> Indonesia: 232 ms</p>

	 <p>Find the shortest path from</p> <p>Soekarno to Indonesia</p> <p>Select searching method: Iterative Deepening Search</p> <p>Show more than one path: Off</p> <p>Find!</p> <p>Found path from Soekarno to Indonesia with 2 article(s) after checking 19 article(s) in 232 ms</p> <p>10 Individual Paths</p> <p>Soekarno</p> <p>Indonesia</p>
3	Jokowi -> Gibran Rakabuming Raka: 193 ms


Find the shortest path from

to

Select searching method:

Show more than one path: ☐ Off

Found path from Jokowi to Gibran Rakabuming Raka with 2 article(s) after checking 16 article(s) in 193 ms

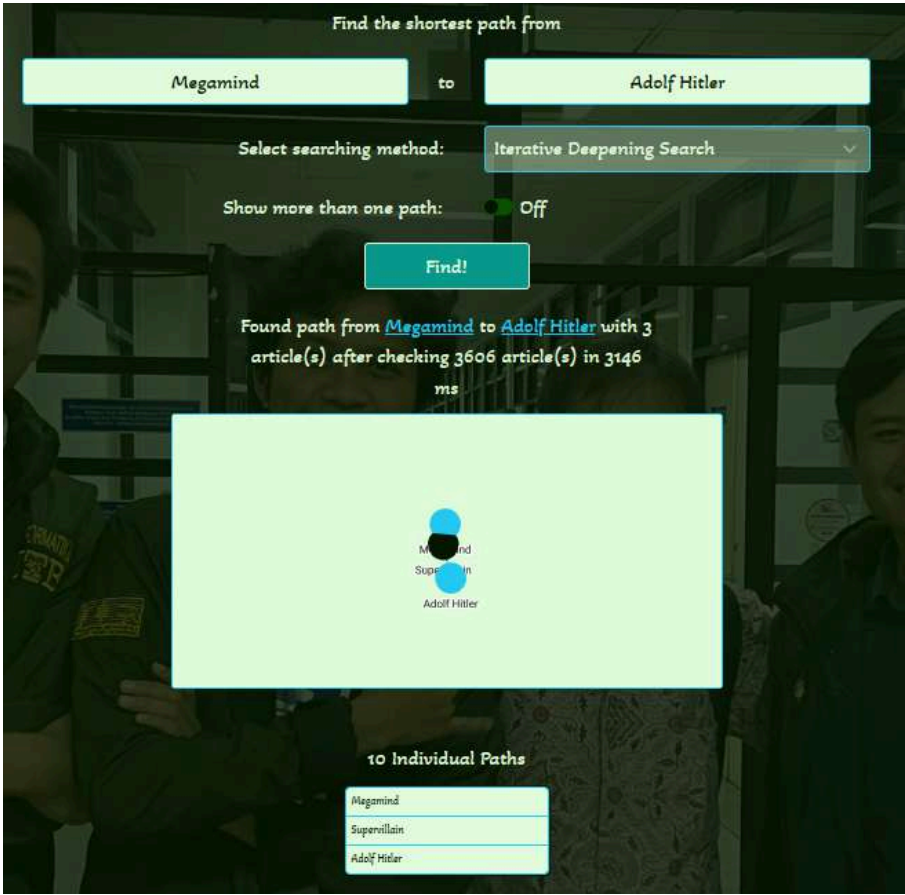


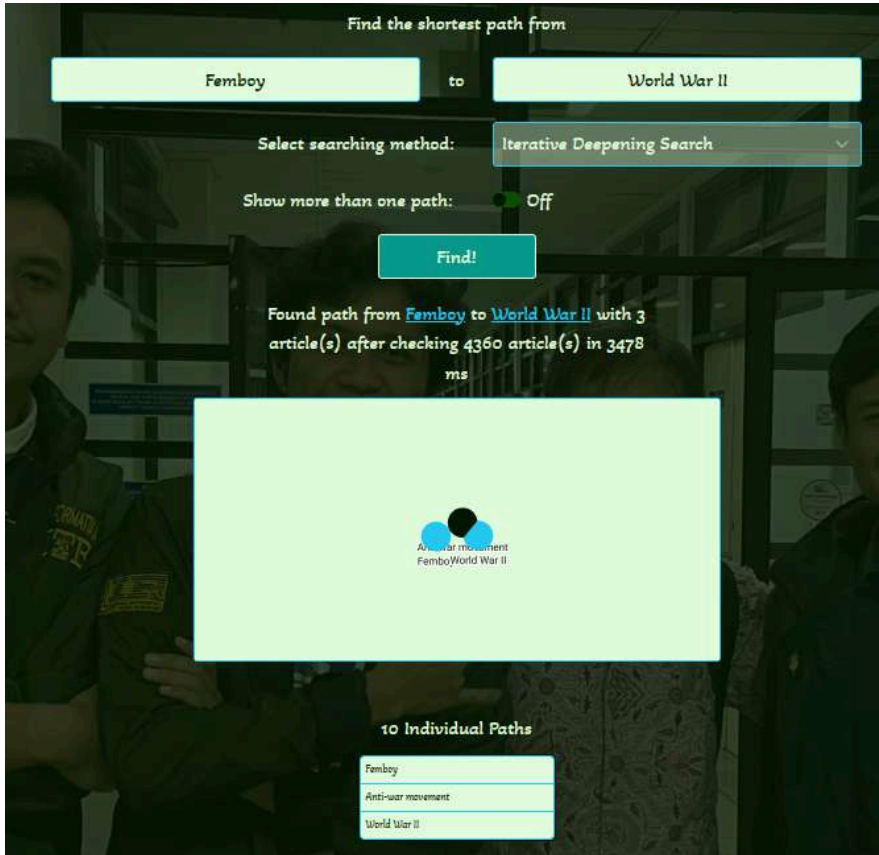
Gibran Rakabuming Raka

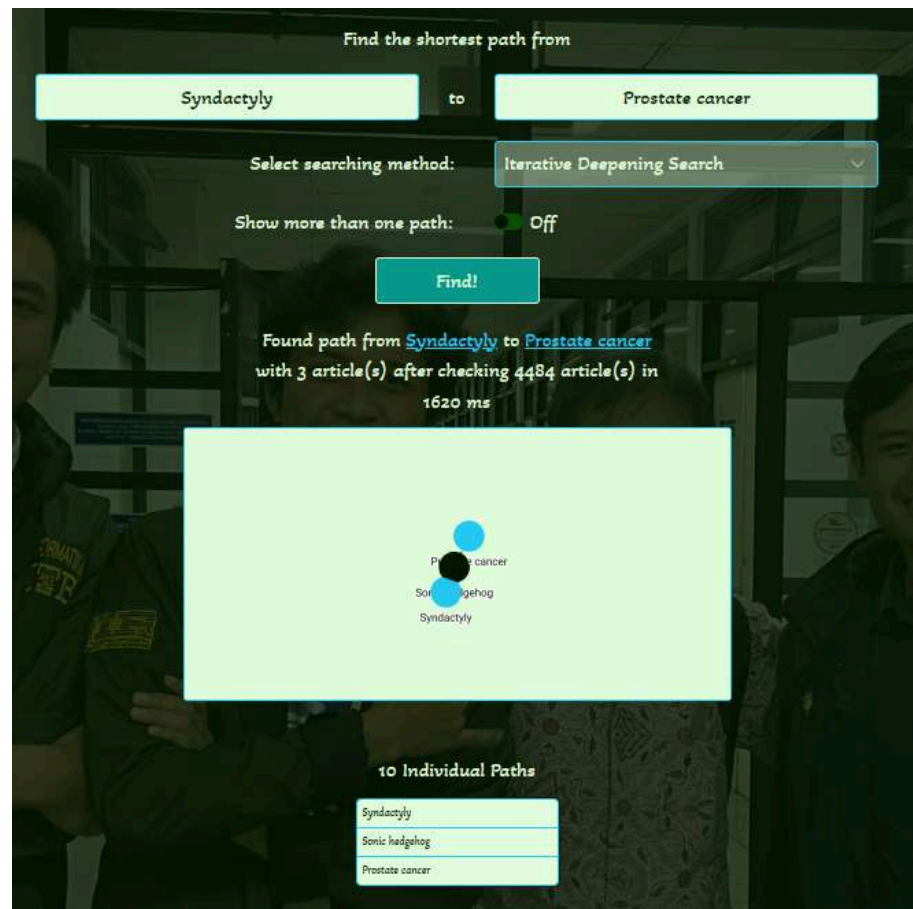
10 Individual Paths

Jokowi
Gibran Rakabuming Raka

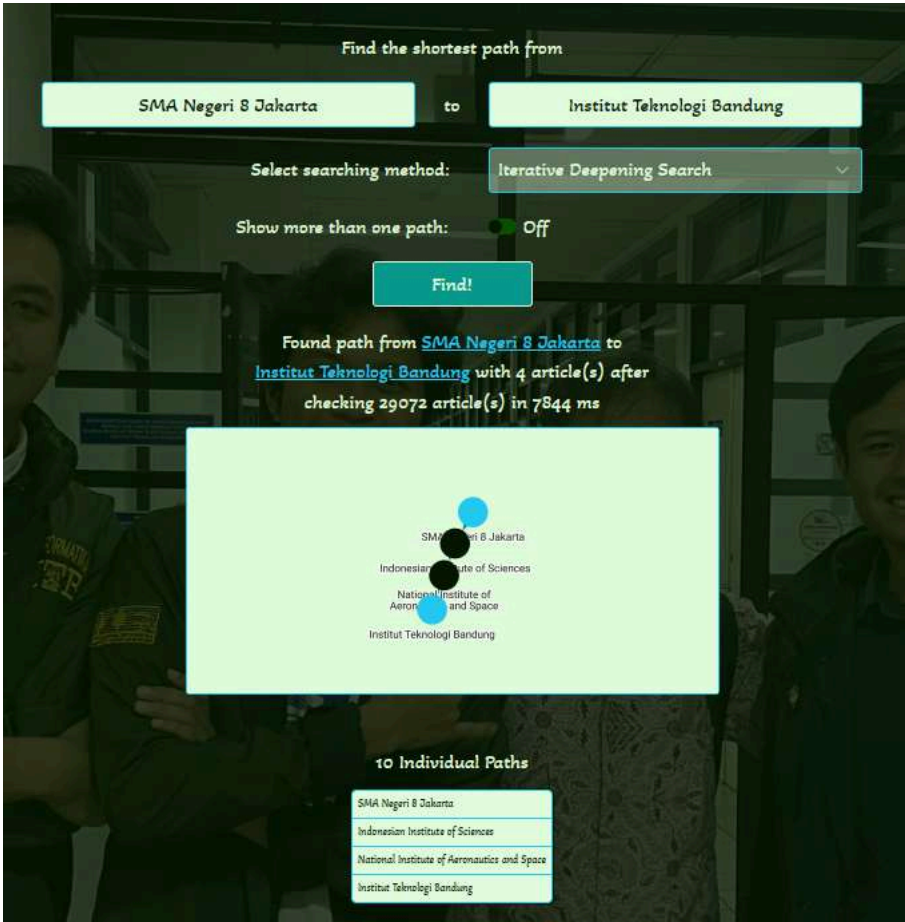
2. Depth 2

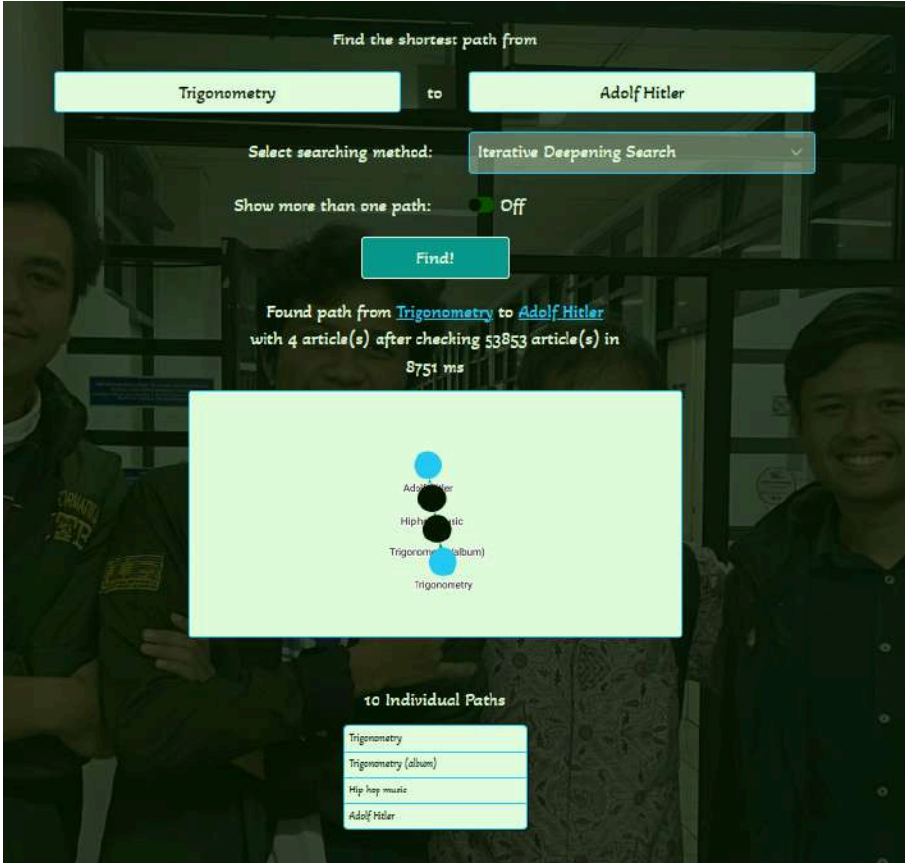
No.	Path
