

**LAPORAN TUGAS BESAR 2**  
**PEMANFAATAN ALGORITMA BFS DAN DFS DALAM**  
**PENCARIAN RECIPE PADA PERMAINAN *LITTLE***  
***ALCHEMY 2***

IF2211– Strategi Algoritma

Kelompok “*Jepangor*”



**Dosen:**

Dr. Ir. Rinaldi, M.T.

Dr. Nur Ulfa Maulidevi, S.T, M.Sc

Monterico Adrian, S.T., M.T.

**Anggota Kelompok:**

Farrell Jabaar Altafataza - 10122057

Muhammad Raihaan Perdana - 13523124

Ardell Aghna Mahendra - 13523151

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

**2025**

# **Daftar Isi**

<b>Daftar Isi.....</b>	<b>1</b>
<b>BAB I: DESKRIPSI TUGAS.....</b>	<b>2</b>
<b>BAB II: LANDASAN TEORI.....</b>	<b>3</b>
Graph Transversal.....	3
Breadth-First Search (BFS).....	3
Depth-First Search (DFS).....	3
<b>BAB III: ANALISIS PEMECAHAN MASALAH.....</b>	<b>4</b>
Fitur Fungsional.....	4
Arsitektur Aplikasi Web.....	5
Contoh Kasus.....	6
<b>BAB IV: IMPLEMENTASI DAN PENGUJIAN.....</b>	<b>8</b>
Spesifikasi Teknis Program.....	8
Struktur Data.....	8
Komponen Frontend.....	10
Komponen Backend.....	11
Tata Cara Penggunaan Program.....	12
Hasil Pengujian.....	15
1. Pengujian Elemen “Hangar”.....	15
2. Pengujian Elemen “Ham”.....	20
3. Pengujian Elemen “Swimming Pool”.....	25
Analisis Hasil Pengujian.....	30
<b>BAB V: PENUTUP.....</b>	<b>32</b>
Kesimpulan.....	32
Saran.....	32
Refleksi.....	32
<b>LAMPIRAN.....</b>	<b>33</b>
<b>DAFTAR PUSTAKA.....</b>	<b>34</b>

## **BAB I: DESKRIPSI TUGAS**

Little Alchemy 2 merupakan permainan berbasis web / aplikasi yang dikembangkan oleh Recloak yang dirilis pada tahun 2017, permainan ini bertujuan untuk membuat 720 elemen dari 4 elemen dasar yang tersedia yaitu air, earth, fire, dan water. Permainan ini merupakan sekuel dari permainan sebelumnya yakni Little Alchemy 1 yang dirilis tahun 2010. Mekanisme dari permainan ini adalah pemain dapat menggabungkan kedua elemen dengan melakukan drag and drop, jika kombinasi kedua elemen valid, akan memunculkan elemen baru, jika kombinasi tidak valid maka tidak akan terjadi apa-apa. Permainan ini tersedia di web browser, Android atau iOS Pada Tugas Besar pertama Strategi Algoritma ini, mahasiswa diminta untuk menyelesaikan permainan Little Alchemy 2 ini dengan menggunakan strategi Depth First Search dan Breadth First Search.

Komponen-komponen dari permainan ini antara lain:

### 1. Elemen dasar

Dalam permainan Little Alchemy 2, terdapat 4 elemen dasar yang tersedia yaitu water, fire, earth, dan air, 4 elemen dasar tersebut nanti akan di-combine menjadi elemen turunan yang berjumlah 720 elemen.

### 2. Elemen turunan

Terdapat 720 elemen turunan yang dibagi menjadi beberapa tier tergantung tingkat kesulitan dan banyak langkah yang harus dilakukan. Setiap elemen turunan memiliki recipe yang terdiri atas elemen lainnya atau elemen itu sendiri.

### 3. Combine Mechanism

Untuk mendapatkan elemen turunan pemain dapat melakukan combine antara 2 elemen untuk menghasilkan elemen baru. Elemen turunan yang telah didapatkan dapat digunakan kembali oleh pemain untuk membentuk elemen lainnya.

## BAB II: LANDASAN TEORI

### Graph Transversal

Penjelajahan graf (*graph traversal*) adalah proses sistematis untuk mengunjungi semua simpul (vertex) dan sisi (edge) dalam sebuah graf. Tujuan dari penjelajahan ini dapat bervariasi, seperti mencari jalur terpendek, mendeteksi siklus, atau membangun struktur data tambahan. Dua algoritma dasar yang umum digunakan untuk penjelajahan graf adalah Breadth-First Search (BFS) dan Depth-First Search (DFS).

#### Breadth-First Search (BFS)

Breadth-First Search (BFS) adalah algoritma penjelajahan graf yang menjelajahi semua simpul pada tingkat kedalaman tertentu sebelum melanjutkan ke tingkat berikutnya. Algoritma ini menggunakan struktur data *queue* untuk melacak simpul yang akan dikunjungi. BFS sangat efektif untuk menemukan jalur terpendek dalam graf tak berbobot.

#### Depth-First Search (DFS)

Depth-First Search (DFS) adalah algoritma penjelajahan graf yang menjelajahi sejauh mungkin sepanjang cabang sebelum melakukan *backtracking*. Algoritma ini dapat diimplementasikan secara rekursif atau menggunakan struktur data *stack*. DFS berguna untuk aplikasi seperti deteksi siklus dan topological sorting.

Pada tugas kali ini, akan dibuat sebuah aplikasi berbasis web untuk pencarian *recipe* elemen dalam permainan Little Alchemy 2 dengan menggunakan strategi BFS dan DFS. Aplikasi ini menawarkan opsi untuk menemukan resep terpendek atau banyaknya resep menuju suatu elemen tertentu. Setelah itu, aplikasi akan mengeluarkan visualisasi dari resep yang ditemukan sebagai sebuah *tree* yang menunjukkan kombinasi elemen yang diperlukan dari elemen dasar.

