

TUGAS BESAR 2
IF2211 - STRATEGI ALGORITMA
PEMANFAATAN ALGORITMA BFS
DAN DFS DALAM PENCARIAN
RECIPE PADA PERMAINAN LITTLE
ALCHEMY 2



Oleh:

Muhammad Ra'if Alkautsar 13523011

Samuel Gerrard Hamongan Girsang 13523064

Lutfi Hakim Yusra 13523084

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2025

DAFTAR ISI

DAFTAR ISI-----	1
BAB I	
DESKRIPSI TUGAS-----	3
BAB II	
LANDASAN TEORI-----	5
1. Graf dan Tree-----	5
2. Breadth-first Search (BFS)-----	6
3. Depth-first Search (DFS)-----	6
4. Back End Web Development-----	7
5. Front End Web Development-----	7
6. Docker-----	8
7. Deployment-----	8
8. Live Update-----	8
BAB III	
ANALISIS PEMECAHAN MASALAH-----	10
1. Pemecahan dan Pemetaan Masalah-----	10
2. Arsitektur Aplikasi Web-----	10
3. Ilustrasi Kasus-----	10
BAB IV	
IMPLEMENTASI DAN PENGUJIAN-----	12
1. Implementasi Back-end-----	12
1. Package recipe-----	12
2. Package scraper-----	20
3. Package server-----	23
4. main.go-----	29
2. Implementasi Front-end-----	29
3. Pengujian-----	31
4. Analisis-----	32
BAB V	
KESIMPULAN, SARAN, DAN REFLEKSI-----	33
LAMPIRAN-----	34
REFERENSI-----	35

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, kami 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*

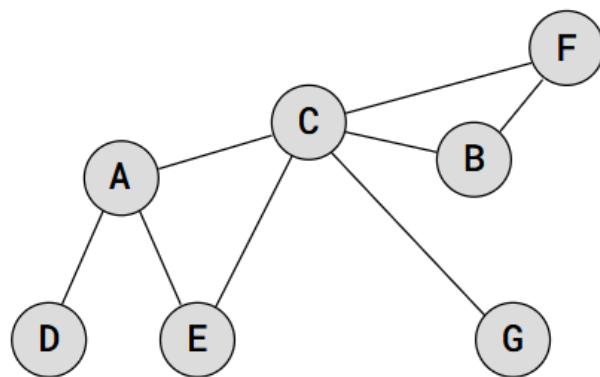
Untuk mendapatkan elemen turunan pemain dapat melakukan *combine* antara 2 elemen untuk menghasilkan elemen baru. Elemen turunan yang telah didapatkan dapat digunakan kembali oleh pemain untuk membentuk elemen lainnya.

BAB II

LANDASAN TEORI

1. Graf dan Tree

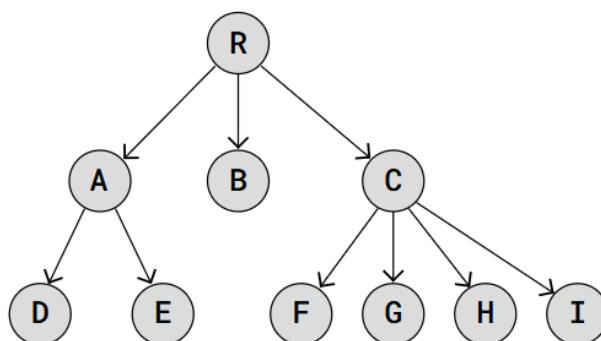
Graf merupakan sebuah struktur data yang terdiri dari simpul (vertices) dan sisi (edges) yang menghubungkan pasangan simpul. Graf digunakan untuk menggambarkan hubungan atau koneksi antara objek. Representasi informasi dari simpul dan sisi ini yang akan menentukan fungsionalitas dari struktur graf tersebut.



Gambar 1 Visualisasi Graph

Sumber: https://www.w3schools.com/dsa/dsa_theory_graphs.php

Tree adalah bentuk khusus dari graf yang tidak memiliki siklus dan terhubung. Selain itu, Tree berbentuk berarah dan *top-down*, yang berarti menjulur dari atas ke bawah. Maka dari itu, Tree umumnya digunakan untuk hubungan-hubungan *Parent* dan *Child*.

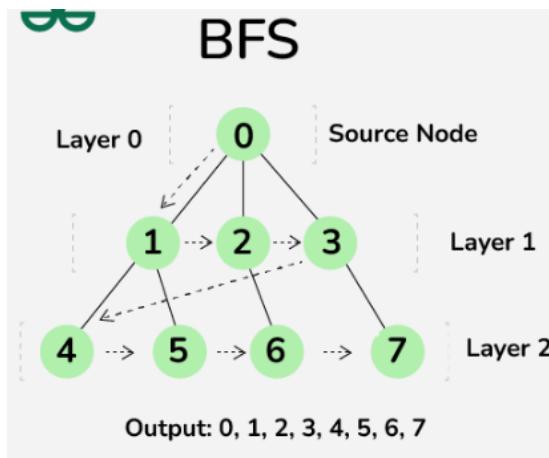


Gambar 2 Visualisasi Tree

Sumber: https://www.w3schools.com/dsa/dsa_theory_trees.php

2. Breadth-first Search (BFS)

Breadth-first Search (BFS) merupakan salah satu algoritma pencarian sebuah elemen pada suatu tree atau graf. Dimulai dari suatu titik, algoritma ini menjelajahi suatu graf dengan cara mengunjungi semua simpul pada tingkat yang sama. Simpul-simpul yang dideteksi lebih awal, akan dieksplorasi terlebih dahulu. Algoritma BFS memiliki sifat First-In-First-Out (FIFO), sehingga implementasi BFS ini menggunakan struktur data Queue.

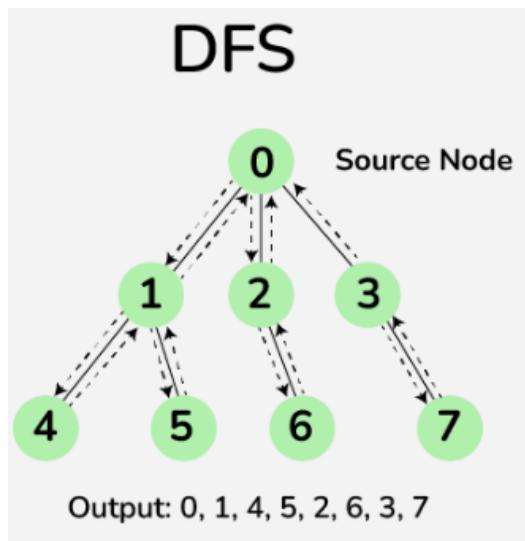


Gambar 3 Visualisasi BFS

Sumber: <https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/>

3. Depth-first Search (DFS)

Depth-first Search (DFS) merupakan salah satu algoritma pencarian yang mengeksplorasi simpul-simpul dalam graf dengan cara mendalam. DFS akan terlebih dahulu mengeksplor simpul anak terlebih dahulu sebelum melanjutkan ke simpul lainnya pada tingkat yang sama. Berbalik dengan BFS, algoritma DFS memiliki sifat Last-In-First-Out (LIFO), sehingga implementasinya dilakukan dalam bentuk struktur data Stack ataupun dilakukan secara rekursif.



Gambar 4 Visualisasi DFS

Sumber: <https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/>

4. Back End Web Development

Back End Web Development adalah pengembangan bagian aplikasi yang tidak terlihat terhadap pengguna tetapi menyangkut logika, pengolahan data, akses basis data, dan komunikasi antara server serta aplikasi. Pada umumnya, back end berinteraksi dengan front end (bagian yang berhubungan dengan visual atau muka depan yang tampak ke user) dengan menyediakan API (*application programming interface*) yang dapat dipanggil oleh front end dengan parameter tertentu melalui protokol seperti HTTP Request. Back End kemudian akan mengembalikan hasil yang dapat digunakan oleh front end.

5. Front End Web Development

Front End Web Development adalah bagian dari pengembangan perangkat lunak yang berkaitan dengan tampilan antarmuka pengguna (UI) dan interaksi pengguna dengan aplikasi. Front end bertanggung jawab untuk menyajikan data atau informasi yang dikirimkan oleh back end dengan cara yang ramah pengguna dan mudah dipahami. Biasanya, front end berinteraksi dengan back end melalui API (Application Programming Interface) yang menyediakan data yang dibutuhkan aplikasi.

Beberapa aspek penting dalam pengembangan front end adalah HTML yang merupakan dasar dari struktur halaman web. HTML (HyperText Markup Language) yang digunakan untuk menentukan elemen-elemen yang ada di halaman web, seperti teks, gambar, tabel, dan formulir. Kemudian, ada pula CSS yang digunakan untuk styling atau penataan elemen-elemen HTML agar tampak menarik dan mudah dibaca. CSS (Cascading Style Sheets) memungkinkan pengembang untuk mengatur tata letak, warna, ukuran font, margin, dan banyak aspek lainnya dari tampilan web. Terakhir adalah JavaScript, yaitu bahasa pemrograman yang digunakan untuk menambahkan

interaktivitas dan logika pada halaman web. JavaScript digunakan untuk merespons tindakan pengguna, seperti klik tombol, pengisian formulir, dan pemuatan data secara dinamis dari server. Berbagai pustaka juga digunakan dalam pengembangan front end seperti React.js dan Vite.

