

IF2211 - Strategi Algoritma

Laporan Tugas Besar 2

*Pemanfaatan Algoritma BFS dan DFS dalam Pencarian Recipe pada
Permainan Little Alchemy 2*



Disusun Oleh:

Muhammad Alfansya	13523005
M. Hazim R. Prajoda	13523009
Adinda Putri	13523071

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1.....	4
DESKRIPSI TUGAS.....	4
BAB 2.....	6
LANDASAN TEORI.....	6
2.1 Pencarian dalam Graf.....	6
2.2. Algoritma Pencarian.....	6
2.2.1. Breadth First Search.....	6
2.2.2. Depth First Search.....	7
2.2.3. Bidirectional.....	8
2.3. Alchendol: Website Pencarian Elemen pada Game Little Alchemy 2.....	9
2.3.1. Little Alchemy 2.....	9
2.3.2. Tujuan Website.....	10
BAB 3.....	11
ANALISIS PEMECAHAN MASALAH.....	11
3.1. Langkah-Langkah Pemecahan Masalah.....	11
3.1.1. Memahami Struktur Permainan.....	11
3.1.2. Memahami Struktur Permainan.....	11
3.1.3. Merancang Representasi Graf.....	11
3.1.4. Mengimplementasikan Algoritma Pencarian.....	12
3.1.5. Membangun Backend API.....	12
3.1.6. Merancang Front End dan Integrasi.....	12
3.1.7. Melakukan Pengujian dan Validasi.....	13
3.2. Pemetaan Masalah ke Algoritma DFS dan BFS.....	13
3.2.1. Breadth-First Search (BFS).....	13
3.2.2. Depth-First Search (DFS).....	14
3.3. Fitur Fungsional dan Arsitektur Aplikasi Web.....	14
3.3.1. Fitur Fungsional.....	14
3.3.1.1. Opsi Pemilihan Algoritma Pencarian (BFS, DFS, Bidirectional).....	14
3.3.1.2. Toggle Mode Pencarian: Shortest Recipe dan Multiple Recipes.....	15
3.3.1.3. Parameter Jumlah Recipe (Multiple Recipe).....	15
3.3.1.4. Visualisasi Recipe dalam Bentuk Tree.....	15
3.3.1.5. Statistik Pencarian (Waktu dan Node yang Dikunjungi).....	15
3.3.2. Arsitektur Web.....	15
3.3.2.1. Frontend (Next.js).....	16
3.3.2.2. Backend (Golang dan Gin).....	16
3.3.2.3. Komunikasi Client-Server.....	17

3.3.2.4. Data - JSON.....	17
3.4. Contoh Ilustrasi Kasus.....	17
3.4.1. Deskripsi Kasus.....	17
3.4.2. Algoritma Pencarian.....	17
3.4.3. Hasil yang Diperoleh.....	18
BAB 4.....	20
IMPLEMENTASI DAN PENGUJIAN.....	20
4.1. Spesifikasi Teknis Program.....	20
4.1.1. Struktur Data.....	20
4.1.3.1. Struktur Data Repository.....	20
4.1.3.2. Struktur Data Program.....	21
4.1.2. Breadth First Search (BFS).....	23
4.1.2.1 MultipleBFS.....	23
4.1.3. Depth First Search (DFS).....	24
4.1.4. Bidirectional.....	26
4.2. Tata Cara Penggunaan Program.....	27
4.2.1 Interface Program.....	27
4.2.1.1 Path Seeker.....	27
4.2.1.2. The Lab Table.....	28
4.2.1.3. The Alchemists.....	29
4.2.1.4. Greets.....	29
4.2.1.5. How to Play.....	30
4.2.2 Fitur.....	31
4.2.2.1 Magic Path.....	31
4.2.2.2 Element Search.....	32
4.2.2.3 Element Preview.....	33
4.3. Pengujian.....	35
4.3.1. Pengujian 1 : Pencarian Resep dengan Shortest Recipe BFS.....	35
4.3.2. Pengujian 2 : Pencarian Resep dengan Multiple Recipes BFS.....	36
4.3.3. Pengujian 3 : Pencarian Recipe dengan Shortest Recipe DFS.....	37
4.3.4. Pengujian 4 : Pencarian Recipe dengan Multiple Recipes DFS.....	38
4.3.5. Pengujian 5 : Pencarian Recipe dengan Shortest Recipe Bidirectional.....	39
4.3.6. Pengujian 6 : Pencarian Recipe dengan Multiple Recipes Bidirectional.....	40
4.3.7. Pengujian 7 : Resep Maksimal yang Tersedia Kurang dari Inputan Max Recipe..	41
4.3.9. Pengujian 8 : Inputan Max Recipe sangat banyak.....	42
4.4. Analisis Hasil Pengujian.....	42
BAB 5.....	44
KESIMPULAN DAN SARAN.....	44
5.1. Kesimpulan.....	44

5.2. Saran.....	44
5.3. Refleksi.....	45
LAMPIRAN.....	46
Tautan Repository GitHub.....	46
Tautan Video.....	46
DAFTAR PUSTAKA.....	47

BAB 1

DESKRIPSI TUGAS



Gambar 1. Little Alchemy 2
(sumber: <https://www.thegamer.com>)

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.



Gambar 2. Elemen dasar pada Little Alchemy 2

(sumber: <https://littlealchemy2.com/>)

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 2

LANDASAN TEORI

2.1 Pencarian dalam Graf

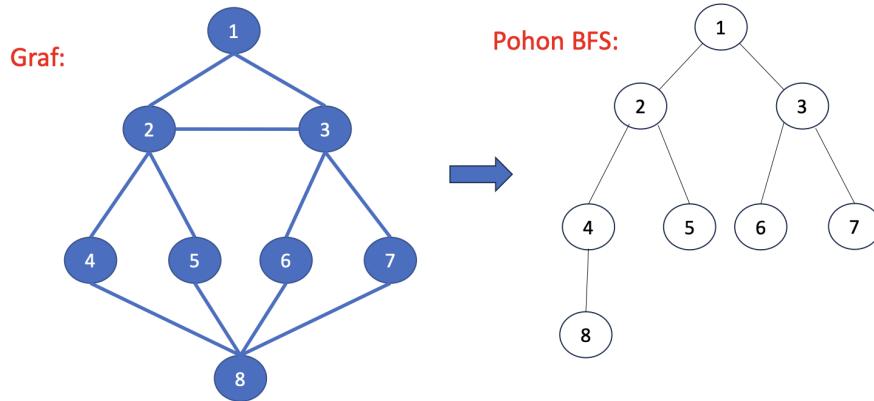
Pencarian dalam graf merupakan teknik penting dalam ilmu komputer yang digunakan untuk menemukan solusi dari suatu permasalahan yang direpresentasikan sebagai simpul (node) dan hubungan antar simpul (edge). Dalam konteks pengembangan aplikasi seperti Little Alchemy 2 Recipe Finder, setiap elemen dapat dianggap sebagai simpul, dan kombinasi dua elemen (resep) menjadi sisi (edge) yang menghubungkan mereka. Proses pencarian ini dikenal sebagai traversal graf, yaitu suatu cara sistematik untuk menelusuri semua simpul guna menemukan jalur atau solusi yang sesuai.

Terdapat dua pendekatan utama dalam pencarian graf: uninformed (blind search) dan informed (heuristic search). Pendekatan uninformed, seperti Breadth-First Search (BFS) dan Depth-First Search (DFS), tidak menggunakan informasi tambahan tentang posisi tujuan, sehingga melakukan pencarian secara menyeluruh. Di sisi lain, pencarian informed seperti A* memanfaatkan heuristik untuk memperkirakan jarak ke tujuan, sehingga lebih efisien dalam beberapa kasus. Dalam implementasi aplikasi berbasis web ini, representasi graf dapat bersifat statis (semua kombinasi sudah tersedia) atau dinamis (dibentuk saat pengguna melakukan pencarian), tergantung bagaimana data resep diakses dan diolah selama runtime.

2.2. Algoritma Pencarian

2.2.1. Breadth First Search

Breadth-First Search (BFS) adalah algoritma pencarian atau traversing pada graf yang menjelajahi simpul-simpul secara melebar, dimulai dari satu simpul awal (biasanya disebut simpul v). BFS bekerja dengan mengunjungi simpul awal terlebih dahulu, kemudian menjelajahi semua simpul yang bertetangga langsung dengannya. Setelah itu, algoritma berlanjut dengan mengunjungi simpul-simpul yang bertetangga dengan simpul-simpul yang baru saja dikunjungi, dan proses ini diulang hingga semua simpul yang dapat dicapai telah dikunjungi. BFS biasanya menggunakan struktur data antrian (queue) untuk mengelola urutan kunjungan simpul.



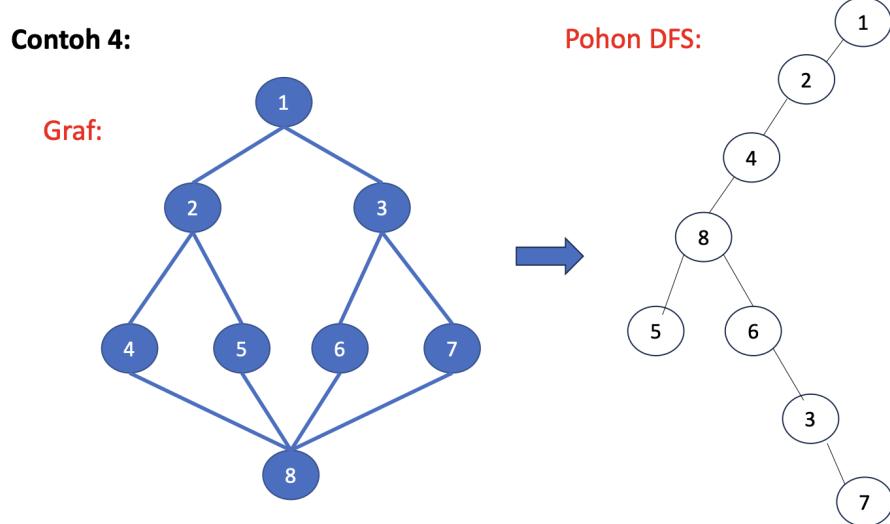
Gambar 3. Contoh Pohon BFS

(sumber:

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-\(2025\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf)

2.2.2. Depth First Search

Depth-First Search (DFS) adalah algoritma pencarian atau traversing pada graf yang menjelajahi simpul-simpul secara mendalam, yaitu dengan menyusuri satu jalur sejauh mungkin sebelum berbalik (backtrack) untuk menjelajahi jalur lainnya. Proses dimulai dari simpul awal v , kemudian algoritma akan mengunjungi satu simpul tetangga w dan langsung melakukan DFS dari w tersebut. Jika mencapai simpul u yang semua tetangganya sudah dikunjungi, maka algoritma akan kembali (backtrack) ke simpul sebelumnya yang masih memiliki tetangga yang belum dikunjungi. Proses ini terus berlanjut hingga semua simpul yang dapat dicapai telah dikunjungi. DFS biasanya diimplementasikan dengan bantuan stack atau rekursi.



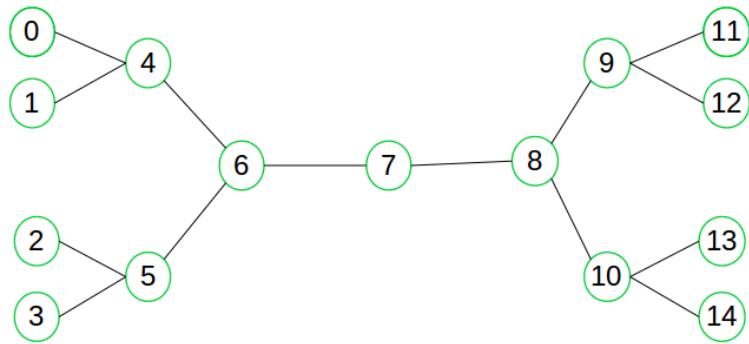
Gambar 4. Contoh Pohon DFS

(sumber:

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-\(2025\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf)

2.2.3. Bidirectional

Pencarian dua arah (*bidirectional search*) adalah algoritma pencarian pada graf yang bertujuan menemukan jalur terpendek dari simpul awal ke simpul tujuan dengan cara melakukan dua pencarian sekaligus—satu pencarian maju dari simpul awal ke tujuan, dan satu pencarian mundur dari simpul tujuan ke awal. Algoritma ini menghentikan pencarian ketika kedua arah pencarian bertemu di satu simpul yang sama, sehingga jalur telah ditemukan. Pendekatan ini jauh lebih efisien dibandingkan pencarian satu arah seperti BFS atau DFS, karena dapat mengurangi ruang pencarian secara drastis dari kompleksitas $O(b^d)$ menjadi $O(b^{d/2}) + O(b^{d/2})$, di mana b adalah faktor percabangan dan d adalah kedalaman solusi. Bidirectional search sangat cocok digunakan saat simpul awal dan tujuan sudah pasti dan unik, serta jika graf memiliki percabangan yang seragam di kedua arah.



Gambar 5. Contoh Pohon Bidirectional

(sumber: <https://www.geeksforgeeks.org/bidirectional-search/>)

2.3. Alchendol: Website Pencarian Elemen pada Game Little Alchemy 2

2.3.1. Little Alchemy 2

Little Alchemy 2 adalah sebuah permainan berbasis kombinasi elemen yang dikembangkan oleh Recloak dan dirilis pada tahun 2017 sebagai sekuel dari *Little Alchemy* versi pertama. Permainan ini mengajak pemain untuk memulai dengan empat elemen dasar, yaitu air, tanah, api, dan udara yang kemudian dapat dikombinasikan dua per dua untuk membentuk elemen baru. Setiap hasil kombinasi yang valid akan menambah koleksi elemen pemain, dan elemen-elemen baru tersebut dapat terus digunakan dalam kombinasi selanjutnya untuk membentuk total hingga 720 elemen yang tersedia dalam permainan. Mekanisme permainan ini bersifat eksploratif dan eksponensial, karena satu elemen bisa berasal dari berbagai kombinasi berbeda. Tantangan utamanya terletak pada menemukan jalur kombinasi (recipe) yang tepat, sehingga permainan ini sangat relevan sebagai studi kasus untuk algoritma pencarian dalam graf.