## BAB III: ANALISIS PEMECAHAN MASALAH

Berikut merupakan langkah-langkah yang dilakukan dalam analisis pemecahan masalah dari tugas ini:

### 1. Pengumpulan Data

Data elemen dan *recipe* diambil dari Fandom Little Alchemy 2 dengan web scraping menggunakan library goquery di Golang. Data disimpan dalam format JSON dan menggunakan struktur data map dengan nama element disimpan dalam “Key” dan daftar resep disimpan dalam “Value”.

### 2. Pemetaan Masalah

Permasalahan dipetakan menjadi sebuah graf berarah dimana *node* pada graf merupakan pemetaan dari elemen, *edge* merupakan hubungan kombinasi dari dua elemen, dan empat elemen dasar (air, api, tanah, dan udara) menjadi *root node* dari graf. Pemetaan ini dilakukan agar permasalahan tersebut dapat diselesaikan dengan metode algoritma BFS, DFS dan Bidirectional Search.

### 3. Algoritma Pencarian Resep

Algoritma pencarian yang digunakan adalah BFS, DFS, dan Bidirectional Search.

Algoritma BFS akan menelusuri jalur dengan pendekatan level-by-level dimulai dari *node* awal atau elemen-elemen dasar. Algoritma ini diimplementasikan dengan struktur data *queue* untuk mengeksplorasi node secara berlapis. Setiap node dalam antrian menyimpan informasi tentang jalur kombinasi yang sudah dilalui serta daftar elemen yang sudah dikunjungi untuk menghindari pengulangan. BFS mengeksplorasi *node* pada tingkat kedalaman yang sama terlebih dahulu sebelum melanjutkan ke tingkat yang lebih dalam, yang membuatnya efektif dalam menemukan jalur dengan jumlah langkah kombinasi paling sedikit.

Sementara itu, Algoritma DFS akan menelusuri satu jalur hingga selesai sebelum kembali dan mencoba jalur lainnya. Algoritma ini diimplementasikan secara rekursif dengan pendekatan penelusuran mendalam pada setiap cabang sebelum beralih ke cabang lainnya. Karena sifatnya yang mengeksplorasi setiap kemungkinan kombinasi secara menyeluruh, DFS lebih cocok untuk menemukan berbagai variasi resep.

Selanjutnya, Algoritma Bidirectional Search secara bersamaan menggabungkan pencarian dari dua arah yaitu dari elemen dasar menuju target dan dari target menuju elemen dasar. Ketika kedua jalur bertemu pada satu *node*, sistem menggabungkan kedua jalur tersebut menjadi resep yang utuh. Metode ini dapat secara signifikan meningkatkan waktu pencarian untuk graf yang besar.

## Fitur Fungsional

### 1. Pemilihan Elemen Target

Pengguna dapat memilih elemen yang ingin diketahui resep kombinasinya antara dua elemen lain dari daftar elemen yang tersedia.

## 2. Pemilihan Algoritma Pencarian

Aplikasi menyediakan opsi kepada pengguna untuk memilih algoritma pencarian yang diinginkan, yaitu Bread-First Search (BFS), Depth-First Search (DFS), dan Bidirectional Search. Pilihan ini dapat disesuaikan dengan strategi yang diinginkan pengguna, baik mencari jalur terpendek maupun mengeksplorasi banyaknya resep maksimal yang diinginkan pengguna.

## 3. Mode Pencarian

Pengguna juga bisa memilih mode pencarian yang tersedia, yaitu pencarian jalur terpendek atau beberapa jalur berbeda sebanyak maksimal resep yang diinginkan. Pilihan tersebut dapat dipilih melalui *toggle* interaktif yang disediakan. Jika mode pencarian yang dipilih adalah banyaknya resep, maka pengguna dapat memasukkan banyak maksimal resep yang ingin ditemukan melalui input tambahan.

## 4. Visualisasi dan Informasi Hasil

Setelah melakukan pencarian, aplikasi akan menampilkan hasil berupa jumlah resep, jumlah *node* yang dikunjungi, waktu pencarian, daftar resep yang ditemukan, dan visualisasi proses pembentukan elemen.