6. Docker

Docker adalah sebuah platform yang digunakan untuk membuat, menjalankan, dan mengelola suatu aplikasi dalam suatu container. Container dalam docker adalah suatu environment terisolasi yang berisikan aplikasi beserta semua dependensinya. Container ini mirip seperti sebuah Virtual Machine, tetapi lebih ringan karena hanya berbagi kernel dengan sistem operasi. Container ini juga memungkinkan aplikasi berjalan dengan konsisten di mana saja dan mengatasi masalah “Works on my machine”. Docker memiliki berbagai komponen utama, yaitu:

- **Dockerfile**: File konfigurasi yang berisi instruksi untuk membangun image.
- **Docker Image**: Blueprint dari container. Bersifat read-only.
- **Docker Container**: Hasil menjalankan image. Ini adalah instance yang berjalan.

7. Deployment

Deployment adalah proses mengirim dan menjalankan aplikasi dari lingkungan development ke lingkungan produksi. Deployment memungkinkan aplikasi yang sudah dikembangkan dapat digunakan oleh pengguna akhir. Secara umum, proses deployment terdiri dari:

- **Persiapan**: Memastikan semua yang diperlukan (kode, library, file konfigurasi, dll.) siap.
- **Pengujian**: Melakukan pengujian fungsional, integrasi, dan kinerja untuk memastikan aplikasi berfungsi dengan baik.
- **Konfigurasi**: Mengkonfigurasi lingkungan produksi agar sesuai dengan kebutuhan aplikasi.
- **Instalasi**: Memasang dan menjalankan aplikasi di lingkungan produksi.

8. Live Update

Live Update adalah fitur yang memungkinkan tampilan atau informasi yang ditampilkan pada antarmuka pengguna (UI) untuk diperbarui secara langsung sesuai dengan perubahan data yang terjadi di sisi server, tanpa perlu memuat ulang halaman atau permintaan ulang dari pengguna. Fitur ini sangat berguna dalam aplikasi-aplikasi yang memerlukan interaksi waktunya nyata, seperti dalam aplikasi visualisasi graf atau pohon, di mana struktur data atau informasi diubah atau diperbarui secara terus-menerus.

Pada aplikasi ini, proses pencarian elemen pada pohon (tree) atau graf (graph) dilakukan dengan algoritma pencarian seperti Depth-first Search (DFS) atau Breadth-first Search (BFS). Selama proses pencarian, visualisasi pohon akan dibangun secara bertahap sesuai dengan progres pencarian. Dengan fitur Live Update, setiap kali ada perubahan pada hasil pencarian (misalnya simpul baru ditemukan atau informasi baru diperbarui), UI akan langsung memperbarui tampilan pohon dengan elemen baru tersebut tanpa menunggu seluruh proses pencarian selesai.

Implementasi Live Update dapat dicapai dengan menggunakan teknologi seperti Server-Sent Events (SSE) atau WebSockets untuk mengirimkan pembaruan secara langsung dari server ke klien. Dengan SSE, server mengirimkan pembaruan dalam bentuk stream, yang memungkinkan klien untuk menerima data secara real-time. Klien (browser) dapat menggunakan JavaScript untuk mendengarkan event yang dikirimkan oleh server dan memperbarui UI sesuai dengan pembaruan tersebut.

BAB III

ANALISIS PEMECAHAN MASALAH

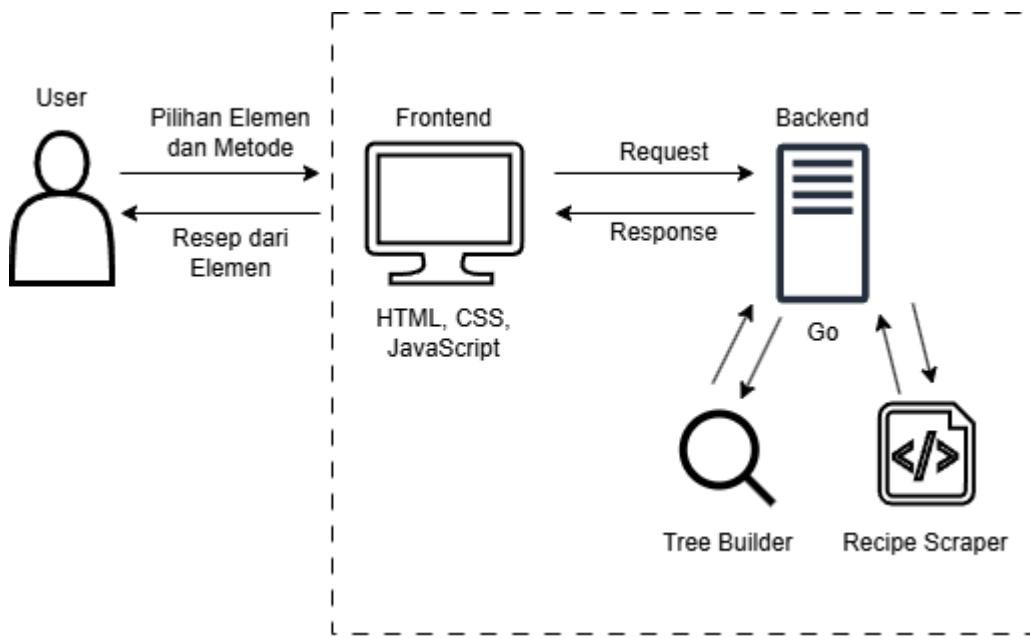
1. Pemecahan dan Pemetaan Masalah

Masalah yang dihadapi pada tugas ini adalah membangun sebuah pohon resep yang dapat digunakan oleh pengguna untuk membuat sebuah elemen yang diinginkannya. Pembangunan pohon resep harus memanfaatkan prinsip struktur data Tree dan juga algoritma BFS dan DFS yang telah dibahas di bab sebelumnya. Maka dari itu, elemen dan resep harus dapat direpresentasikan oleh struktur data Tree.

Pada pohon resep, sebuah simpul merepresentasikan sebuah elemen, dan sisi merepresentasikan hubungan ‘membuat’. Pada Little Alchemy 2, sebuah elemen terdiri dari nama, tier, dan beberapa resep yang dapat digunakan untuk membuat elemen tersebut. Agar keutuhan resep tidak dihilangkan pada Tree, resep-resep diwujudkan dalam bentuk *nested array*, yaitu array dalam array. Setiap array *child* yang dimiliki sebuah elemen merupakan pointer terhadap dua simpul (elemen), yang mewakili hubungan bahan yang dibutuhkan untuk membangun elemen tersebut. Selain itu, untuk menghindari terjadinya loop, kami diharuskan mengimplementasikan heuristik yang wajibkan sebuah elemen terdiri dari elemen dengan *tier* yang lebih rendah (*tier* 0: Fire, Water, Earth, Air). Heuristik ini efektif dalam pencegahan loop, tetapi terdapat permasalahan yang akan dibahas di bab berikutnya. Terakhir, kita harus mampu menentukan jumlah resep yang valid dari sebuah pohon resep. Sebuah resep lengkap dianggap valid ketika mampu membangun sebuah elemen dimulai dari elemen *tier* 0. Jumlah resep lengkap yang dapat dibuat pada sebuah elemen dari salah satu resepnya merupakan jumlah semua resep lengkap yang dapat digunakan untuk elemen pertama dikali dengan jumlah semua resep lengkap yang dapat digunakan untuk elemen kedua. Pada akhirnya, metode rekursif dibuat yang memeriksa setiap resep yang mungkin pada pohon dengan wajibkan semua elemen dalam traversal itu dapat dicapai dari elemen *tier* 0.

Setelah pemetaan itu selesai, algoritma BFS dan DFS dapat secara sepenuhnya diimplementasikan kepada pembangunan pohon resep. Algoritma BFS dimulai dengan melihat setiap resep yang mungkin digunakan untuk menciptakan elemen tersebut, yang kemudian setiap elemen pada setiap resep tersebut akan diperiksa juga semua elemen yang mungkin. Pendekatan ini menggunakan Queue sebagai penentu arah penjelajahan algoritma. Algoritma DFS dimulai dengan mendalami sebuah resep hingga semua elemen pada resep tersebut dapat dibuat dengan segala cara yang mungkin. Pendekatan ini menggunakan Stack sebagai penentu arah penjelajahan algoritma.

2. Arsitektur Aplikasi Web



Aplikasi web memiliki arsitektur standar dengan adanya sisi front end dan juga back end. Pengguna berinteraksi dengan front end dan memilih elemen serta metode yang ingin digunakan. Front end lalu akan mengirimkan request ke back end server berdasarkan parameter yang diberikan oleh pengguna dan mengembalikan data ke front end. Front end lalu menggunakan data tersebut untuk menampilkan hasil kepada pengguna.

Sisi back end secara garis besar terdiri atas *recipe finder* atau *tree builder* yang menggunakan algoritma BFS serta DFS untuk membangun pohon resep dari suatu elemen dan *recipe scraper* untuk melakukan scraping resep seluruh elemen dari situs wiki fandom dari Little Alchemy 2. Scraping dilakukan setiap kali aplikasi dijalankan. Hasil scraping resep tersebut digunakan oleh front end untuk menampilkan seluruh elemen yang bisa dicari resepnya kepada pengguna dan digunakan oleh back end pada proses pembangunan pohon.