1	<p>Megamind -> Adolf Hitler: 3.146 ms</p> 
2	<p>Femboy -> World War II: 3.478 ms</p>

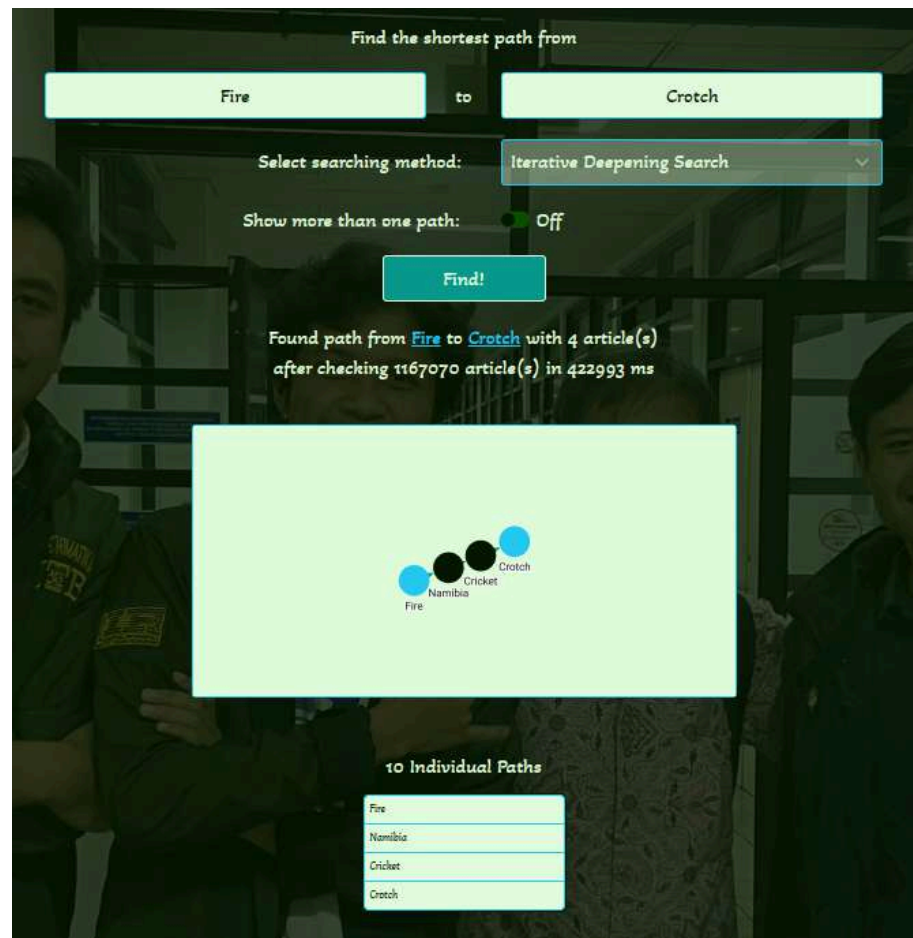
	 <p>The screenshot shows the Syndactyly search interface. At the top, it says "Find the shortest path from". Below this are two input fields: "Femboy" and "World War II", separated by a "to" label. Underneath, there's a "Select searching method:" dropdown menu set to "Iterative Deepening Search". A toggle switch for "Show more than one path:" is currently "Off". A blue "Find!" button is centered below these options. The results section states: "Found path from Femboy to World War II with 3 article(s) after checking 4360 article(s) in 3478 ms". Below this is a large white box containing a graph visualization with three nodes: "Femboy", "Anti-war movement", and "World War II", connected by lines. At the bottom, it says "10 Individual Paths" and lists the three nodes in a table.</p>
3	Syndactyly -> Prostate cancer: 4.484 ms



3. Depth 3

No.	Path
1	<p>SMA Negeri 8 Jakarta -> Institut Teknologi Bandung: 7.844 ms</p>  <p>Find the shortest path from</p> <p>SMA Negeri 8 Jakarta to Institut Teknologi Bandung</p> <p>Select searching method: Iterative Deepening Search</p> <p>Show more than one path: Off</p> <p>Find!</p> <p>Found path from SMA Negeri 8 Jakarta to Institut Teknologi Bandung with 4 article(s) after checking 29072 article(s) in 7844 ms</p> <p>Graph showing the path:</p> <ul style="list-style-type: none"> SMA Negeri 8 Jakarta (Blue dot) Indonesian Institute of Sciences (Black dot) National Institute of Aeronautics and Space (Black dot) Institut Teknologi Bandung (Blue dot) <p>10 Individual Paths</p> <ul style="list-style-type: none"> SMA Negeri 8 Jakarta Indonesian Institute of Sciences National Institute of Aeronautics and Space Institut Teknologi Bandung
2	<p>Trigonometry -> Adolf Hitler: 53.853 ms</p>