# Arsitektur Aplikasi Web

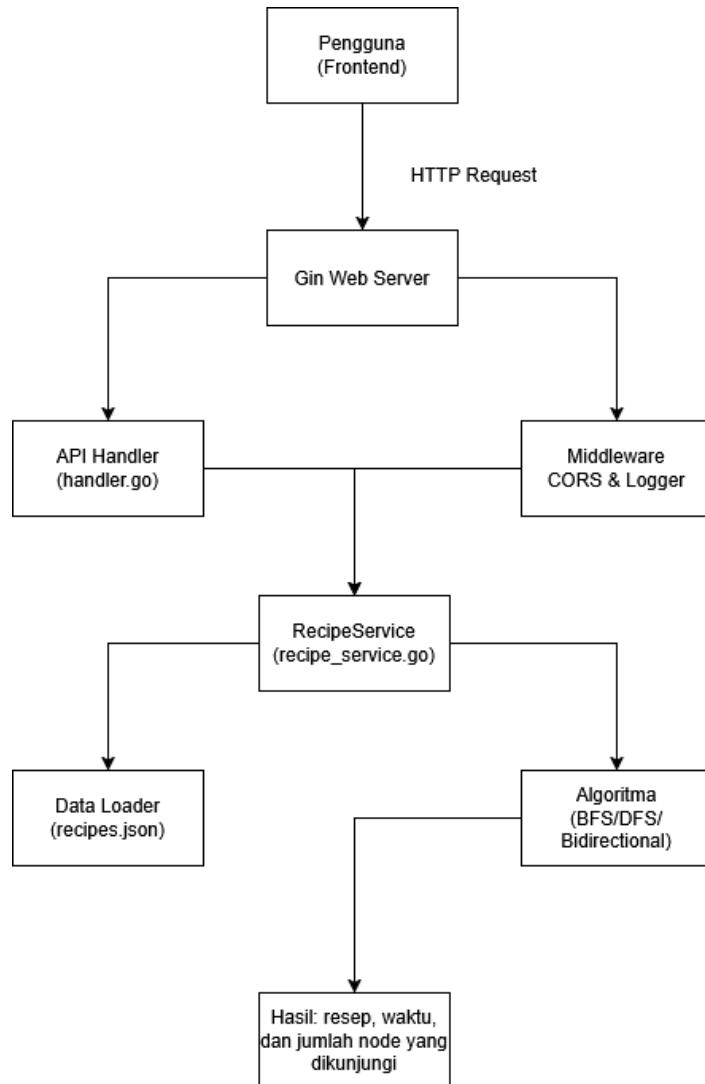
## 1. Frontend

Bagian frontend dibuat menggunakan framework [Next.js](#) dengan TypeScript. *Interface* pengguna meliputi beberapa komponen seperti SearchForm, ElementSelector, dan AlgorithmSelector untuk mengatur input yang diinginkan pengguna. Setelah pengguna melakukan input parameter, frontend akan mengirimkan permintaan kepada HTTP POST kepada backend melalui endpoint /api/find-recipes, lalu menampilkan hasil menggunakan komponen ResultInfo dan RecipeTree.

## 2. Backend

Backend dibuat menggunakan bahasa pemrograman Go dan framework Gin. Terdapat 3 algoritma pencarian (BFS, DFS, Bidirectional) yang diimplementasikan. Backend juga

memuat file data resep yang diolah menjadi struktur graf. Setelah frontend mengirimkan permintaan, backend akan menjalankan proses pencarian sesuai parameter yang diberikan dan memberikan output dalam bentuk JSON kepada frontend.

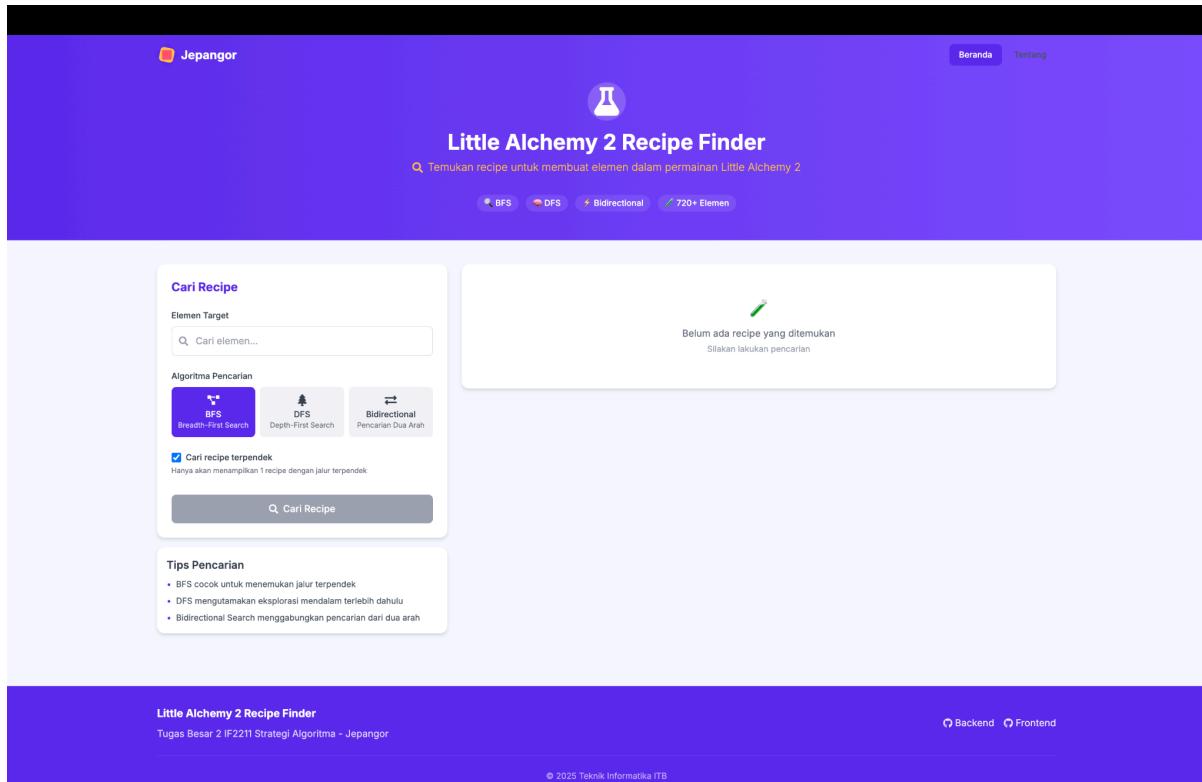


## Contoh Kasus

Misalkan terdapat suatu pengguna yang ingin mencari resep terpendek dari elemen “Brick” dan memilih algoritma BFS, maka sistem akan memulai pencarian dari elemen-elemen dasar lalu

menelusuri secara level-by-level untuk menemukan kombinasi yang terpendek. Setelah proses selesai, akan ditemukan bahwa “Brick” dapat dibentuk dengan dua langkah kombinasi yaitu membentuk “Mud” dari kombinasi “Water” dan “Earth”, lalu membentuk “Brick” dari kombinasi “Mud” dan “Fire”. Setelah menemukan hasil tersebut, sistem akan menghentikan pencarian karena pengguna hanya meminta resep terpendek. Sistem akan mengeluarkan *output* berupa urutan kombinasi yang diperlukan, jumlah *node* yang dikunjungi, dan waktu eksekusi.

## BAB IV: IMPLEMENTASI DAN PENGUJIAN



### Spesifikasi Teknis Program

#### Struktur Data

##### 1. Graph (model.Graph)

```
type Graph map[string] [] [] string
```

Struktur utama yang merepresentasikan graf elemen dan kombinasinya. Map dengan key berupa nama elemen dan value berupa array dari kombinasi (pasangan elemen) yang dapat membentuk elemen tersebut

Contoh: "Brick" -> [["Mud", "Fire"], ["Clay", "Stone"]]

##### 2. Tier Mapping (model.TierMap)

```
type TierMap map[string] int
```

Menyimpan tingkatan (tier) dari setiap elemen untuk validasi kombinasi. Implementasinya adalah Map dengan key berupa nama elemen dan value berupa level tier. Kegunaannya adalah memastikan bahwa elemen dengan tier lebih tinggi tidak digunakan untuk membuat elemen dengan tier lebih rendah.