3. Ilustrasi Kasus

- Website melakukan scraping untuk mendapatkan semua resep yang mungkin
- Hasil scraping diolah dengan menghilangkan resep yang tidak memenuhi persyaratan tier
- Pengguna meminta 1 pohon resep untuk elemen “Life” dengan proses BFS.
- Aplikasi menerima permintaan tersebut dan memeriksa semua resep yang mungkin untuk membentuk life.
- Belum terdapat pohon resep yang valid dari pohon tersebut, sehingga setiap elemen yang diperlukan oleh setiap resep life akan diperiksa juga semua resepnya.
- Pencarian akan berulang terus hingga akhirnya terbentuk sebuah tree yang valid

- g. Pengguna menerima pohon tersebut dan mampu membuat resep tersebut pada web Little Alchemy 2

BAB IV

IMPLEMENTASI DAN PENGUJIAN

1. Implementasi Back-end

Pemrograman aplikasi pembangunan pohon resep dari permainan Little Alchemy 2 ini meliputi dua sisi, yaitu Back-end dan Front-end. Back-end memanfaatkan bahasa Go sebagai bahasa pemrogramannya, dan mengurus proses *scraping* dan pembuatan pohon resep. Dalam implementasi, Back-end aplikasi ini berjalan dengan 3 package, dan 1 program utama.

1. Package recipe

o Struct Recipe

Atribut	Tipe	Deskripsi
Name	string	Nama dari elemen
Recipes	[][]string	Resep-resep untuk membentuk sebuah elemen. Setiap resep terdiri dari dua elemen
Tier	int	Tier dari elemen

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

Berperan sebagai elemen dari Little Alchemy 2. Terdapat penambahan `json:{attribute}` untuk pencocokan dalam konversi JSON menjadi Recipe.

o Struct RecipeTreeNode

Atribut	Tipe	Deskripsi
Name	string	Nama dari elemen
Children	[][*RecipeTreeNode]	Kumpulan dari kombinasi dua elemen yang dapat membentuk elemen tersebut

```

type RecipeTreeNode struct {
    Name      string
    Children [][][]*RecipeTreeNode
}

```

RecipeTreeNode berperan sebagai simpul elemen pada pohon resep. Children berupa *nested array* untuk menjaga integritas setiap resep pada setiap elemen.

- Method

Nama	Parameter	Hasil	Deskripsi
IsBaseElement	name string	bool	Mengecek apakah elemen tersebut merupakan elemen tier 0
	<pre> func IsBaseElement(name string) bool { recipe, exists := RecipeMap[name] if !exists { return false } if recipe.Tier == 0 { return true } return false } </pre>		
ReadJSON	filename string	map[string]Recipe	Membaca JSON dan mengembalikan map resep yang lengkap dan memenuhi persyaratan

```

func ReadJson(filename string) (map[string]Recipe, error) {
    file, err := os.Open(filename)
    if err != nil {
        return nil, fmt.Errorf("error opening file: %w", err)
    }
    defer file.Close()

    decoder := json.NewDecoder(file)
    var recipes []Recipe
    if err := decoder.Decode(&recipes); err != nil {
        return nil, fmt.Errorf("error decoding JSON: %w", err)
    }

    recipeMap := make(map[string]Recipe)

    for _, recipe := range recipes {
        recipeMap[recipe.Name] = recipe
    }

    for name, recipe := range recipeMap {
        validRecipes := [][]string{}

        for _, ingredients := range recipe.Recipes {
            valid := true

            for _, ingredient := range ingredients {
                ingredientRecipe, exists := recipeMap[ingredient]
                if exists && ingredientRecipe.Tier >= recipe.Tier {
                    valid = false
                    break
                }
            }

            if valid {
                validRecipes = append(validRecipes, ingredients)
            }
        }

        if len(validRecipes) > 0 {
            recipe.Recipes = validRecipes
            recipeMap[name] = recipe
        } else {
            delete(recipeMap, name)
        }
    }

    return recipeMap, nil
}

```

CalculateTotalCompleteRecipes	root *RecipeTreeNode	int	Menghitung semua resep lengkap yang mungkin
-------------------------------	-------------------------	-----	---

```

func CalculateTotalCompleteRecipes(root *RecipeTreeNode) int {
    if root == nil {
        return 0
    }

    if IsBaseElement(root.Name) {
        return 1
    }

    total := 0
    for _, group := range root.Children {
        if len(group) != 2 {
            continue // Skip if not a valid recipe group
        }
        leftCount := CalculateTotalCompleteRecipes(group[0])
        rightCount := CalculateTotalCompleteRecipes(group[1])
        if leftCount > 0 && rightCount > 0 {
            total += leftCount * rightCount
        }
    }
    return total
}

```

BuildRecipeTreeDFS	root *RecipeTreeNode, recipeMap map[string]Recipe maxRecipes int, stopChan chan bool, wg *sync.WaitGroup, mu *sync.Mutex, nodesVisited *int, treeChan chan *RecipeTreeNode	-	Melakukan pencarian DFS. Implementasi dilakukan dengan menggunakan stack. sync.WaitGroup digunakan untuk mengaplikasikan konkurensi, stopChan digunakan untuk menghentikan semua thread, dan treeChan berperan untuk mengirimkan pohon. Pengecekan jumlah pohon valid berlaku tiap ketemu base element
--------------------	--	---	---

```

func BuildRecipeTreeDFS(
    root *RecipeTreeNode,
    recipeMap map[string]Recipe,
    maxRecipes int,
    stopChan chan bool, // Channel to signal stopping
    wg *sync.WaitGroup, // WaitGroup for goroutines
    mu *sync.Mutex, // Mutex to safely modify shared variables
    nodesVisited *int,
    treeChan chan *RecipeTreeNode,
) {
    defer wg.Done()

    stack := []*RecipeTreeNode{root}

    for len(stack) > 0 {
        select {
        case <-stopChan:
            return // Stop further search if signal is received
        default:
        }
        node := stack[len(stack)-1]
        stack = stack[:len(stack)-1]

        recipe, exists := recipeMap[node.Name]
        if !exists {
            continue
        }

        var children [][]*RecipeTreeNode
        var childWg sync.WaitGroup

        for _, r := range recipe.Recipes {
            childWg.Add(1)

            go func(r []string) {
                defer childWg.Done()

                var childNodes []*RecipeTreeNode
                for _, name := range r {
                    childNode := &RecipeTreeNode{Name: name}
                    childNodes = append(childNodes, childNode)

                    mu.Lock()
                    stack = append(stack, childNode)
                    mu.Unlock()
                }

                if len(r) == 2 && IsBaseElement(r[0]) && IsBaseElement(r[1]) {
                    mu.Lock()
                    treeChan <- root
                    time.Sleep(500 * time.Millisecond)
                    if CalculateTotalCompleteRecipes(root) >= maxRecipes {
                        stopChan <- true
                        return
                    }
                    mu.Unlock()
                }

                mu.Lock()
                children = append(children, childNodes)
                mu.Unlock()
            }(r)
        }

        childWg.Wait()

        mu.Lock()
        SetChildren(node, children)
        mu.Unlock()

        select {
        case <-stopChan:
            return
        default:
        }
        mu.Lock()
        *nodesVisited++
        mu.Unlock()
    }
}

```

BuildRecipeTreeBF S	<pre> root *RecipeTreeNode, recipeMap map[string]Recipe maxRecipes int, stopChan chan bool, wg *sync.WaitGroup, mu *sync.Mutex, nodesVisited *int, treeChan chan *RecipeTreeNode </pre>	-	<p>Melakukan pencarian DFS. Implementasi dilakukan dengan menggunakan queue.</p> <p>sync.WaitGroup digunakan untuk mengaplikasikan konkurensi, stopChan digunakan untuk menghentikan semua thread, dan treeChan berperan untuk mengirimkan pohon. Pengecekan jumlah pohon valid berlaku tiap ketemu base element</p>
------------------------	---	---	--

```

func BuildRecipeTreeBFS(
    root *RecipeTreeNode,
    recipeMap map[string]Recipe,
    maxRecipes int,
    stopChan chan bool,
    wg *sync.WaitGroup,
    mu *sync.Mutex,
    nodesVisited *int,
    treeChan chan *RecipeTreeNode,
) {
    defer wg.Done()

    queue := []*RecipeTreeNode{root}
    for len(queue) > 0 {
        select {
        case <-stopChan:
            return
        default:
        }
        node := queue[0]
        queue = queue[1:]

        recipe, exists := recipeMap[node.Name]
        if !exists {
            continue
        }

        var children [][]*RecipeTreeNode
        var childWg sync.WaitGroup

        for _, r := range recipe.Recipes {
            childWg.Add(1)

            go func(r []string) {
                defer childWg.Done()

                var childNodes []*RecipeTreeNode
                for _, name := range r {
                    childNode := &RecipeTreeNode{Name: name}
                    childNodes = append(childNodes, childNode)

                    mu.Lock()
                    queue = append(queue, childNode)
                    mu.Unlock()
                }

                if len(r) == 2 && IsBaseElement(r[0]) && IsBaseElement(r[1]) {
                    mu.Lock()
                    if CalculateTotalCompleteRecipes(root) >= maxRecipes {
                        stopChan <- true // Send stop signal
                        return
                    }
                    mu.Unlock()
                }

                mu.Lock()
                children = append(children, childNodes)
                mu.Unlock()
            }(r)
        }

        childWg.Wait()

        mu.Lock()
        SetChildren(node, children)
        mu.Unlock()

        select {
        case <-stopChan:
            return
        default:
        }
        mu.Lock()
        *nodesVisited++
        if *nodesVisited%6 == 0 {
            treeChan <- root
            time.Sleep(500 * time.Millisecond)
        }
        mu.Unlock()
    }
}

