LAPORAN TUGAS BESAR IF2211
STRATEGI ALGORITMA

Tugas Besar 2 Strategi Algoritma

Stefan Mattew Susanto	13523020
Hanif Kalyana Aditya	13523041
Aramazaya	13523082

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2025

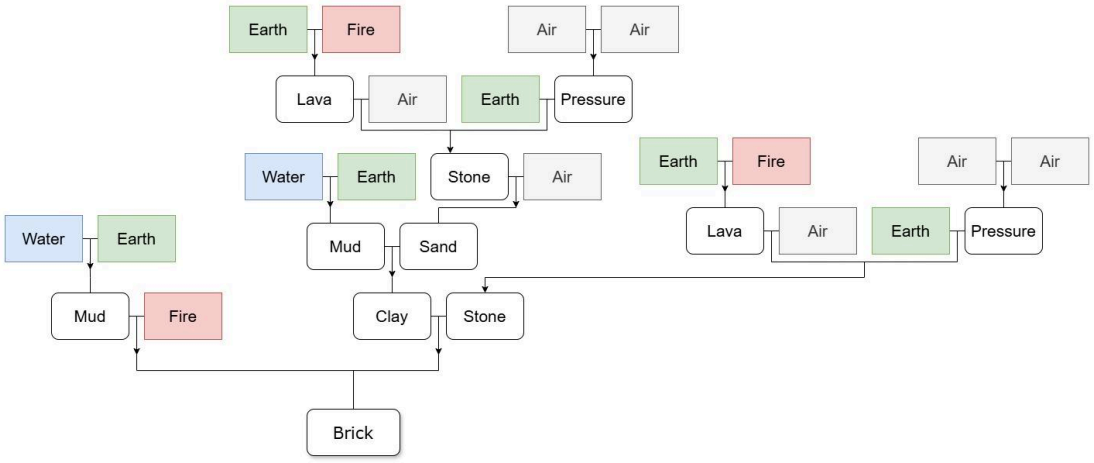
DAFTAR ISI

BAB I

DESKRIPSI TUGAS

Spesifikasi Wajib

- Buatlah aplikasi pencarian *recipe* elemen dalam permainan Little Alchemy 2 dengan menggunakan strategi **BFS dan DFS**.
- Tugas dikerjakan berkelompok dengan anggota **minimal 2 orang** dan **maksimal 3 orang**, boleh lintas kelas dan lintas kampus.
- Aplikasi berbasis **web**, untuk *frontend* dibangun menggunakan bahasa **Javascript** dengan *framework* **Next.js atau React.js**, dan untuk *backend* menggunakan bahasa **Golang**.
- Untuk *repository frontend* dan *backend* diperbolehkan **digabung** maupun **dipisah.**
- Untuk data elemen beserta resep dapat diperoleh dari ***scraping*** *website* Fandom Little Alchemy 2.
- Terdapat opsi pada *aplikasi* untuk **memilih algoritma** BFS atau DFS (juga *bidirectional* jika membuat bonus)
- Terdapat *toggle button* untuk memilih untuk menemukan **sebuah *recipe*** (Silahkan yang mana saja) ~~**terpendek** (*output* dengan rute terpendek~~) atau **mencari banyak *recipe* *(multiple recipe)*** menuju suatu elemen tertentu. Apabila pengguna ingin mencari banyak *recipe* maka terdapat cara bagi pengguna untuk **memasukkan parameter banyak *recipe*** maksimal yang ingin dicari. Aplikasi boleh mengeluarkan *recipe* apapun asalkan berbeda dan memenuhi banyak yang diinginkan pengguna (apabila mungkin).
- Mode pencarian *multiple recipe* wajib dioptimasi menggunakan ***multithreading.***
- Elemen yang digunakan pada suatu *recipe* harus berupa elemen dengan ***tier* lebih rendah** dari elemen yang ingin dibentuk.
- Aplikasi akan **memvisualisasikan *recipe*** yang ditemukan sebagai sebuah *tree* yang menunjukkan kombinasi elemen yang diperlukan dari elemen dasar. Agar lebih jelas perhatikan contoh berikut



Gambar 3. Contoh visualisasi *recipe* elemen

Gambar diatas menunjukkan contoh visualisasi *recipe* dari elemen *Brick*. Setiap elemen bersebelahan menunjukkan elemen yang perlu dikombinasikan. Amati bahwa *leaf* dari *tree* selalu berupa elemen dasar. Apabila dihitung, gambar diatas menunjukkan 5 buah *recipe* untuk *Brick* (karena *Brick* dapat dibentuk dengan kombinasi *Mud+Fire* atau *Clay+Stone*, begitu pula Stone yang dapat dibentuk oleh kombinasi *Lava+Air* atau *Earth+Pressure*). Visualisasi pada aplikasi tidak perlu persis seperti contoh diatas, tetapi pastikan bahwa *recipe* ditampilkan dengan jelas.