##### 3. Recipe (model.Recipe)

```

type Recipe struct {
    ID int `json:"id"`
    Nodes []RecipeNode `json:"nodes"`
    Links []RecipeLink `json:"links"`
}

```

Merepresentasikan resep lengkap dengan struktur tree. Komponen dalam struktur ini adalah: ID (Pengidentifikasi unik untuk recipe), Nodes (Kumpulan node yang merepresentasikan elemen dalam recipe), Links (Hubungan antara node yang menunjukkan alur kombinasi).

#### 4. Recipe Node dan Recipe Link

```

type RecipeNode struct {
    ID string `json:"id"`
    Label string `json:"label"`
    Level int `json:"level"`
}

type RecipeLink struct {
    Source string `json:"source"`
    Target string `json:"target"`
}

```

RecipeNode: Merepresentasikan elemen dengan ID, nama elemen, dan level dalam tree.

RecipeLink: Merepresentasikan hubungan antara dua node (source->target)

#### 5. Visualization Tree (modelTreeNode)

```

type TreeNode struct {
    ID string `json:"id"`
    Name string `json:"name"`
    Combine []TreeNode `json:"combine,omitempty"`
}

```

Struktur khusus untuk visualisasi tree di frontend. Implementasinya adalah node rekursif dengan children yang disimpan dalam array Combine. Kegunaannya adalah digunakan oleh komponen RecipeTree.tsx untuk visualisasi hasil pencarian.

#### 6. Search Parameters (model.SearchRequest dan model.SearchResponse)

```

type SearchRequest struct {
    TargetElement string `json:"targetElement"`
    Algorithm string `json:"algorithm"`
    MultipleRecipe bool `json:"multipleRecipe"`
    MaxRecipes int `json:"maxRecipes"`
}

type SearchResponse struct {
    Target string `json:"target"`
    Algorithm string `json:"algorithm"`
    Time float64 `json:"time"`
    VisitedNodes int `json:"visitedNodes"`
}

```

```

    Recipes []Recipe `json:"recipes"`
    TreeData TreeNode `json:"treeData,omitempty"`
}

```

SearchRequest: Parameter input dari pengguna untuk pencarian

SearchResponse: Hasil lengkap pencarian termasuk data performa dan visualisasi

## 7. Struktur Pencarian (BFS, DFS, Bidirectional)

Dalam algoritma pencarian, digunakan beberapa struktur bantu yaitu:

- Queue/Stack: Implementasi dengan slice di Go untuk BFS(queue) dan DFS(stack)
- Visited Map: Map untuk melacak node yang sudah dikunjungi
- Path Tracking: Mekanisme untuk melacak jalur dari elemen target ke elemen dasar

## 8. Multithreading untuk Multiple Recipe

```

func (s *RecipeService) FindMultipleRecipes(params
model.SearchRequest) model.SearchResponse {
    var wg sync.WaitGroup
    var mutex sync.Mutex
    var totalVisitedNodes int
    var combinedRecipes []model.Recipe
    // ...
    for i := 0; i < numGoroutines; i++ {
        wg.Add(1)
        go func(idx int) {
            defer wg.Done()
            // ... pencarian recipe per goroutine
            mutex.Lock()
            totalVisitedNodes += visitedNodes
            combinedRecipes = append(combinedRecipes,
recipes...)
            mutex.Unlock()
        }(i)
    }
    wg.Wait()
    // ...
}

```

Implementasi multithreading untuk optimasi pencarian multiple recipe

Komponennya adalah:

- sync.WaitGroup: untuk menunggu semua goroutine selesai
- sync.Mutex: untuk sinkronisasi akses ke data bersama

## Komponen Frontend

Frontend dibuat menggunakan [Next.js](#) dengan TypeScript dan styling dari Tailwind CSS. Komponen-komponen dari frontend adalah sebagai berikut:

### 1. SearchForm.tsx

SearchForm.tsx adalah komponen utama di mana pengguna berinteraksi dengan sistem. Komponen ini memiliki berbagai elemen form seperti dropdown pemilihan target elemen, selector algoritma, toggle untuk memilih pencarian resep terpendek atau pencarian banyak resep, serta input jumlah maksimal resep yang ingin ditemukan. Komponen ini bertindak sebagai pusat koordinasi dari seluruh alur pencarian.

## 2. ElementSelector.tsx

ElementSelector.tsx adalah komponen yang menyediakan tampilan dropdown sehingga pengguna dapat memilih target elemen yang ingin dicari resepnya.

## 3. AlgorithmSelector.tsx

AlgorithmSelector.tsx adalah komponen yang menampilkan pilihan algoritma yang tersedia, yaitu BFS, DFS, dan Bidirectional Search. Pengguna nantinya akan memilih salah satu dari tiga algoritma tersebut.

## 4. RecipeTree.tsx

RecipeTree.tsx adalah komponen yang menampilkan hasil pencarian resep dengan visualisasi *tree*.

## 5. ResultInfo.tsx

ResultInfo.tsx adalah komponen yang menampilkan informasi terkait ringkasan hasil pencarian, termasuk jumlah resep, *nodes* yang dikunjungi, dan waktu pencarian.

## 6. Header.tsx dan NavBar.tsx

Komponen-komponen yang menjadi layout tambahan agar memperindah dan menyusun tampilan halaman web.

## Komponen Backend

Backend dibuat menggunakan bahasa pemrograman Go dengan framework gin, serta dijalankan menggunakan Dockerfile. Komponen-komponen dari backend adalah sebagai berikut:

### 1. main.go

Merupakan entry point dari website. File ini menginisialisasi router Gin, mengatur middleware, dan mendefinisikan endpoint-endpoint HTTP yang akan menangani request dari frontend. File ini juga menghubungkan seluruh komponen yang diperlukan seperti handler dan service.

### 2. bfs.go

Berisi implementasi algoritma Breadth-First Search (BFS) yang digunakan untuk pencarian jalur dari satu node ke node lainnya pada graf. Algoritma ini digunakan untuk menjelajahi simpul secara menyeluruh berdasarkan level.