```

PruneTree	node *RecipeTreeNode	-	Memotong bagian dari tree-nya yang tidak membentuk recipe yang valid
<pre>func PruneTree(node *RecipeTreeNode) { var newChildren [][]*RecipeTreeNode for _, recipe := range node.Children { bothBase := true for _, child := range recipe { if CalculateTotalCompleteRecipes(child) == 0 { bothBase = false fmt.Println("Pruning", child.Name) break } } if bothBase { newChildren = append(newChildren, recipe) } } SetChildren(node, newChildren) for _, recipe := range newChildren { for _, child := range recipe { PruneTree(child) } } }</pre>			
stopSearch	stopChan chan bool, wg *sync.WaitGroup	-	Sigap menerima signal stop, untuk menghentikan proses.
<pre>func StopSearch(stopChan chan bool, wg *sync.WaitGroup) { defer wg.Done() <-stopChan fmt.Println("Stopping the search!") }</pre>			
setChildren	node *RecipeTreeNode children [][]*RecipeTreeNode	-	Memasang children pada node.
<pre>func SetChildren(node *RecipeTreeNode, children [][]*RecipeTreeNode) { node.Children = children }</pre>			

2. Package scraper

- **struct ElementWithRecipes**

Nama	Tipe	Deskripsi
Element	string	Nama dari elemen
Tier	int	Tier dari elemen
ImageURL	string	Image URL yang diambil dari wiki umtuk elemen tersebut
Recipes	[][]string	Resep-resep untuk membuat elemen tersebut

```
type ElementWithRecipes struct {
    Element string `json:"element"`
    Tier    int    `json:"tier" // <-
    ImageURL string `json:"image_url"`
    Recipes [][]string `json:"recipes"`
}
```

- **Method**

Nama	Parameter	Output	Deskripsi
ExcludedElements	-	map[string]bool	Memberikan map sebuah string semua elemen yang mengecualikan Time, Ruins dan juga dari extension Myths and Monsters

```

func ExcludedElements() map[string]bool {
    excluded := make(map[string]bool)
    c := colly.NewCollector()
    c.OnHTML("li.category-page__member", func(e *colly.HTMLElement) {
        name := strings.TrimSpace(e.ChildText("a.category-page__member-link"))
        if name != "" {
            excluded[name] = true
        }
    })
    if err := c.Visit("https://little-alchemy.fandom.com/wiki/Category:Myths_and_Monsters"); err != nil {
        log.Fatal(err)
    }
    excluded["Time"] = true
    excluded["Ruins"] = true
    return excluded
}

```

FindRecipes	-	-	Membuat sebuah JSON yang mewakili semua elemen berdasarkan struct Element WithRecipes
-------------	---	---	---

```

func FindRecipes() {
    excluded := ExcludedElements()
    baseEls := map[string]bool{"Fire": true, "Earth": true, "Water": true, "Air": true}

    var elements []ElementWithRecipes

    c := colly.NewCollector()

    c.OnHTML("h3", func(e *colly.HTMLElement) {
        headline := e.ChildText("span.mw-headline")

        if headline == "Starting elements" {
            tableSel := e.DOM.NextUntil("h3").FilterFunction(func(_ int, s *goquery.Selection) bool {
                return s.Is("table.list-table.col-list.icon-hover")
            }).First()

            tableSel.Find("tbody tr").Each(func(_ int, row *goquery.Selection) {
                name := strings.TrimSpace(row.Find("td:nth-of-type(1) a").Text())
                if name == "" || excluded[name] || !baseEls[name] {
                    return
                }
                imgURL, _ := row.Find("td:nth-of-type(1) a").Attr("href")
                if imgURL == "" {
                    imgURL = "No image"
                }

                elements = append(elements, ElementWithRecipes{
                    Element: name,
                    Tier: 0,
                    ImageURL: imgURL,
                    Recipes: [][]string{},
                })
            })
            return
        }

        if strings.HasPrefix(headline, "Tier ") {
            return
        }
        parts := strings.Fields(headline)
        tier, err := strconv.Atoi(parts[1])
        if err != nil {
            return
        }

        tableSel := e.DOM.NextUntil("h3").FilterFunction(func(_ int, s *goquery.Selection) bool {
            return s.Is("table.list-table.col-list.icon-hover")
        }).First()

        tableSel.Find("tbody tr").Each(func(_ int, row *goquery.Selection) {
            name := strings.TrimSpace(row.Find("td:nth-of-type(1) a").Text())
            if name == "" || excluded[name] || baseEls[name] {
                return
            }
            imgURL, _ := row.Find("td:nth-of-type(1) a").Attr("href")
            if imgURL == "" {
                imgURL = "No image"
            }

            var recipes [][]string
            row.Find("td:nth-type(2) li").Each(func(_ int, li *goquery.Selection) {
                var comps []string
                li.Find("a").Each(func(_ int, a *goquery.Selection) {
                    t := strings.TrimSpace(a.Text())
                    if t != "" && !excluded[t] {
                        comps = append(comps, t)
                    }
                })
                if len(comps) == 2 {
                    recipes = append(recipes, comps)
                }
            })
            elements = append(elements, ElementWithRecipes{
                Element: name,
                Tier: tier,
                ImageURL: imgURL,
                Recipes: recipes,
            })
        })
    })

    if err := c.Visit("https://little-alchemy.fandom.com/wikit/Elements_(Little_Alchemy_2)"); err != nil {
        log.Fatal(err)
    }

    out, err := json.MarshalIndent(elements, "", " ")
    if err != nil {
        log.Fatal(err)
    }
    if err := os.WriteFile("recipes.json", out, 0644); err != nil {
        log.Fatal("Failed to write recipes.json:", err)
    }
}

```

3. Package server

- o NodeDTO

Nama	Tipe	Deskripsi
Name	string	Nama dari elemen pada pohon
Recipes	[]RecipeDTO	Kumpulan resep dari elemen

```
type NodeDTO struct {
    Name     string      `json:"name"`
    Recipes []RecipeDTO `json:"recipes"`
}
```

- o RecipeDTO

Nama	Tipe	Deskripsi
Inputs	[]NodeDTO	Elemen yang dimiliki sebuah resep

```
type RecipeDTO struct {
    Inputs []NodeDTO `json:"inputs"`
}
```

- o TreeResponse

Nama	Tipe	Deskripsi
Tree	NodeDTO	Pohon yang dikirim lewat JSON
TimeTaken	int64	Waktu yang dibutuhkan untuk memproses pohon
NodesVisited	int	Jumlah node yang dilewati
RecipesFound	int	Jumlah resep yang ditemukan
MethodUsed	string	BFS/DFS

```

type TreeResponse struct {
    Tree           NodeDTO `json:"tree"`
    TimeTaken     int64   `json:"timeTaken" //`
    NodesVisited  int     `json:"nodesVisited"`
    RecipesFound int     `json:"recipesFound"`
    MethodUsed    string  `json:"methodUsed"`
}

```

Digunakan sebagai basis penunjukkan pohon

- **Method**

Nama	Parameter	Output	Deskripsi
buildDTO	node *recipe.RecipeTreeNode	NodeDTO	Mengubah sebuah simpul RecipeTreeNode menjadi NodeDTO
<pre> func buildDTO(node *recipe.RecipeTreeNode) NodeDTO { dto := NodeDTO{ Name: node.Name, Recipes: make([]RecipeDTO, 0), } for _, group := range node.Children { if len(group) == 0 { continue } inputs := make([]NodeDTO, 0, len(group)) for _, child := range group { inputs = append(inputs, buildDTO(child)) } dto.Recipes = append(dto.Recipes, RecipeDTO{Inputs: inputs}) } return dto } </pre>			
writeJSON	w http.ResponseWriterWriter, v interface{}	-	Mengubah DTO menjadi sebuah JSON yang dikirim ke FrontEnd untuk ditampilkan

```

func writeJSON(w http.ResponseWriter, v interface{}) {
    w.Header().Set("Access-Control-Allow-Origin", "*")
    w.Header().Set("Content-Type", "application/json")

    payload, err := json.Marshal(v)
    if err != nil {
        log.Printf("[writeJSON] X marshal error: %v\n", err)
        http.Error(w, "json encode error", http.StatusInternalServerError)
        return
    }

    log.Printf("[writeJSON] → sending %d bytes: %s\n", len(payload), payload)

    if _, err := w.Write(payload); err != nil {
        log.Printf("[writeJSON] X write error: %v\n", err)
    }
}

```

parseCount	r *http.Request	int	Mengambil jumlah dari ?count=N dari request front-end. Default ke 1
------------	-----------------	-----	--

```

func parseCount(r *http.Request) int {
    if s := r.URL.Query().Get("count"); s != "" {
        if c, err := strconv.Atoi(s); err == nil {
            return c
        }
    }
    return 1
}

```

parseStream	r *http.Request	bool	Membaca ?stream = 1 untuk mengaktifkan SSE (untuk live update)
-------------	-----------------	------	--

```

func parseStream(r *http.Request) bool {
    return r.URL.Query().Get("stream") == "1"
}