Gambar 6. Little Alchemy 2

(sumber: <https://littlealchemy2.com/>)

2.3.2. Tujuan Website

Website Alchendol dibangun sebagai media interaktif untuk menerapkan dan menguji algoritma pencarian graf, khususnya Breadth-First Search (BFS), Depth-First Search (DFS), dan Bidirectional, dalam konteks permainan *Little Alchemy 2*. Permainan ini menantang pemain untuk menemukan kombinasi dua elemen untuk menghasilkan elemen baru, hingga total 720 elemen. Alchendol bertujuan untuk membantu pengguna menemukan *jalur kombinasi elemen* (recipe tree) dari elemen dasar menuju elemen target yang diinginkan.

Melalui Alchendol, pengguna dapat mengeksplorasi bagaimana algoritma pencarian bekerja dalam membentuk pohon solusi dari berbagai kombinasi yang mungkin. Selain itu, aplikasi ini menyediakan pilihan untuk mencari satu solusi terpendek atau beberapa solusi berbeda menuju elemen target, yang divisualisasikan dalam bentuk pohon resep yang jelas dan informatif. Dengan demikian, aplikasi ini tidak hanya menyelesaikan permainan secara sistematis, tetapi juga menjadi sarana edukatif untuk memahami struktur pencarian graf dan kompleksitasnya.

BAB 3

ANALISIS PEMECAHAN MASALAH

3.1. Langkah-Langkah Pemecahan Masalah

Dalam membangun website Alchendol sebagai solusi pencarian kombinasi elemen pada permainan *Little Alchemy 2*, diperlukan serangkaian langkah sistematis untuk memecah kompleksitas permasalahan menjadi komponen-komponen yang dapat dianalisis dan diimplementasikan secara terstruktur. Langkah-langkah pemecahan masalah ini disusun sebagai berikut:

3.1.1. Memahami Struktur Permainan

Langkah awal yang dilakukan adalah memahami struktur dan mekanisme permainan *Little Alchemy 2*. Permainan ini melibatkan proses eksploratif di mana pemain dapat menggabungkan dua elemen untuk menciptakan elemen baru. Kombinasi yang valid menghasilkan elemen turunan, dan elemen tersebut bisa digunakan kembali untuk kombinasi selanjutnya. Dalam konteks algoritma, proses ini dapat direpresentasikan sebagai sebuah graf pencarian, di mana setiap simpul adalah elemen, dan sisi adalah pasangan kombinasi yang menghasilkan elemen baru. Pemahaman ini menjadi dasar untuk menyusun pemodelan data dan strategi pencarian.

3.1.2. Memahami Struktur Permainan

Setelah memahami bentuk umum permainan, langkah selanjutnya adalah memperoleh data kombinasi elemen (recipe). Data tersebut tidak disediakan dalam bentuk resmi oleh pengembang, sehingga perlu dilakukan scraping dari halaman wiki *Little Alchemy 2* menggunakan library goquery dalam bahasa Go. Proses ini mengekstrak kombinasi elemen-elemen yang menghasilkan elemen baru, kemudian menyusunnya ke dalam format JSON. File JSON ini akan menjadi sumber data utama yang digunakan oleh backend dalam membentuk graf dan memproses pencarian.

3.1.3. Merancang Representasi Graf

Data hasil scraping selanjutnya direpresentasikan dalam bentuk graf. Setiap elemen dianggap sebagai simpul dalam graf, dan setiap kombinasi dua elemen yang membentuk elemen baru akan direpresentasikan sebagai sisi atau relasi dalam struktur

tersebut. Representasi ini menggunakan struktur data seperti peta (map) dan daftar ketetanggaan (adjacency list) agar algoritma pencarian dapat bekerja secara efisien. Selain itu, perlu juga disimpan informasi kebalikan (reverse mapping) untuk mendukung visualisasi recipe tree dari target ke elemen dasar.

3.1.4. Mengimplementasikan Algoritma Pencarian

Dengan graf yang sudah terbentuk, langkah berikutnya adalah mengimplementasikan algoritma Breadth-First Search (BFS) dan Depth-First Search (DFS) untuk mencari jalur dari elemen dasar menuju elemen target. BFS digunakan untuk mencari jalur dengan jumlah langkah paling sedikit, sedangkan DFS cocok digunakan untuk eksplorasi menyeluruh yang mendalam. Untuk melengkapi fitur bonus, juga ditambahkan algoritma Bidirectional Search, yang bekerja dengan mencari dari dua arah sekaligus (dari elemen dasar dan elemen target) untuk mempercepat waktu pencarian.

3.1.5. Membangun Backend API

Aplikasi backend dikembangkan menggunakan bahasa Golang dengan framework Gin. Backend bertugas menerima permintaan pencarian dari frontend berupa target elemen dan pilihan algoritma, menjalankan proses pencarian pada graf, dan mengembalikan hasil berupa struktur pohon resep dalam format JSON. Backend ini juga menangani logika untuk pencarian satu solusi (terpendek) maupun banyak solusi (multiple recipe) dengan menggunakan multithreading, serta mencatat metrik seperti waktu eksekusi dan jumlah node yang dikunjungi.

3.1.6. Merancang Front End dan Integrasi

Frontend dibangun menggunakan framework Next.js atau React.js untuk menampilkan fitur interaktif seperti input elemen target, pemilihan algoritma, dan opsi jumlah solusi. Hasil pencarian ditampilkan dalam bentuk tree visualisasi yang menggambarkan jalur kombinasi dari elemen dasar menuju target. Visualisasi ini dirancang agar intuitif dan informatif, memperlihatkan hubungan antar elemen secara bertingkat. Selain itu, pengguna juga dapat melihat statistik pencarian seperti node visited dan searching time secara real time.

3.1.7. Melakukan Pengujian dan Validasi

Langkah terakhir adalah melakukan pengujian menyeluruh terhadap sistem yang telah dibangun. Pengujian dilakukan dengan mencoba berbagai elemen target yang memiliki tingkat kesulitan bervariasi, dan memverifikasi bahwa sistem mampu menghasilkan kombinasi yang valid, lengkap, serta dapat divisualisasikan secara benar. Analisis hasil pengujian juga digunakan untuk mengevaluasi performa algoritma dan efektivitas visualisasi yang telah diterapkan.

3.2. Pemetaan Masalah ke Algoritma DFS dan BFS

Permainan *Little Alchemy 2* dapat dimodelkan sebagai sebuah masalah pencarian jalur pada graf tidak berarah. Setiap elemen dalam permainan direpresentasikan sebagai simpul (node), sementara kombinasi dua elemen yang menghasilkan elemen baru direpresentasikan sebagai sisi (edge) yang menghubungkan pasangan simpul ke simpul hasil. Karena satu elemen dapat berasal dari berbagai pasangan kombinasi, struktur graf yang dibangun bersifat bercabang banyak (multi-parent), dan traversal terhadap graf ini menjadi kompleks. Untuk menyelesaikan masalah pencarian recipe, digunakan dua pendekatan algoritmik utama, yaitu Breadth-First Search (BFS) dan Depth-First Search (DFS), yang masing-masing memiliki karakteristik dan keunggulan tersendiri dalam menjelajahi ruang solusi.

3.2.1. Breadth-First Search (BFS)

Algoritma BFS dipilih karena kemampuannya dalam menemukan jalur terpendek dari elemen target menuju elemen-elemen dasar berdasarkan jumlah langkah kombinasi. Dalam konteks permainan ini, BFS melakukan pencarian dari elemen target, kemudian menelusuri semua kombinasi yang dapat menghasilkan elemen tersebut, lalu melanjutkan pencarian ke elemen-elemen penyusunnya. Setiap level pencarian mewakili satu langkah kombinasi.

BFS menyimpan informasi pasangan pembentuk (parent pair) pada setiap langkah agar pohon solusi dapat dibangun kembali dari hasil ke akar. Karena BFS menjelajahi kombinasi secara menyeluruh pada setiap level, ia menjamin bahwa solusi pertama yang ditemukan adalah jalur dengan jumlah langkah paling sedikit, sehingga cocok digunakan untuk mode pencarian “shortest recipe”. Meski begitu, BFS memerlukan memori yang

besar dan waktu eksekusi yang lebih lama pada elemen-elemen dengan banyak kemungkinan kombinasi.

3.2.2. Depth-First Search (DFS)

DFS digunakan untuk mengeksplorasi seluruh jalur kombinasi dari elemen target hingga ke elemen dasar secara mendalam. DFS juga bekerja mundur dari elemen target, namun berbeda dengan BFS yang menjelajahi per level, DFS langsung menelusuri satu cabang kombinasi hingga mentok ke elemen dasar sebelum kembali dan menelusuri cabang lain.

Pendekatan ini cocok untuk menemukan banyak jalur alternatif yang valid menuju suatu elemen target, karena DFS tidak berhenti pada satu solusi saja. Dalam implementasinya, DFS menyimpan urutan pasangan pembentuk elemen selama traversal dan melakukan backtracking jika jalur tersebut tidak berhasil mencapai elemen dasar. DFS cenderung lebih hemat memori dibandingkan BFS, tetapi tidak menjamin solusi yang ditemukan merupakan jalur terpendek. Pendekatan ini lebih sesuai digunakan pada mode pencarian “multiple recipe”.

3.3. Fitur Fungsional dan Arsitektur Aplikasi Web

3.3.1. Fitur Fungsional

Website *Alchendol* dirancang untuk menyediakan pengalaman interaktif dalam pencarian recipe elemen pada *Little Alchemy 2*, serta memperlihatkan bagaimana algoritma pencarian bekerja dalam konteks graf. Fitur-fitur berikut dirancang untuk mendukung kebutuhan tersebut.

3.3.1.1. Opsi Pemilihan Algoritma Pencarian (BFS, DFS, Bidirectional)

Pengguna dapat memilih salah satu dari tiga algoritma pencarian yang tersedia: Breadth-First Search (BFS), Depth-First Search (DFS), atau Bidirectional Search. Fitur ini memberikan fleksibilitas sekaligus memperkenalkan karakteristik dari masing-masing algoritma. Dengan adanya pilihan ini, pengguna dapat membandingkan efektivitas serta pendekatan pencarian yang dilakukan oleh tiap algoritma dalam menghasilkan recipe dari elemen dasar menuju target.

3.3.1.2. Toggle Mode Pencarian: Shortest Recipe dan Multiple Recipes

Setelah memilih algoritma, pengguna dapat menentukan mode pencarian. Jika memilih mode “recipe terpendek”, aplikasi akan mencari jalur kombinasi yang paling efisien dalam jumlah langkah. Sebaliknya, jika memilih mode “multiple recipe”, pengguna dapat mengeksplorasi beberapa jalur berbeda yang dapat menghasilkan elemen target. Fitur ini penting untuk menunjukkan bahwa satu elemen dapat memiliki lebih dari satu jalur pembentukan yang valid serta memungkinkan eksplorasi yang lebih luas terhadap struktur graf permainan.

3.3.1.3. Parameter Jumlah Recipe (Multiple Recipe)

Pada mode multiple recipe, pengguna dapat menentukan jumlah maksimal solusi yang ingin ditampilkan. Fitur ini memberikan kendali kepada pengguna atas kedalaman eksplorasi, sekaligus membatasi beban komputasi. Pencarian multiple recipe ini juga akan dioptimalkan melalui penerapan multithreading pada sisi backend agar performanya tetap efisien meskipun eksplorasi dilakukan secara masif.

3.3.1.4. Visualisasi Recipe dalam Bentuk Tree

Hasil pencarian ditampilkan dalam bentuk pohon (tree) yang menggambarkan urutan kombinasi dari elemen dasar hingga ke elemen target. Visualisasi ini dirancang untuk membantu pengguna memahami struktur solusi secara hierarkis. Dengan pendekatan tree, pengguna dapat melihat elemen mana saja yang perlu digabung, di level berapa, serta dari mana saja jalur solusi tersebut bercabang.

3.3.1.5. Statistik Pencarian (Waktu dan Node yang Dikunjungi)

Selain menampilkan hasil akhir, aplikasi juga menyajikan metrik tambahan berupa waktu pencarian dan jumlah node yang dikunjungi selama proses traversal. Informasi ini berguna untuk mengevaluasi performa dari algoritma yang digunakan serta mengukur efisiensi pencarian terhadap elemen target tertentu.

3.3.2. Arsitektur Web

Website *Alchendol* dibangun dengan arsitektur client-server yang memisahkan tanggung jawab antara sisi frontend (client) dan backend (server). Pemisahan ini

memungkinkan sistem berjalan secara modular, di mana frontend bertugas sebagai antarmuka pengguna, sementara backend menangani logika algoritma dan pengolahan data.

3.3.2.1. Frontend (Next.js)

Frontend dibangun menggunakan framework Next.js (berbasis React.js) untuk menyediakan halaman interaktif yang responsif dan dinamis. Pengguna dapat mengakses aplikasi melalui browser, memilih algoritma pencarian (BFS, DFS, atau Bidirectional), mengatur mode pencarian (terpendek atau multiple), serta memasukkan elemen target yang ingin dicari. Setelah permintaan dikirim, frontend akan menerima data hasil pencarian dalam format JSON dan merendernya dalam bentuk visualisasi pohon (tree) menggunakan library D3.js atau struktur custom.