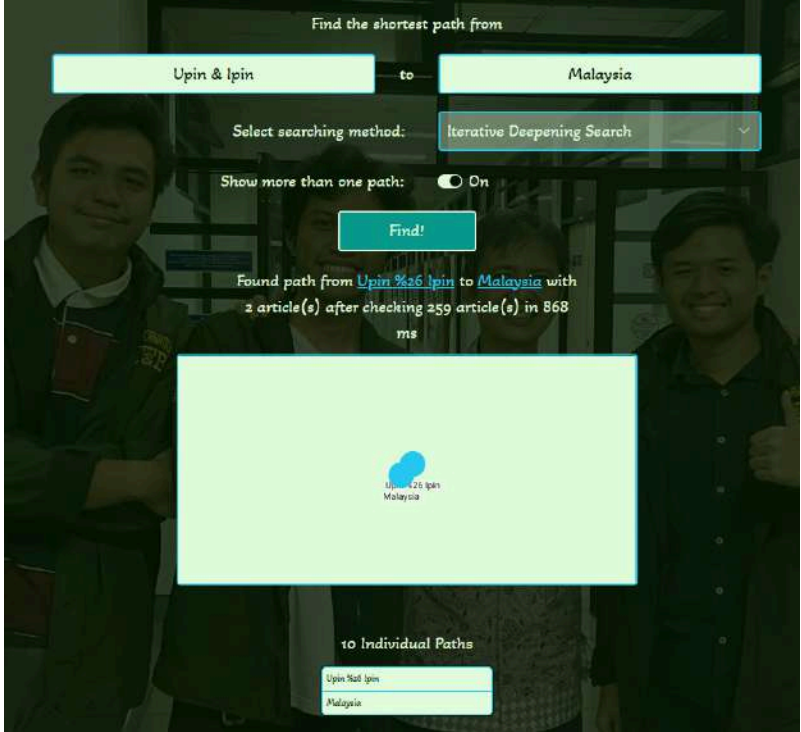
	 <p>The screenshot shows a web application interface for finding the shortest path between two terms. At the top, it says "Find the shortest path from". Below this are two input fields: "Trigonometry" and "Adolf Hitler", separated by the word "to". Underneath, there's a "Select searching method:" label with a dropdown menu set to "Iterative Deepening Search". Below that is a "Show more than one path:" label with a toggle switch set to "Off". A green "Find!" button is centered below these options. The results section states: "Found path from <u>Trigonometry</u> to <u>Adolf Hitler</u> with 4 article(s) after checking 53853 article(s) in 8751 ms". A central box displays a graph with four nodes: "Adolf Hitler" (blue circle), "Hip hop music" (black circle), "Trigonometry (album)" (black circle), and "Trigonometry" (blue circle). Lines connect "Adolf Hitler" to "Hip hop music", "Hip hop music" to "Trigonometry (album)", and "Trigonometry (album)" to "Trigonometry". Below the graph, it says "10 Individual Paths" and lists four items in a box: "Trigonometry", "Trigonometry (album)", "Hip hop music", and "Adolf Hitler".</p>
3	Fire -> Crotch: 422.933 ms