```

recipesHandler	w http.ResponseWriter iter, r *http.Request	-	Menerima request untuk membaca RecipesJSON dan
----------------	--	---	--

			membalaskan status message
<pre>func recipesHandler(w http.ResponseWriter, r *http.Request) { w.Header().Set("Access-Control-Allow-Origin", "*") w.Header().Set("Content-Type", "application/json") data, err := os.ReadFile("recipes.json") if err != nil { http.Error(w, "cannot read recipes.json", http.StatusInternalServerError) return } log.Printf("→ [recipesHandler] loaded %d bytes\n", len(data)) log.Printf("→ [recipesHandler] preview:\n%s\n", truncate(data, 200)) w.Write(data) }</pre>			
dfsHandler	w http.ResponseWriter, r *http.Request	-	Menerima request untuk memulai pembangunan resep pohon dengan metode DFS, disertai streaming jika diinginkan.

```

func dfsHandler(w http.ResponseWriter, r *http.Request) {
    target := r.URL.Query().Get("target")
    if target == "" {
        http.Error(w, "missing target", http.StatusBadRequest)
        return
    }
    maxRecipes := parseCount(r)
    streaming := parseStream(r)

    log.Printf("→ [dfsHandler] target=%q maxRecipes=%d stream=%v\n", target, maxRecipes, streaming)

    recipe.VisitedMap = make(map[string]*recipe.RecipeTreeNode)

    root := &recipe.RecipeTreeNode{Name: target}
    stopChan := make(chan bool)
    wg := &sync.WaitGroup{}
    mu := &sync.Mutex{}
    treeChan := make(chan *recipe.RecipeTreeNode, 10)

    start := time.Now()
    var nodesVisited int

    wg.Add(1)
    go recipe.BuildRecipeTreeDFS(root, recipe.RecipeMap, maxRecipes, stopChan, wg, mu, &nodesVisited,
        treeChan)

    if streaming {
        w.Header().Set("Access-Control-Allow-Origin", "*")
        w.Header().Set("Content-Type", "text/event-stream")
        w.Header().Set("Cache-Control", "no-cache")
        flusher, ok := w.(http.Flusher)
        if !ok {
            http.Error(w, "streaming unsupported", http.StatusInternalServerError)
            return
        }

        go func() {
            wg.Wait()
            close(treeChan)
        }()
        for node := range treeChan {
            dto := buildDTO(node)
            elapsed := time.Since(start).Milliseconds()
            found := recipe.CalculateTotalCompleteRecipes(root)

            sse := TreeResponse{
                Tree:      dto,
                TimeTaken: elapsed,
                NodesVisited: nodesVisited,
                RecipesFound: found,
                MethodUsed:   "DFS",
            }
            data, _ := json.Marshal(sse)
            fmt.Fprintf(w, "data: %s\n\n", data)
            flusher.Flush()
        }
        return
    }

    go func() {
        wg.Wait()
        close(treeChan)
    }()
    wg.Wait()
    elapsed := time.Since(start).Milliseconds()
    recipesFound := recipe.CalculateTotalCompleteRecipes(root)
    recipe.PruneTree(root)
    dto := buildDTO(root)

    resp := TreeResponse{
        Tree:      dto,
        TimeTaken: elapsed,
        NodesVisited: nodesVisited,
        RecipesFound: recipesFound,
        MethodUsed:   "DFS",
    }
    writeJSON(w, resp)
}

```

bfsHandler	w http.ResponseWriter riter, r *http.Request	-	Menerima request untuk memulai pembangunan resep pohon
------------	---	---	--

		dengan metode BFS, disertai streaming jika diinginkan.
		<pre> func bfsHandler(w http.ResponseWriter, r *http.Request) { target := r.URL.Query().Get("target") if target == "" { http.Error(w, "missing target", http.StatusBadRequest) return } maxRecipes := parseCount(r) streaming := parseStream(r) log.Printf("→ [bfsHandler] target=%q maxRecipes=%d stream=%v\n", target, maxRecipes, streaming) recipe.VisitedMap = make(map[string]*recipe.RecipeTreeNode) root := &recipe.RecipeTreeNode{Name: target} stopChan := make(chan bool) wg := &sync.WaitGroup{} mu := &sync.Mutex{} treeChan := make(chan *recipe.RecipeTreeNode, 10) start := time.Now() var nodesVisited int wg.Add(1) go recipe.BuildRecipeTreeDFS(root, recipe.RecipeMap, maxRecipes, stopChan, wg, mu, &nodesVisited, treeChan) if streaming { w.Header().Set("Access-Control-Allow-Origin", "*") w.Header().Set("Content-Type", "text/event-stream") w.Header().Set("Cache-Control", "no-cache") flusher, ok := w.(http.Flusher) if !ok { http.Error(w, "streaming unsupported", http.StatusInternalServerError) return } go func() { wg.Wait() close(treeChan) }() for node := range treeChan { dto := buildDTO(node) elapsed := time.Since(start).Milliseconds() found := recipe.CalculateTotalCompleteRecipes(root) sse := TreeResponse{ Tree: dto, TimeTaken: elapsed, NodesVisited: nodesVisited, RecipesFound: found, MethodUsed: "BFS", } data, _ := json.Marshal(sse) fmt.Fprintf(w, "data: %s\n\n", data) flusher.Flush() } return } go func() { wg.Wait() close(treeChan) }() } wg.Wait() elapsed := time.Since(start).Milliseconds() recipesFound := recipe.CalculateTotalCompleteRecipes(root) recipe.PruneTree(root) dto := buildDTO(root) resp := TreeResponse{ Tree: dto, TimeTaken: elapsed, NodesVisited: nodesVisited, RecipesFound: recipesFound, MethodUsed: "BFS", } writeJSON(w, resp) } </pre>

Start	-	-	Menginisiasi Back-end untuk mendengar request
<pre>func Start() { var err error recipe.RecipeMap, err = recipe.ReadJson("recipes.json") if err != nil { log.Fatalf("Failed to load recipes.json: %v", err) } http.HandleFunc("/api/recipes", recipesHandler) http.HandleFunc("/api/dfs", dfsHandler) http.HandleFunc("/api/bfs", bfsHandler) http.HandleFunc("/api/bidirectional", func(w http.ResponseWriter, r *http.Request) { http.Error(w, "Bidirectional not implemented", http.StatusNotImplemented) }) log.Println("Server listening on :8080") log.Fatal(http.ListenAndServe(":8080", nil)) }</pre>			

4. main.go

main	-	-	Menyiapkan server untuk menerima request dari FE
<pre>func main() { log.Println("Scraping recipes...") scraper.FindRecipes() log.Println("Finished scraping; wrote recipes.json") var err error recipe.RecipeMap, err = recipe.ReadJson("recipes.json") if err != nil { log.Fatalf("Failed to load recipes.json: %v", err) } log.Printf("Loaded %d recipes.\n", len(recipe.RecipeMap)) log.Println("Starting HTTP server on :8080") server.Start() }</pre>			

2. Implementasi Front-end

Aplikasi web dibangun menggunakan JavaScript, HTML, dan CSS. Library yang digunakan dalam proyek ini adalah React disertai dengan server pengembangan lokal Vite untuk mempercepat proses pengembangan.

N o	Komponen	States	Deskripsi
1	App	- [treeData, setTreeData]	Komponen utama yang

.		<ul style="list-style-type: none"> - [metrics, setMetrics] - [loading, setLoading] - [useLiveUpdate, setUseLiveUpdate] 	mengandung logika aplikasi secara keseluruhan.
---	--	---	--

```

● ● ●

1 import React, { useState, useRef } from "react";
2 import SearchWindow from "./SearchWindow.jsx";
3 import RecipeTree from "./Tree.jsx";
4
5 function App() {
6   const [treeData, setTreeData] = useState(null);
7   const [metrics, setMetrics] = useState({
8     timeTaken: 0,
9     nodesVisited: 0,
10    recipesFound: 0,
11    methodUsed: "",
12  });
13  const sourceRef = useRef(null);
14
15  const [loading, setLoading] = useState(false);
16  const [useLiveUpdate, setUseLiveUpdate] = useState(false);
17
18  const handleSearch = ({
19    selectedItem,
20    algorithm,
21    numRecipes,
22    liveUpdate,
23  }) => {
24    setUseLiveUpdate(liveUpdate);
25    setLoading(true);
26
27    if (sourceRef.current) {
28      sourceRef.current.close();
29      sourceRef.current = null;
30    }
31
32    const path =
33      algorithm === "dfs"
34        ? "dfs"
35        : algorithm === "bfs"
36        ? "bfs"
37        : "bidirectional";
38
39    const params = new URLSearchParams({
40      target: selectedItem.name,
41      count: numRecipes,
42      stream: liveUpdate ? "1" : "0",
43    });
44
45    const url = `${import.meta.env.VITE_API_BASE_URL}/api/${path}?${params}`;
46
47    if (liveUpdate) {
48      const es = new EventSource(url);
49      sourceRef.current = es;
50
51      es.onmessage = (e) => {
52        const { tree, timeTaken, nodesVisited, recipesFound, methodUsed } =
53          JSON.parse(e.data);
54        setTreeData(tree);
55        setMetrics({ timeTaken, nodesVisited, recipesFound, methodUsed });
56      };