### 3. dfs.go

Berisi implementasi algoritma Depth-First Search (DFS), sebuah algoritma penelusuran graf yang menjelajahi jalur sedalam mungkin sebelum mundur kembali (backtracking). Cocok untuk eksplorasi graf yang memerlukan pencarian jalur lebih panjang terlebih dahulu.

### 4. bidirectional.go

Mengimplementasikan algoritma pencarian Bidirectional Search. Algoritma ini melakukan pencarian dari dua arah secara simultan (dari node awal dan akhir) dan mempertemukan kedua pencarian tersebut di tengah untuk efisiensi waktu.

### 5. handler.go

Berisi fungsi-fungsi yang menangani request dari client (seperti frontend). Handler menerima input dari HTTP request, memanggil service yang sesuai, dan mengembalikan hasilnya dalam bentuk response JSON. Handler ini menghubungkan lapisan routing dengan service logic.

### 6. middleware.go

Berisi middleware untuk mengatur logika tambahan seperti logging request dan pengaturan CORS. Middleware ini dijalankan sebelum request masuk ke handler utama, untuk memastikan keamanan dan manajemen request yang baik.

### 7. recipe.go

Mendefinisikan struktur data (struct) yang merepresentasikan informasi terkait resep makanan, termasuk bahan, langkah-langkah, dan metadata lain. Struct ini juga digunakan untuk serialisasi/deserialisasi data JSON.

### 8. recipe\_service.go

Berfungsi sebagai lapisan logika bisnis (business logic layer) untuk pengolahan data resep dan pemanggilan algoritma pencarian rute. File ini mengatur alur data antara handler dan algoritma pencarian seperti BFS, DFS, atau Bidirectional.

## Tata Cara Penggunaan Program

Pertama kita mengakses website Jepangor melalui link [berikut](#). Hal pertama yang akan ditampilkan di website adalah halaman utama yang mana pengguna dapat memilih elemen dan mencari recipe. Untuk lebih detailnya, cara untuk mencari recipe adalah sebagai berikut:

1. Lihat ke sisi sebelah kiri bawah.

**Cari Recipe**

Elemen Target

Algoritma Pencarian

  
**BFS**  
 Breadth-First Search

  
**DFS**  
 Depth-First Search

  
**Bidirectional**  
 Pencarian Dua Arah

Cari recipe terpendek  
Hanya akan menampilkan 1 recipe dengan jalur terpendek

**Cari Recipe**

- Klik pada bagian cari elemen dan pilih elemen yang anda mau.

Elemen Target

- Selanjutnya pilih algoritma pencarian yang diinginkan untuk mencari recipe.

Algoritma Pencarian

  
**BFS**  
 Breadth-First Search

  
**DFS**  
 Depth-First Search

  
**Bidirectional**  
 Pencarian Dua Arah

- Setelah memilih elemen dan memilih algoritma pencarian, klik bagian Cari recipe (warna abu-abu) yang berada di bawah.

**Cari Recipe**

- Hasil dari pencarian Recipe akan ditampilkan di sisi kanan website Jepangor.

> Contoh pencarian Recipe elemen airplane menggunakan BFS.

Selanjutnya, pada halaman Tentang atau *About* pengguna dapat melihat penjelasan tentang algoritma BFS dan DFS dan juga Bidirectional Search.

Algoritma	Karakteristik
BFS	<ul style="list-style-type: none"> <li>Menjamin jalur terpendek</li> <li>Membutuhkan lebih banyak memori</li> <li>Eksplorasi level demi level</li> <li>Implementasi dengan antrian (FIFO)</li> </ul>
DFS	<ul style="list-style-type: none"> <li>Eksplorasi mendalam terlebih dahulu</li> <li>Effisien dalam penggunaan memori</li> <li>Backtracking untuk alternatif jalur</li> <li>Implementasi dengan stack (LIFO)</li> </ul>
Bidirectional Search	<ul style="list-style-type: none"> <li>Pencarian dari dua arah sekaligus</li> <li>Mempertemukan dua frontier search</li> <li>Potensial mengurangi jumlah node yang dieksplorasi</li> <li>Kompleksitas ruang yang lebih besar</li> </ul>

Selain itu, pengguna juga bisa memilih untuk melihat penjelasan tentang aplikasi yang dibuat oleh Jepangor dengan mengklik bagian yang bertulisan “Aplikasi”.

Dan terakhir, pengguna dapat melihat tim pengembang yaitu anggota dari kelompok Jepangor dengan mengklik bagian yang bertulisan “Tim”.

The screenshot shows the 'Tim Pengembang' (Development Team) section of the Jepangor application. It displays three team members in cards:

- Muhammad Raihaan Perdana** (13523124), FullStack & Algorithm Developer.
- Ardell Aghna Mahendra** (13523151), UI/UX & Frontend Developer.
- Farrel Jabaar Altafataza** (10122057), Scraper & Algorithm Developer.

Below the team cards, there is a section for the 'Jepangor Team' which includes the name, degree, and year of graduation.

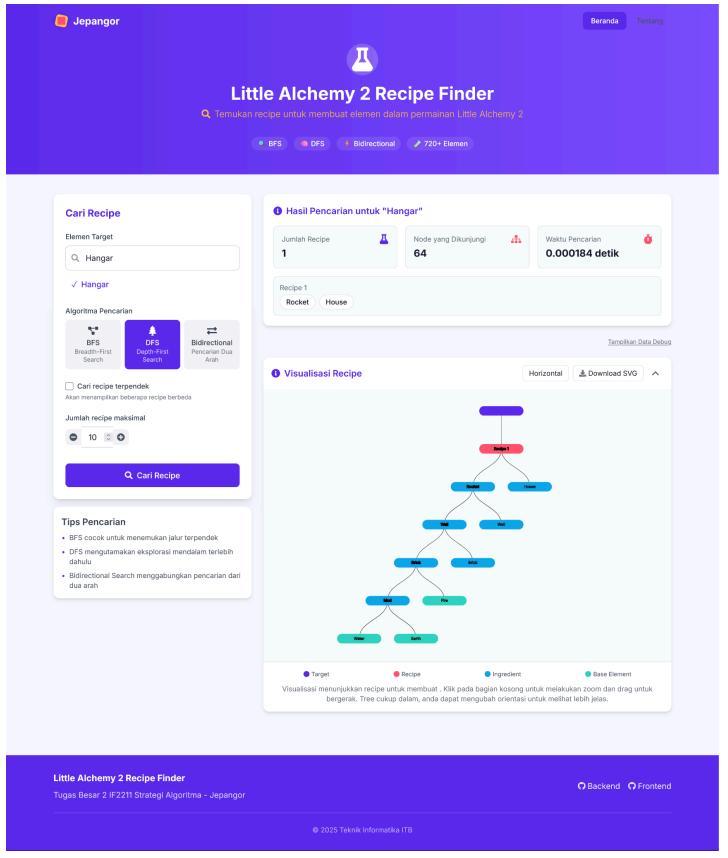
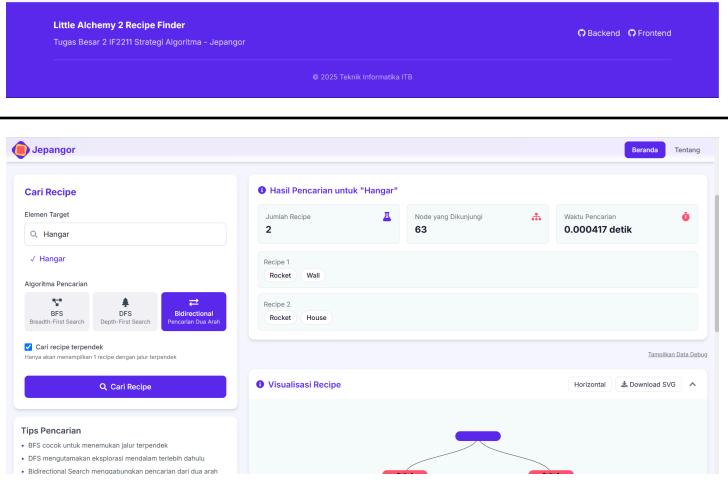
## Hasil Pengujian

### 1. Pengujian Elemen “Hangar”

Algoritma	Banyak Resep	Hasil
BFS	Terpendek	<p>The screenshot shows the search results for the target element "Hangar" using the BFS algorithm. The results are as follows:</p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 1</li> <li>Node yang Dikunjungi: 33</li> <li>Waktu Pencarian: 0.000161 detik</li> <li>Recipe 1: Rocket -&gt; Wall</li> </ul> <p>Below the results, there is a visualization of the search path as a tree diagram.</p>

BFS	5	<p>The screenshot shows the Jepangor search interface. In the search bar, 'Hangar' is entered. Under 'Algoritma Pencarian', 'BFS Breadth-First Search' is selected. The search results show two recipes: 'Rocket' and 'Wall'. A visual tree diagram illustrates the search space, with nodes colored by type: purple for target ('Hangar'), red for recipe, blue for ingredient, and green for base element.</p>
BFS	10	<p>The screenshot shows the Jepangor search interface. In the search bar, 'Hangar' is entered. Under 'Algoritma Pencarian', 'BFS Breadth-First Search' is selected. The search results show two recipes: 'Rocket' and 'Wall'. A visual tree diagram illustrates the search space, with nodes colored by type: purple for target ('Hangar'), red for recipe, blue for ingredient, and green for base element.</p>

DFS	Terpendek	<p>The screenshot shows the Jepangor search interface. In the search bar, 'Hangar' is entered. Under 'Algoritma Pencarian', 'DFS Depth-First Search' is selected. The search results show a single recipe found in 15 nodes, with a search time of 0.000090 detik. The visualized tree shows a depth-first search path.</p>
DFS	5	<p>The screenshot shows the Jepangor search interface. In the search bar, 'Hangar' is entered. Under 'Algoritma Pencarian', 'DFS Depth-First Search' is selected. The search results show a single recipe found in 31 nodes, with a search time of 0.000154 detik. The visualized tree shows a depth-first search path. A note indicates that selecting 'Cari recipe terpendek' will result in multiple recipes being found.</p>

DFS	10	 <p><b>Hasil Pencarian untuk "Hangar"</b></p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 1</li> <li>Node yang Dikunjungi: 64</li> <li>Waktu Pencarian: 0.000184 detik</li> </ul> <p><b>Visualisasi Recipe</b></p> <p>Visualisasi menunjukkan resep untuk membuat. Klik pada bagian kosong untuk melakukan zoom dan drag untuk bergerak. Tree cukup dalam, anda dapat mengubah orientasi untuk melihat lebih jelas.</p>
Bidirectional	Terpendek	 <p><b>Hasil Pencarian untuk "Hangar"</b></p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 2</li> <li>Node yang Dikunjungi: 63</li> <li>Waktu Pencarian: 0.000417 detik</li> </ul> <p><b>Visualisasi Recipe</b></p> <p>Visualisasi menunjukkan resep untuk membuat. Klik pada bagian kosong untuk melakukan zoom dan drag untuk bergerak. Tree cukup dalam, anda dapat mengubah orientasi untuk melihat lebih jelas.</p>

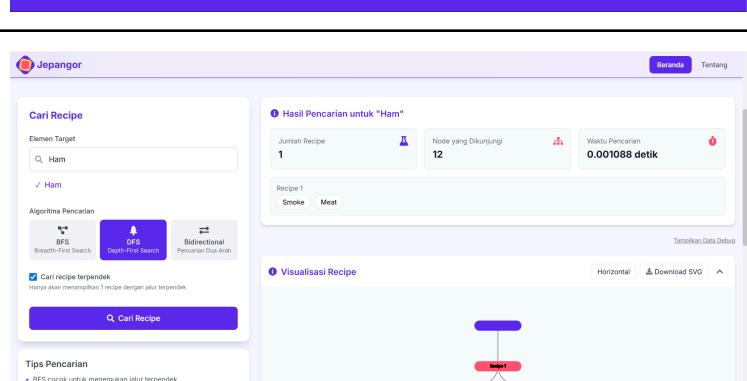
Bidirectional	5	
---------------	---	--

Bidirectional	10	<p>The screenshot shows the search interface for 'Hangar'. The search results table indicates 2 recipes found, with 1,896 nodes explored and a search time of 0.082215 seconds. The visualizations section displays a search tree for 'Ham'.</p>
---------------	----	--

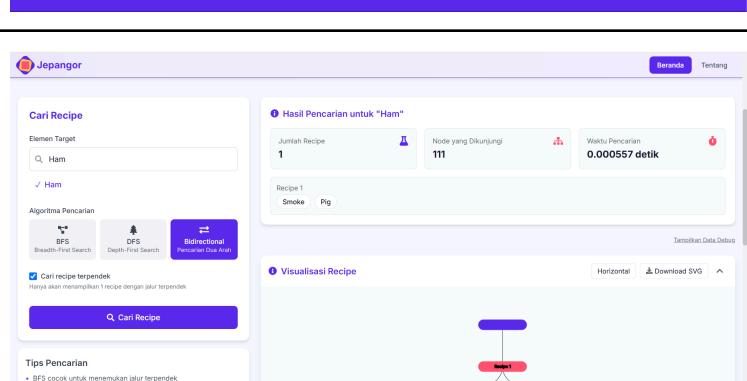
## 2. Pengujian Elemen “Ham”

Algoritma	Banyak Resep	Hasil
BFS	Terpendek	<p>The screenshot shows the search interface for 'Ham'. The search results table indicates 1 recipe found, with 6 nodes explored and a search time of 0.000055 seconds. The visualizations section displays a search tree for 'Ham'.</p>

BFS	5	<p>The screenshot shows the Little Alchemy 2 Recipe Finder interface. The search bar at the top contains 'Ham'. Below it, there are three search algorithm buttons: BFS (Breath-First Search), DFS (Depth-First Search), and Bidirectional Search. The 'BFS' button is highlighted. The search results are displayed in a card titled 'Hasil Pencarian untuk "Ham"'. It shows two recipes found: 'Smoke + Meat' and 'Smoke + Pig'. The search took 0.000112 seconds and explored 13 nodes. A visual tree diagram labeled 'Visualisasi Recipe' illustrates the search path from base elements to the target 'Ham'. The legend indicates: Target (purple dot), Recipe (red dot), Ingredient (blue dot), and Base Element (green dot). The bottom of the page includes navigation links like 'Home', 'About', 'Contact', and copyright information.</p>
-----	---	--

BFS	10	 <p><b>Hasil Pencarian untuk "Ham"</b></p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 1</li> <li>Node yang Dikunjungi: 12</li> <li>Waktu Pencarian: 0.001088 detik</li> </ul> <p><b>Visualisasi Recipe</b></p> <pre> graph TD     Target((Ham)) --- Recipe1[Smoke + Meat]   </pre> <p>Visualisasi menunjukkan recipe untuk membuat. Klik pada bagian kosong untuk melakukan zoom dan drag untuk bergerak. Tree cukup dalam, anda dapat mengubah orientasi untuk melihat lebih jelas.</p>
DFS	Terpendek	

DFS	5	
-----	---	--

DFS	10	 <p>The screenshot shows the search results for "Ham" using the Bidirectional Search algorithm. It displays one recipe found:</p> <ul style="list-style-type: none"> <li><b>Recipe 1:</b> Smoke + Pig</li> </ul> <p>The search statistics are as follows:</p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 1</li> <li>Node yang Dikunjungi: 111</li> <li>Waktu Pencarian: 0.000557 detik</li> </ul> <p>A visual tree diagram is shown below the search results, illustrating the search space and the path taken by the Bidirectional Search algorithm.</p>
Bidirectional	Terpendek	

Bidirectional	5	
---------------	---	--

Bidirectional	10	<p>The screenshot shows the Little Alchemy 2 Recipe Finder application. At the top, there's a search bar with the query "Ham". Below it, a section titled "Hasil Pencarian untuk 'Ham'" displays five recipes: "Smoke" + "Meat" = "Ham", "Fire" + "Air" = "Ham", "Campfire" + "Time" = "Ham", "Smoke" + "Pig" = "Ham", and "Campfire" + "Water" = "Ham". A "Cari Recipe" button is present. To the right, a "Visualisasi Recipe" section shows a search tree starting from "Ham" at the root, branching down through various elements like "House", "Lake", "Food", "Meat", "Pork", etc. A legend identifies the colors: purple for Target, red for Recipe, blue for Ingredient, and teal for Base Element. The bottom of the screen shows the application's footer with links for Backend and Frontend.</p>
---------------	----	--

### 3. Pengujian Elemen “Swimming Pool”

Algoritma	Banyak Resep	Hasil
BFS	Terpendek	<p>The screenshot shows the Little Alchemy 2 Recipe Finder application. At the top, there's a search bar with the query "Swimming pool". Below it, a section titled "Hasil Pencarian untuk 'Swimming pool'" displays one recipe: "House" + "Lake" = "Swimming pool". A "Cari Recipe" button is present. To the right, a "Visualisasi Recipe" section shows a search tree starting from "Swimming pool" at the root, which has only one node labeled "House" under "Lake". A legend identifies the colors: purple for Target, red for Recipe, blue for Ingredient, and teal for Base Element. The bottom of the screen shows the application's footer with links for Backend and Frontend.</p>

BFS	5	<p><b>Hasil Pencarian untuk "Swimming pool"</b></p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 1</li> <li>Node yang Dikunjungi: 35</li> <li>Waktu Pencarian: 0.000182 detik</li> </ul> <p><b>Visualisasi Recipe</b></p> <pre> graph TD     Root[House] --&gt; Lake[Lake]     Root --&gt; Sand[Sand]     Lake --&gt; Water[Water]     Lake --&gt; Salt[Salt]     Water --&gt; SwimmingPool[Swimming pool]     Salt --&gt; SwimmingPool     Sand --&gt; SwimmingPool   </pre>
BFS	10	<p><b>Hasil Pencarian untuk "Swimming pool"</b></p> <ul style="list-style-type: none"> <li>Jumlah Recipe: 1</li> <li>Node yang Dikunjungi: 44</li> <li>Waktu Pencarian: 0.000148 detik</li> </ul> <p><b>Visualisasi Recipe</b></p> <pre> graph TD     Root[House] --&gt; Lake[Lake]     Root --&gt; Sand[Sand]     Lake --&gt; Water[Water]     Lake --&gt; Salt[Salt]     Water --&gt; SwimmingPool[Swimming pool]     Salt --&gt; SwimmingPool     Sand --&gt; SwimmingPool   </pre>

DFS	Terpendek	
DFS	5	

DFS	10	
Bidirectional	Terpendek	

Bidirectional	5	
---------------	---	--

Bidirectional	10	
---------------	----	--

## Analisis Hasil Pengujian

Pengujian dilakukan dengan menggunakan tiga algoritma pencarian yaitu Breadth First Search (BFS), Depth First Search (DFS), dan Bidirectional Search. Setiap algoritma diuji dengan variasi mode jumlah resep maksimal yaitu terpendek, 5, dan 10. Tiga elemen digunakan sebagai sampel pengujian, yakni *Hangar*, *Ham*, dan *Swimming Pool*. Secara umum, algoritma BFS dan DFS menunjukkan performa yang efisien dari segi waktu eksekusi dan jumlah *nodes* yang dikunjungi. Waktu eksekusi yang ditempuh berkisar di sekitar satuan mikrodetik dan hanya mengunjungi puluhan *nodes*. Akan tetapi, performa kedua algoritma tersebut sangat bergantung kepada kedalaman solusi. BFS cenderung menyebar secara luas dari elemen dasar dan DFS menjelajahi jalur yang lebih dalam terlebih dahulu. Hal ini membuat DFS pada konteks tertentu hanya bisa menemukan sebagian resep dari total kemungkinan resep yang ada. Di sisi lain, BFS cenderung lebih stabil tetapi tetap terbatas pada jalur terdekat. Algoritma Bidirectional Search menunjukkan performa paling optimal untuk mencari jumlah resep terbanyak. Hal ini bisa terjadi karena terdapat beberapa elemen yang resepnya membutuhkan elemen dengan *tier* lebih tinggi dari *tier* target elemennya. Sifat Bidirectional Search yang mencari dari dua arah berhasil menghilangkan

hambatan tersebut. Akan tetapi, waktu eksekusi dan jumlah *nodes* yang dikunjungi cenderung lebih besar jika dibandingkan dengan BFS dan DFS.

## **BAB V: PENUTUP**

### **Kesimpulan**

Berdasarkan analisis hasil pengujian, dapat disimpulkan bahwa setiap algoritma memiliki kelebihannya masing-masing. BFS dan DFS terbukti efisien dari segi waktu dan jumlah *nodes* yang dikunjungi sehingga algoritma tersebut ideal untuk pencarian resep dengan cepat dan jumlah resep yang cenderung lebih sedikit. Di sisi lain, Bidirectional Search unggul dalam mengeksplorasi kombinasi resep secara menyeluruh tetapi tidak efisien dari segi waktu dan jumlah *nodes* yang dikunjungi. Dapat disimpulkan bahwa pemilihan algoritma harus disesuaikan dengan strategi pencarian dan kebutuhan dari pengguna.

### **Saran**

Untuk pengembangan selanjutnya, kami harap di web ditambahkan fitur-fitur lain seperti fitur di mana pengguna dapat mendrag elemen dan menyatukan dengan elemen lain seperti pada web asli little alchemy 2.

### **Refleksi**

Dengan mengerjakan tugas besar 2 mata kuliah Strategi Algoritma ini, kami merasa pemahaman terkait algoritma BFS dan DFS dan juga pengalaman terkait dengan pembuatan web bertambah. Pada tugas besar 2 ini juga, kami melatih komunikasi antar anggota tim di mana setiap anggota tim di tugas besar 2 ini selalu aktif dari awal hingga akhir pengerjaan.

## LAMPIRAN

No	Poin	Ya	Tidak
1	Aplikasi dapat dijalankan.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	Aplikasi dapat memperoleh data recipe melalui scraping.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	Algoritma Depth First Search dan Breadth First Search dapat menemukan recipe elemen dengan benar.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	Aplikasi dapat menampilkan visualisasi recipe elemen yang dicari sesuai dengan spesifikasi.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	Aplikasi mengimplementasikan multithreading.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	Membuat laporan sesuai dengan spesifikasi.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
7	Membuat bonus video dan diunggah pada Youtube.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
8	Membuat bonus algoritma pencarian Bidirectional	<input checked="" type="checkbox"/>	<input type="checkbox"/>
9	Membuat bonus Live Update	<input type="checkbox"/>	<input checked="" type="checkbox"/>
10	Aplikasi di-containerize dengan Docker.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
11	Aplikasi di-deploy dan dapat diakses melalui internet.	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Tautan Repository Github:

- Frontend: [https://github.com/fliegenhaan/Tubes2\\_FE\\_Jepangor.git](https://github.com/fliegenhaan/Tubes2_FE_Jepangor.git)
- Backend: [https://github.com/fliegenhaan/Tubes2\\_BE\\_Jepangor.git](https://github.com/fliegenhaan/Tubes2_BE_Jepangor.git)

Tautan Video (Bonus): <https://youtu.be/pdd0t1Yrkwo>

Tautan Website: <https://jepangor.vercel.app/>

## **DAFTAR PUSTAKA**

WsCubeTech. *DFS vs BFS Algorithm (All Differences With Example)*. Diakses pada 8 Mei 2025 dari <https://www.wscubetech.com/resources/dsa/dfs-vs-bfs>

VisuAlgo. (n.d.). *Penjelajahan Graf (Depth/Breadth First Search)*. Diakses pada 10 Mei 2025 dari <https://visualgo.net/id/dfs bfs>