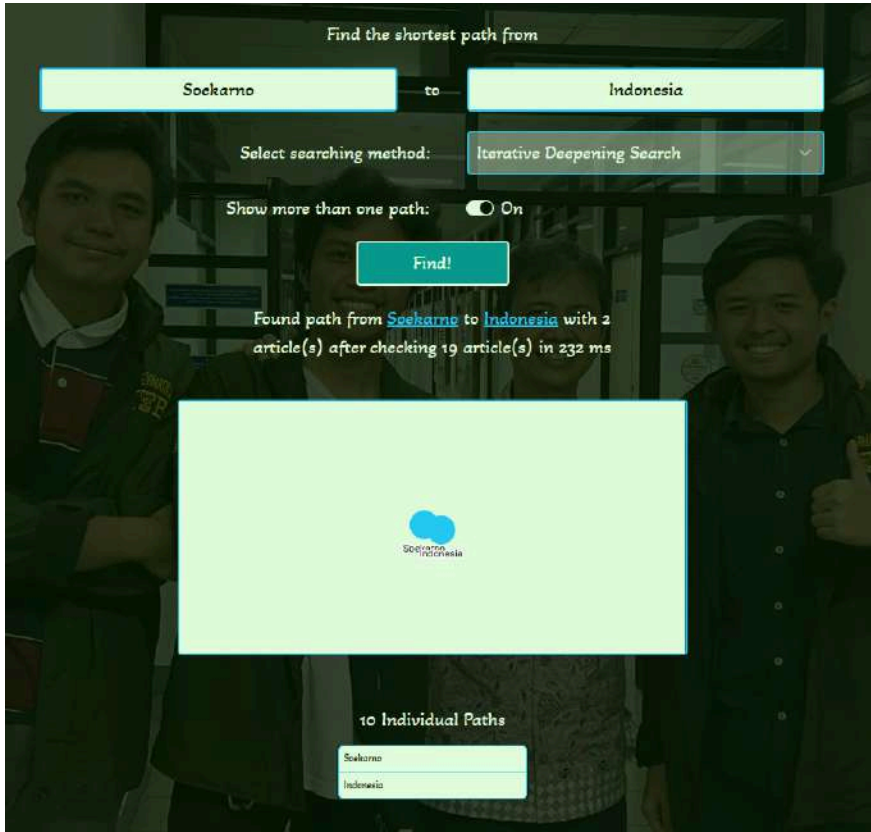


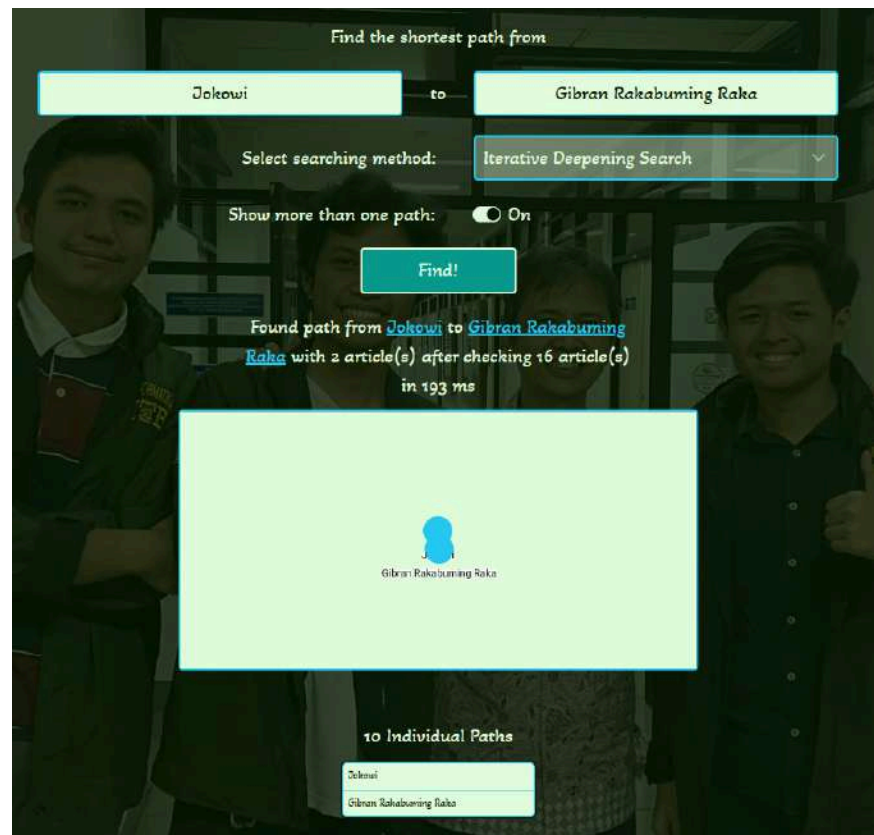
4. Depth 4

No.	Path
1	<p data-bbox="521 317 1133 352">Martin Garrix -> Velebit uprising: 640.341 ms</p> <div data-bbox="532 415 1414 1304"><p data-bbox="862 428 1081 449">Find the shortest path from</p><div data-bbox="573 474 1370 520"><div data-bbox="573 474 935 520">Martin Garrix</div> to <div data-bbox="1008 474 1370 520">Velebit uprising</div></div><p data-bbox="776 552 971 573">Select searching method:</p><div data-bbox="1008 541 1370 583">Iterative Deepening Search</div><p data-bbox="760 606 1068 630">Show more than one path: <input type="checkbox"/> Off</p><div data-bbox="894 653 1049 695">Find!</div><p data-bbox="784 720 1157 800">Found path from <u>Martin Garrix</u> to <u>Velebit uprising</u> with 5 article(s) after checking 2163587 article(s) in 640341 ms</p><div data-bbox="712 814 1230 1073"><pre>graph LR; A((Martin Garrix)) --- B((Scared to Be Lonely)); B --- C((New Year's Eve)); C --- D((League of Communists of Yugoslavia)); D --- E((Velebit uprising));</pre></div><p data-bbox="829 1127 1114 1148">Showing at most 10 individual paths</p><div data-bbox="875 1165 1065 1297"><div data-bbox="875 1165 1065 1186">Martin Garrix</div><div data-bbox="875 1192 1065 1213">Scared to Be Lonely</div><div data-bbox="875 1220 1065 1241">New Year's Eve</div><div data-bbox="875 1247 1065 1268">League of Communists of Yugoslavia</div><div data-bbox="875 1274 1065 1295">Velebit uprising</div></div></div>

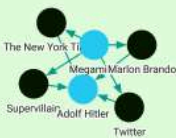
ii. More Than One Path
1. Depth 1

No.	Path
1	<p>Upin & Ipin -> Malaysia: 868 ms</p> 
2	Soekarno -> Indonesia: 232 ms

	
3	Jokowi -> Gibran Rakabuming Raka: 193 ms



2. Depth 2

No.	Path
1	<div><div>Megamind</div> to <div>Adolf Hitler</div><div>Select searching method: Iterative Deepening Search</div><div>Show more than one path: <input checked="" type="checkbox"/> On</div><div>Find!</div><div>Found path from Megamind to Adolf Hitler with 6 article(s) after checking 46966 article(s) in 4008 ms</div><div></div><div>10 Individual Paths</div><div><div><div>Megamind</div><div>Supervillain</div><div>Adolf Hitler</div></div><div><div>Megamind</div><div>The New York Times</div><div>Adolf Hitler</div></div><div><div>Megamind</div><div>Twitter</div><div>Adolf Hitler</div></div><div><div>Megamind</div><div>Marlon Brando</div><div>Adolf Hitler</div></div></div></div>
2	Femboy -> World War II: 3.547 ms

Find the shortest path from

to

Select searching method:

Show more than one path: ☒ On

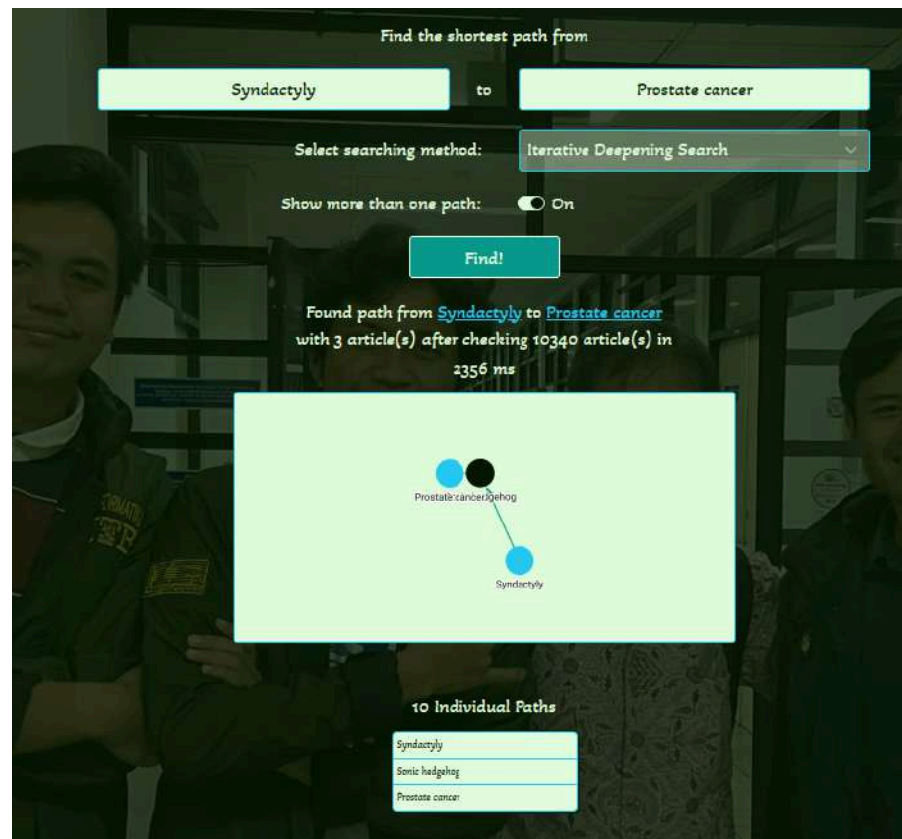
Found path from Femboy to World War II with 30 article(s) after checking 41812 article(s) in 3547 ms