```

```

57
58     es.onerror = (err) => {
59         console.error("SSE error:", err);
60         es.close();
61     };
62 } else {
63     fetch(url, { headers: { Accept: "application/json" } })
64     .then((res) => {
65         if (!res.ok) throw new Error(res.statusText);
66         return res.json();
67     })
68     .then(({ tree, timeTaken, nodesVisited, recipesFound, methodUsed }) => {
69         setTreeData(tree);
70         setMetrics({ timeTaken, nodesVisited, recipesFound, methodUsed });
71     })
72     .catch((err) => {
73         console.error("Fetch tree error:", err);
74         alert("Gagal mengambil data resep: " + err.message);
75     })
76     .finally(() => {
77         setLoading(false);
78     });
79 }
80 );
81
82 const handleBack = () => {
83     if (sourceRef.current) {
84         sourceRef.current.close();
85         sourceRef.current = null;
86     }
87     setTreeData(null);
88 };
89
90 return (
91     <div className="App">
92         {!treeData ? (
93             <SearchWindow onSearch={handleSearch} />
94         ) : (
95             <RecipeTree
96                 data={treeData}
97                 {...metrics}
98                 onBack={handleBack}
99             />
100        )}
101        {loading && !useLiveUpdate && (
102            <div className="loading-overlay">
103                <p>Loading recipes...</p>
104            </div>
105        )}
106    );
107 }
108
109
110 export default App;
111

```

2	SearchWindow	<ul style="list-style-type: none"> - [items, setItems] - [selectedItem, setSelectedItem] - [algorithm, setAlgorithm] - [numRecipes, 	Modal jendela di mana pengguna bisa memilih elemen yang ingin dibangun resepnya serta menentukan berbagai parameter pencarian
---	--------------	---	---

		<ul style="list-style-type: none"> - [setNumRecipes] - [searchTerm, setSearchTerm] - [liveUpdate, setLiveUpdate] 	seperati metode dan jumlah resep.
--	--	---	-----------------------------------



```

1 // src/SearchWindow.jsx
2 import React, { useState, useEffect } from "react";
3 import "./SearchWindow.css";
4
5 export default function SearchWindow({ onSearch }) {
6   const [items, setItems] = useState(null);
7   const [selectedItem, setSelectedItem] = useState(null);
8   const [algorithm, setAlgorithm] = useState("dfs");
9   const [numRecipes, setNumRecipes] = useState(1);
10  const [searchTerm, setSearchTerm] = useState("");
11  const [liveUpdate, setLiveUpdate] = useState(false);
12
13  useEffect(() => {
14    // ambil daftar elemen + image_url dari backend
15    console.log("VITE_API_BASE_URL:", import.meta.env.VITE_API_BASE_URL);
16    fetch(`${import.meta.env.VITE_API_BASE_URL}/api/recipes`)
17      .then((res) => {
18        if (!res.ok) throw new Error(res.statusText);
19        return res.json();
20      })
21      .then((data) => {
22        // data: [{ element, image_url, recipes }, ...]
23        const mapped = data.map((el, idx) => {
24          // build local path: spaces > underscores
25          const fileName = el.element.replace(/\s+/g, "_") + ".svg";
26          return {
27            id: idx,
28            name: el.element,
29            icon: `/images/${fileName}`,
30          };
31        });
32        setItems(mapped);
33        setSelectedItem(mapped[0]);
34      })
35      .catch((err) => {
36        console.error("Load recipes.json failed:", err);
37        alert("Gagal load daftar elemen dari server");
38      });
39  }, []);
40
41  if (!items)
42    return (
43      <div className="loading-overlay">
44        <p>Loading elements...</p>
45      </div>
46    );
47
48  const filtered = items.filter((it) =>
49    it.name.toLowerCase().includes(searchTerm.toLowerCase())
50  );

```

```

51
52     const handleSearch = () => {
53         onSearch({ selectedItem, algorithm, numRecipes: Number(numRecipes), liveUpdate });
54     };
55
56     return (
57         <div className="overlay">
58             <div className="modal">
59                 <div className="grid-wrapper">
60                     <div className="grid">
61                         {filtered.map((item) => (
62                             <div
63                                 key={item.id}
64                                 className={`grid-item${item.id === selectedItem.id ? " selected" : ""}`}
65                                 onClick={() => setSelectedItem(item)}
66                             >
67                                 <img src={item.icon} alt={item.name} />
68                             </div>
69                         )));
70                         {filtered.length === 0 && (
71                             <p className="no-results">No elements found</p>
72                         )}
73                     </div>
74                 </div>
75             </div>
76             <div className="controls">
77                 <div className="search-bar">
78                     <input
79                         type="text"
80                         placeholder="Search elements..."
81                         value={searchTerm}
82                         onChange={(e) => setSearchTerm(e.target.value)}
83                     />
84                 </div>
85
86                 <br/>
87                 <div className="selected-preview">
88                     <span className="label"><u>Selected</u></span>
89                     <div className="preview-box">
90                         <img src={selectedItem.icon} alt={selectedItem.name} />
91                     </div>
92                     <div className="item-name">{selectedItem.name}</div>
93                 </div>
94                 <div className="radio-group">
95                     <label>
96                         <input
97                             type="radio"
98                             name="algorithm"
99                             value="dfs"
100                         />

```

```

180           value="dfs"
181           checked={algorithm === "dfs"}
182           onChange={() => setAlgorithm("dfs")}
183         />
184         DFS
185       </label>
186       <label>
187         <input
188           type="radio"
189           name="algorithm"
190           value="bfs"
191           checked={algorithm === "bfs"}
192           onChange={() => setAlgorithm("bfs")}
193         />
194         BFS
195       </label>
196     <div className="live-update-toggle">
197       <label>
198         <input
199           type="checkbox"
200           checked={liveUpdate}
201           onChange={(e) => setLiveUpdate(e.target.checked)}
202           />{" "}
203         Live Update
204       </label>
205     </div>
206   </div>
207   <div className="recipes-input">
208     <label>Number of Recipes</label>
209     <input
210       type="number"
211       min="1"
212       value={numRecipes}
213       onChange={(e) => setNumRecipes(e.target.value)}
214     />
215   </div>
216   <button className="search-button" onClick={handleSearch}>
217     SEARCH
218   </button>
219 </div>
220 </div>
221 </div>
222 );
223 }
224

```

3 .	Tree	<ul style="list-style-type: none"> - [translate, setTranslate] - [ready, setReady] 	Visualisasi pohon resep hasil pencarian yang memanfaatkan library React D3.
-----	------	--	---

```
 1 import React, { useMemo, useRef, useState, useLayoutEffect } from "react";
 2 import Tree from "react-d3-tree";
 3 import "./Tree.css";
 4
 5 function convertDTO(node) {
 6   const fileName = node.name.replace(/\ /g, "_") + ".svg";
 7   const treeNode = {
 8     name: node.name,
 9     image: node.name ? `/images/${fileName}` : null,
10   };
11
12   if (Array.isArray(node.recipes)) {
13     const childrenGroups = node.recipes.reduce((groups, recipe) => {
14       if (Array.isArray(recipe.inputs)) {
15         const inputs = recipe.inputs.map(convertDTO);
16         groups.push({ name: "", image: null, children: inputs });
17       }
18       return groups;
19     }, []);
20
21     if (childrenGroups.length > 0) {
22       treeNode.children = childrenGroups;
23     }
24   }
25
26   return treeNode;
27 }
28
29 export default function RecipeTree({
30   data,
31   timeTaken,
32   nodesVisited,
33   recipesFound,
34   methodUsed,
35   onBack,
36 }) {
37   const treeData = useMemo(() => [convertDTO(data)], [data]);
38   const containerRef = useRef(null);
39   const [translate, setTranslate] = useState({ x: 0, y: 0 });
40   const [ready, setReady] = useState(false);
41
42   useLayoutEffect(() => {
43     if (!containerRef.current) return;
44     const { width } = containerRef.current.getBoundingClientRect();
45     setTranslate({ x: width / 2, y: 60 });
46     setReady(true);
47   }, []);
}
```

```

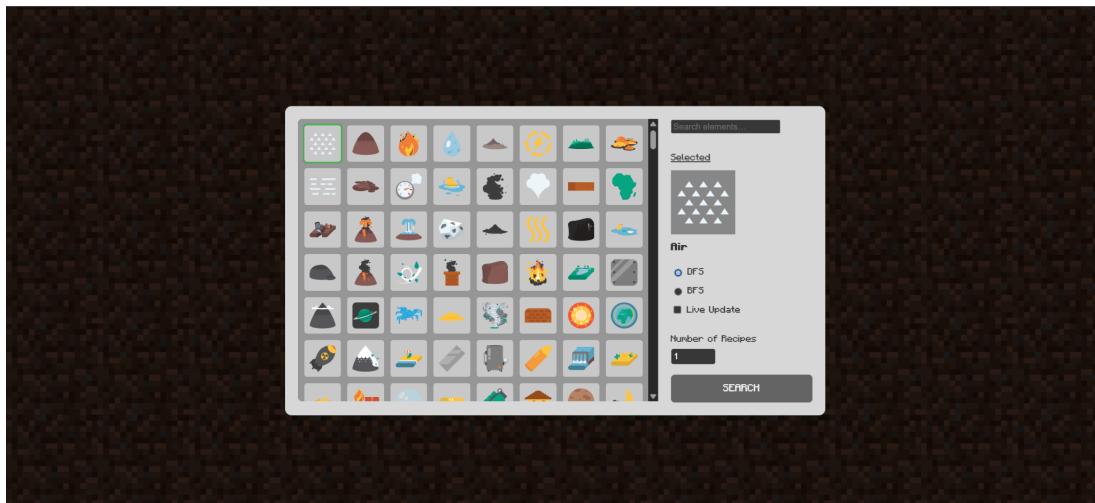
48
49  const renderNode = ({ nodeDatum, transform }) => {
50    const isDummy = nodeDatum.name === "";
51    const cls = isDummy
52      ? "dummy-node"
53      : nodeDatum.children
54      ? "branch-node"
55      : "leaf-node";
56
57    return (
58      <g className={cls} transform={transform}>
59        {!isDummy && nodeDatum.image && (
60          <image className="node-icon" href={nodeDatum.image} />
61        )}
62        {!isDummy && (
63          <rect
64            className="node-rect"
65            x={-26}
66            y={-26}
67            width={52}
68            height={52}
69          />
70        )}
71        {!isDummy && <text className="node-label">{nodeDatum.name}</text>}
72      </g>
73    );
74  };
75
76  return (
77    <div className="tree-container" ref={containerRef}>
78      {ready && (
79        <div
80          style={{
81            width: "100%",
82            height: "100%",
83            transform: "rotate(180deg)",
84          }}
85        >
86          <Tree
87            data={treeData}
88            translate={translate}
89            orientation="vertical"
90            pathFunc="straight"
91            renderCustomNodeElement={renderNode}
92            collapsible={false}
93            enableLegacyTransitions={false}
94            zoomable={true}
95            zoom={1}
96            separation={{ siblings: 1.5, nonSiblings: 2 }}
97            rootOrientation="bottom"
98          />