- Aplikasi juga menampilkan **waktu pencarian** serta **banyak *node*** yang dikunjungi.

Spesifikasi Bonus

- **(maks 5)** Membuat video tentang aplikasi BFS dan DFS pada permainan Little Alchemy 2 di Youtube. Video dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Untuk contoh video tubes stima tahun-tahun sebelumnya dapat dilihat di Youtube dengan kata kunci “Tubes Stima”, “strategi algoritma”, “Tugas besar stima”, dll. **Semakin menarik video, maka semakin banyak poin yang diberikan.**
- **(maks 3)** Aplikasi dijalankan menggunakan **Docker** baik untuk *frontend* maupun *backend.*
- **(maks 3)** Aplikasi **di-*deploy*** ke aplikasi *deployment* (aplikasi *deployment* bebas) agar bisa diakses secara daring
- **(maks 3)** Menambahkan algoritma bukan hanya DFS dan BFS, tetapi juga strategi **bidirectional**
- **(maks 6)** Aplikasi memiliki fitur ***Live Update*** visualisasi *recipe* selama proses pencarian. *Tree* visualisasi akan dibangun bertahap secara ***real time*** sesuai

dengan progress pencarian. Tambahkan ***delay*** pada Live Update karena pencarian dapat berjalan dengan sangat cepat.

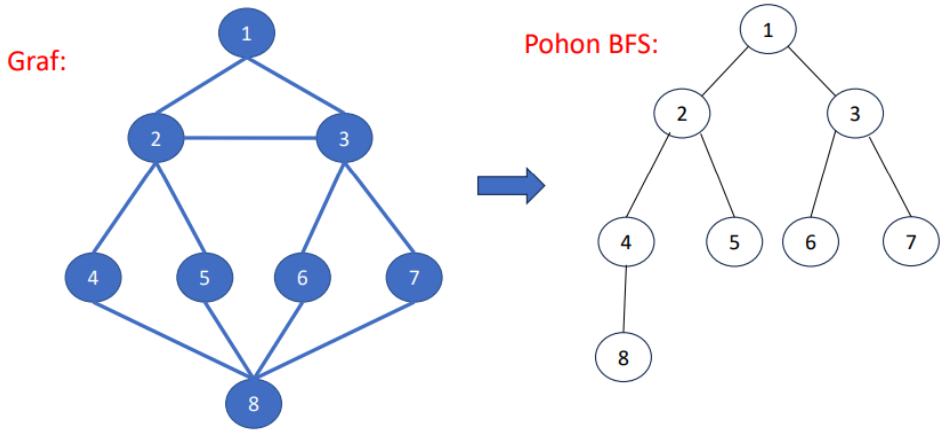
BAB II
LANDASAN TEORI

2.1. Dasar Teori

1.1. Penjelajahan Graf

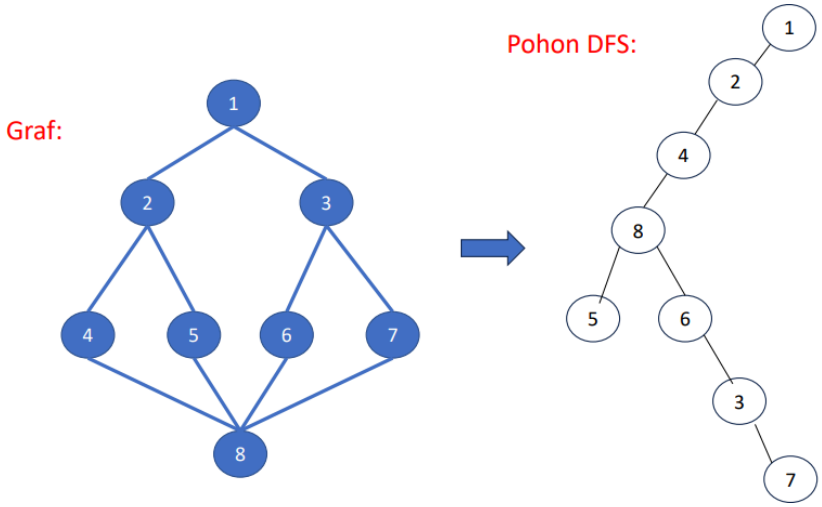
Graf umumnya digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf harus memiliki minimal 1 simpul dan boleh tidak memiliki sisi. Penjelajahan graf dapat dilakukan dengan mengunjungi simpul simpul yang terdapat dalam graf secara sistematik. Penjelajahan graf dapat digunakan dalam menelusuri jalur ataupun banyaknya jalur yang dapat digunakan untuk mencapai suatu simpul tertentu. Serta digunakan untuk mencari jalur yang paling efektif. Dalam penjelajahan graf terdapat 2 pendekatan, yakni Breadth-First Search (BFS) dan juga Depth-First Search (DFS).

1.2. Algoritma BFS dan DFS



Breadth-First Search (BFS) adalah salah satu algoritma penjelajahan graf yang menelusuri graf secara melebar dari simpul awal. Algoritma akan dimulai pada simpul awal (simpul v). Lalu akan menuju ke semua simpul yang bertetangga dengan simpul v terlebih dahulu. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul -simpul yang tadi dikunjungi, demikian seterusnya. BFS cukup efektif untuk menemukan jalur terpendek dalam graf tidak berbobot

karena ia menjelajahi graf berdasarkan kedalaman. Namun algoritma BFS cenderung memerlukan memori yang memori relatif lebih besar.



Depth-First Search (DFS) adalah algoritma penjelajahan graf yang menelusuri simpul-simpul pada graf secara mendalam terlebih dahulu sebelum kembali ke simpul sebelumnya. Misalnya DFS akan dimulai dari simpul v. DFS akan mengunjungi simpul w yang bertetangga dengan simpul v. Mengulangi DFS mulai dari simpul w. Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, akan backtrack ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi. DFS menggunakan memori yang relatif lebih hemat dibandingkan BFS. DFS cocok dalam mencari kemungkinan dari semua solusi yang dalam. Dalam DFS, jalur yang ditemukan belum tentu yang paling efisien atau terdekat

2.2. Penjelasan singkat mengenai aplikasi web yang dibangun.

Aplikasi web ini dirancang untuk membantu pengguna mencari kombinasi elemen dalam permainan Little Alchemy 2 menggunakan algoritma penelusuran graf. Pengguna dapat memasukkan elemen target yang ingin dibentuk, lalu memilih metode pencarian (BFS atau DFS). Pengguna dapat memilih untuk menemukan sebuah recipe atau menemukan banyak recipe. Pengguna yang memilih banayak recipe maka perlu memasukkan jumlah recipe maksimal yang ingin dicari. Aplikasi akan menampilkan jalur kombinasi elemen

yang membentuk target, dilengkapi dengan visualisasi pohon solusi dan pembaruan proses secara real-time. Aplikasi ini dibangun menggunakan bahasa Go untuk backend dan framework Next.js untuk frontend.

.

BAB III
Analisis Pemecahan Masalah

3.1. Langkah-langkah pemecahan masalah.

- Menganalisis permainan Little Alchemy 2
  Memahami elemen- elemen yang terdapat dalam game. Serta memahami tier dan proses pembentukan suatu elemen

- Merepresentasikan data dengan melakukan scraping
  Data pada website *website* Fandom Little Alchemy 2 akan diolah untuk dimasukkan ke dalam file database. Data yang diolah adalah data elemen-elemen dan tiernya serta gambar/icon dari tiap elemen. Data elemen diolah dan dimasukkan ke dalam file json. Sedangkan, data gambar akan dimasukkan dalam bentuk png. Contoh data hasil scraping dapat di lihat di lampiran.

- Pemodelan data
  Data elemen akan diolah dan dibentuk menjadi suatu graf tak berarah.

- Implementasi penjelajahan graf dengan algoritma DFS daan BFS.
  Terdapat 2 algoritma yang digunakan untuk melakukan pencarian suatu elemen, yaitu DFS dan BFS.

- Visualisasi data
  Setelah algoritma selesai dilakukan maka recipe elemen akan digambarkan dalam bentuk tree pada frontend.

3.2. Proses pemetaan masalah menjadi elemen-elemen algoritma DFS dan BFS.

- Pada aplikasi web ini, elemen berperan sebagai node/ simpul dari suatu graf, misalnya air, water.
- Kombinasi dari dua elemen berperaan sebagai sisi pada graf, contohnya Earth+Air → Dust.

- Elemen dasar didefinisikan menjadi startElement terdapat 4 yakni Air, Water, Fire, Earth. Elemen dasar ini sebagai root/ awal proses penjelajahan graf dimulai.
- Elemen target yang ingin dibentuk dari elemen-elemen lainnya sebagai simpul tujuan.
- Proses akan dilanjutkan ke tahap pemrosesan dengan DFS dan BFS.
- Pada BFS, elemen mulanya akan mencari dari elemen dasar lalu ia akan mengeksplor jalur secara melebar. BFS akan mencari berdasarkan tingkatan tier lebih dulu.
- Pada DFS, elemen akan dimulai dari elemen dasar secara multithreading, lalu akan mengeskplor satu jalur lebih dulu sampai ujung lalu kembali backtrack dan mencari jalur lain.

3.3. Fitur fungsional dan arsitektur aplikasi web yang dibangun.

Fitur fungsional:

- Pencarian recipe elemen menggunakan algoritma BFS dan DFS.
- Pilihan untuk mencari sebuah recipe atau banyak recipe(multiple).
- Input jumlah maksimal recipe yang dicari pada mode multiple recipe.
- Visualisasi recipe dalam bentuk tree.
- Penampilan waktu pencarian dan jumlah node yang dikunjungi.

Arsitektur

- Backend
  Backend menggunakan bahasa golang dalam implementasinya. Pada backend, program akan menerima request dari frontend dan menjalankan algoritma BFS/DFS, lalu akan mengembalikan hasil ke frontend.

- Frontend
  Frontend dibangun dengan bahasa javascript dengan framework Next.js. Pada frontend, program akan menampilkan interface kepada pengguna, menerima input dari user, mengirimkan request ke backend, menerima hasil dari backend , dan menampilkan visualisasi hasil

3.4. Contoh ilustrasi kasus.

Misalnya target elemen yang ingin dicari adalah Dust.

Dust merupakan elemen yang terbentuk dari elemen Earth+Air.

Dengan BFS:

1. Elemen akan dimulai dari elemen dasar Air, Earth, Fire, Water.

2. Elemen akan digabungkan dengan elemen dasar yang lain
    - Fire + Air → Smoke
    - Fire + Water→Steam
    - Air + Water→Mist
    - Earth + Fire→Lava
    - **Air +Earth→ Dust** (hasil recipe ditemukan)
3. Elemen akan dilanjutkan pada tier selanjutnya sesuai dengan tier target element. Pada kasus ini tidak dilanjutkan karena tier dust adalah tier 1 element.
4. Recipe berhasil ditemukan dan akan ditampilkan

Dengan DFS:

1. Elemen akan dimulai pada salah satu elemen dasar misalnya elemen Fire. Tetapi proses akan dilakukan secara multithreading.
2. Mencari kombinasi dengan Fire secara mendalam ke tier selanjutnya sampai tier maksimal yaitu tier target element.
    - Fire + Water → Steam
    - Steam +..
    - Dan seterusnya
3. Ketika elemen mencari dari kombinasi Fire tidak ditemukan maka akan backtrack
4. Lalu pada kombinasi dengan Air:
    - Air + Fire →Steam
    - …
    - Backtrack
    - Air + Earth → Dust.
5. Recipe ditemukan dan tree akan ditampilkan.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi teknis program

Sturktur Data:

```
├── README.md
│
├── docker-compose.yml
│
├── doc/
│   └── Tubes2_Aku Suka Yang Little-little.pdf
│
└── src/
    ├── backend/
    │   ├── bfs
    │   │   ├── main.go
    │   │   ├── go.mod
    │   │   └── dockerfile
    │   ├── dfs
    │   │   ├── main.go
    │   │   ├── go.mod
    │   │   └── dockerfile
    │   └── shared/
    │       ├── algorithm
    │       │   ├── bfs.go
    │       │   └── dfs.go
    │       ├── data
    │       │   ├── images/
    │       │   ├── elements.json
    │       │   └── tiers.json
    │       ├── handdler
    │       │   └── scrape.go
    │       ├── model
    │       │   └── element.go
    │       ├── utility
    │       │   └── loader.go
    │       ├── go.mod
    │       └── go.sum
    └── Frontend/
        ├── data
```

```
    ├── images/
    ├── elements.json
    └── tiers.json
├── public
├── src
    ├── app
        ├── layout.js
        ├── global.css
        └── page.js
    ├── component
        ├── ElementCardSelector.js
        ├── ResultTree.js
        ├── SearchButton.js
        └── SearchButton.js
├── dockerfile
├── next.config.mjs
├── postcss.config.mjs
├── estlint.config.mjs
├── package-lock.json
├── package.json
└── jsconfig.json
```

Fungsi dan Prosedur pada backend:

- Scrapper.go

```
func downloadImage(url, name string) (string, error) { … }

func RunScrapperAndSave() error { … }
```

- loader.go

```
func LoadElementsFromFile(path string) (*model.ElementsDatabase, error)
{ … }

func SortByTier(db *model.ElementsDatabase) *model.ElementsDatabase {
}
```

```
func ParseTier(tierStr string) int {
… }
```

- bfs.go

```
func iterativeExpansion(path []model.Recipe, db *model.ElementsDatabase,
startElements []string, step chan<- *SearchProgress) []model.Recipe
 {... }
func BFSWithOptions(db *model.ElementsDatabase, startElements []string,
targetElement string, bannedTargetRecipes []int, maxPaths int, result
chan<- *BFSResult, progress chan<- *SearchProgress)
 {...}

func BFSSingle(db *model.ElementsDatabase, startElements []string,
targetElement string, result chan<- *BFSResult, progress chan<-
*SearchProgress)
 {...}

func BFSMultipleThreaded(db *model.ElementsDatabase, startElements
[]string, targetElement string, maxPaths int, timeoutSeconds int, result
chan<- *BFSResult) {...}
func keysFromMap(m map[string]bool) []string {...}

func generateFixedShuffles(elements []string) [][]string
 {...}

func min(a, b int) int {...}

func isDuplicatePath(newPath []model.Recipe, existingPaths
[][]model.Recipe) bool {...}


func isValidTierProgression(recipe model.Recipe, resultElement
model.Element, db *model.ElementsDatabase) bool {...}


func Driver(db *model.ElementsDatabase, targetElement string, maxPaths
```

```
int, step chan<- *SearchProgress) *BFSResult {...}
```

- dfs.go

```
func DFS(db *model.ElementsDatabase, startElements []string,
targetElement string, maxPath int, result chan<- *DFSResult, step chan<-
*SearchProgress) { …}

func expandPath(path []model.Recipe, db *model.ElementsDatabase,
startElements []string) []model.Recipe {..}

func MultiDFS(db *model.ElementsDatabase, targetElement string, maxPath
int, step chan<- *SearchProgress) *DFSResult {
```

4.2. Tata cara Penggunaan Program

- Interface Program
    - Input Elemen Target :Terdapat kolom input untuk memasukkan nama elemen target.
    - Pilihan Algoritma: Pengguna memilih algoritma pencarian: BFS atau DFS.
    - Pilihan Mode Pencarian: Pengguna memilih antara single recipe atau multiple recipe
    - Input Jumlah Maksimal Resep: Jika pengguna memilih mode multiple , pengguna memasukkan jumlah maksimal resep yang ingin ditampilkan.
    - Visualisasi Resep: Resep yang telah dihasilkan akan ditampilkan dalam bentuk tree.
    - Informasi Pencarian: Aplikasi akan menampilkan waktu pencarian dan jumlah node yang dikunjungi.
- Fitur-Fitur yang Disediakan
    - Pencarian resep elemen.
    - Visualisasi resep dalam bentuk tree.
    - Pilihan algoritma pencarian (BFS dan DFS).
    - Pilihan mode pencarian (single/multiple).

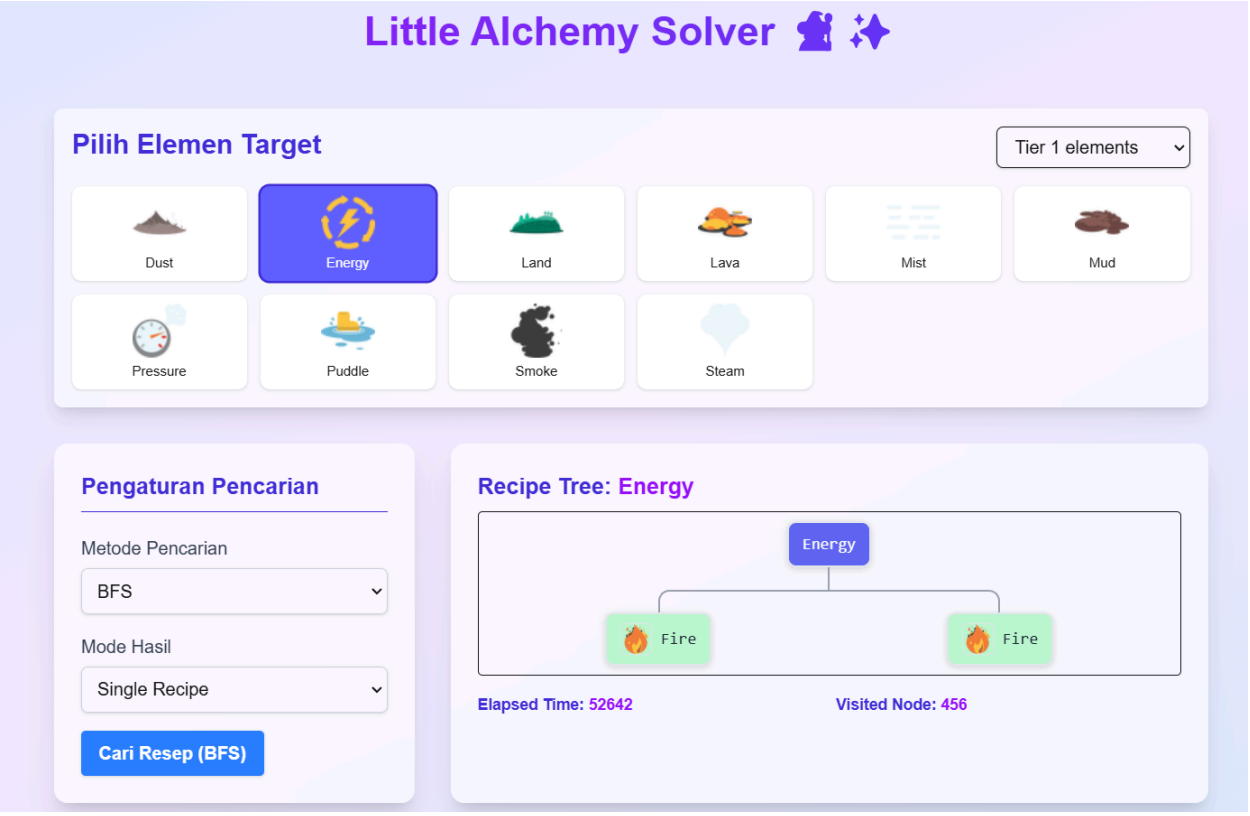
- Multithreading untuk pencarian semua jalur.

- Cara menjalankan Program

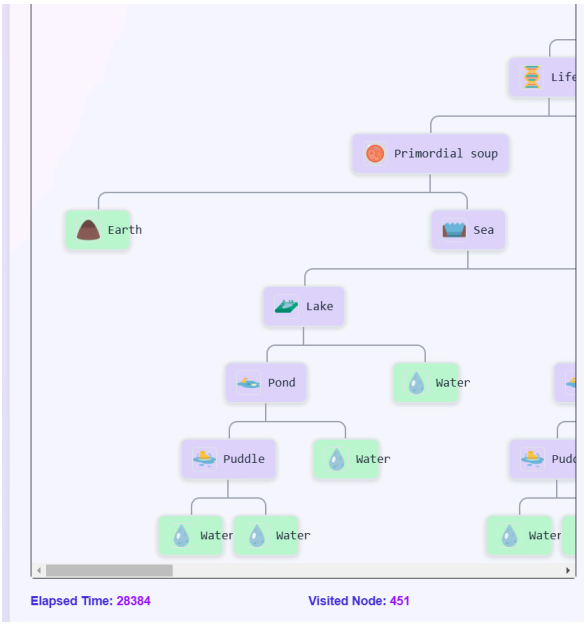
1. Clone repository
2. Buka terminal pada direktori backend
3. jalankan perintah "go run main.go"
4. Bka terminal baru di folder drontend
5. lakukan perintah "npm run dev”

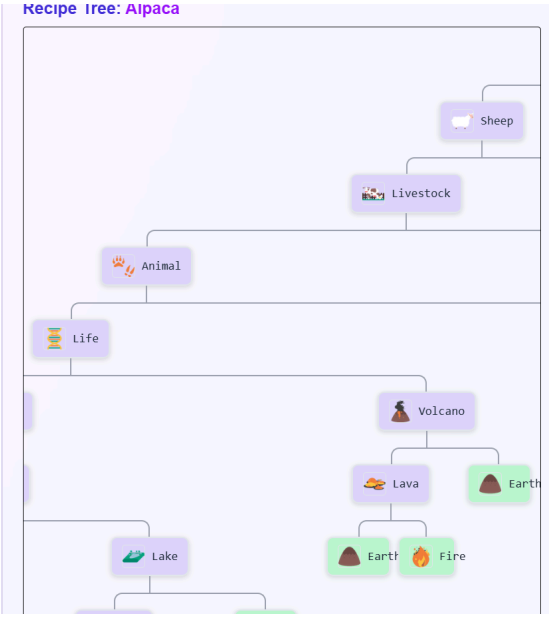
4.3. Hasil Pengujian

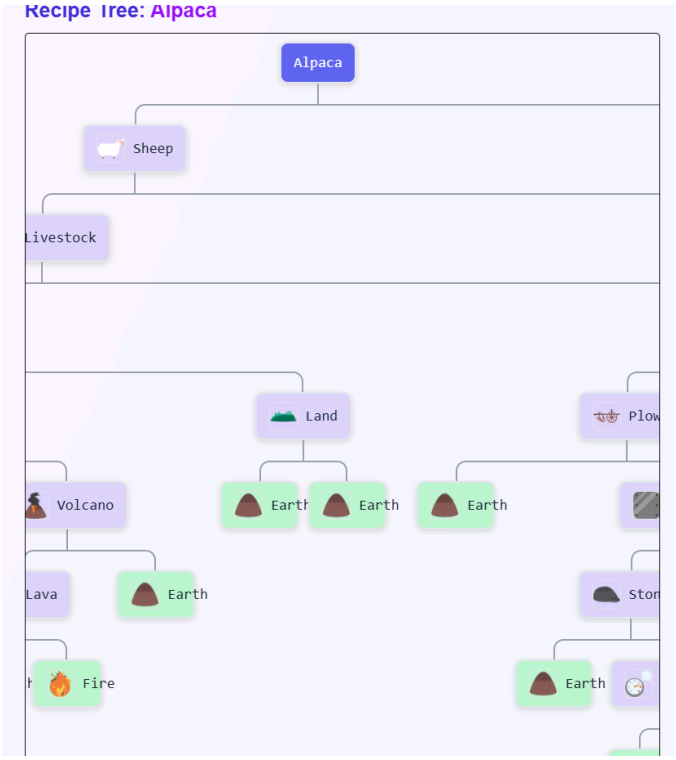
- Pencarian Elemen “Energy”



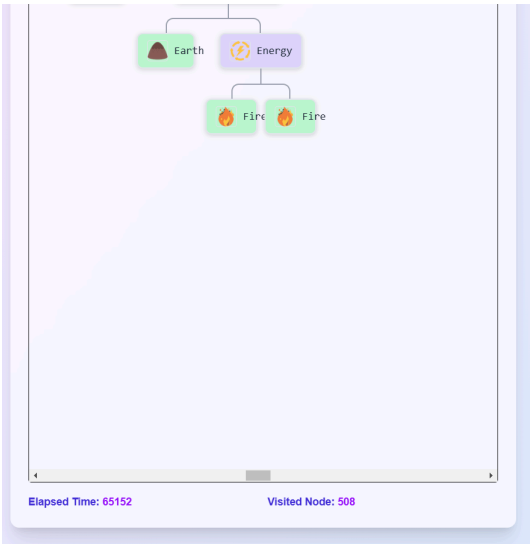
- Pencarian Elemen “Alpaca”

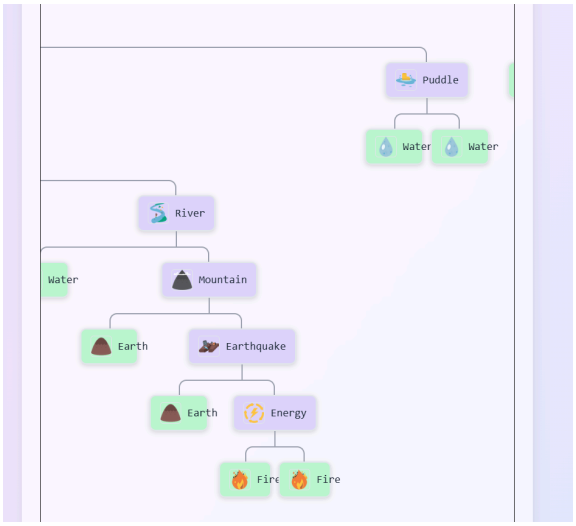






- Pencarian Elemen “Picnic”





4.4. Analisis Hasil Pengujian

Analisis Pencarian Elemen “Energy”

Elemen energy dapat dibentuk dari dua elemen dasar yaitu fire + fire dan tree berhenti sampai di elemen tersebut

Analisis Pencarian Elemen “Alpaca”

Elemen alpaca dapat dibentuk dari beberapa resep menggunakan elemen dasar

Elapsed time : 28384

Node : 451

Analisis Pencarian Elemen “Picnic”

Elemen alpaca dapat dibentuk dari beberapa resep menggunakan elemen dasar

Elapsed time : 65152

Node : 508

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Program ini berhasil memodelkan proses pencarian elemen dalam *Little Alchemy 2* sebagai masalah penelusuran graf. Dengan menggunakan algoritma BFS dan DFS, program mampu menemukan kombinasi elemen menuju target secara efisien. BFS unggul dalam menemukan jalur terpendek, sementara DFS mengeksplorasi kemungkinan yang lebih dalam. Melalui implementasi ini, kami memahami penerapan strategi algoritma pencarian serta pentingnya efisiensi dan visualisasi dalam menyelesaikan masalah kompleks.

5.2. Saran

Program yang kami buat masih jauh dari sempurna. Untuk kedepannya, bisa lebih memperhatikan struktur program, flow program yang lebih jelas, dan juga menyisihkan waktu lebih banyak untuk bug fixing secara total.

5.3. Komentar

Kami mengucapkan terima kasih kepada asisten-asisten yang bertanggung jawab atas pelaksanaan tugas besar ini dan karena telah memberikan kesan yang terbaik untuk tugas pertama di Teknik Informatika ITB juga memberi gambaran akan bagaimana tugas-tugas besar lain yang akan dihadapi kedepannya. Kami juga mengucapkan terima kasih kepada mahasiswa Teknik Informatika ITB lain yang telah memberikan semangat dan dukungan sehingga pada akhirnya tugas besar ini dapat selesai dengan baik.

5.4. Refleksi

Tugas besar ini telah memberikan kami pengalaman yang sangat mengesankan dari senang hingga tekanan. Kami menghadapi berbagai macam kendala seperti alokasi waktu yang kurang efektif akibat kesibukannya masing-masing, belum terbiasa dengan bahasa golang sendiri, dan kinerja dari kami sendiri yang kurang baik. Untuk kedepannya, diharapkan tugas besar ini bisa menjadi motivasi untuk kami agar kami lebih berusaha kedepannya baik dalam pengerjaan tugas maupun menghadapi hidup

LAMPIRAN

Link Repository GitHub:

https://github.com/StefanMattew/Tubes2_Aku-Suka-Yang-Little-little.git

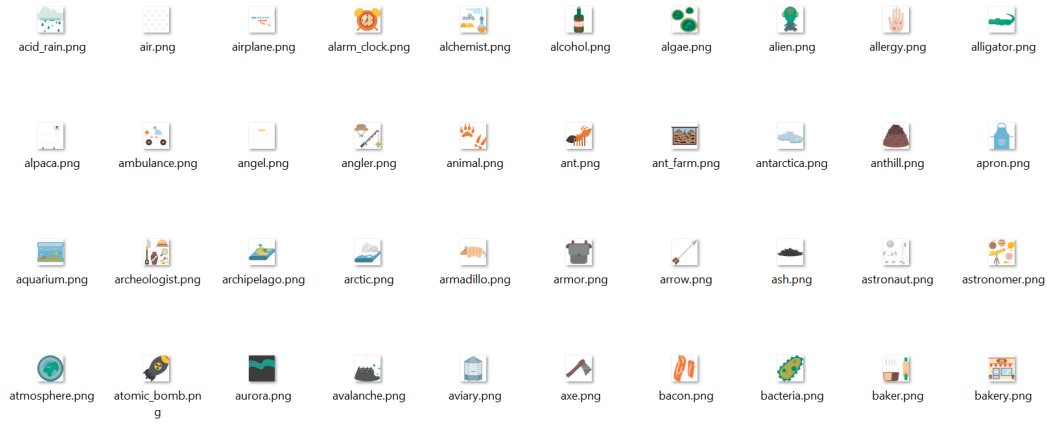
Tabel Check:

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

Contoh Data Hasil Scraping:


```
"Air": {
  "combos": [
    [
      "Fire",
      "Mist"
    ]
  ],
  "image": "air.png",
  "tier": "Starting elements"
},
```

```
"Allergy": {
  "combos": [
    [
      "Human",
      "Dust"
    ],
    [
      "Human",
      "Pollen"
    ]
  ],
  "image": "allergy.png",
  "tier": "Tier 8 elements"
},
```

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

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