10 Individual Paths

Femboy	Femboy	Femboy
Dress	BDSM	Femina
World War II	World War II	World War II
Femboy	Femboy	Femboy
Cross-dressing	Fruit (slang)	Kinsey scale
World War II	World War II	World War II
Femboy	Femboy	Femboy
Gender	Lesbian	Bohla
World War II	World War II	World War II
Femboy	Fuck	
	World War II	

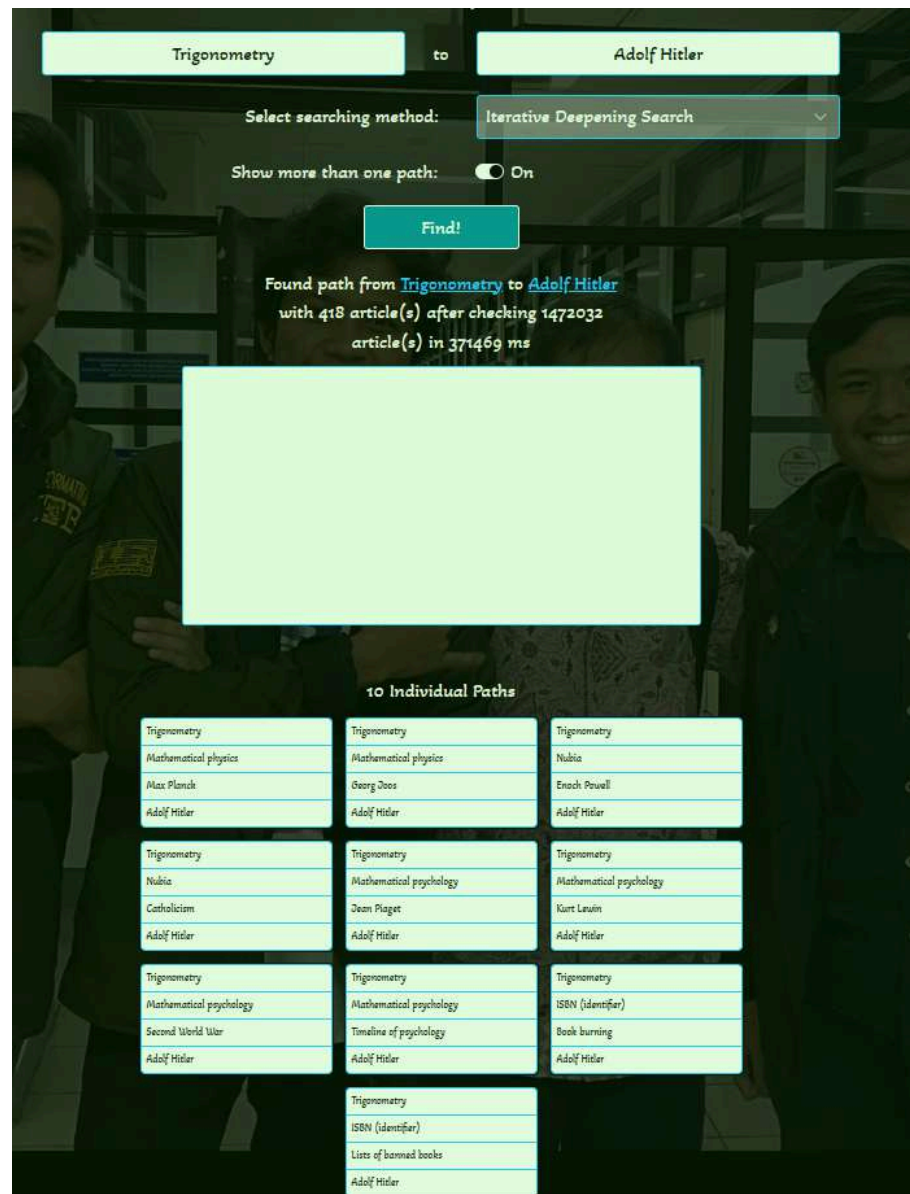
3

Syndactyly -> Prostate cancer: 2.356 ms



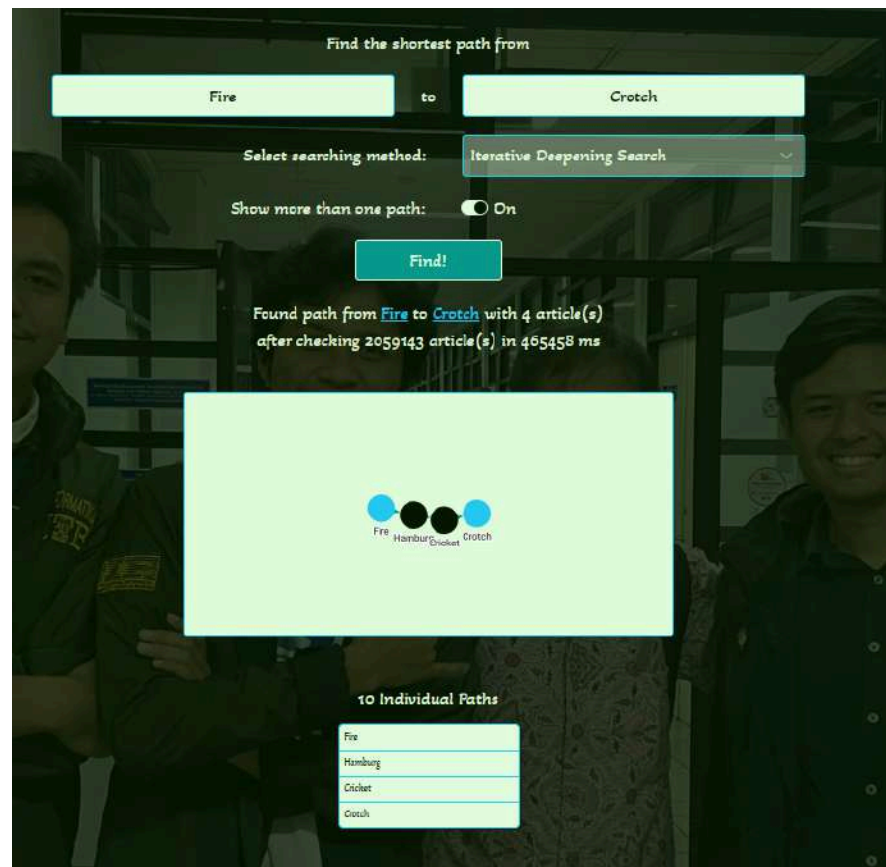
3. Depth 3

No.	Path
1	<div><div><div><div>Find the shortest path from</div><div><div>SMA Negeri 8 Jakarta</div>to<div>Institut Teknologi Bandung</div></div><div>Select searching method:<div>Iterative Deepening Search</div></div><div>Show more than one path:<div><div></div>On</div></div><div>Find!</div><div><div>Found path from SMA Negeri 8 Jakarta to Institut Teknologi Bandung with 12 article(s) after checking 1614593 article(s) in 453519 ms</div><div></div><div>10 Individual Paths</div><div><div><div>SMA Negeri 8 Jakarta</div><div>Indonesian Institute of Sciences</div><div>National Institute of Aeronautics and Space</div><div>Institut Teknologi Bandung</div></div><div><div>SMA Negeri 8 Jakarta</div><div>Bandung Institute of Technology</div><div>Rachmat Witoelar</div><div>Institut Teknologi Bandung</div></div><div><div>SMA Negeri 