```

```

99         </div>
100    )}
101
102    <div className="info-box">
103      <h4>Search Info</h4>
104      <p>
105        <strong>Time taken:</strong> {(timeTaken / 1000).toFixed(3)} ms
106      </p>
107      <p>
108        <strong>Nodes visited:</strong> {nodesVisited}
109      </p>
110      <p>
111        <strong>Recipes found:</strong> {recipesFound}
112      </p>
113      <p>
114        <strong>Method used:</strong> {methodUsed}
115      </p>
116      <button className="back-button" onClick={onBack}>
117        ← New Search
118      </button>
119    </div>
120  </div>
121  );
122}
123

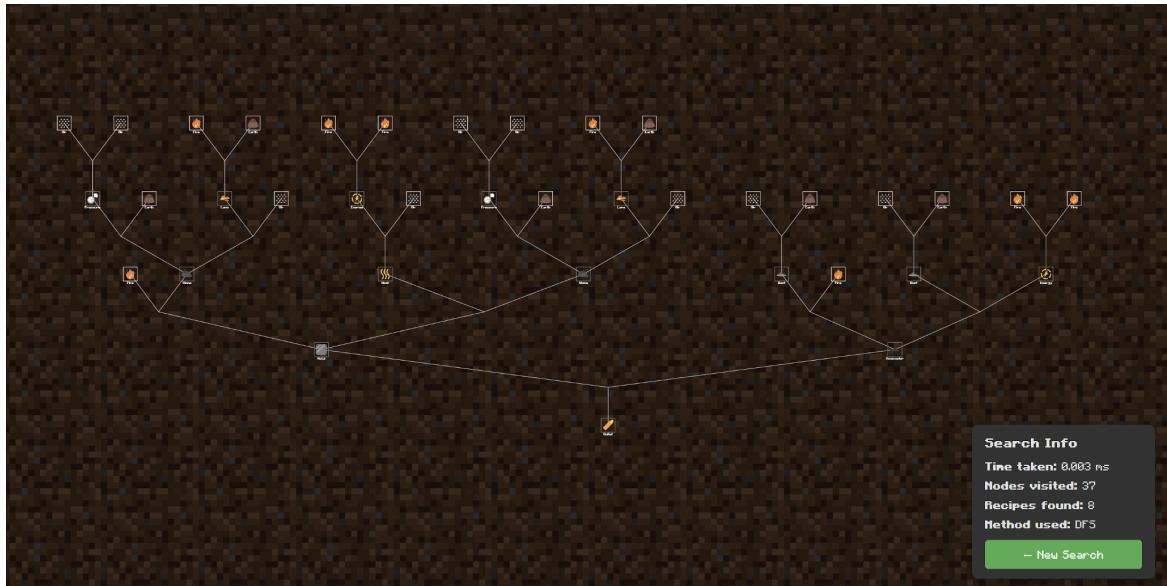
```



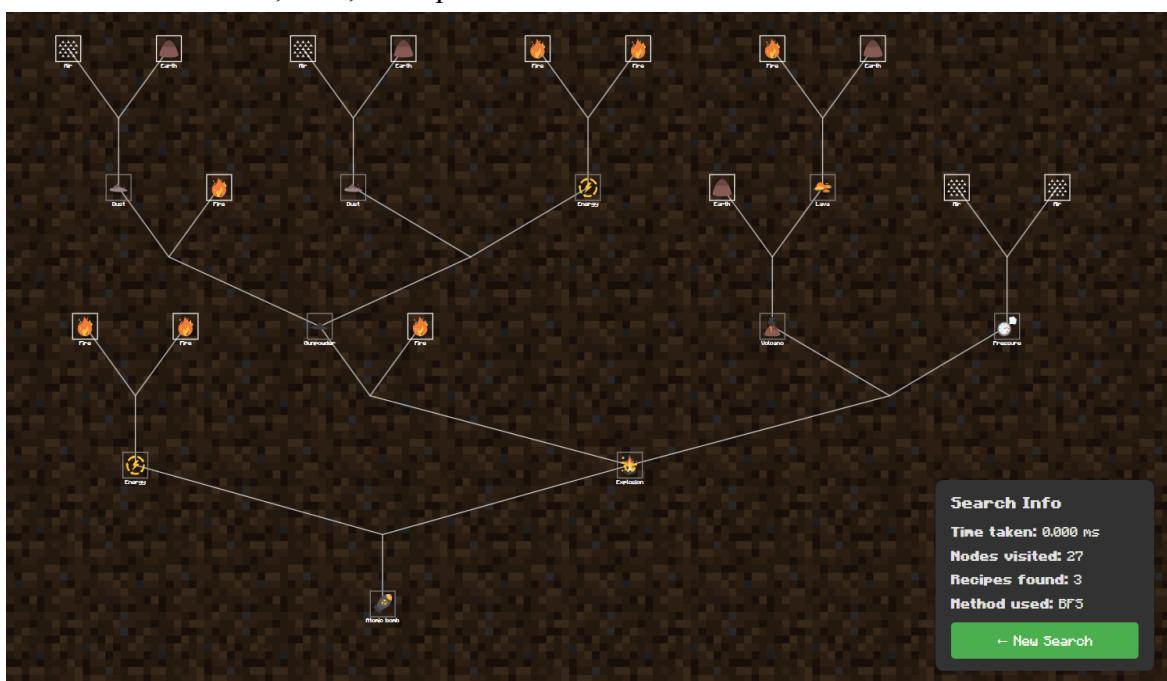
Ini merupakan front-end yang telah dibangun. Penggunaan aplikasi dapat dilakukan dengan menentukan elemen yang ingin dibuat serta parameter proses pembuatan pohon resep di sebelahnya. Ketika sudah sesuai dengan keinginan, cukup menekan tombol `SEARCH` untuk memulai pencarian.