3.3.2.2. Backend (Golang dan Gin)

Backend dikembangkan menggunakan bahasa Go dengan framework Gin untuk menangani routing dan API. Backend menyediakan endpoint seperti /api/search, yang menerima parameter seperti algo, target, dan maxRecipes. Berdasarkan input tersebut, backend akan memuat struktur data dari file elements.json (hasil scraping dari wiki), lalu menjalankan algoritma pencarian yang sesuai: BFS, DFS, atau Bidirectional.

Untuk mode pencarian multiple recipe, backend mengaktifkan multithreading menggunakan fitur goroutine. Setiap jalur pencarian atau kombinasi potensial dapat dieksekusi dalam thread terpisah (goroutine), sehingga pencarian beberapa recipe dapat dilakukan secara paralel. Dengan menggunakan channel dan sinkronisasi, hasil dari beberapa thread ini akan dikumpulkan dan difilter sesuai batas maksimal recipe yang diminta. Pendekatan ini secara signifikan mempercepat proses pencarian dibandingkan pendekatan sekvensial biasa, terutama pada elemen target yang memiliki banyak kemungkinan jalur kombinasi.

Backend juga mencatat statistik seperti waktu pencarian dan jumlah node yang dikunjungi, lalu mengembalikannya bersama struktur pohon hasil pencarian dalam format JSON ke sisi frontend.

3.3.2.3. Komunikasi Client-Server

Arsitektur aplikasi memungkinkan frontend dan backend berkomunikasi melalui permintaan HTTP. Ketika pengguna mengisi form pencarian dan menekan tombol “Search”, frontend akan mengirim permintaan GET atau POST ke endpoint backend dengan parameter yang sesuai. Backend memproses pencarian dan mengembalikan response berisi tree resep dalam bentuk JSON. Frontend kemudian memproses response ini dan menampilkannya secara visual ke pengguna.

3.3.2.4. Data - JSON

Data kombinasi elemen diperoleh melalui proses scraping dari halaman Little Alchemy 2 Wiki menggunakan pustaka goquery. Data ini kemudian dikonversi menjadi file elements.json yang berisi daftar elemen dan resep pembentuknya. Backend memuat file ini saat startup dan menggunakannya sebagai dasar untuk membentuk graf dan melakukan traversal.

3.4. Contoh Ilustrasi Kasus

3.4.1. Deskripsi Kasus

Sebagai ilustrasi cara kerja sistem, diasumsikan pengguna ingin mencari jalur kombinasi untuk membentuk elemen “Ninja Turtle”. Pada tampilan aplikasi, pengguna memilih algoritma Bidirectional dan mode pencarian “multiple recipe” dengan jumlah maksimal recipe yang dicari sebanyak 2 recipe. Tujuan dari pencarian ini adalah untuk mengeksplorasi berbagai kemungkinan jalur pembentukan elemen secara efisien dan cepat, tanpa harus menelusuri keseluruhan graf.

3.4.2. Algoritma Pencarian

Algoritma Bidirectional Search bekerja dengan pendekatan dua arah secara paralel, yang dirancang untuk mempercepat proses pencarian jalur dalam graf. Dalam konteks aplikasi *Alchendol*, algoritma ini memulai pencarian dari dua titik: satu dari elemen target, yaitu “Ninja Turtle”, yang ditelusuri ke atas ke arah komponen pembentuknya; dan satu lagi dari elemen dasar, seperti “Air”, “Water”, “Earth”, dan “Fire”, yang ditelusuri ke bawah ke arah kemungkinan kombinasi hasil. Setiap arah

pencarian menggunakan prinsip Breadth-First Search (BFS) untuk menjelajahi level demi level dari kedua sisi graf.

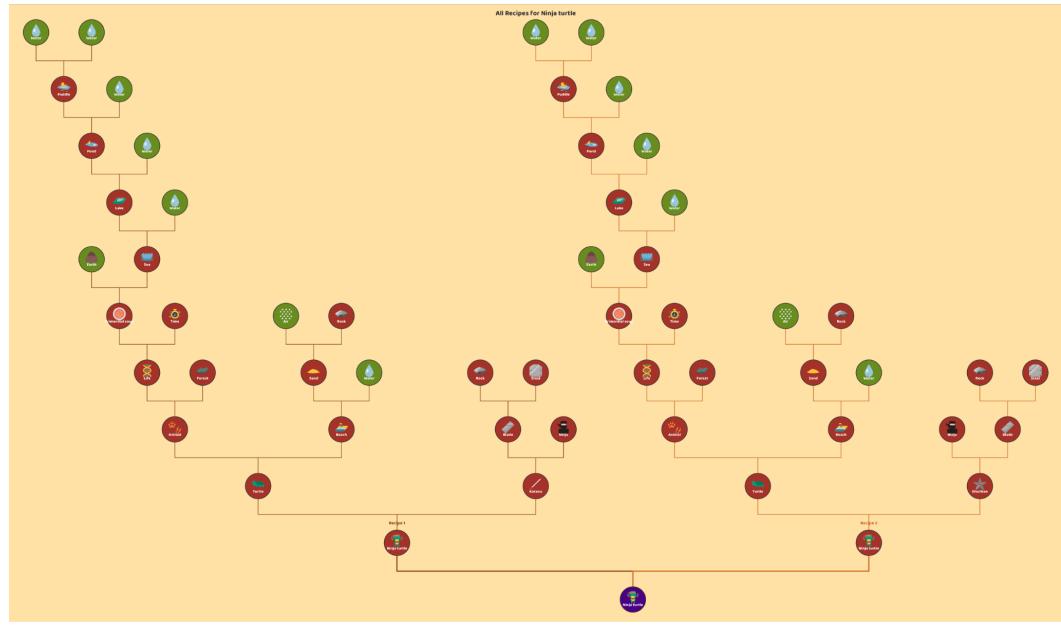
Proses traversal dijalankan secara bersamaan dari kedua arah ini, dengan masing-masing membangun frontier (daftar node yang sedang dijelajahi). Dari arah target, sistem menelusuri kombinasi seperti “Ninja + Turtle” yang membentuk “Ninja Turtle”, lalu melanjutkan pencarian ke elemen-elemen yang dapat membentuk “Ninja” dan “Turtle”. Di sisi lain, arah dari elemen dasar akan mengeksplorasi jalur kombinasi dari elemen-elemen awal ke elemen-elemen turunan seperti “Beach”, “Human”, “Egg”, atau “Katana”.

Ketika dua jalur pencarian dari arah berbeda bertemu di satu simpul yang sama, misalnya ketika pencarian dari elemen dasar mencapai “Human”, sementara pencarian dari target juga bertemu pada node tersebut, algoritma akan menyambungkan kedua jalur tersebut menjadi satu lintasan lengkap dari elemen dasar ke elemen target. Teknik pencocokan pertemuan simpul ini (frontier overlap) memungkinkan sistem menghentikan pencarian lebih awal dibandingkan pencarian satu arah seperti DFS atau BFS murni.

Dengan pendekatan dua arah ini, Bidirectional Search secara signifikan meningkatkan efisiensi pencarian, terutama pada elemen-elemen kompleks yang memiliki kedalaman kombinasi yang besar. Dalam kasus pencarian elemen “Ninja Turtle”, algoritma ini mampu menemukan dua jalur solusi berbeda dengan eksplorasi node yang lebih sedikit dan waktu komputasi yang lebih singkat dibanding pendekatan lainnya.

3.4.3. Hasil yang Diperoleh

Sistem berhasil menemukan dua jalur kombinasi berbeda untuk menghasilkan “Ninja Turtle”, ditunjukkan pada visualisasi berikut. Masing-masing pohon merepresentasikan satu recipe unik yang dibentuk dari pasangan-pasangan elemen yang valid. Struktur visual ini memperlihatkan bagaimana elemen dasar seperti Water, Earth, Rock, dan Animal saling berinteraksi dalam beberapa tahap kombinasi hingga membentuk elemen target.



Gambar 7. Hasil Pencarian Ninja Turtle

(sumber: Penulis)

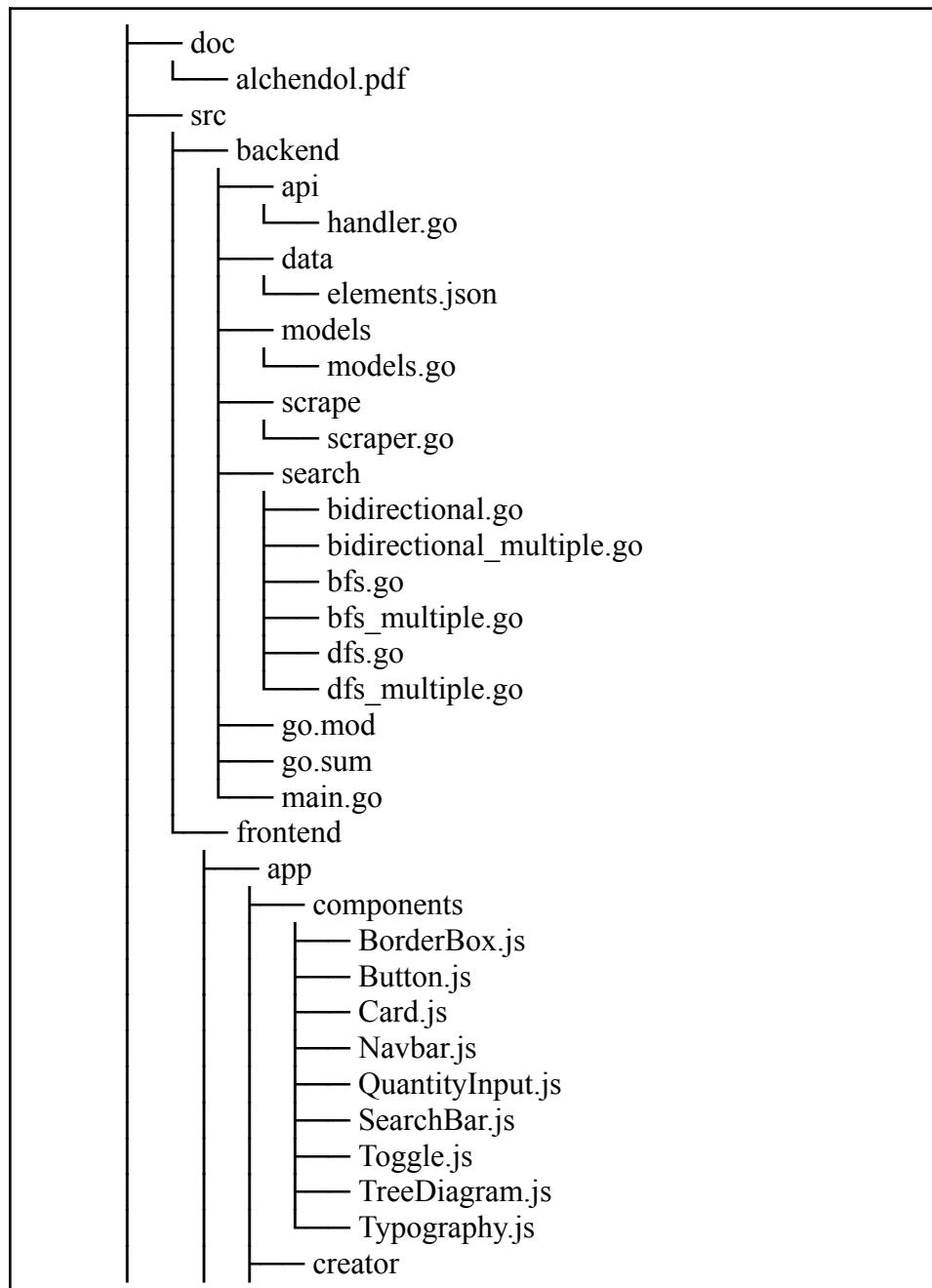
BAB 4

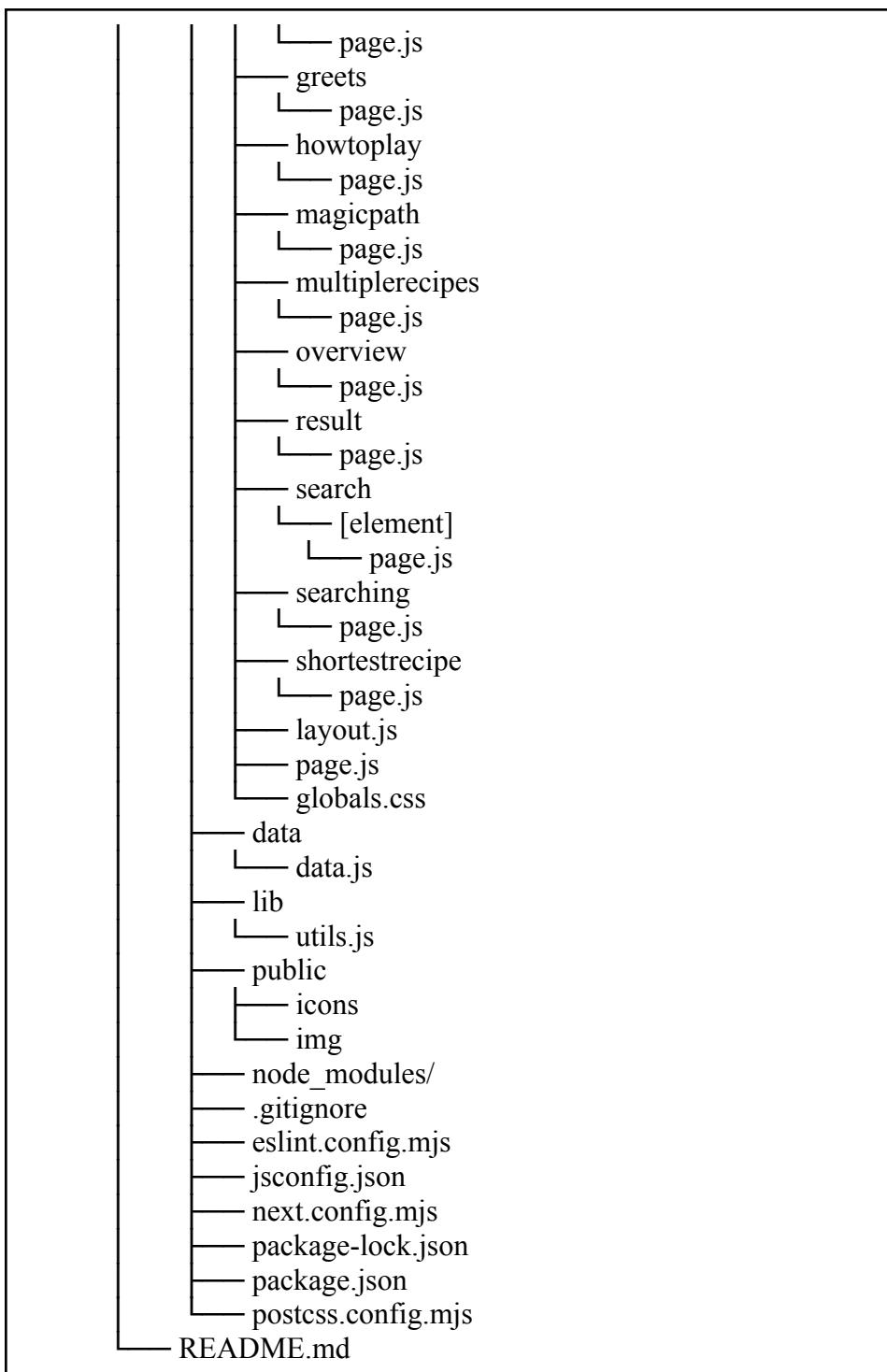
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi Teknis Program

4.1.1. Struktur Data

4.1.3.1. Struktur Data Repository





4.1.3.2. Struktur Data Program

Tiga struktur data ini berperan penting dalam merepresentasikan elemen dan membentuk pohon resep pada sistem pencarian *Little Alchemy 2*. Struktur

Element merepresentasikan satu elemen dalam permainan, berisi nama elemen, daftar kombinasi yang bisa membentuknya, serta tingkat (tier) yang menunjukkan urutan atau kompleksitas penciptaannya. Elemen dengan tier 0 dianggap sebagai elemen dasar.

Selanjutnya, RecipeNode digunakan untuk membentuk pohon resep secara rekursif, di mana setiap node mewakili satu elemen dan komponen penyusunnya. Struktur ini berguna untuk menelusuri dan membangun kembali jalur penciptaan dari elemen dasar hingga elemen target. Sementara itu, RecipeTree menyusun hasil pencarian dalam bentuk pohon yang lebih terstruktur, terdiri dari elemen utama (root), dua komponen pembentuk (left dan right), serta daftar anak (children) yang menunjukkan rekursi bertingkat. Struktur ini dirancang khusus untuk mempermudah visualisasi dan penyajian resep di sisi tampilan (UI). Singkatnya, Element menyimpan informasi elemen dasar, RecipeNode menyusun logika pembuatan secara rekursif, dan RecipeTree menyajikan hasil pencarian dalam format yang mudah ditampilkan.

```
type Element struct {
    Name      string      `json:"name"`
    Recipes   []string    `json:"recipes"`
    Tier     int         `json:"tier"`
}

type RecipeNode struct {
    Element    string      `json:"element"`
    Components []RecipeNode `json:"components,omitempty"`
    IsBasic    bool        `json:"isBasic"`
}

type RecipeTree struct {
    Root      string      `json:"root"`
    Left     string      `json:"Left"`
    Right    string      `json:"Right"`
    Tier     string      `json:"Tier"`
    Children  []RecipeTree `json:"children"`
}
```

4.1.2. Breadth First Search (BFS)

4.1.2.1 MultipleBFS

Melakukan pencarian jalur kombinasi menuju elemen target dengan metode Breadth-First Search secara paralel dan multithreaded. Fungsi ini dirancang untuk menghasilkan banyak resep (multiple recipes) dengan memproses berbagai kombinasi secara efisien hingga mencapai batas maksimum resep yang ditentukan.

4.1.2.2 BFS

Melakukan pencarian satu jalur tercepat (shortest recipe) menuju elemen target menggunakan metode Breadth-First Search. Fungsi ini bekerja secara sistematis dan memprioritaskan kombinasi dengan tier terendah untuk menghasilkan pohon resep yang paling optimal.

4.1.2.3 bfsSearchComponentToBasics

Digunakan oleh BFS untuk menelusuri komponen penyusun suatu elemen hingga mencapai elemen dasar. Fungsi ini membangun bagian-bagian RecipeTree dari elemen kompleks ke elemen dasar menggunakan BFS.

4.1.2.4 bfsProcessCombinationWithVariations

Memproses satu pasangan komponen dan mengeksplorasi semua variasi kombinasinya dengan pendekatan BFS. Hasil berupa RecipeTree dikirim ke channel untuk dikumpulkan sebagai resep unik.

4.1.2.5 findBFSCOMPONENTVariations

Menemukan berbagai versi alternatif dari satu elemen komponen menggunakan BFS. Fungsi ini menghasilkan beberapa RecipeTree yang berbeda untuk satu elemen, sesuai dengan variasi kombinasinya.

4.1.2.6 efficientBfsSearchComponent

Mencari satu jalur pembuatan elemen dari elemen dasar dengan BFS secara efisien dan caching. Fungsi ini mengembalikan RecipeTree tunggal yang mewakili satu rute logis dari bawah ke atas.

4.1.2.7 buildTreeFromNodeMap

Membangun struktur pohon resep lengkap dari peta node yang dihasilkan selama proses BFS. Ini digunakan setelah seluruh traversal selesai untuk merekonstruksi pohon dari data mentah.

4.1.2.8 buildCompleteTree

Serupa dengan buildTreeFromNodeMap, namun digunakan dalam pencarian BFS tunggal. Fungsi ini merekonstruksi RecipeTree dari nodeMap secara rekursif dan mendalam.

4.1.2.9 deepCopyRecipeTree

Membuat salinan mendalam dari sebuah RecipeTree agar struktur aslinya tidak terpengaruh oleh perubahan data di tempat lain.

4.1.2.10 generateBFSTreeKey

Menghasilkan string unik sebagai representasi dari struktur RecipeTree, digunakan untuk mendeteksi dan menghindari duplikasi resep.

4.1.2.11 generateBFSTreeKeyHelper

Fungsi bantu rekursif untuk generateBFSTreeKey, menyusun representasi unik dari setiap node dan child-nya.

4.1.2.12 processRecipes

Mengorganisasi data resep menjadi dua struktur penting: recipeMap (peta kombinasi) dan tierMap (tingkatan elemen), sebagai dasar untuk pencarian BFS.

4.1.2.13 IsBasicElementWithTier

Memeriksa apakah elemen tertentu merupakan elemen dasar berdasarkan peta tingkatan (tierMap).

4.1.2.14 createBasicElementLeaf

Membuat node RecipeTree kosong untuk elemen dasar (tier 0), tanpa komponen penyusun.

4.1.2.15 createRecipeTreeNode

Membuat satu node RecipeTree lengkap dari data kombinasi: elemen utama, komponen kiri dan kanan, serta tingkatannya.

4.1.3. Depth First Search (DFS)

4.1.3.1 DFS

Melakukan pencarian satu jalur resep menuju elemen target menggunakan algoritma DFS (Depth-First Search). Hanya menghasilkan satu RecipeTree yang valid dengan pencarian terdalam terlebih dahulu.

4.1.3.2 dfsSearchRecipe

Fungsi rekursif utama untuk menjelajahi jalur resep secara DFS. Mencari jalur dari target ke elemen dasar dan membangun RecipeNode untuk setiap kombinasi valid.

4.1.3.3 ConvertToRecipeTree

Mengonversi hasil DFS (RecipeNode) ke dalam bentuk RecipeTree agar dapat ditampilkan atau dikembalikan melalui API.

4.1.3.4 createCompleteRecipeTree

Membangun RecipeTree lengkap dari kombinasi dua komponen dengan memastikan tiap jalur mengarah hingga elemen dasar, digunakan dalam validasi hasil DFS.

4.1.3.5 MultipleDFS

Melakukan pencarian banyak resep menuju elemen target menggunakan DFS paralel dengan multithreading. Menghasilkan banyak variasi jalur resep (multiple recipes) secara efisien dan dibatasi oleh maxRecipes.

4.1.3.6 processComponentVariations

Menelusuri dan membangun berbagai variasi pohon untuk satu elemen secara rekursif dengan pendekatan DFS. Digunakan oleh worker pada MultipleDFS.

4.1.3.7 convertRecipeTreeToNode

Mengubah RecipeTree kembali menjadi RecipeNode agar dapat disimpan di cache, membantu menghindari pencarian ulang.

4.1.3.8 generateCompleteTreeKey

Membuat key unik berbasis struktur lengkap dari RecipeTree, untuk memastikan hasil tidak duplikat dalam pencarian multiple.

4.1.3.9 generateCompleteTreeKeyHelper

Fungsi bantu rekursif untuk menyusun representasi tekstual unik dari RecipeTree, digunakan oleh generateCompleteTreeKey.

4.1.4. Bidirectional

4.1.4.5 MultipleBidirectional

Melakukan pencarian kombinasi resep dari dua arah secara paralel dan multithreaded untuk menghasilkan banyak variasi resep (multiple recipes). Menggunakan channel dan goroutine untuk efisiensi, serta membatasi jumlah hasil berdasarkan maxRecipes.

4.1.4.6 processRecipeCombinationWithMultipleVariations

Memproses satu kombinasi resep (komponen kiri dan kanan) dan mengeksplorasi semua kemungkinan variasi pada tiap level menggunakan pencarian dua arah. Setiap kombinasi yang valid dikirim ke result channel.

4.1.4.7 findAllRecipeVariations

Menemukan semua variasi resep yang valid untuk sebuah elemen secara rekursif, sambil menghindari siklus dengan visited map. Fungsi ini memastikan bahwa setiap hasil kombinasi tidak berulang dan tidak menyalahi aturan tier.

4.1.4.8 countTreeNodes

Menghitung total jumlah node dalam sebuah RecipeTree, digunakan untuk menyortir hasil pencarian berdasarkan kompleksitas pohon (jumlah langkah/tingkatan).

4.1.4.9 generateTreeKey

Membuat key unik berbasis struktur top-level dari sebuah RecipeTree (Root, Left, Right), digunakan untuk mendeteksi duplikasi hasil.

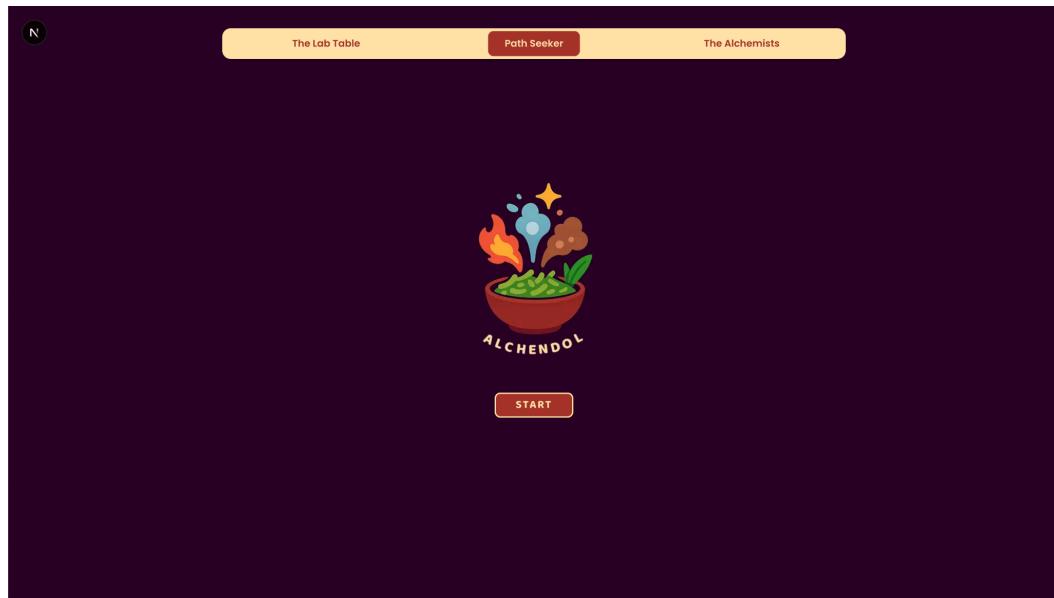
4.1.4.10 getTierAsString

Mengambil nilai tier dari suatu elemen dalam bentuk string, berdasarkan data dari elementMap.

4.2. Tata Cara Penggunaan Program

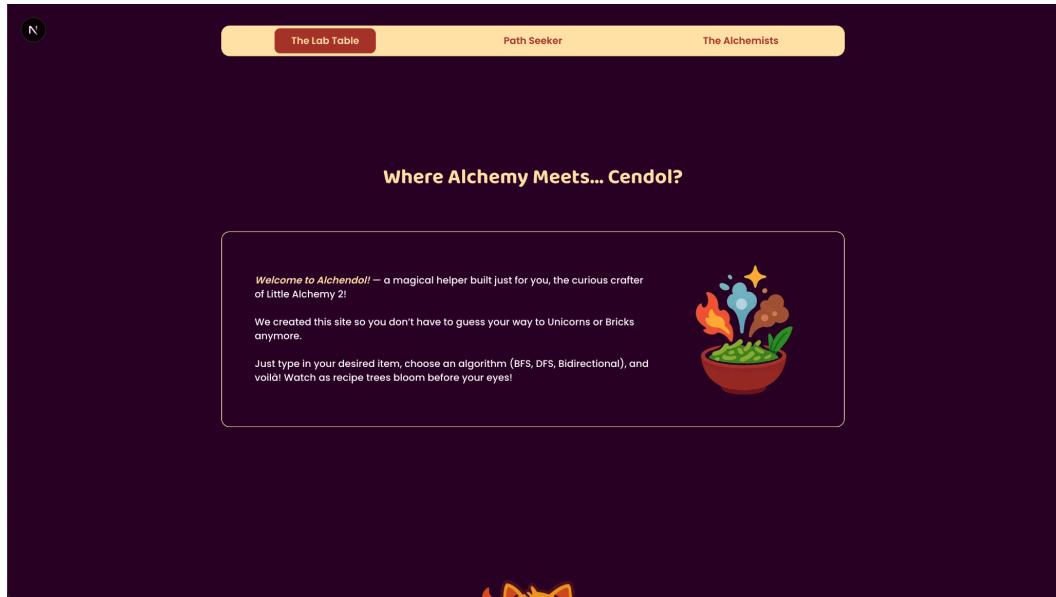
4.2.1 Interface Program

4.2.1.1 Path Seeker



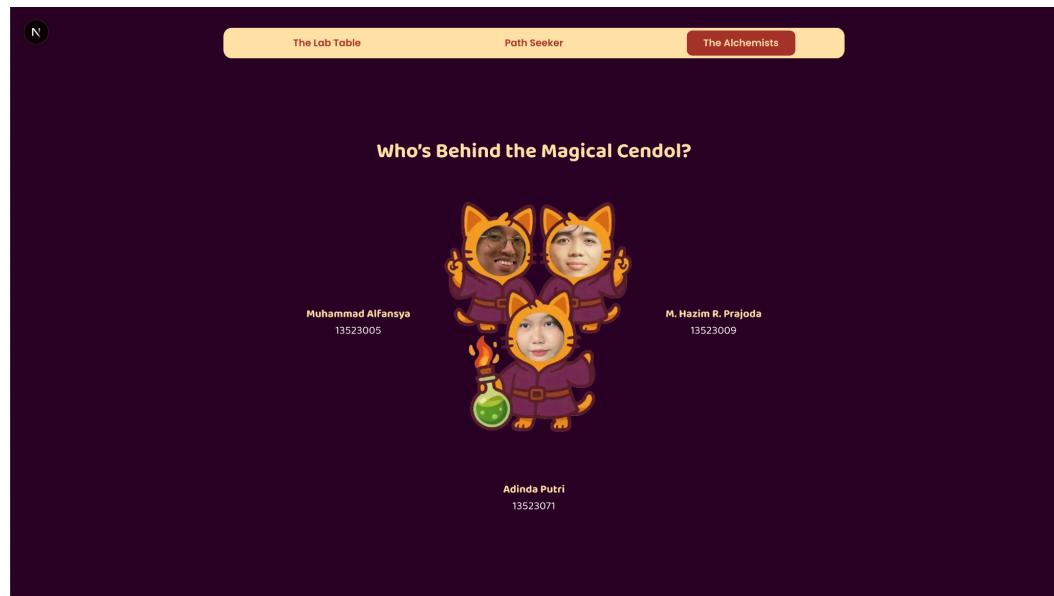
Halaman Path Seeker merupakan tampilan utama yang mengarahkan pengguna untuk memulai proses pencarian kombinasi elemen dalam permainan *Little Alchemy 2*. Dengan desain sederhana dan elemen visual khas Alchendol di tengah layar, pengguna disambut oleh ilustrasi tematik serta tombol Start yang menjadi titik awal eksplorasi. Setelah tombol ini diklik, pengguna akan diarahkan menuju halaman input pencarian, tempat mereka dapat memilih algoritma dan mode pencarian untuk menemukan jalur pembuatan elemen target. Tampilan ini berfungsi sebagai gerbang masuk ke fitur utama aplikasi.

4.2.1.2. The Lab Table



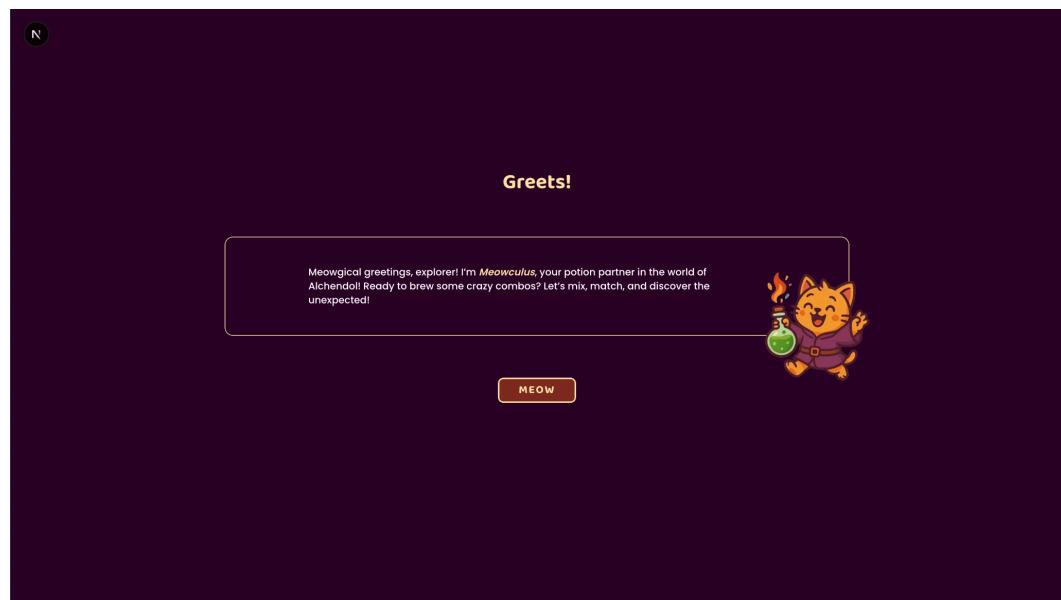
The Lab Table adalah halaman beranda interaktif dari aplikasi Alchendol yang menyajikan gambaran umum tentang aplikasi secara menyeluruh. Halaman ini dilengkapi dengan tampilan visual yang menarik, termasuk maskot aplikasi, Meowculus, yang merepresentasikan karakter ilmuwan alkimia, serta informasi ringkas mengenai cara kerja Alchendol. Pengguna juga dapat melihat video demo aplikasi pada halaman ini, yang berisi penjelasan fitur, alur penggunaan, dan cara kerja pencarian elemen dalam bentuk visual yang mudah dipahami. The Lab Table berperan sebagai titik awal eksplorasi pengguna sebelum masuk ke fitur utama aplikasi.

4.2.1.3. The Alchemists



The Alchemists adalah halaman yang berisi informasi tentang siapa saja yang membuat aplikasi Alchendol. Di sini ditampilkan nama dan NIM anggota kelompok yang terlibat dalam pengembangan aplikasi. Halaman ini cukup sederhana namun bermakna, sebagai pengingat bahwa di balik fitur-fitur dan algoritma yang digunakan, ada kerja sama nyata antar anggota tim untuk menyelesaikan proyek ini bersama-sama.

4.2.1.4. Greets



Setelah pengguna menekan tombol Start pada halaman Path Seeker, mereka akan diarahkan ke halaman Greets. Di halaman ini, pengguna disambut oleh Meowculus, maskot Alchendol yang berperan sebagai partner ramuan digital mereka. Dengan gaya penyampaian yang ramah dan penuh semangat, Meowculus memberikan sapaan singkat sekaligus ajakan untuk mulai bereksperimen mencampur elemen. Tombol Meow di bagian bawah berfungsi untuk melanjutkan ke proses berikutnya dalam pencarian.

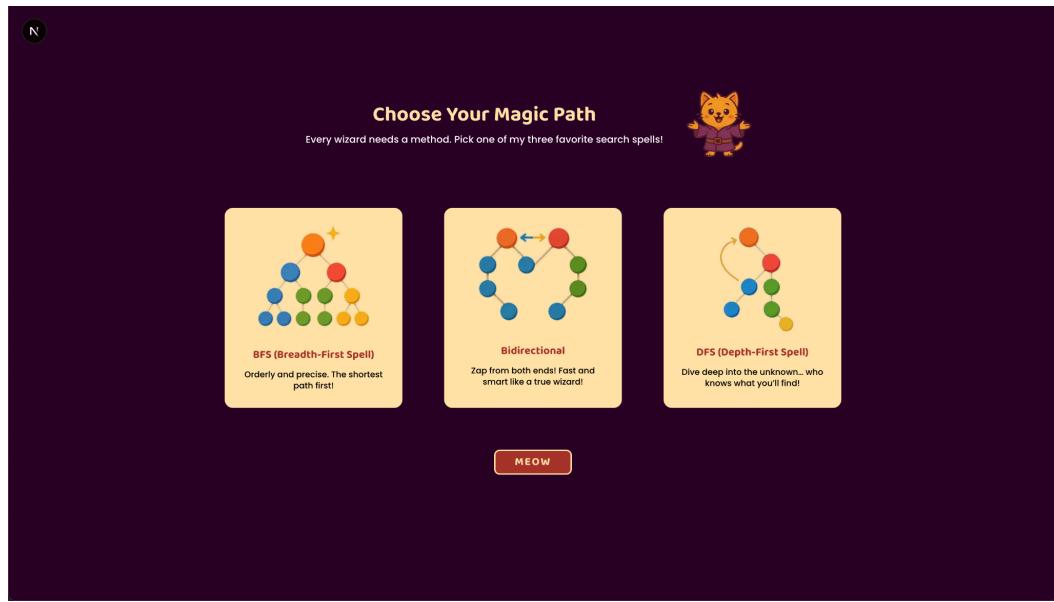
4.2.1.5. How to Play



Setelah pengguna mengklik tombol Meow pada halaman Greets, mereka akan diarahkan ke halaman How to Play. Di halaman ini, Meowculus akan memandu pengguna memahami cara kerja aplikasi Alchendol. Meowculus menjelaskan langkah-langkah dasar pencarian elemen, mulai dari memilih magic path berupa algoritma pencarian (DFS, BFS, atau Bidirectional), memilih mode pencarian berupa shortest recipe atau multiple recipe, memilih elemen target yang ingin dicari, hingga mendapatkan hasil. Halaman ini dirancang agar pengguna memahami alur penggunaan aplikasi sebelum masuk ke pencarian sesungguhnya, sehingga pengalaman eksplorasi mereka menjadi lebih terarah dan menyenangkan.

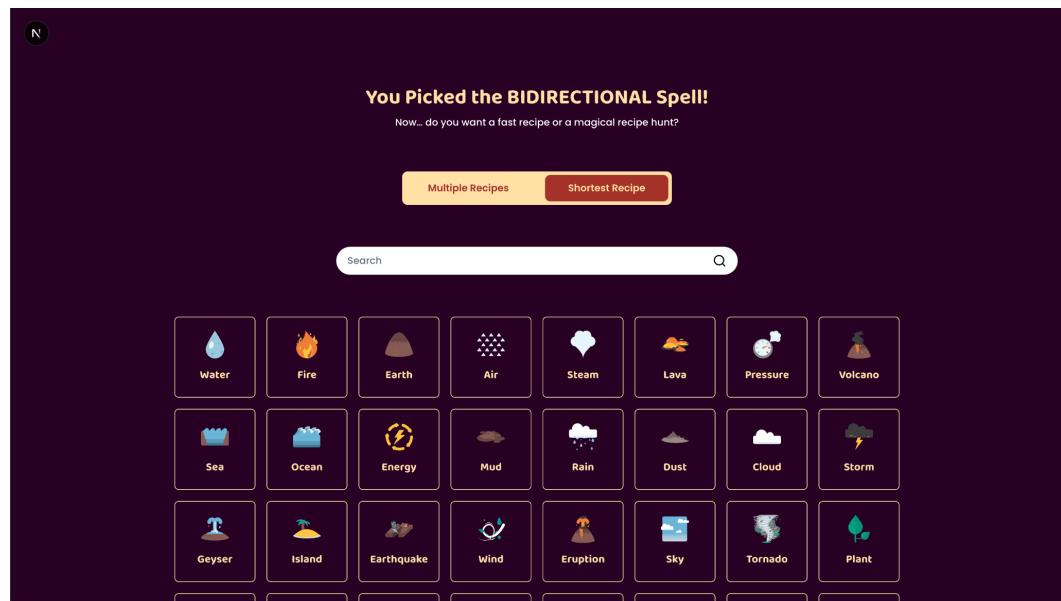
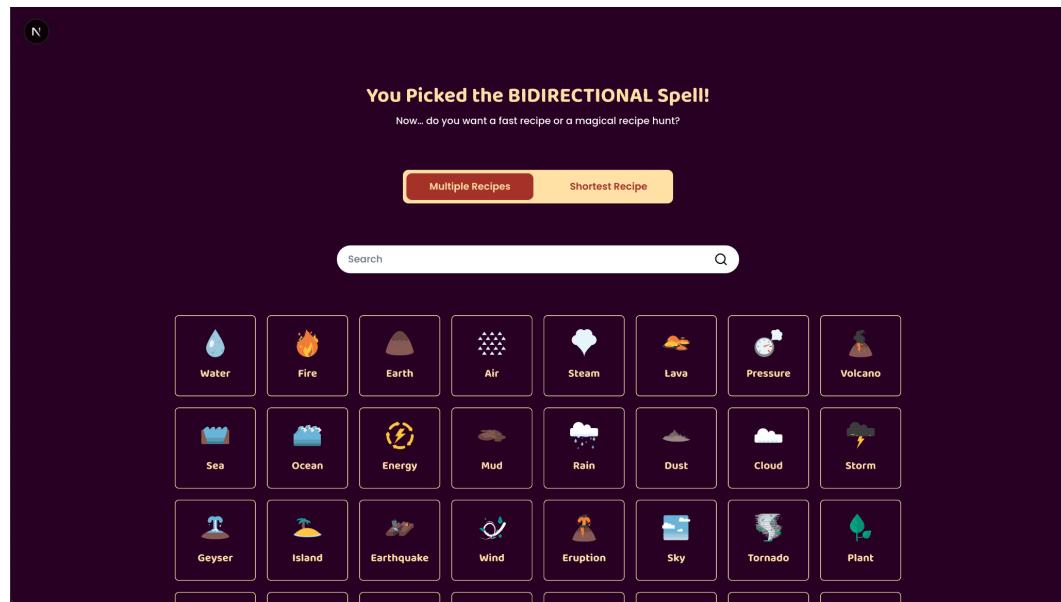
4.2.2 Fitur

4.2.2.1 Magic Path



Setelah pengguna mengklik tombol Meow pada halaman Greets, mereka akan diarahkan ke halaman How to Play. Di halaman ini, Meowculus akan memandu pengguna memahami cara kerja aplikasi Alchendol. Meowculus menjelaskan langkah-langkah dasar pencarian elemen, mulai dari memilih magic path berupa algoritma pencarian (DFS, BFS, atau Bidirectional), memilih mode pencarian berupa shortest recipe atau multiple recipe, memilih elemen target yang ingin dicari, hingga mendapatkan hasil. Halaman ini dirancang agar pengguna memahami alur penggunaan aplikasi sebelum masuk ke pencarian sesungguhnya, sehingga pengalaman eksplorasi mereka menjadi lebih terarah dan menyenangkan.

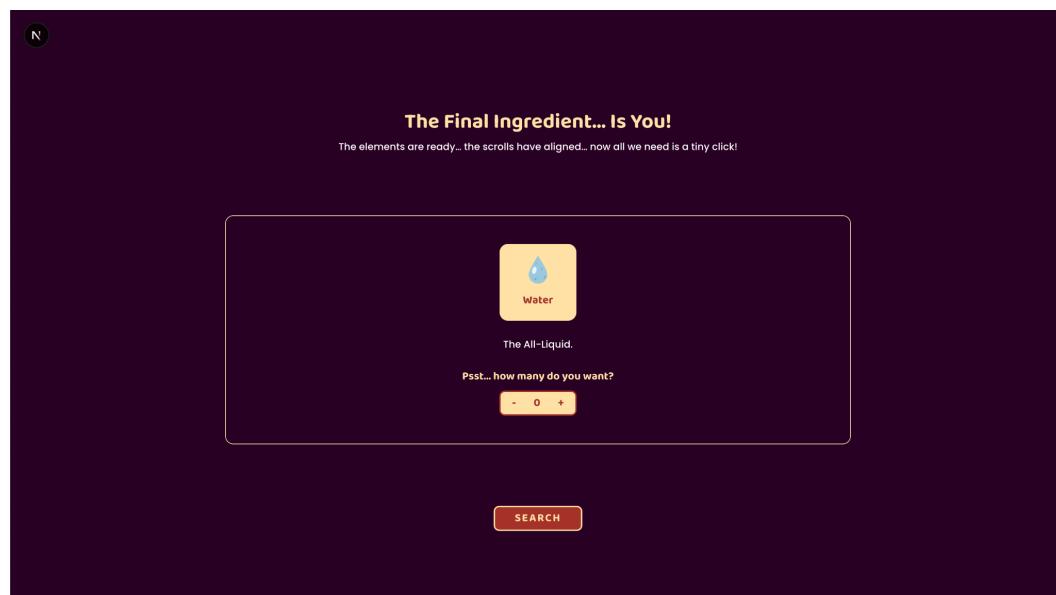
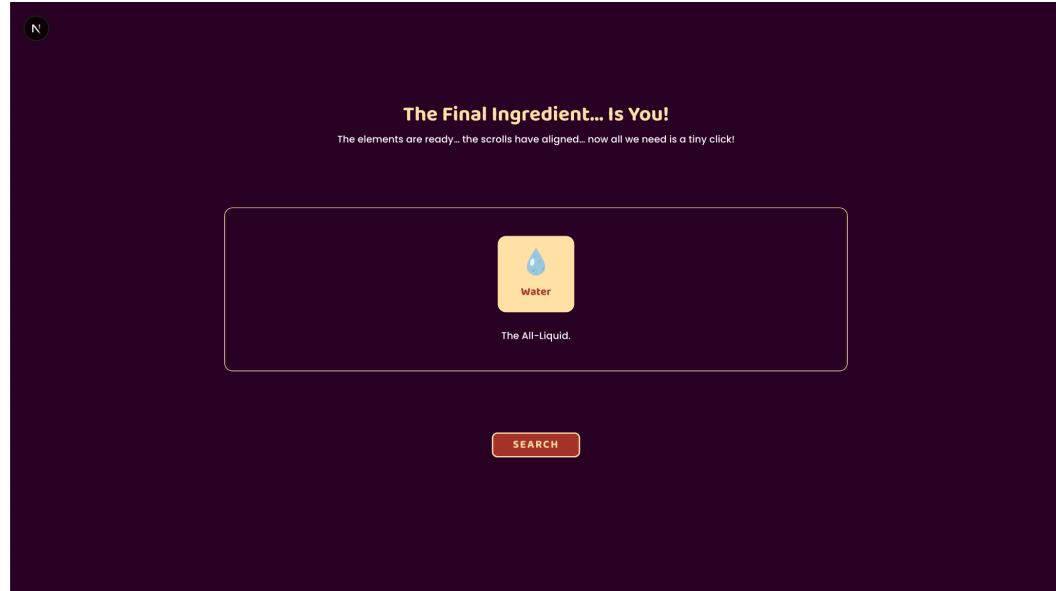
4.2.2.2 Element Search



Setelah pengguna memilih algoritma pencarian dan mode pencarian pada halaman Magic Path, mereka akan diarahkan ke halaman ini dengan menekan tombol Meow. Halaman ini merupakan titik awal pencarian elemen berdasarkan mode dan jenis path yang telah ditentukan sebelumnya, baik itu *Multiple Recipes* maupun *Shortest Recipe*. Pengguna dapat memilih elemen target dari daftar elemen yang tersedia. Jika mode yang dipilih adalah *Multiple Recipes*, maka pengguna akan menentukan jumlah recipe yang ingin dicari untuk setiap elemen,

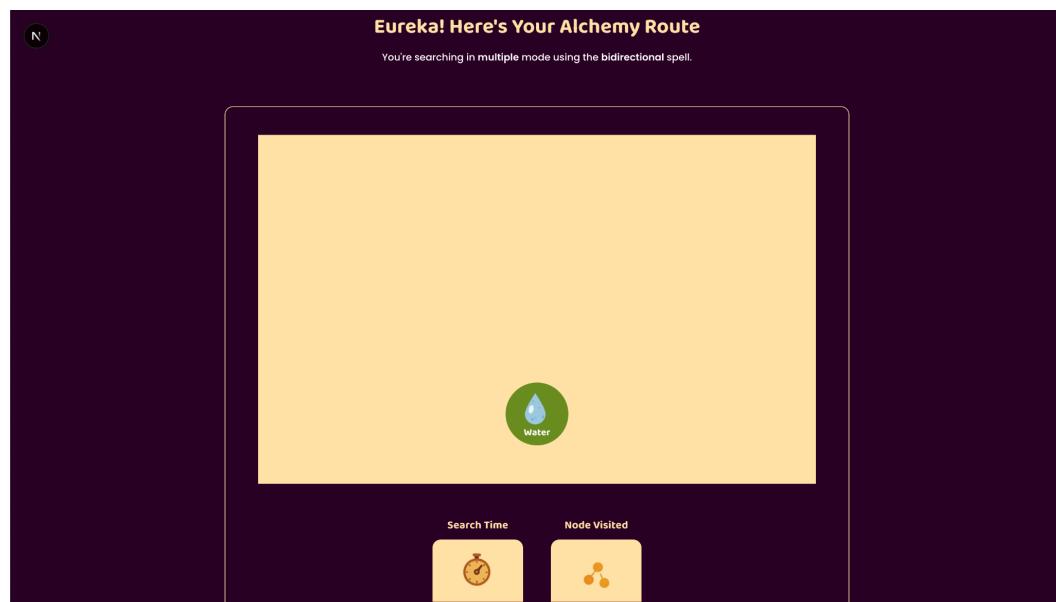
yang kemudian diproses secara paralel oleh backend menggunakan multithreading. Sementara itu, jika mode *Shortest Recipe* dipilih, pencarian akan difokuskan untuk menemukan satu jalur kombinasi terpendek menuju elemen target. Halaman ini menjadi penghubung antara pemilihan strategi pencarian dan proses eksplorasi elemen dalam aplikasi Alchendol.

4.2.2.3 Element Preview



Halaman ini merupakan preview pencarian sebelum pengguna benar-benar menjalankan proses pencarian kombinasi elemen. Setelah memilih elemen target dan mode pencarian (multiple/single), pengguna diarahkan ke halaman ini sebagai langkah konfirmasi akhir. Jika mode pencarian yang dipilih adalah multiple recipes, maka akan ditampilkan komponen Quantity Input untuk menentukan jumlah maksimal recipe yang ingin ditemukan; sedangkan pada mode shortest recipe, komponen ini tidak muncul. Halaman ini juga menampilkan detail elemen yang dipilih serta tombol Search untuk memulai proses pencarian ke backend. Tulisan tematik “The Final Ingredient... Is You!” menambahkan kesan bahwa pengguna adalah bagian penting dalam proses eksplorasi alkimia.

4.2.2.4 Result



Setelah pengguna menekan tombol Search pada halaman preview, mereka akan diarahkan ke halaman hasil pencarian. Di halaman ini, sistem menampilkan visualisasi pohon kombinasi (recipe tree) berdasarkan elemen target yang dipilih dan konfigurasi pencarian sebelumnya. Pohon ini menunjukkan jalur pembuatan elemen dari elemen dasar secara bertahap dan interaktif. Selain itu, halaman ini juga menyajikan statistik pencarian, yaitu waktu pencarian (Search Time) dan jumlah simpul (Node) yang dikunjungi, untuk memberi gambaran performa dan

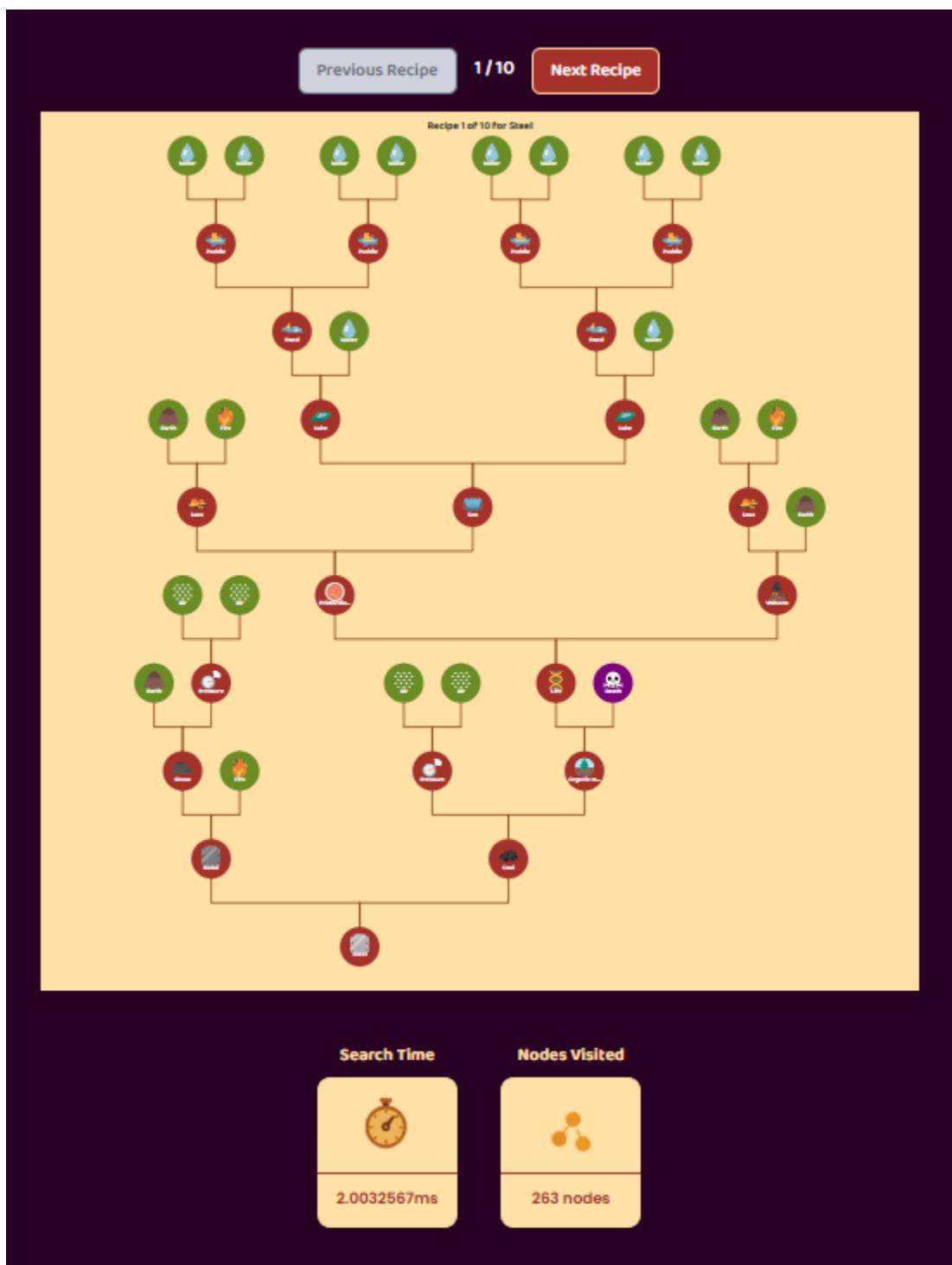
efisiensi algoritma yang digunakan. Halaman ini menjadi output utama dari Alchendol yang menggabungkan aspek visual, informatif, dan interaktif dari hasil eksplorasi.

4.3. Pengujian

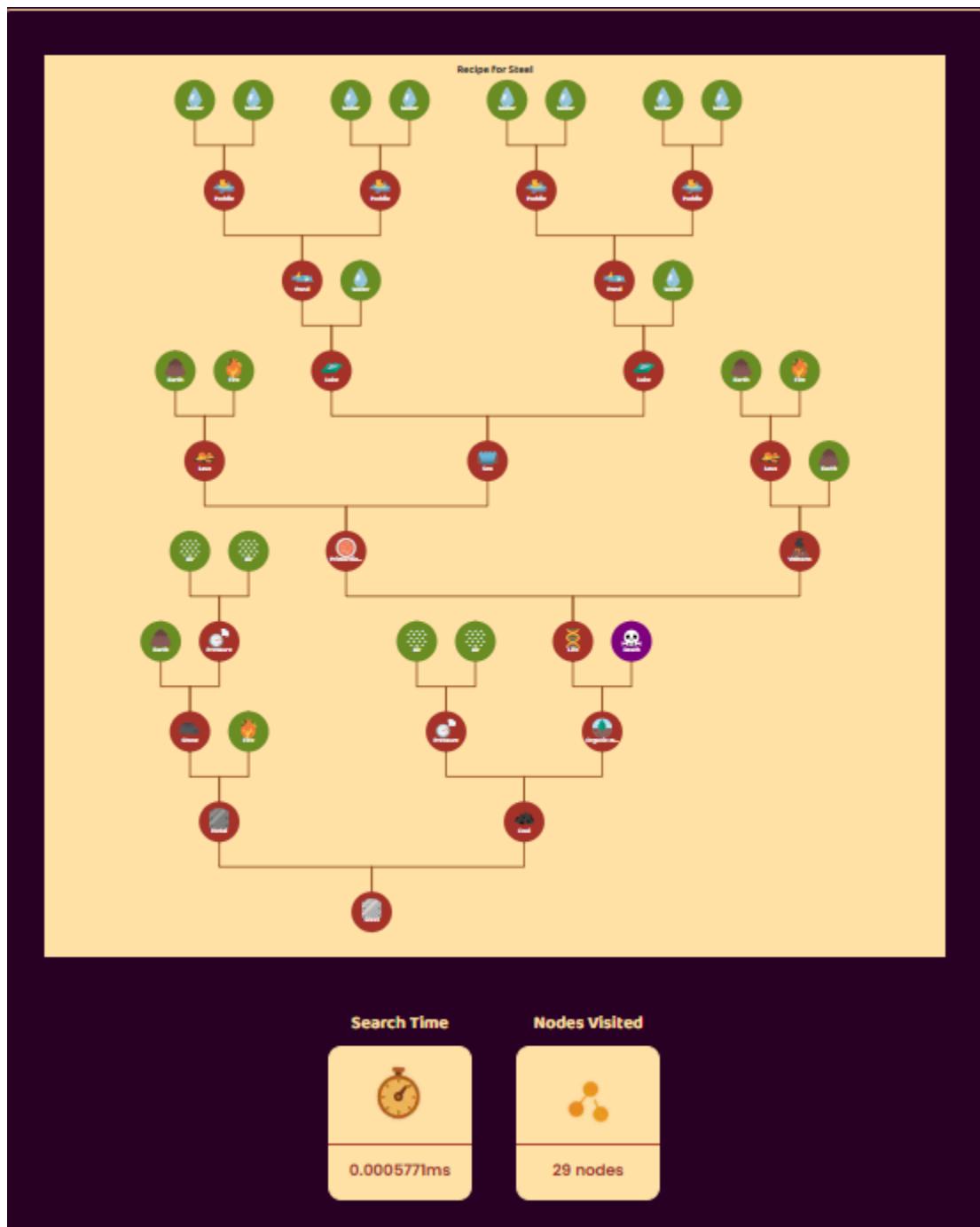
4.3.1. Pengujian 1 : Pencarian Resep dengan Shortest Recipe BFS



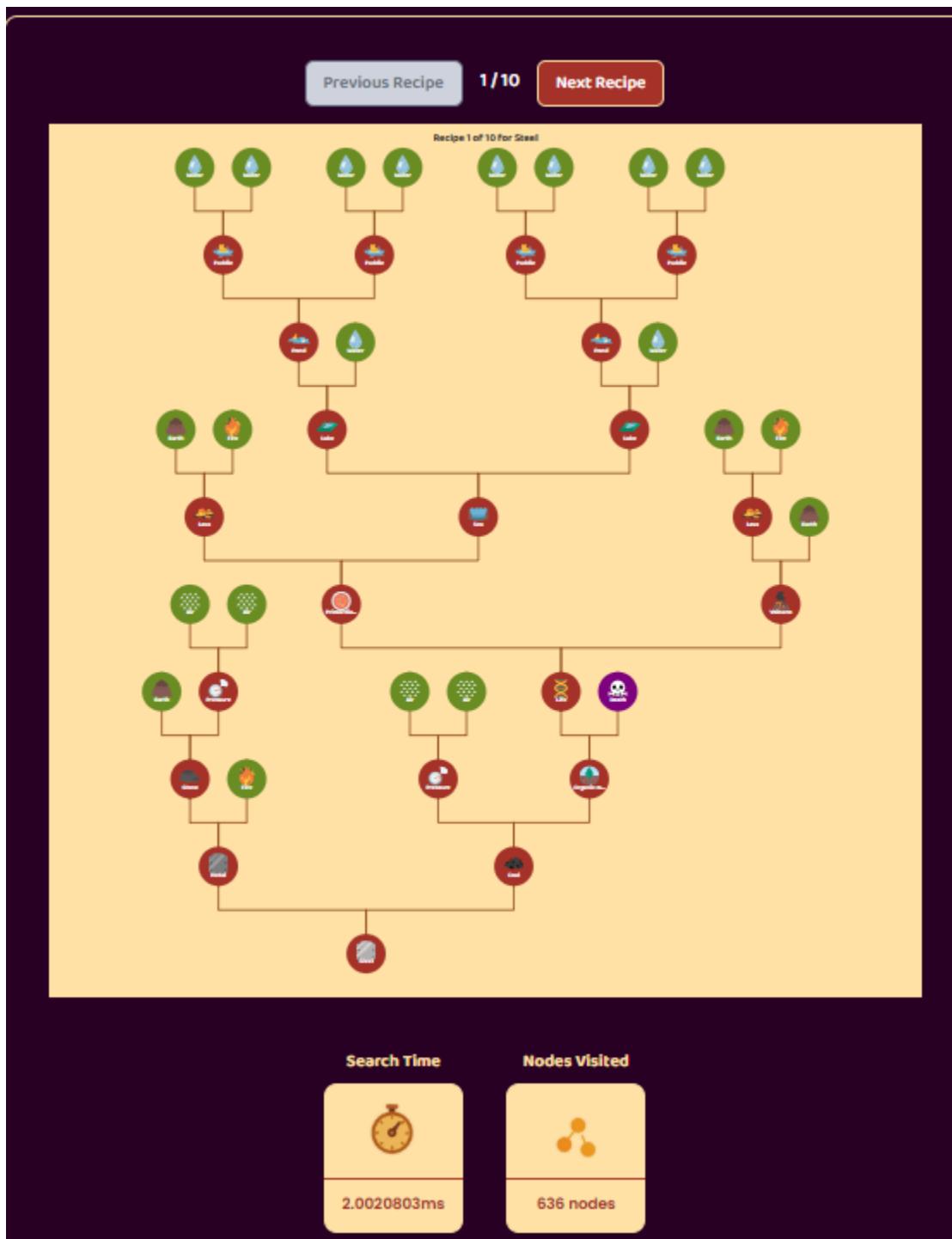
4.3.2. Pengujian 2 : Pencarian Resep dengan Multiple Recipes BFS



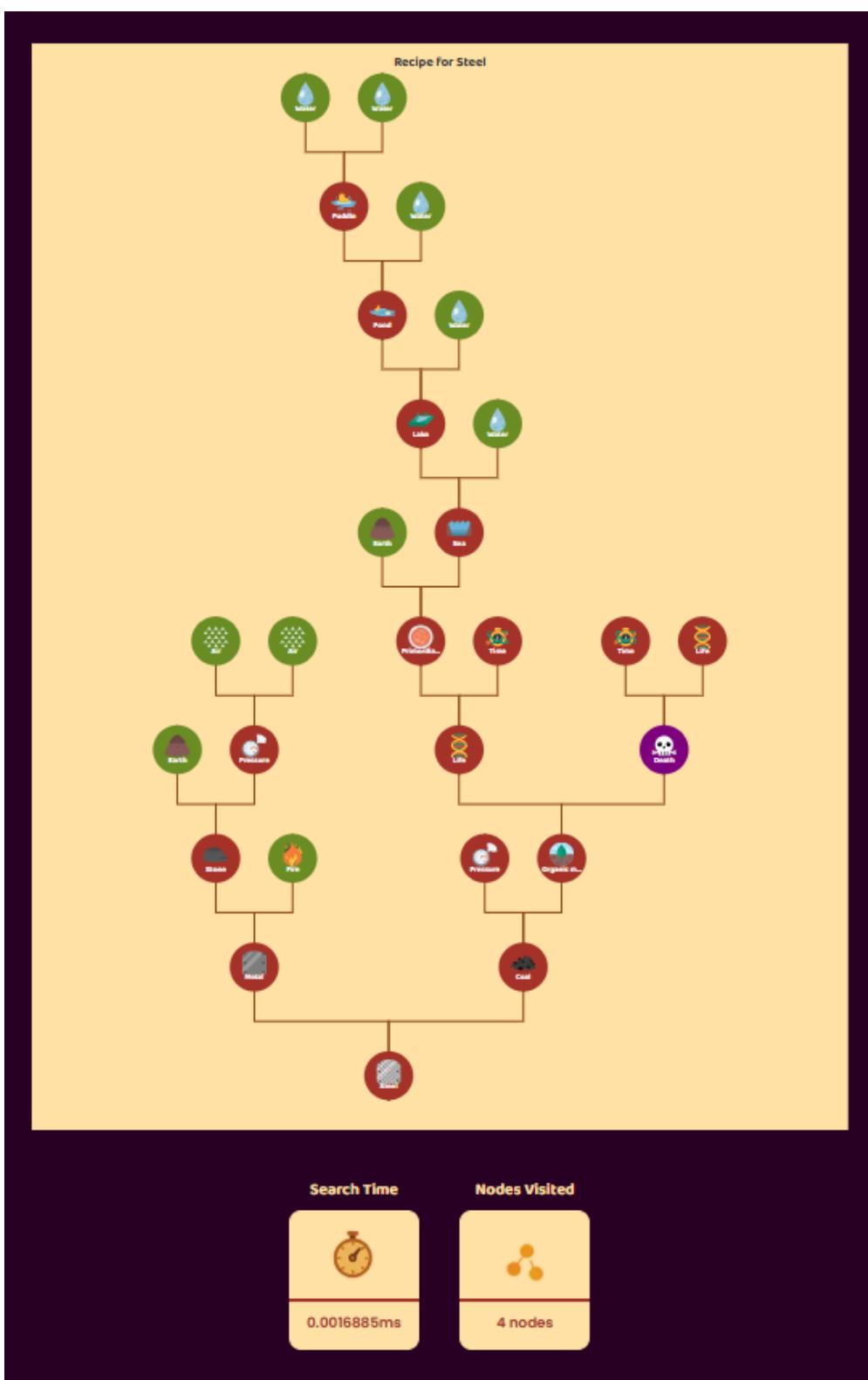
4.3.3. Pengujian 3 : Pencarian Recipe dengan Shortest Recipe DFS



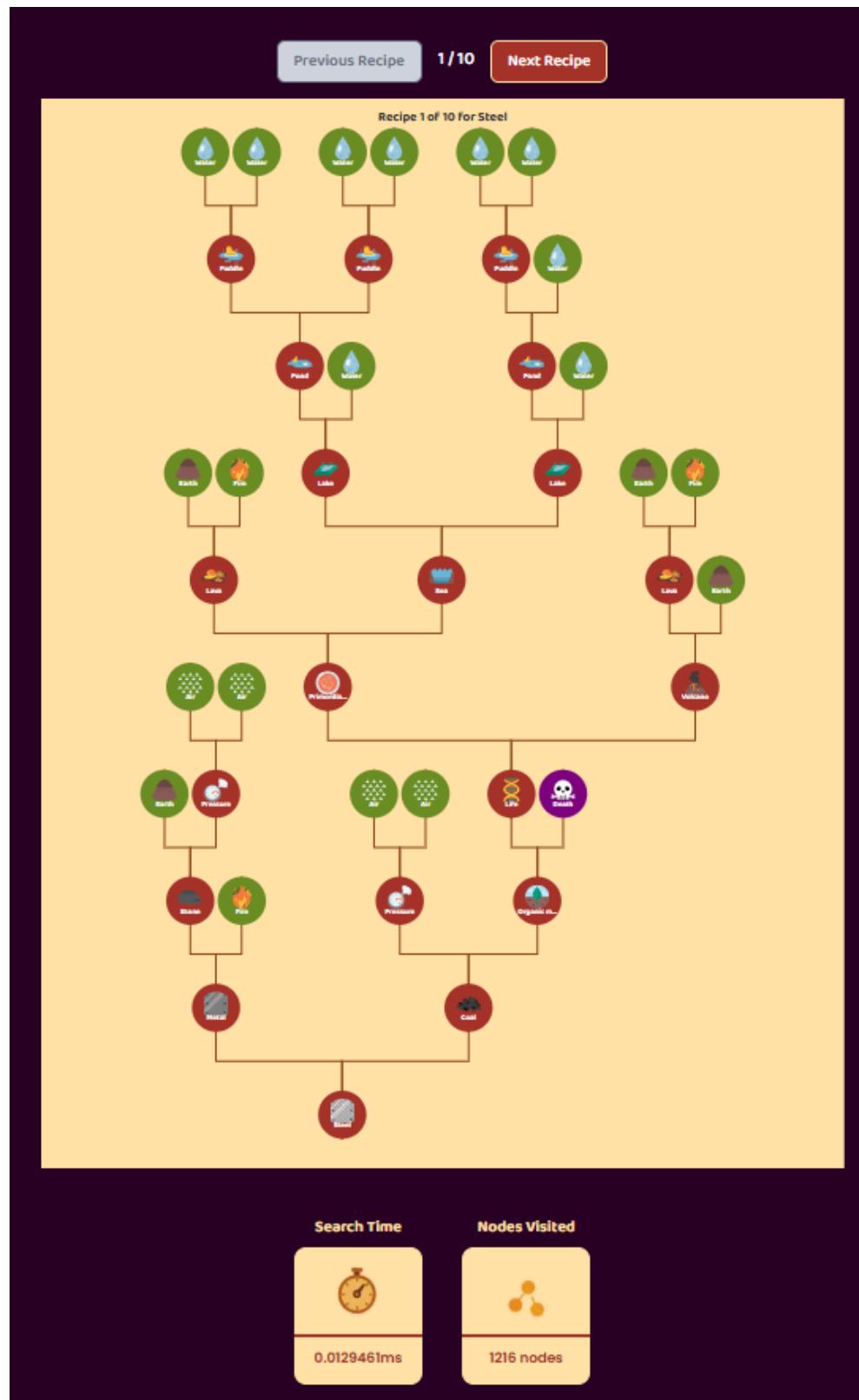
4.3.4. Pengujian 4 : Pencarian Recipe dengan Multiple Recipes DFS



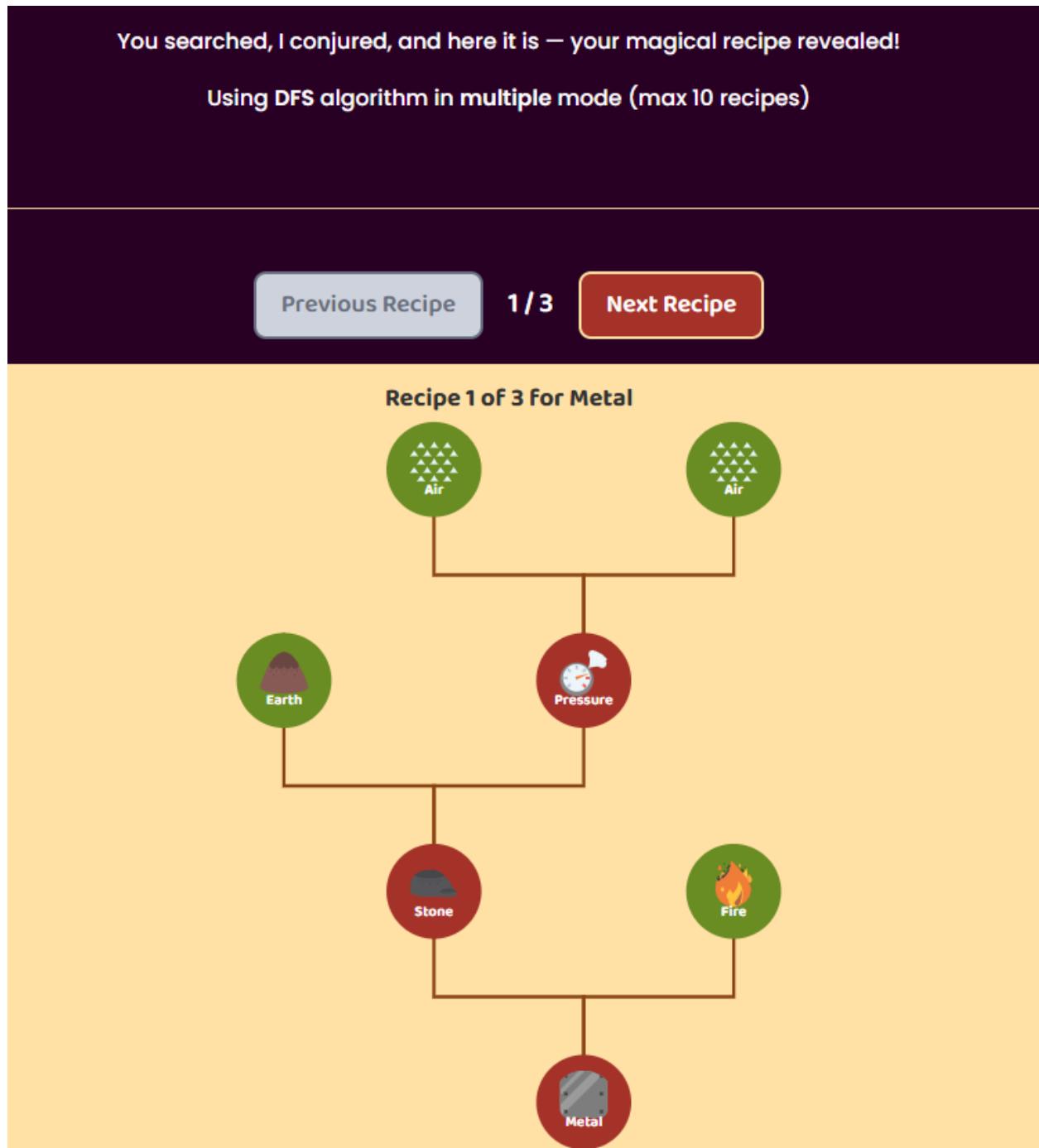
4.3.5. Pengujian 5 : Pencarian Recipe dengan Shortest Recipe Bidirectional



4.3.6. Pengujian 6 : Pencarian Recipe dengan Multiple Recipes Bidirectional



4.3.7. Pengujian 7 : Resep Maksimal yang Tersedia Kurang dari Inputan Max Recipe



4.3.9. Pengujian 8 : Inputan Max Recipe sangat banyak

The screenshot shows the Picnic app's search interface. At the top, there is a yellow button labeled "Picnic" with a small icon. Below it is a text input field containing the word "Picnic". A message below the input says "Not just eating outdoors, but eating outdoors with a purpose.". The next section asks "How many recipes do you want to discover?", with a numeric input set to "300" and a range slider from "-" to "+". A warning message "OOH MY PCC! Are you sure you want to do this?" is displayed in red. A large orange "SEARCH" button is at the bottom of this section.

You searched, I conjured, and here it is — your magical recipe revealed!

Using DFS algorithm in multiple mode (max 300 recipes)

1 / 114

Previous Recipe Next Recipe

A large grid of nodes representing the search space, showing a complex tree structure with many branches and nodes. The grid is mostly yellow with some green and red nodes.

Search Time	Nodes Visited
2.0031187ms	486 nodes

4.4. Analisis Hasil Pengujian

Berdasarkan hasil pengujian, Aplikasi dan fitur utama sudah bisa berjalan dengan baik. Pada pencarian single recipe untuk resep steel, pencarian secara Bidirectional mendapatkan hasil

dengan node yang dikunjungi lebih sedikit dan pencarian secara BFS mendapat hasil dengan node yang dikunjungi paling banyak.

Pada pencarian multiple recipes, pencarian secara bidirectional menjadi yang paling tidak efisien dengan node yang dikunjungi menjadi paling banyak dan pencarian secara BFS menjadi yang paling efisien dengan node yang dikunjungi menjadi yang paling sedikit dikunjungi

Jika inputan MaxRecipe yang dimasukkan lebih banyak dari jumlah variasi resep yang tersedia, resep yang ditampilkan akan sesuai dengan maksimal variasi yang tersedia. Masukan pengguna sudah divalidasi, hanya bisa berupa bilangan bulat dan tidak negatif.

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dalam Tugas Besar ini, kelompok Alchendol berhasil membangun aplikasi web Alchendol, yang dilengkapi dengan tiga algoritma pencarian graf, yaitu Breadth-First Search (BFS), Depth-First Search (DFS), dan Bidirectional Search. Masing-masing algoritma memiliki karakteristik dan keunggulan yang berbeda dalam mencari jalur kombinasi elemen di permainan *Little Alchemy 2*. BFS terbukti efisien dalam menemukan *recipe* terpendek karena eksplorasi dilakukan secara melebar. DFS unggul dalam menemukan berbagai jalur alternatif karena pencarinya yang mendalam. Sementara itu, Bidirectional Search menunjukkan performa yang jauh lebih cepat karena pencarian dilakukan dari dua arah secara simultan.

Secara keseluruhan, dapat disimpulkan bahwa tidak ada satu algoritma yang selalu unggul dalam segala kondisi. Setiap strategi pencarian bekerja paling efektif dalam situasi tertentu, misalnya, BFS cocok untuk solusi pendek, DFS untuk eksplorasi luas, dan Bidirectional untuk efisiensi waktu. Penerapan ketiga algoritma ini dalam Alchendol memperlihatkan bagaimana pendekatan strategi algoritma dapat diterjemahkan ke dalam sebuah website yang interaktif, edukatif, dan aplikatif.

5.2. Saran

Dari Tugas Besar 2 IF2211 Strategi Algoritma ini, kami menyarankan agar ke depannya dilakukan eksplorasi lebih lanjut terhadap variasi algoritma pencarian graf, khususnya yang melibatkan heuristik seperti A* atau algoritma pencarian berbasis biaya (*cost-based search*). Hal ini dapat membuka kemungkinan solusi yang lebih efisien dan adaptif terhadap kompleksitas struktur *recipe* dalam *Little Alchemy 2*. Selain itu, visualisasi pohon yang telah diterapkan pada aplikasi Alchendol masih dapat dikembangkan lebih interaktif, seperti dengan fitur zoom, animasi progresif pencarian, atau penandaan node yang terlibat dalam solusi akhir, guna meningkatkan pengalaman pengguna dan aspek edukatif dari aplikasi.

5.3. Refleksi

Melalui penggerjaan Tugas Besar 2 IF2211 Strategi Algoritma ini, kami menyadari bahwa menggabungkan konsep algoritma pencarian dengan pengembangan aplikasi web bukanlah hal yang mudah. Kami belajar banyak tentang bagaimana teori yang dipelajari di kelas, seperti BFS dan DFS, dapat diimplementasikan dalam dunia nyata. Tantangan terbesar tidak hanya datang dari sisi algoritma backend, tetapi juga dari sinkronisasi frontend-backend, pengelolaan data hasil scraping, hingga optimasi performa multiple recipes melalui multithreading. Kedepannya, kami berharap dapat lebih memahami integrasi antara strategi algoritma dan pengembangan sistem secara menyeluruh, serta mengembangkan aplikasi yang tidak hanya berfungsi dengan baik, tetapi juga memberikan pengalaman pengguna yang menarik dan bermanfaat.

LAMPIRAN

No	Poin	Ya	Tidak
1	Aplikasi dapat dijalankan.	✓	
2	Aplikasi dapat memperoleh data <i>recipe</i> melalui scraping.	✓	
3	Algoritma DFS dan BFS dapat menemukan <i>recipe</i> elemen dengan benar.	✓	
4	Aplikasi menampilkan visualisasi recipe elemen sesuai spesifikasi	✓	
5	Aplikasi mengimplementasikan multithreading.	✓	
6	Membuat laporan sesuai dengan spesifikasi.	✓	
7	Membuat bonus video dan diunggah pada YouTube.	✓	
8	Membuat bonus algoritma pencarian Bidirectional.	✓	
9	Membuat bonus Live Update.		✓
10	Aplikasi di- <i>containerize</i> dengan Docker.	✓	
11	Aplikasi di- <i>deploy</i> dan dapat diakses melalui internet.	✓	

Tautan Repository GitHub

https://github.com/adndax/Tubes2_alchendol

Tautan Video

<https://www.youtube.com/watch?v=vxrFMGibedg&si=laPnpG1nCSRkVlj>

DAFTAR PUSTAKA

Munir, R. & Maulidevi, N. U. 2025. “Breadth/Depth First Search (Bagian 1)”. [Online]. Tersedia:

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-\(2025\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf) [9 Mei 2025].

Munir, R. & Maulidevi, N. U. 2025. “Breadth/Depth First Search (Bagian 2)”. [Online].

Tersedia:

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-\(2025\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian2.pdf) [9 Mei 2025].

Munir, R. 2025. “Tugas Besar 2: Pemanfaatan Algoritma BFS dan DFS dalam Pencarian Recipe pada Permainan Little Alchemy 2”. [Online]. Tersedia:

[≡ Spesifikasi Tugas Besar 2 Stima 2024/2025](#) . [9 Mei 2025].

GeeksforGeeks. 2024. “Bidirectional Search”. [Online]. Tersedia:

<https://www.geeksforgeeks.org/bidirectional-search/>. [9 Mei 2025].