8 Jakarta</div><div>Indonesia National Science Olympiad</div><div>Cempasar</div><div>Institut Teknologi Bandung</div></div><div><div>SMA Negeri 8 Jakarta</div><div>University of Indonesia</div><div>Universiti Brunei Darussalam</div><div>Institut Teknologi Bandung</div></div><div><div>SMA Negeri 8 Jakarta</div><div>University of Indonesia</div><div>De La Salle Catholic University, Manado</div><div>Institut Teknologi Bandung</div></div><div><div>SMA Negeri 8 Jakarta</div><div>University of Indonesia</div><div>Astra Manufacturing Polytechnic</div><div>Institut Teknologi Bandung</div></div></div></div></div></div></div>
2	Trigonometry -> Adolf Hitler: 371.469 ms

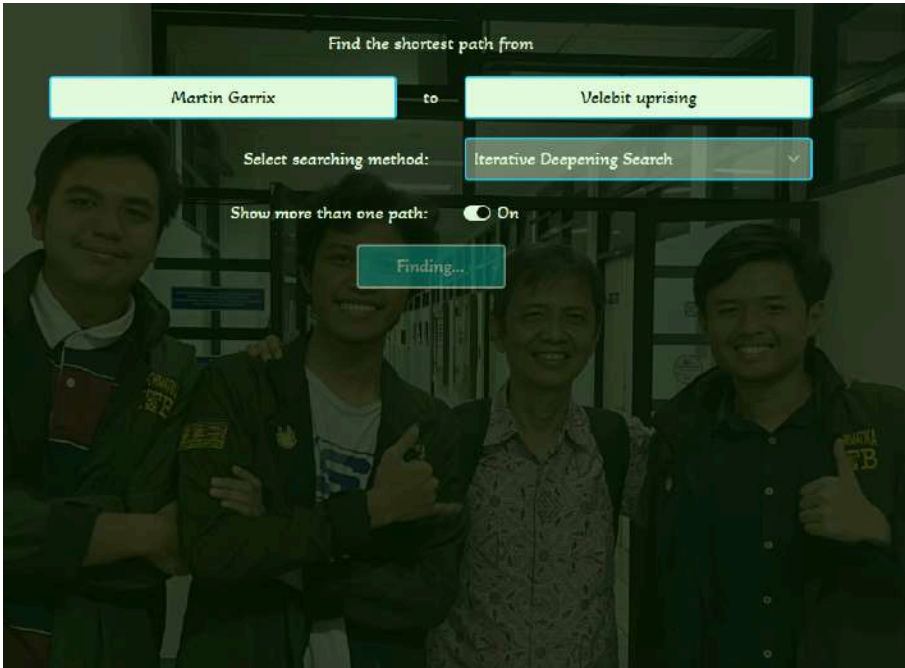


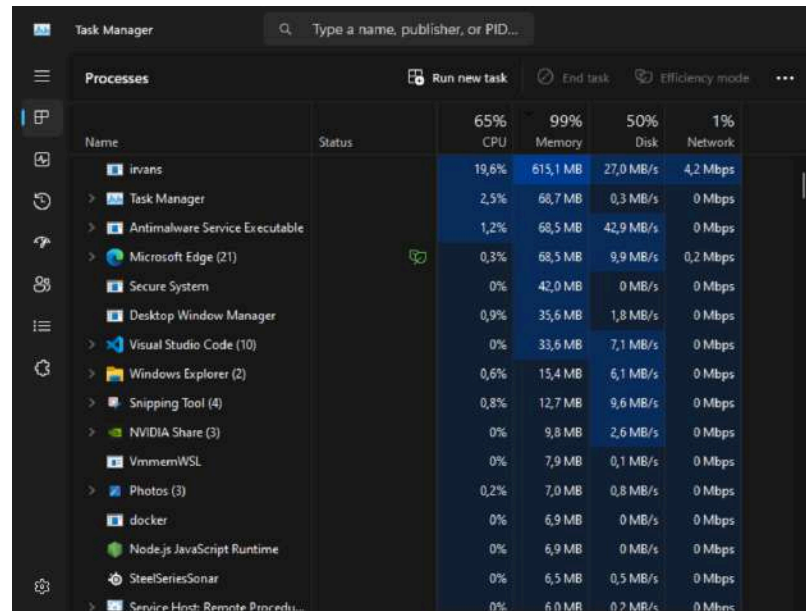
3

Fire -> Crotch: 465.458 ms



4. Depth 4

No.	Path
1	<div data-bbox="516 317 1419 365">Martin Garrix -> Velebit uprising: Null</div> <div data-bbox="522 415 1422 1079"></div> <div data-bbox="516 1121 862 1157">Kapasitas memory penuh:</div>



The screenshot shows the Windows Task Manager interface with the 'Processes' tab selected. The search bar at the top contains the text 'Type a name, publisher, or PID...'. The table below lists various running processes with their respective CPU, Memory, Disk, and Network usage.