3. Pengujian

- Bullet, DFS, 8 recipe



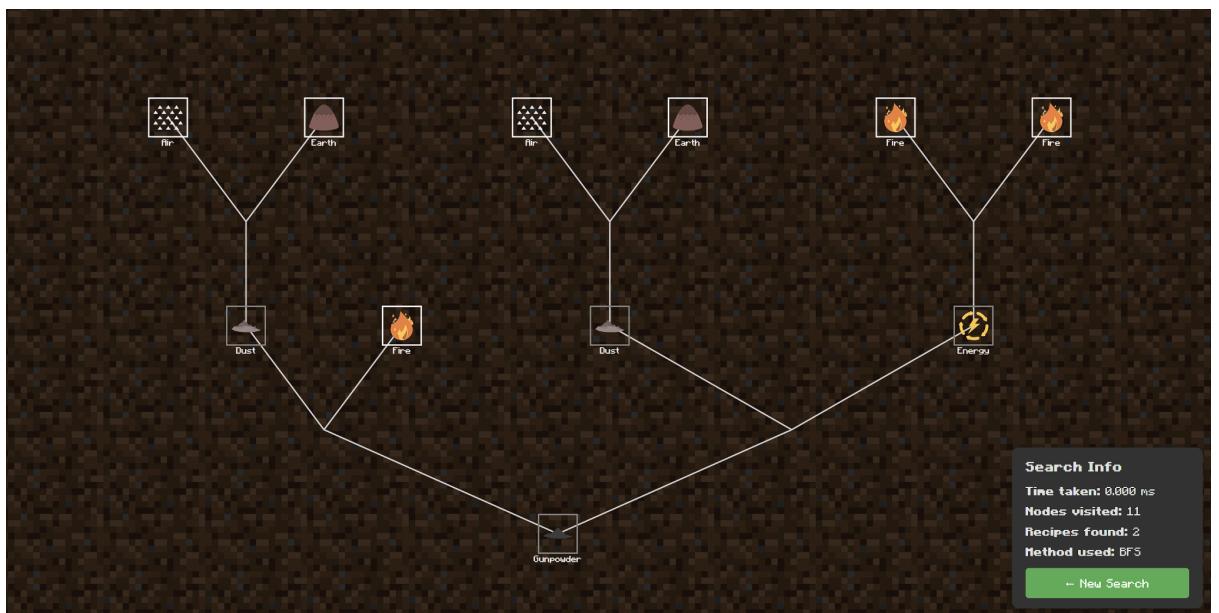
- Atomic bomb, BFS, 3 recipe



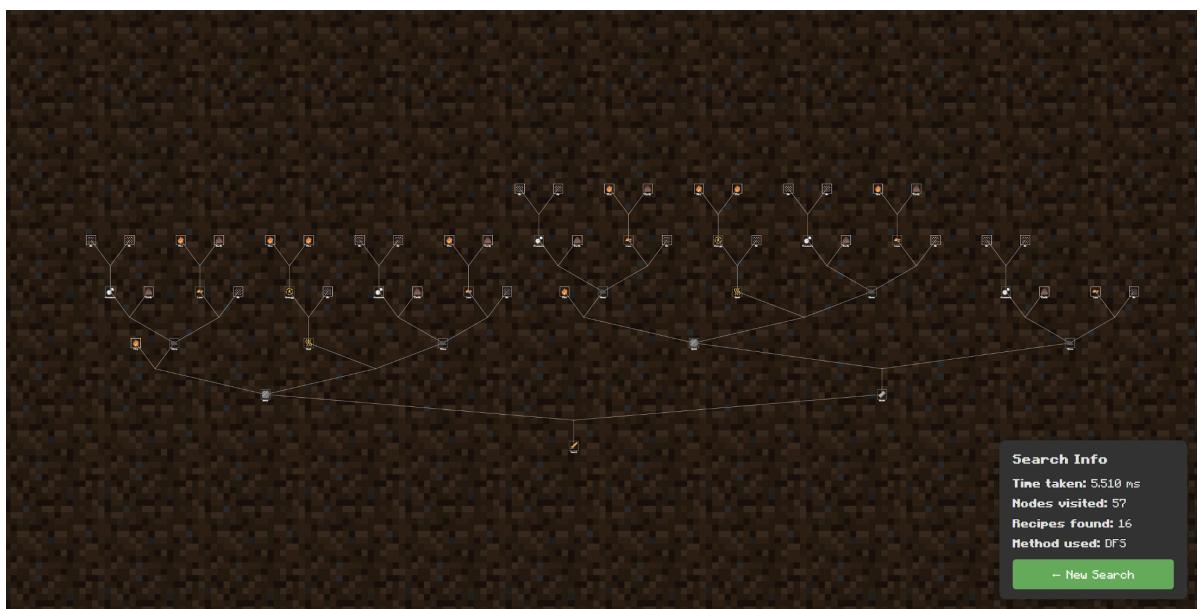
- Gun, DFS, 16 recipe



- Gunpowder, BFS, 5 Recipe



- Sword, DFS, 16 Recipe



4. Analisis

Menurut pengujian yang dilakukan, pohon resep dapat dibuat dengan cepat. Pohon resep tersebut dapat digunakan oleh pengguna untuk mencapai sebuah elemen yang diinginkan. Jumlah resep yang mungkin biasanya melebihi maxRecipe, tapi ini merupakan limitasi penggunaan pohon. Ketika melakukan pencarian secara BFS maupun DFS, jumlah kemungkinan resep yang valid dari sebuah elemen merupakan hasil perkalian jumlah pohon di kanan dengan jumlah pohon di kiri. Jika kanan sudah menemukan beberapa jumlah dan kiri masih belum, pohon tersebut masih dianggap memiliki 0 resep yang mungkin. Ini menyebabkan ketika pohon di kiri akhirnya mendapatkan 1 resep, penggabungan akan langsung banyak. Untuk elemen yang kompleks dengan tier tinggi, biasanya akan mengakibatkan tree tersebut menjadi sangat besar. Ini merupakan limitasi metode pengecekan validitas pohon resep karena melakukan traversal keseluruhan pohon. Untuk keadaan dimana pohon tersebut sangat besar, pengecekan akan menjadi cukup lambat, sehingga pencarian masih tetap dilakukan sebelum stop. Delay pengecekan validitas juga berpengaruh pada jumlah resep yang dikembalikan, karena behaviour konkurensi pada program cukup volatile.

Pada pencarian elemen, terdapat beberapa elemen yang terhambat dikarenakan heuristik minimasi tier dan penghilangan elemen Time. Contohnya, pada Wood (*tier* 9), resep Wood yang memanfaatkan bahan *tier* 8 kebawah hanya satu, dan resep tersebut memanfaatkan Time. Dengan menghilangkan kedua batasan tersebut, Wood tidak akan memiliki resep, mengakibatkan Wood menjadi sebuah *dead-end* dalam pencarian beberapa resep yang membutuhkannya.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Dalam tugas besar ini, kami sudah membuat suatu website yang mampu membantu pemain baru game Little Alchemy 2 untuk menemukan elemen yang ingin mereka buat. Pencarian resep dilakukan dengan menampilkan sebuah pohon resep yang dibentuk menggunakan algoritma BFS (Breadth-First Search) ataupun DFS (Depth-First Search). Pengguna juga dapat menentukan jumlah resep yang mereka inginkan untuk menuju elemen tersebut. Pengguna juga dapat menggunakan fitur Live Update untuk melihat pembentukan resep suatu elemen secara real time. Pemetaan elemen dan resep terhadap struktur data pohon sudah sesuai implementasinya pada pohon resep yang dihasilkan.

5.2 Saran

Selama penggerjaan, ada beberapa momen yang disayangkan akibat perubahan spesifikasi. Sehingga progres yang sudah ada perlu dirombak atau diubah banyak akibat perubahan-perubahan tersebut. Sebaiknya, dilakukan pengujian yang pasti agar pembuatan spesifikasi menjadi lebih pasti sehingga perubahan yang diperlukan minim dan tidak mengganggu progres yang sudah lewat. Selain itu, pada bagian QNA, terdapat jawaban yang tidak konsisten yang menciptakan kebingungan, terkadang juga pertanyaan yang diberikan di QNA memakan waktu yang lama untuk dijawab sehingga menciptakan keraguan dalam penggerjaan yang menghambat progres.

5.3 Refleksi

Dalam proses penggerjaan, banyak hal yang telah dipelajari dari melakukan tugas ini. Wawasan mengenai pemanfaatan algoritma graf dan map bertambah luas ketika mengatur pemetaan permasalahan menjadi elemen-elemen struktur data Tree dan algoritmanya. Selain memperdalam algoritmanya, kami mempelajari banyak dari pembuatan aplikasi web dengan Front-end dan Back-end. Dalam pembangunan back-end, kami mempelajari banyak mengenai bahasa Go dan juga cara setup server back-end. Dalam pembangunan front-end, kami mempelajari hal baru dalam proses mengatur UI yang baik dan proses penampilan pohon pada front-end. Akhirnya, kami juga mempelajari cara melakukan *containerization* dan *deployment* sebuah aplikasi web.

Melihat proses penggerjaan kami, sebaiknya penggerjaan dilakukan dengan mengatur waktu yang baik. Penggerjaan tugas ini diselingi sangat banyak tugas lainnya, sehingga cukup kewalahan dalam penggerjaan. Selain itu, testing sebaiknya dilakukan berulang kali untuk beberapa *edge case* untuk mengetahui integritas program yang dibangun.

LAMPIRAN

Pranala ke repository yang berisi kode program.

https://github.com/mraifalkautsar/Tubes2_FE_CraftingTable

(Front End)

https://github.com/Henshou/Tubes2_BE_CraftingTable

(Back End)

Pranala ke video bonus.

<https://youtu.be/mSLQqL4iBl8>

Pranala ke link deployed.

<https://tubes2fecraftingtable-production.up.railway.app/>

Poin	Ya	Tidak
1. Aplikasi dapat dijalankan.	✓	
2. Aplikasi dapat memperoleh <i>data recipe</i> melalui scraping.	✓	
3. Algoritma <i>Depth First Search</i> dan <i>Breadth First Search</i> dapat menemukan <i>recipe</i> elemen dengan benar.	✓	
4. Aplikasi dapat menampilkan visualisasi <i>recipe</i> elemen yang dicari sesuai dengan spesifikasi.	✓	
5. Aplikasi mengimplementasikan multithreading.	✓	
6. Membuat laporan sesuai dengan spesifikasi.	✓	
7. Membuat bonus video dan diunggah pada Youtube.	✓	
8. Membuat bonus algoritma pencarian <i>Bidirectional</i> .		✓
9. Membuat bonus <i>Live Update</i> .	✓	
10. Aplikasi di- <i>containerize</i> dengan Docker.	✓	
11. Aplikasi di- <i>deploy</i>	✓	

REFERENSI

All elements in Little Alchemy 2 -

[https://little-alchemy.fandom.com/wiki/Elements_\(Little_Alchemy_2\)](https://little-alchemy.fandom.com/wiki/Elements_(Little_Alchemy_2))

“Go By Example”, Go Documentation- <https://gobyexample.com/>

giflib example usage - <https://gist.github.com/suzumura-ss/a5e922994513e44226d33c3a0c2c60d1>

GIF file format - https://en.wikipedia.org/wiki/GIF#Animated_GIF

Structural Similarity Index -

https://www.researchgate.net/publication/342435717_Understanding_SSIM