Name	Status	CPU	Memory	Disk	Network
irvans		19,6%	615,1 MB	27,0 MB/s	4,2 Mbps
Task Manager		2,5%	68,7 MB	0,3 MB/s	0 Mbps
Antimalware Service Executable		1,2%	68,5 MB	42,9 MB/s	0 Mbps
Microsoft Edge (21)		0,3%	68,5 MB	9,9 MB/s	0,2 Mbps
Secure System		0%	42,0 MB	0 MB/s	0 Mbps
Desktop Window Manager		0,9%	35,6 MB	1,8 MB/s	0 Mbps
Visual Studio Code (10)		0%	33,6 MB	7,1 MB/s	0 Mbps
Windows Explorer (2)		0,6%	15,4 MB	6,1 MB/s	0 Mbps
Snipping Tool (4)		0,8%	12,7 MB	9,6 MB/s	0 Mbps
NVIDIA Share (3)		0%	9,8 MB	2,6 MB/s	0 Mbps
VmmemWSL		0%	7,9 MB	0,1 MB/s	0 Mbps
Photos (3)		0,2%	7,0 MB	0,8 MB/s	0 Mbps
docker		0%	6,9 MB	0 MB/s	0 Mbps
Node.js JavaScript Runtime		0%	6,9 MB	0 MB/s	0 Mbps
SteelSeriesSonar		0%	6,5 MB	0,5 MB/s	0 Mbps
Service Host: Remote Proce...		0%	6,0 MB	0,2 MB/s	0 Mbps

Program berhasil menemukan 3 path:

```

Recurring
Finished Recursing
Depth: 1
Recurring
Finished Recursing
Depth: 2
Recurring
Finished Recursing
Depth: 3
Found endURL when querying!
Found endURL when querying!
Found endURL when querying!

```


4.4. Analisis Hasil Pengujian

Dari hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa solusi tunggal selalu memiliki waktu eksekusi yang lebih cepat dibandingkan dengan solusi banyak, baik pada algoritma Breadth-First Search (BFS) maupun Iterative Deepening Search (IDS). Fenomena ini terjadi karena pada solusi tunggal, pencarian dihentikan setelah menemukan minimal satu solusi sehingga program dapat langsung mengembalikan jalur solusi tersebut. Sebaliknya, pada solusi banyak, pencarian akan terus dilakukan hingga semua solusi ditemukan, yang menyebabkan waktu eksekusi lebih lama.

Kompleksitas waktu untuk algoritma BFS adalah $O(b^d)$ dengan b adalah maksimum percabangan dari suatu simpul dan d adalah kedalaman dari solusi terbaik. Di sisi lain, kompleksitas waktu untuk algoritma IDS adalah $O(b^d)$ dengan definisi b dan d sama seperti algoritma BFS. Dapat dilihat bahwa kedua algoritma secara teori akan memiliki performa yang sama satu sama lain. Kenyataannya, sesuai hasil pengujian, didapatkan algoritma BFS memiliki waktu eksekusi yang lebih cepat dibandingkan dengan algoritma IDS.

Perbedaan performa teori dengan hasil pengujian dapat dijelaskan dari beberapa aspek. Pencarian dengan algoritma BFS memiliki waktu eksekusi rata-rata yang lebih cepat dibandingkan dengan algoritma IDS. Hal ini karena pencarian dengan algoritma BFS akan mencari solusi berdasarkan banyaknya cabang pada kedalaman tertentu, sedangkan algoritma IDS akan mencari solusi dengan mengecek masing-masing path yang dibentuk dari suatu graf lalu dilakukan backtracking berdasarkan jumlah depth yang ditentukan. Hal ini tentunya akan membuat waktu eksekusi rata-rata dari algoritma BFS menjadi lebih cepat. Algoritma IDS lebih unggul jika solusi berada pada path pertama yang ditelusuri.

Pencarian dengan algoritma IDS juga memiliki redundansinya tersendiri. Setiap memperdalam pencarian, algoritma IDS akan mengulangi pencarian pada kedalaman sebelumnya sehingga akan terlihat tidak efisien dibandingkan dengan algoritma BFS. Walaupun notasi big-O nya sama dengan algoritma BFS, pada kenyataannya $T(n)$ pada algoritma IDS akan lebih besar dari algoritma BFS karena akan ada pencarian berulang pada tiap kedalaman yang ditelusuri. Perbedaan performa ini akan semakin terasa seiring bertambahnya kedalaman pencarian.

Pada kedalaman empat, pencarian dengan metode multiple path dengan algoritma IDS dan BFS akan memerlukan memory yang sangat banyak sehingga menyebabkan process di-kill secara otomatis. Hal ini disebabkan oleh banyaknya artikel yang dikunjungi, yaitu sekitar sepertiga dari banyaknya artikel di wikipedia, yang mengakibatkan terjadinya stack overflow.

Bab 5

Kesimpulan, Saran, dan Refleksi

5.1. Kesimpulan

Pada kasus kedalaman yang kurang dari empat, algoritma BFS membutuhkan waktu yang lebih sedikit dibandingkan dengan IDS. Selain itu, solusi tunggal juga membutuhkan waktu yang lebih sedikit daripada solusi banyak.

5.2. Saran

Saran untuk kami sendiri, sebaiknya program diadakan *caching* sehingga tidak perlu dilakukan peng-*query*-an kembali pada link wikipedia yang sudah pernah di-*query*. Saran untuk asisten, sebaiknya tidak melakukan revisi QnA pada H-3 deadline dan walaupun ada revisi QnA, mohon dibuat pengumuman khusus di suatu tempat yang berbeda agar mudah di-*notice*.

5.3. Refleksi

Tugas besar ini membantu kami untuk mengeksplor pengembangan web terutama dengan library React. Melalui tugas besar ini kami juga dapat mengeksplorasi program yang menggunakan multi proses seperti goroutine.

Lampiran

- Tautan Repository GitHub:
https://github.com/SandWithCheese/Tubes2_TheIrvans
- Tautan Video:
<https://youtu.be/w5tLoWAn1vU>
- Tautan Figma:
<https://www.figma.com/file/CAM35iyMo1gAgVUSz8bXrU/Tubes-Dua-Stima?type=design&node-id=0%3A1&mode=design&t=4IKADuK7KGM4YLcT-1>

Daftar Pustaka

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2008-2009/Makalah2008/Makalah0809-054.pdf>

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

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf>