

LAPORAN TUGAS KECIL
IF2211 Strategi Algoritma
Penyelesaian Permainan Word Ladder
Menggunakan Algoritma UCS, Greedy Best First
Search, dan A*



Disusun oleh:

Angelica Kierra Ninta Gurning **(13522048)**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

| | |
|---|-----------|
| DAFTAR ISI..... | 1 |
| BAB I..... | 2 |
| Analisis dan Implementasi..... | 2 |
| 1.1. Algoritma UCS (Uniform Cost Search)..... | 2 |
| 1.2. Algoritma GBFS (Greedy Best First Search)..... | 3 |
| 1.3. Algoritma A* (A Star)..... | 5 |
| 1.4. Analisis Lanjutan..... | 7 |
| BAB II..... | 9 |
| Penjelasan Struktur Kode..... | 9 |
| 2.1. Abstraksi Class dan Method yang dibuat..... | 9 |
| 2.2. Detail Class dan Method..... | 10 |
| BAB III..... | 18 |
| Hasil Implementasi..... | 18 |
| 3.1. Tangkapan Layar Algoritma UCS (Uniform Cost Search)..... | 18 |
| 3.2. Tangkapan Layar Algoritma GBFS (Greedy Best First Search)..... | 22 |
| 3.3. Tangkapan Layar Algoritma A* (A Star)..... | 29 |
| BAB IV..... | 35 |
| Analisis Perbandingan Algoritma..... | 35 |
| 4.1. Perbandingan Optimalitas..... | 35 |
| 4.2. Perbandingan Waktu Eksekusi..... | 37 |
| 4.3. Perbandingan Memori yang Dibutuhkan..... | 39 |
| BAB V..... | 42 |
| Implementasi Bonus..... | 42 |
| 5.1. Tampilan GUI..... | 42 |
| 5.2. Kode Utama GUI..... | 44 |
| BAB VII..... | 47 |
| Lampiran..... | 47 |

BAB I

Analisis dan Implementasi

1.1. Algoritma UCS (Uniform Cost Search)

Algoritma UCS (*Uniform Cost Search*) merupakan algoritma pencarian *uninformed search* atau sering disebut sebagai *blind search*. Artinya, algoritma UCS tidak memiliki informasi tambahan mengenai kondisi pencarian selain dari yang disediakan pada definisi masalah.

Algoritma UCS merupakan perluasan dari algoritma BFS, yaitu pencarian dengan melihat *cost* atau nilai dari tiap-tiap rute yang ada. Algoritma ini akan melakukan ekspansi pada simpul sesuai dengan *cost path* dari simpul pertama. UCS digunakan untuk mencari solusi yang optimal dan akan memprioritaskan *cost path* dengan biaya yang minimum.

Langkah-langkah implementasi pada kasus Word Ladder adalah sebagai berikut:

1. Kata-kata akan dianggap seperti simpul pada suatu graf/pohon
2. Kata pertama (*start word*) akan dianggap sebagai *root node* (simpul akar)
3. Kata tujuan (*end word*) akan dianggap sebagai *goal mode* (simpul tujuan)
4. Akan ada dua buah *list* yang akan dicatat, yaitu pertama adalah *list* berisi kumpulan simpul yang sudah diekspansi dan yang kedua merupakan *list* simpul hidup
5. Jika sebuah simpul sudah di ekspansi, dan ternyata simpul tersebut merupakan anak (*successor*) dari simpul lain, maka simpul tersebut tidak akan dimasukkan ke dalam simpul hidup
6. Simpul hidup akan diurutkan berdasarkan *cost* paling rendah, pada kasus Word Ladder *cost* yang diperhitungkan merupakan jumlah huruf yang diganti dari simpul akar sampai simpul tersebut.
7. Simpul dengan *cost* paling rendah akan diekspansi terlebih dahulu, sehingga *list* akan berbentuk seperti *priority queue*
8. Jika simpul merupakan *top of queue* pada simpul hidup, maka simpul itu akan di ekspansi, dengan cara mencari anak-anak dari

simpul tersebut. Pada kasus Word Ladder, anak-anak simpul tersebut merupakan kata-kata yang bisa dibuat dengan cara mengganti satu huruf. Program akan melakukan traversal untuk tiap huruf pada kata, dan pada tiap iterasi akan dilakukan penggantian satu huruf sehingga membentuk kata-kata baru.

9. Sebagai ilustrasi, misal terdapat simpul dengan kata ‘hit’, maka program akan pertama mencari semua kata yang ada jika ,’h’ diganti , lalu jika ‘i’ diganti, lalu jika ‘t’ diganti, kumpulan dari kata-kata tersebut akan menjadi anak-anak dari ‘hit’ dan akan dimasukkan ke dalam simpul hidup untuk di expansi.
10. Jika simpul yang diekspansi merupakan simpul tujuan, maka program akan berhenti dan mengembalikan rute yang ada.
11. Program akan terus melakukan ekspansi hingga simpul hidup telah diekspansi semua, jika simpul hidup sudah diekspansi semua dan simpul tujuan masih belum ditemui, maka rute tersebut tidak dapat ditemukan.

1.2. Algoritma GBFS (Greedy Best First Search)

Algoritma GBFS (*Greedy Best First Search*) merupakan algoritma pencarian *informed search*. Artinya, algoritma ini menggunakan pengetahuan terhadap masalah yang ada. Algoritma GBFS tidak memperhitungkan *cost*, melainkan memperkirakan sebuah *heuristic cost*, yaitu sebuah nilai perkiraan dari simpul menuju simpul tujuan.

GBFS akan memprioritaskan simpul yang terlihat lebih menjanjikan untuk menuju simpul tujuan. GBFS akan mendahulukan simpul dengan *heuristic cost* yang lebih minimal. *Heuristic cost* dihitung dengan fungsi heuristic.

Langkah-langkah implementasi pada kasus Word Ladder adalah sebagai berikut:

1. Kata-kata akan dianggap seperti simpul pada suatu graf/pohon
2. Kata pertama (*start word*) akan dianggap sebagai *root node* (simpul akar)

3. Kata tujuan (*end word*) akan dianggap sebagai *goal mode* (simpul tujuan)
4. Akan ada dua buah *list* yang akan dicatat, yaitu pertama adalah *list* berisi kumpulan simpul yang sudah diekspansi dan yang kedua merupakan *list* simpul hidup
5. Jika sebuah simpul sudah di ekspansi, dan ternyata simpul tersebut merupakan anak (*successor*) dari simpul lain, maka simpul tersebut tidak akan dimasukkan ke dalam simpul hidup
6. Simpul hidup akan diurutkan berdasarkan *heuristic cost* paling rendah, pada kasus Word Ladder, fungsi heuristik yang akan diterapkan adalah jumlah huruf dari suatu simpul yang harus diubah sehingga menjadi sama seperti simpul tujuan.
7. Simpul dengan *heuristic cost* paling rendah akan diekspansi terlebih dahulu, sehingga *list* akan berbentuk seperti *priority queue*
8. Jika simpul merupakan *top of queue* pada simpul hidup, maka simpul itu akan di ekspansi, dengan cara mencari anak-anak dari simpul tersebut. Pada kasus World Ladder, fungsi heuristik yang akan melakukan traversal pada tiap huruf dan memeriksa apakah huruf sama dengan huruf pada simpul tujuan.
9. Sebagai ilustrasi, misal terdapat simpul dengan kata ‘short’ dan simpul tujuan merupakan kata ‘taken’ . Agar ‘short’ menjadi ‘taken’ , huruf ‘s’ harus diubah menjadi ‘t’, ‘h’ menjadi ‘a’, dan seterusnya. Fungsi akan membandingkan dua karakter pada posisi tertentu dan menambahkan *heuristic cost* jika berbeda. Untuk ‘short’ ke ‘taken’ , nilai *heuristic cost*-nya adalah 5, karena tidak ada huruf yang sama pada posisi yang sama.
10. Jika simpul yang diekspansi merupakan simpul tujuan, maka program akan berhenti dan mengembalikan rute yang ada.
11. Program akan terus melakukan ekspansi hingga simpul hidup telah diekspansi semua, jika simpul hidup sudah diekspansi semua dan simpul tujuan masih belum ditemui, maka rute tersebut tidak dapat ditemukan.

1.3. Algoritma A* (A Star)

Algoritma A* (A Star) merupakan algoritma pencarian *informed search*. Artinya, algoritma ini menggunakan pengetahuan terhadap masalah yang ada. Algoritma A* akan mengevaluasi tiap simpul dengan mempertimbangkan *cost* dan *heuristic cost*. Algoritma akan mempertimbangkan *cost* dari simpul akar hingga simpul dan juga *heuristic cost* dari simpul hingga simpul tujuan.

Algoritma A* akan memprioritaskan nilai *cost* dan *heuristic* yang paling minimal. A* biasa digunakan untuk mendapatkan solusi yang optimal dan juga efisien.

Langkah-langkah implementasi pada kasus Word Ladder adalah sebagai berikut:

1. Kata-kata akan dianggap seperti simpul pada suatu graf/pohon
2. Kata pertama (*start word*) akan dianggap sebagai *root node* (simpul akar)
3. Kata tujuan (*end word*) akan dianggap sebagai *goal mode* (simpul tujuan)
4. Akan ada dua buah *list* yang akan dicatat, yaitu pertama adalah *list* berisi kumpulan simpul yang sudah diekspansi dan yang kedua merupakan *list* simpul hidup
5. Jika sebuah simpul sudah di ekspansi, dan ternyata simpul tersebut merupakan anak (*successor*) dari simpul lain, maka simpul tersebut tidak akan dimasukkan ke dalam simpul hidup
6. Simpul hidup akan diurutkan berdasarkan hasil dari fungsi evaluasi $f(n)$. Fungsi evaluasi merupakan jumlah dari *cost* ($g(n)$) dan *heuristic cost* ($h(n)$).
7. Simpul dengan $f(n)$ paling rendah akan diekspansi terlebih dahulu, sehingga *list* akan berbentuk seperti *priority queue*
8. Jika simpul merupakan *top of queue* pada simpul hidup, maka simpul itu akan di ekspansi, dengan cara mencari anak-anak dari simpul tersebut. Fungsi akan mengevaluasi $g(n)$ dan $h(n)$ pada simpul tersebut

9. Pada kasus Word Ladder $g(n)$ yang diperhitungkan merupakan jumlah huruf yang diganti dari simpul akar sampai simpul tersebut. Program akan melakukan traversal untuk tiap huruf pada kata, dan pada tiap iterasi akan dilakukan penggantian satu huruf sehingga membentuk kata-kata baru.
10. Pada kasus Word Ladder, $h(n)$ yang diperhitungkan adalah jumlah huruf dari suatu simpul yang harus diubah sehingga menjadi sama seperti simpul tujuan. Program akan melakukan traversal pada tiap huruf dan memeriksa apakah huruf sama dengan huruf pada simpul tujuan.
11. Sebagai ilustrasi, misal terdapat simpul dengan kata ‘hit’, simpul tujuan merupakan kata ‘log’, simpul akar ‘hot’. *Cost* ($g(n)$) untuk ‘hit’ adalah 1 karena memerlukan 1 pergantian kata, yaitu dari ‘hot’ ke ‘hit’ (mengganti ‘o’ menjadi ‘i’). Sedangkan *heuristic cost* ($h(n)$) untuk ‘hit’ adalah 3, karena terdapat 3 huruf yang berbeda dari ‘hit’ dan ‘log’ pada posisi yang sama. Sehingga hasil dari fungsi evaluasi simpul ‘hit’ adalah 4 ($f(n) = g(n) + h(n)$).
12. Jika simpul yang diekspansi merupakan simpul tujuan, maka program akan berhenti dan mengembalikan rute yang ada.
13. Program akan terus melakukan ekspansi hingga simpul hidup telah diekspansi semua, jika simpul hidup sudah diekspansi semua dan simpul tujuan masih belum ditemui, maka rute tersebut tidak dapat ditemukan.

1.4. Analisis Lanjutan

Berikut merupakan analisis lanjutan, mengenai algoritma A*, UCS, dan GBFS:

1. $f(n)$ pada $g(n)$ tidaklah sama pada algoritma A*, $f(n)$ merupakan fungsi untuk heuristik yang digunakan untuk mengevaluasi $g(n)$ dan $h(n)$. Dalam konteks ini $g(n)$ merupakan jarak yang diperlukan sampai simpul n , $h(n)$ adalah jarak perkiraan yang diperlukan untuk simpul n hingga simpul tujuan. $f(n)$ terkomposisi oleh $g(n)$ dan $h(n)$

2. Heuristik yang digunakan pada A^* *admissible*, karena sesuai dengan salindia kuliah, fungsi heuristik yang *admissible* adalah merupakan *lower bound* atau minimum. $f(n)$ akan mengestimasi jumlah *cost* terkecil. Jika fungsi heuristik selalu *underestimate cost* yang asli maka nilai heuristik akan selalu lebih kecil dari nilai nilai asli, sehingga solusi yang ditemukan juga optimal. Heuristik yang digunakan pada sistem ini adalah mencari berapa banyak huruf yang harus diganti hingga menjadi simpul tujuan, hal ini bersifat *admissible* karena fungsi selalu mencari batas bawah.
3. Pada kasus Word Ladder, UCS dan BFS akan sama karena *cost* untuk setiap perubahan simpul adalah seragam, sehingga *cost* dapat dianggap seperti kedalaman pada BFS. Karena sifat UCS yang selalu mulai dari *cost* yang paling rendah, maka UCS akan mengekspansi mulai dari simpul-simpul dengan *cost* 1 lalu 2 ,3, dan seterusnya. Hal ini sama dengan BFS yang mencari pada kedalaman 1 lalu 2, dan seterusnya.
4. Secara teoritis, A^* lebih efisien dibandingkan dengan UCS. Meskipun keduanya menghasilkan hasil yang optimal, jumlah simpul yang dikunjungi A^* akan lebih sedikit dibandingkan dengan UCS. Hal ini terjadi karena A^* merupakan *informed search*, sedangkan UCS merupakan *blind search*. Artinya, A^* memiliki informasi tambahan, yaitu heuristik/perkiraan untuk mencapai simpul tujuan dengan lebih cepat. UCS hanya memiliki *cost* yang diperlukan untuk menuju simpul n, sedangkan A^* juga memiliki informasi tersebut ditambah dengan perkiraan *cost* untuk dari simpul n ke simpul tujuan. A^* akan mencapai simpul tujuan dengan lebih efisien karena sudah memiliki perkiraan untuk menuju simpul tujuan.
5. Secara teoritis, GBFS tidak akan memberikan solusi yang optimal, karena hanya mengandalkan estimasi menuju simpul tujuan. Sebagai contoh terdapat simpul n dengan *cost* 1 untuk menuju tujuan, namun *cost* untuk menuju simpul n adalah 5. Dibandingkan dengan suatu

simpul m dengan $cost$ 2 untuk menuju tujuan , namun $cost$ untuk menuju simpul m adalah 2. GBFS akan lebih memilih untuk menuju simpul n dibanding m (meskipun m lebih optimal), karena n lebih dekat menuju simpul tujuan.

BAB II

Penjelasan Struktur Kode

2.1. Abstraksi Class dan Method yang dibuat

Untuk program berbasis GUI akan dijalankan melalui WordLadderGUI.java, sedangkan untuk program berbasis CLI akan dijalankan melalui WordLadderCLI.java.

Pada program berbasis CLI, terdapat WordLadder.java yang merupakan kelas untuk membuat suatu instansiasi permainan WordLadder. Penerimaan input dan cara mengeksekusi program akan dilakukan pada WordLadder.java

Untuk program berbasis GUI, penerimaan input akan dilakukan di WordLadderGUI.java, penampilan hasil akan dilakukan pada ResultDisplayPanel.java (akan diinstansiasi melalui WordLadderGUI.java). Terdapat kelas tambahan untuk membuat komponen-komponen yang diperlukan, seperti RoundedButton.java, RoundedLabel.java dan RoundedTextField.java.

Untuk algoritma pencarian akan dipisahkan menjadi 3 kelas, yaitu AStar.java, GBFS.java, dan UCS.java. Ketiganya merupakan turunan dari kelas abstrak Search.java. Hal ini dilakukan untuk mendukung polimorfisme saat pemanggilan fungsi algoritma.

Terdapat kelas pembantu yang digunakan untuk membuat struktur data dan juga memuat kamus yang akan dipakai. Node.java merupakan kelas yang dibuat untuk membuat struktur Node yang akan digunakan untuk pencarian. Kelas Result.java merupakan kelas yang dibuat untuk membuat struktur Result (solusi dan jumlah node yang dikunjungi). Dictionary.java berfungsi untuk memuat kamus yang sudah dipetakan menjadi (kata : turunan kata). MapDictionary.java berfungsi untuk membaca kamus biasa dan memetakannya pada file baru yang akan digunakan untuk pencarian yang lebih optimal.

Struktur data yang digunakan untuk kamus merupakan sebuah Map<String, List<String>>, yaitu mapping antara kata dan turunan anaknya. Hal ini dilakukan untuk optimisasi karena tipe data *hash map* memiliki lookup yang cepat.

2.2. Detail Class dan Method

| WordLadderCLI.java | |
|--|---------------------------|
| Method | |
| Nama | Fungsi |
| public static void main(String[] args) | Menjalankan program utama |

| WordLadderGUI.java | |
|---|--|
| Attribut | |
| Nama | Fungsi |
| private JLabel startLabel, endLabel, algorithmLabel, titleLabel; | untuk menampilkan tulisan label |
| private JTextField startField, endField; | untuk menampilkan kotak masukkan untuk kata |
| private JComboBox<String> algorithmComboBox; | untuk menampilkan pilihan algoritma |
| private JButton submitButton; | untuk menampilkan tombol |
| private src.utils.Dictionary dictionary; | untuk menyimpan kamus |
| public static Font customFont; | untuk menyimpan font |
| public static String[] colpal = {"#6ad09d", "#e27eab", "#f9d8e0", "#eb954a", "#ffeab4"}; | kumpulan hexcode warna yang digunakan |
| Method | |
| Nama | Fungsi |
| public WordLadderGUI() | untuk menginisiasi gui |
| private class handleButtonClick implements ActionListener { @Override public void actionPerformed(ActionEvent e) { private boolean isValidInput(String input) | untuk mengeksekusi pencarian saat tombol diklik (dikemas dalam kelas baru) |

| | |
|---|-----------------------------|
| private void showResultPopup(List<String> solutions, long executionTime ,String startWord, String endWord, int nodesVisited, long memory) | untuk menampilkan hasil |
| private Font loadCustomFont(String fontName) throws IOException, FontFormatException | untuk memuat font dari file |
| public static void main(String[] args) | untuk menjalankan program |

| WordLadder.java | |
|--|---|
| Attribut | |
| Nama | Fungsi |
| private int algorithmID; | untuk menyimpan pilihan algoritma |
| private String startWord; | untuk menyimpan kata awal |
| private String endWord; | untuk menyimpan kata akhir |
| private long executionTime; | untuk menyimpan waktu eksekusi |
| private long memoryUsed; | untuk menyimpan memori yang dipakai |
| private Dictionary dictionary; | untuk menyimpan kamus |
| private Result result; | untuk menyimpan hasil |
| Method | |
| Nama | Fungsi |
| public WordLadder() | untuk menerima input dan menginisiasi game |
| public void printInfo() | untuk menampilkan hasil |
| private int getValidAlgorithm(Scanner scanner) | untuk mendapatkan masukkan pilihan algoritma yang valid |
| private String getValidWord(Scanner scanner, String prompt) | untuk mendapatkan masukkan kata yang valid |
| public void run() | untuk mencari hasil |

| Search.java | |
|---|--|
| Method | |
| Nama | Fungsi |
| public abstract Result findSolution(String startWord, String endWord, Dictionary dictionary); | method abstrak untuk mencari solusi |
| protected List<String> getPath(Node node) | untuk mengubah dari List<Node> menjadi List<String> |

| A*.java | |
|---|--|
| Method | |
| Nama | Fungsi |
| public Result findSolution(String startWord, String endWord, Dictionary dictionary) | untuk mencari solusi menggunakan algoritma A* |
| private int getHeuristic(String startWord, String endWord) | untuk mencari <i>heuristic cost</i> |

| UCS.java | |
|---|---|
| Method | |
| Nama | Fungsi |
| public Result findSolution(String startWord, String endWord, Dictionary dictionary) | untuk mencari solusi menggunakan algoritma UCS |

| GBFS.java | |
|-----------|--------|
| Method | |
| Nama | Fungsi |

| | |
|---|---|
| public Result findSolution(String startWord, String endWord, Dictionary dictionary) | untuk mencari solusi menggunakan algoritma GBFS |
| private int getHeuristic(String startWord, String endWord) | untuk mencari <i>heuristic cost</i> |

| Node.java | |
|--|---|
| Attribut | |
| Nama | Fungsi |
| private String word; | untuk menyimpan kata |
| private Node parent; | untuk menyimpan predesor |
| private int price; | untuk menyimpan cost yang dibutuhkan (tipe tergantung algoritma yang digunakan) |
| Method | |
| Nama | Fungsi |
| public Node(String word, Node parent, int price) | konstruktor |
| public int getPrice() | mengembalikan price |
| public String getWord() | mengembalikan kata |
| public Node getParent() | mengembalikan predecessor |

| Result.java | |
|--------------------------------|---------------------------------------|
| Attribut | |
| Nama | Fungsi |
| private List<String> solution; | menyimpan hasil solusi |
| private int numOfVisitedNodes; | menyimpan jumlah node yang dikunjungi |
| Method | |
| Nama | Fungsi |

| | |
|---|---|
| public Result(List<String> solution, int numOfVisitedNodes) | konstruktor |
| public List<String> getSolution() | mengembalikan solusi |
| public int getNumOfVisitedNodes() | mengembalikan jumlah node yang dikunjungi |

| MapDictionary.java | |
|---|---|
| Attribut | |
| Nama | Fungsi |
| private Set<String> dictionaryWords; | dictionary biasa yang belum dipetakan |
| Method | |
| Nama | Fungsi |
| public MapDictionary() | konstruktor |
| public void loadDictionary(String dictionaryFilePath) | untuk memuat kamus biasa dan menjadikannya attribut |
| public void mapAndWrite(String outputPath) | untuk menulis ke file baru berisi kata dan anak-anaknya |
| private List<String> getChild(String word) | untuk mencari anak-anak dari sebuah kata |

| Dictionary.java | |
|--|-----------------------------------|
| Attribut | |
| Nama | Fungsi |
| private Map<String, List<String>> dictionary | dictionary yang sudah dipetakan |
| Method | |
| Nama | Fungsi |
| public Dictionary(String filePath) | konstruktor untuk menginstansiasi |

| | |
|---|--|
| | kamus |
| public Map<String, List<String>> getDictionary() | untuk mengembalikan kamus |
| public void loadDictionary(String filePath) | untuk memuat kamus yang sudah dipetakan dari file txt |
| public boolean validWord(String word) | untuk memvalidasi sebuah kata ada di dalam kamus atau tidak |

| ResultDisplayPanel.java | |
|--|--|
| Attribut | |
| Nama | Fungsi |
| private List<String> solutions; | untuk menyimpan hasil |
| private String endWord; | untuk menyimpan kata awal |
| private String startWord; | untuk menyimpan kata akhir |
| private long executionTime; | untuk menyimpan waktu eksekusi |
| private long memoryUsed; | untuk menyimpan memori yang dipakai |
| private int nodesVisited; | untuk menyimpan jumlah node yang dikunjungi |
| Method | |
| Nama | Fungsi |
| public ResultDisplayPanel(List<String> solutions, long executionTime ,String startWord, String endWord, int nodesVisited, long memory) | untuk membuat tampilan hasil |
| private JPanel createInfoPanel() | untuk membuat panel tampilan info |
| private JPanel createResultPanel() | untuk membuat panel tampilan rute |

| RoundedButton.java | |
|---|---|
| Attribut | |
| Nama | Fungsi |
| private int cornerRadius; | untuk menyimpan radius ujung |
| private Color defaultColor; | untuk menyimpan warna default |
| private Color hoverColor; | untuk menyimpan warna saat hover |
| Method | |
| Nama | Fungsi |
| public RoundedButton(String text, Color defaultColor, Color hoverColor, int cornerRadius) | konstruktor |
| private void init() | sebagai inisialisasi dan mengganti konfigurasi awal JButton dan juga menambahkan listener untuk hover |
| public void addNotify() | override untuk mengganti warna background |
| protected void paintComponent(Graphics g) | override untuk menggambar bentuk rounded |

| RoundedLabel.java | |
|--|--|
| Attribut | |
| Nama | Fungsi |
| private int cornerRadius; | untuk menyimpan radius ujung |
| private Color backgroundColor; | untuk menyimpan warna background |
| Method | |
| Nama | Fungsi |
| public RoundedLabel(String text, int alignment, int cornerRadius, Color backgroundColor) | konstruktor |
| protected void paintComponent(Graphics g) | override untuk menggambar bentuk rounded |

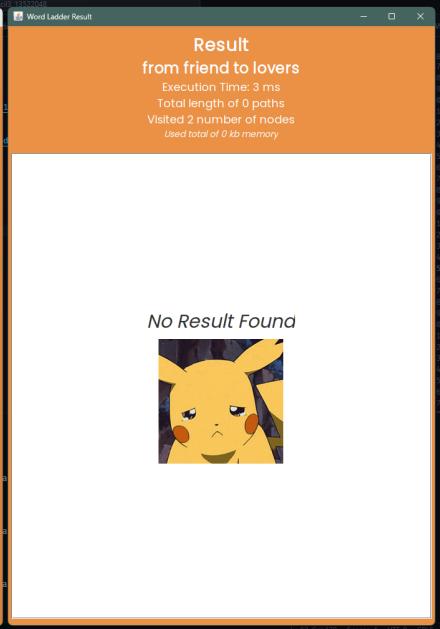
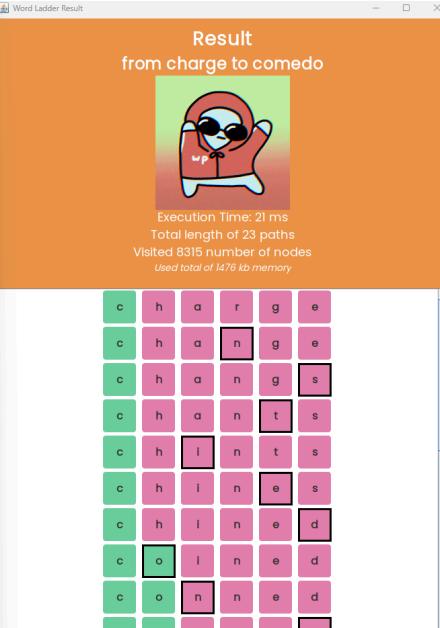
| RoundedTextField.java | |
|--|--|
| Attribut | |
| Nama | Fungsi |
| private int cornerRadius; | untuk menyimpan radius ujung |
| Method | |
| Nama | Fungsi |
| public RoundedTextField(int columns, int cornerRadius) | konstruktor |
| protected void paintComponent(Graphics g) | override untuk menggambar bentuk rounded |

BAB III

Hasil Implementasi

3.1. Tangkapan Layar Algoritma UCS (Uniform Cost Search)

Tabel 1. Hasil implementasi menggunakan algoritma UCS

| | | | |
|---|--|-------------------------|--------------------------|
|  |  | Kata Awal : “friend” | Kata Akhir : “lovers” |
|  |  | Kata Awal : “charge” | Kata Akhir : “comedo” |



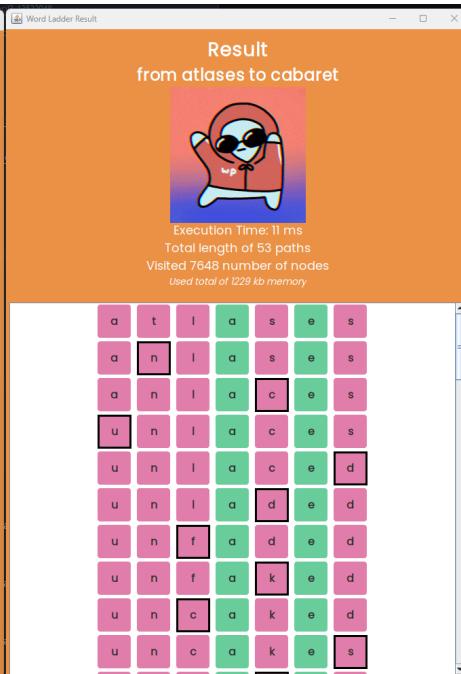
Word Ladder

Start Word: atlasses

End Word: cabaret

Algorithm: UCS

Submit



Result
from atlases to cabaret

Execution Time: 11 ms
Total length of 53 paths
Visited 7648 number of nodes
Used total of 1229 kb memory

| | | | | | | |
|---|---|---|---|---|---|---|
| a | t | i | a | s | e | s |
| a | n | i | a | s | e | s |
| a | n | i | a | c | e | s |
| u | n | i | a | c | e | s |
| u | n | i | a | c | e | d |
| u | n | i | a | d | e | d |
| u | n | f | a | d | e | d |
| u | n | f | a | k | e | d |
| u | n | c | a | k | e | d |
| u | n | c | a | k | e | s |



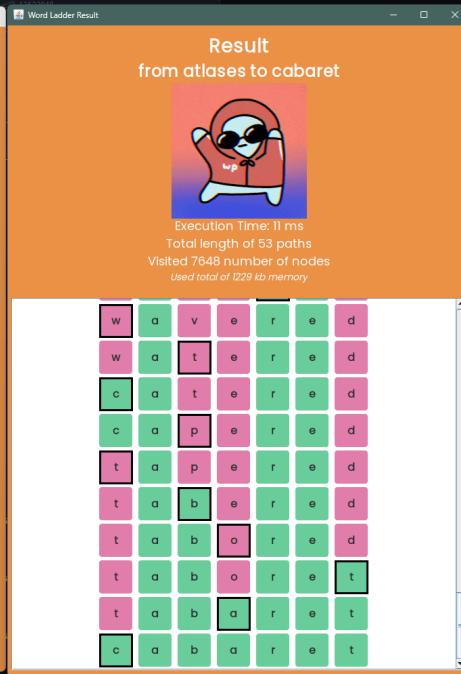
Word Ladder

Start Word: atlasses

End Word: cabaret

Algorithm: UCS

Submit



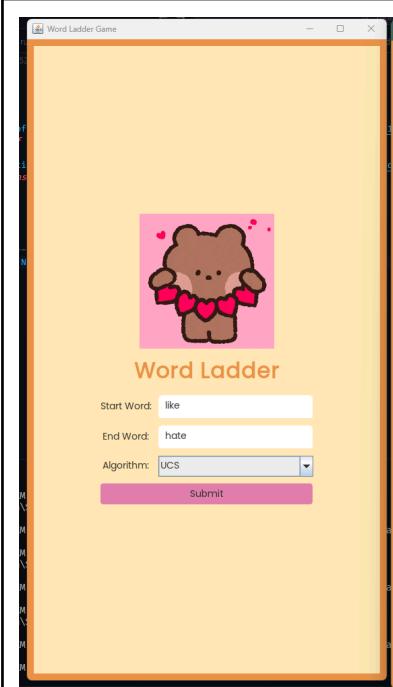
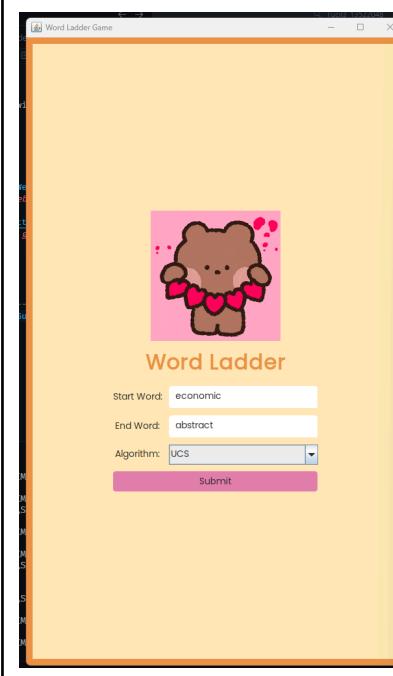
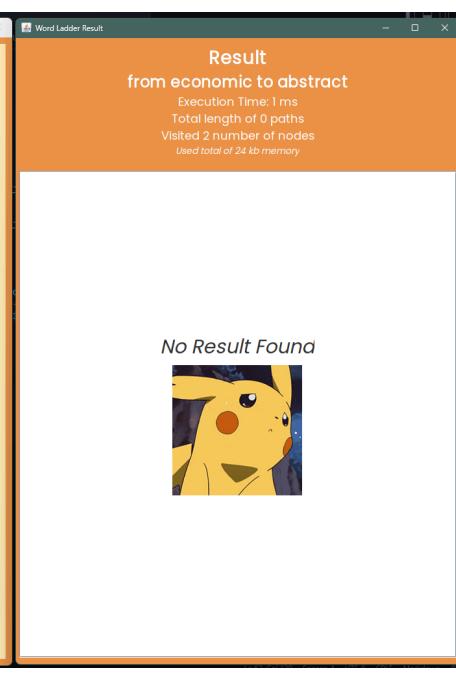
Result
from atlases to cabaret

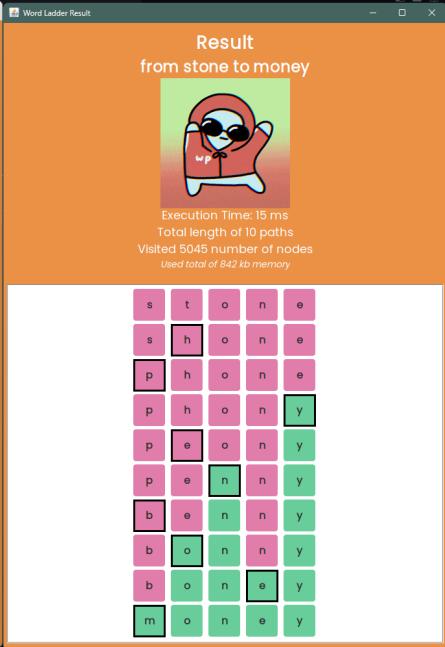
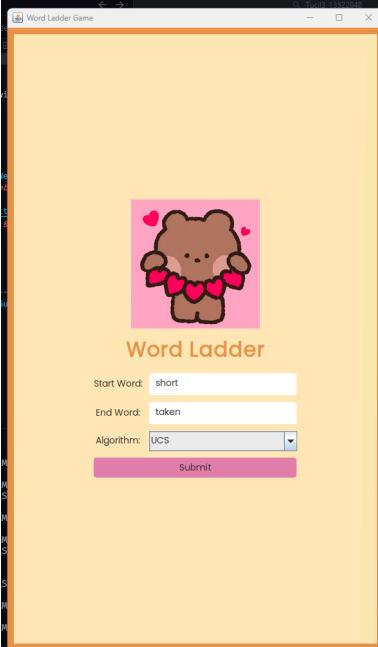
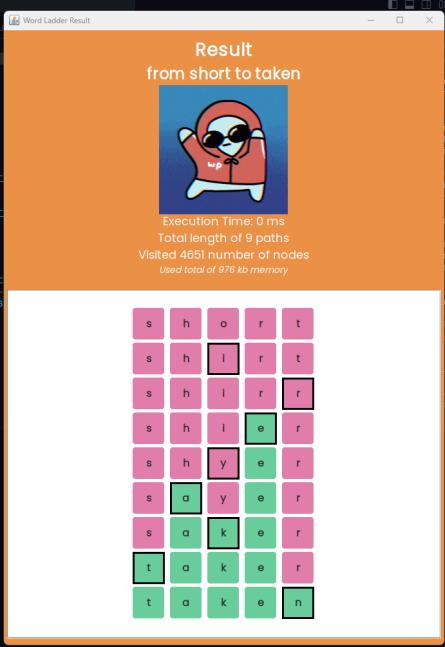
Execution Time: 11 ms
Total length of 53 paths
Visited 7648 number of nodes
Used total of 1229 kb memory

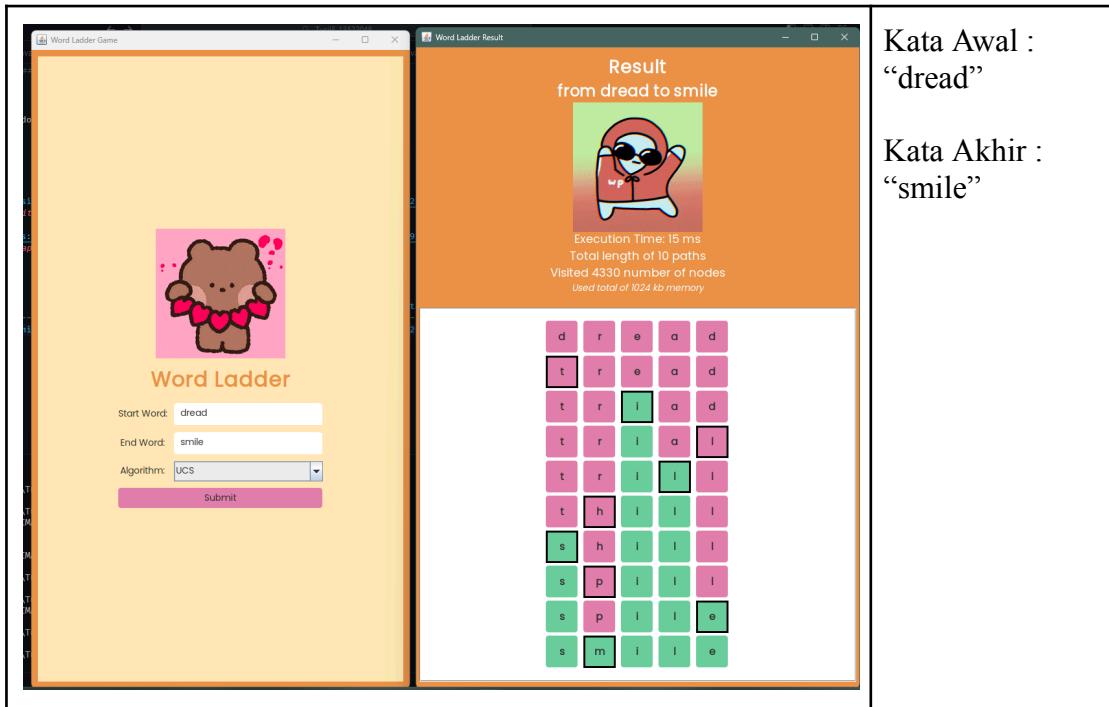
| | | | | | | |
|---|---|---|---|---|---|---|
| w | a | v | e | r | e | d |
| w | a | t | e | r | e | d |
| c | a | t | e | r | e | d |
| c | a | p | e | r | e | d |
| t | a | p | e | r | e | d |
| t | a | b | e | r | e | d |
| t | a | b | o | r | e | d |
| t | a | b | o | r | e | t |
| c | a | b | a | r | e | t |

Kata Awal :
“atlases”

Kata Akhir :
“cabaret”

| | | | | | | | | | | | | | | | | | | |
|--|--|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
|  |  <p>Result from like to hate</p> <p>Execution Time: 1 ms Total length of 4 paths Visited 498 number of nodes Used total of 143 kb memory</p> <table border="1" data-bbox="826 651 977 797"> <tbody> <tr><td>I</td><td>I</td><td>k</td><td>e</td></tr> <tr><td>I</td><td>I</td><td>t</td><td>e</td></tr> <tr><td>I</td><td>a</td><td>t</td><td>e</td></tr> <tr><td>h</td><td>a</td><td>t</td><td>e</td></tr> </tbody> </table> | I | I | k | e | I | I | t | e | I | a | t | e | h | a | t | e | <p>Kata Awal : “like”</p> <p>Kata Akhir : “hate”</p> |
| I | I | k | e | | | | | | | | | | | | | | | |
| I | I | t | e | | | | | | | | | | | | | | | |
| I | a | t | e | | | | | | | | | | | | | | | |
| h | a | t | e | | | | | | | | | | | | | | | |
|  |  <p>Result from economic to abstract</p> <p>Execution Time: 1 ms Total length of 0 paths Visited 2 number of nodes Used total of 24 kb memory</p> <p>No Result Found</p>  | <p>Kata Awal : “economic”</p> <p>Kata Akhir : “abstract”</p> | | | | | | | | | | | | | | | | |

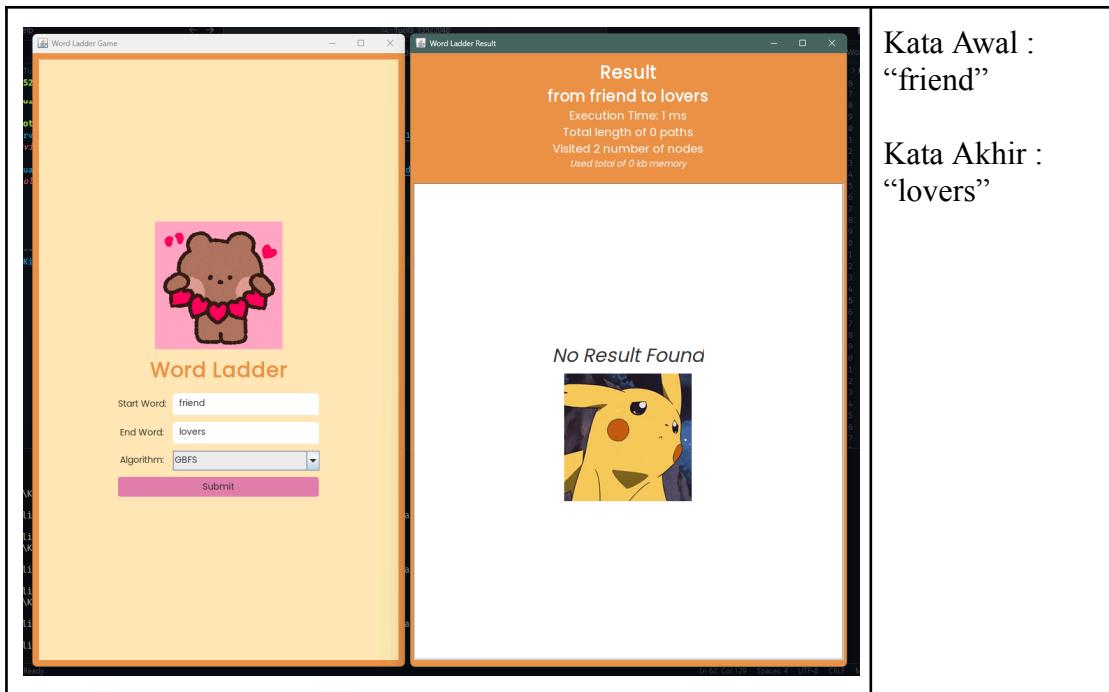
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|--|
|  <p>Word Ladder</p> <p>Start Word: stone</p> <p>End Word: MONEY</p> <p>Algorithm: UCS</p> <p>Submit</p> |  <p>Result from stone to money</p> <p>Execution Time: 15 ms Total length of 10 paths Visited 5045 number of nodes Used total of 842 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>s</td><td>t</td><td>o</td><td>n</td><td>e</td></tr> <tr><td>s</td><td>h</td><td>o</td><td>n</td><td>e</td></tr> <tr><td>p</td><td>h</td><td>o</td><td>n</td><td>e</td></tr> <tr><td>p</td><td>h</td><td>o</td><td>n</td><td>y</td></tr> <tr><td>p</td><td>e</td><td>o</td><td>n</td><td>y</td></tr> <tr><td>p</td><td>e</td><td>n</td><td>n</td><td>y</td></tr> <tr><td>b</td><td>e</td><td>n</td><td>n</td><td>y</td></tr> <tr><td>b</td><td>o</td><td>n</td><td>n</td><td>y</td></tr> <tr><td>b</td><td>o</td><td>n</td><td>e</td><td>y</td></tr> <tr><td>m</td><td>o</td><td>n</td><td>e</td><td>y</td></tr> </tbody> </table> | s | t | o | n | e | s | h | o | n | e | p | h | o | n | e | p | h | o | n | y | p | e | o | n | y | p | e | n | n | y | b | e | n | n | y | b | o | n | n | y | b | o | n | e | y | m | o | n | e | y | <p>Kata Awal : “stone”</p> <p>Kata Akhir : “money”</p> |
| s | t | o | n | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | o | n | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| p | h | o | n | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| p | h | o | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| p | e | o | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| p | e | n | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| b | e | n | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| b | o | n | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| b | o | n | e | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| m | o | n | e | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  <p>Word Ladder</p> <p>Start Word: short</p> <p>End Word: taken</p> <p>Algorithm: UCS</p> <p>Submit</p> |  <p>Result from short to taken</p> <p>Execution Time: 0 ms Total length of 9 paths Visited 4651 number of nodes Used total of 976 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>s</td><td>h</td><td>o</td><td>r</td><td>t</td></tr> <tr><td>s</td><td>h</td><td>i</td><td>r</td><td>t</td></tr> <tr><td>s</td><td>h</td><td>i</td><td>r</td><td>r</td></tr> <tr><td>s</td><td>h</td><td>i</td><td>e</td><td>r</td></tr> <tr><td>s</td><td>h</td><td>y</td><td>e</td><td>r</td></tr> <tr><td>s</td><td>a</td><td>y</td><td>e</td><td>r</td></tr> <tr><td>s</td><td>a</td><td>k</td><td>e</td><td>r</td></tr> <tr><td>t</td><td>a</td><td>k</td><td>e</td><td>r</td></tr> <tr><td>t</td><td>a</td><td>k</td><td>e</td><td>n</td></tr> </tbody> </table> | s | h | o | r | t | s | h | i | r | t | s | h | i | r | r | s | h | i | e | r | s | h | y | e | r | s | a | y | e | r | s | a | k | e | r | t | a | k | e | r | t | a | k | e | n | <p>Kata Awal : “short”</p> <p>Kata Akhir : “taken”</p> | | | | | |
| s | h | o | r | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | i | r | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | i | r | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | i | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | y | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | a | y | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | a | k | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | k | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | k | e | n | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |



Kata Awal :
“dread”

Kata Akhir :
“smile”

3.2.Tangkapan Layar Algoritma GBFS (Greedy Best First Search)



Kata Awal :
“friend”

Kata Akhir :
“lovers”



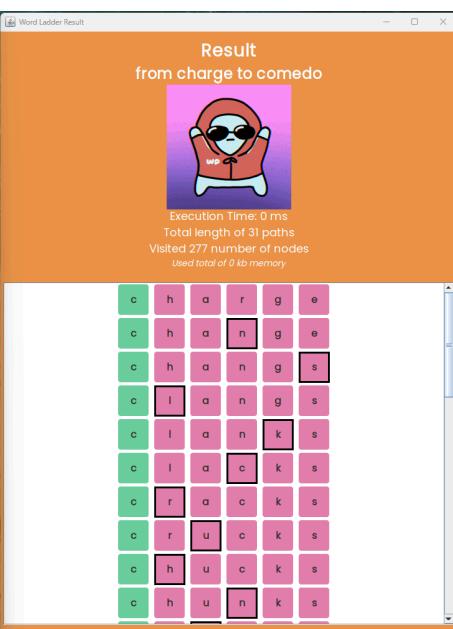
Word Ladder

Start Word: charge

End Word: comedo

Algorithm: GBFS

Submit



Result
from charge to comedo

Execution Time: 0 ms
Total length of 31 paths
Visited 277 number of nodes
Used total of 0 kb memory

| | | | | | |
|---|---|---|---|---|---|
| c | h | a | r | g | e |
| c | h | a | n | g | e |
| c | h | a | n | g | s |
| c | i | a | n | g | s |
| c | i | a | n | k | s |
| c | i | a | c | k | s |
| c | r | a | c | k | s |
| c | r | u | c | k | s |
| c | h | u | c | k | s |
| c | h | u | n | k | s |



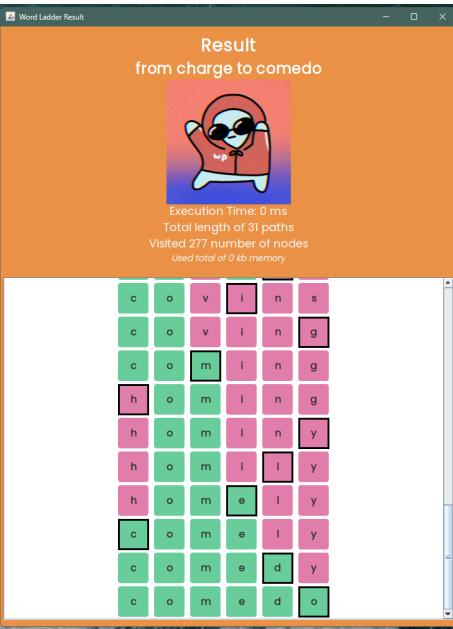
Word Ladder

Start Word: charge

End Word: comedo

Algorithm: GBFS

Submit



Result
from charge to comedo

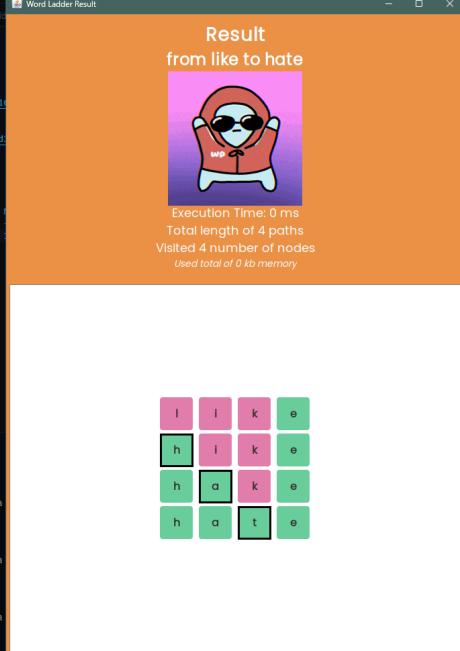
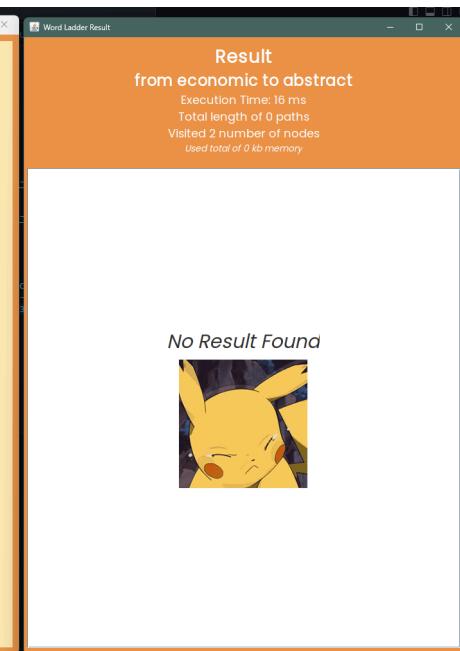
Execution Time: 0 ms
Total length of 31 paths
Visited 277 number of nodes
Used total of 0 kb memory

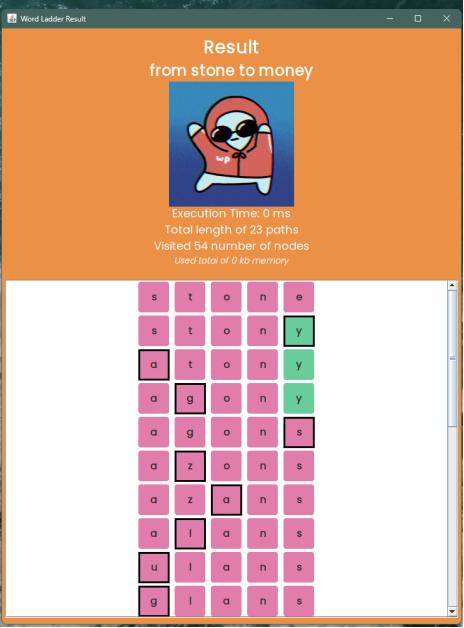
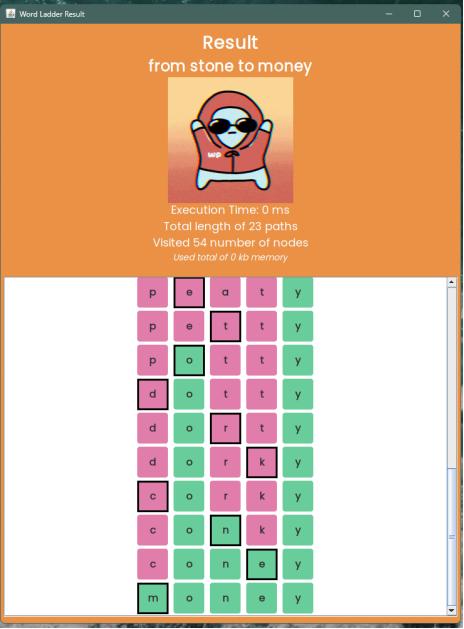
| | | | | | |
|---|---|---|---|---|---|
| c | o | v | i | n | s |
| c | o | v | i | n | g |
| c | o | m | i | n | g |
| h | o | m | i | n | y |
| h | o | m | i | l | y |
| h | o | m | e | l | y |
| c | o | m | e | l | y |
| c | o | m | e | d | y |
| c | o | m | e | d | o |

Kata Awal :
“charge”

Kata Akhir :
“comedo”

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| <p>Word Ladder</p> <p>Start Word: atlasses</p> <p>End Word: cabaret</p> <p>Algorithm: GBFS</p> <p>Submit</p> | <p>Result from atlases to cabaret</p> <p>Execution Time: 9 ms Total length of 154 paths Visited 3078 number of nodes Used total of 1024 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>a</td><td>t</td><td>l</td><td>a</td><td>s</td><td>e</td><td>s</td></tr> <tr><td>a</td><td>n</td><td>l</td><td>a</td><td>s</td><td>e</td><td>s</td></tr> <tr><td>a</td><td>n</td><td>l</td><td>a</td><td>c</td><td>e</td><td>s</td></tr> <tr><td>u</td><td>n</td><td>l</td><td>a</td><td>c</td><td>e</td><td>s</td></tr> <tr><td>u</td><td>n</td><td>l</td><td>a</td><td>d</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>f</td><td>a</td><td>d</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>f</td><td>a</td><td>k</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>b</td><td>a</td><td>k</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>b</td><td>a</td><td>s</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>b</td><td>a</td><td>s</td><td>e</td><td>d</td></tr> </tbody> </table> | a | t | l | a | s | e | s | a | n | l | a | s | e | s | a | n | l | a | c | e | s | u | n | l | a | c | e | s | u | n | l | a | d | e | d | u | n | f | a | d | e | d | u | n | f | a | k | e | d | u | n | b | a | k | e | d | u | n | b | a | s | e | d | u | n | b | a | s | e | d | <p>Kata Awal : “atlases”</p> <p>Kata Akhir : “cabaret”</p> |
| a | t | l | a | s | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | n | l | a | s | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | n | l | a | c | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | l | a | c | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | l | a | d | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | f | a | d | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | f | a | k | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | b | a | k | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | b | a | s | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | b | a | s | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Word Ladder</p> <p>Start Word: atlasses</p> <p>End Word: cabaret</p> <p>Algorithm: GBFS</p> <p>Submit</p> | <p>Result from atlases to cabaret</p> <p>Execution Time: 9 ms Total length of 154 paths Visited 3078 number of nodes Used total of 1024 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>w</td><td>a</td><td>v</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>w</td><td>a</td><td>t</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>c</td><td>a</td><td>t</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>c</td><td>a</td><td>p</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>p</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>o</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>o</td><td>r</td><td>e</td><td>t</td></tr> <tr><td>c</td><td>a</td><td>b</td><td>a</td><td>r</td><td>e</td><td>t</td></tr> </tbody> </table> | w | a | v | e | r | e | d | w | a | t | e | r | e | d | c | a | t | e | r | e | d | c | a | p | e | r | e | d | t | a | p | e | r | e | d | t | a | b | e | r | e | d | t | a | b | o | r | e | d | t | a | b | o | r | e | t | c | a | b | a | r | e | t | | | | | | | | |
| w | a | v | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| w | a | t | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | a | t | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | a | p | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | p | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | o | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | o | r | e | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | a | b | a | r | e | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|--|--|
|   | <p>Kata Awal : “like”</p> <p>Kata Akhir : “hate”</p> |
|   | <p>Kata Awal : “economic”</p> <p>Kata Akhir : “abstract”</p> |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-------------------------------|--------------------------------|
|  <p>Word Ladder</p> <p>Start Word: stone</p> <p>End Word: money</p> <p>Algorithm: GBFS</p> <p>Submit</p> |  <p>Result from stone to money</p> <p>Execution Time: 0 ms Total length of 23 paths Visited 54 number of nodes Used total of 0 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>s</td><td>t</td><td>o</td><td>n</td><td>e</td></tr> <tr><td>s</td><td>t</td><td>o</td><td>n</td><td>y</td></tr> <tr><td>a</td><td>t</td><td>o</td><td>n</td><td>y</td></tr> <tr><td>a</td><td>g</td><td>o</td><td>n</td><td>y</td></tr> <tr><td>a</td><td>g</td><td>o</td><td>n</td><td>s</td></tr> <tr><td>a</td><td>z</td><td>o</td><td>n</td><td>s</td></tr> <tr><td>a</td><td>z</td><td>a</td><td>n</td><td>s</td></tr> <tr><td>a</td><td>i</td><td>a</td><td>n</td><td>s</td></tr> <tr><td>u</td><td>i</td><td>a</td><td>n</td><td>s</td></tr> <tr><td>g</td><td>i</td><td>a</td><td>n</td><td>s</td></tr> </tbody> </table> | s | t | o | n | e | s | t | o | n | y | a | t | o | n | y | a | g | o | n | y | a | g | o | n | s | a | z | o | n | s | a | z | a | n | s | a | i | a | n | s | u | i | a | n | s | g | i | a | n | s | Kata Awal : “stone” | Kata Akhir : “money” |
| s | t | o | n | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | t | o | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | t | o | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | g | o | n | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | g | o | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | z | o | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | z | a | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | i | a | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | i | a | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| g | i | a | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  <p>Word Ladder</p> <p>Start Word: stone</p> <p>End Word: money</p> <p>Algorithm: GBFS</p> <p>Submit</p> |  <p>Result from stone to money</p> <p>Execution Time: 0 ms Total length of 23 paths Visited 54 number of nodes Used total of 0 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>p</td><td>e</td><td>a</td><td>t</td><td>y</td></tr> <tr><td>p</td><td>e</td><td>t</td><td>t</td><td>y</td></tr> <tr><td>p</td><td>o</td><td>t</td><td>t</td><td>y</td></tr> <tr><td>d</td><td>o</td><td>r</td><td>t</td><td>y</td></tr> <tr><td>d</td><td>o</td><td>r</td><td>t</td><td>y</td></tr> <tr><td>d</td><td>o</td><td>r</td><td>k</td><td>y</td></tr> <tr><td>c</td><td>o</td><td>r</td><td>k</td><td>y</td></tr> <tr><td>c</td><td>o</td><td>n</td><td>k</td><td>y</td></tr> <tr><td>c</td><td>o</td><td>n</td><td>e</td><td>y</td></tr> <tr><td>m</td><td>o</td><td>n</td><td>e</td><td>y</td></tr> </tbody> </table> | p | e | a | t | y | p | e | t | t | y | p | o | t | t | y | d | o | r | t | y | d | o | r | t | y | d | o | r | k | y | c | o | r | k | y | c | o | n | k | y | c | o | n | e | y | m | o | n | e | y | | |
| p | e | a | t | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| p | e | t | t | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| p | o | t | t | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| d | o | r | t | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| d | o | r | t | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| d | o | r | k | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | r | k | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | n | k | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | n | e | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| m | o | n | e | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |



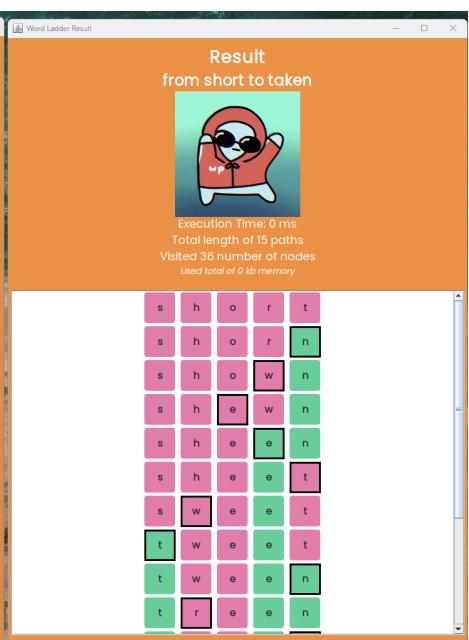
Word Ladder

Start Word: short

End Word: taken

Algorithm: GBFS

Submit



Result
from short to taken

Execution Time: 0 ms
Total length of 15 paths
Visited 36 number of nodes
Used total of 0 kb memory

| | | | | |
|---|---|---|---|---|
| s | h | o | r | t |
| s | h | o | r | n |
| s | h | o | w | n |
| s | h | e | w | n |
| s | h | e | e | n |
| s | h | e | e | t |
| s | w | e | e | t |
| t | w | e | e | t |
| t | w | e | e | n |
| t | r | e | e | t |



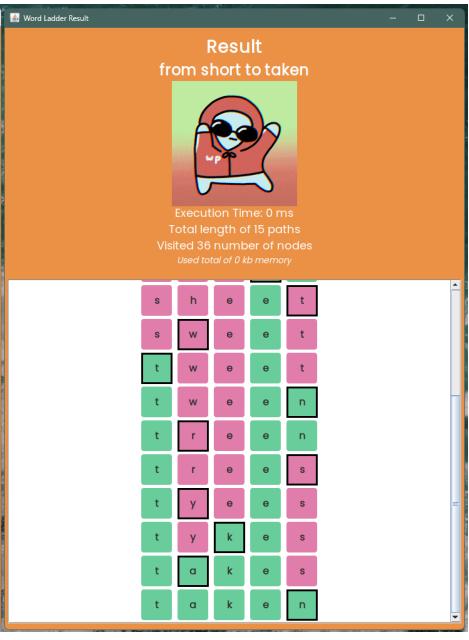
Word Ladder

Start Word: short

End Word: taken

Algorithm: GBFS

Submit



Result
from short to taken

Execution Time: 0 ms
Total length of 15 paths
Visited 36 number of nodes
Used total of 0 kb memory

| | | | | |
|---|---|---|---|---|
| s | h | e | e | t |
| s | w | e | e | t |
| t | w | e | e | t |
| t | r | e | e | n |
| t | r | e | e | s |
| t | y | e | e | s |
| t | y | k | e | s |
| t | a | k | e | s |
| t | a | k | e | n |

Kata Awal :
“short”

Kata Akhir :
“taken”



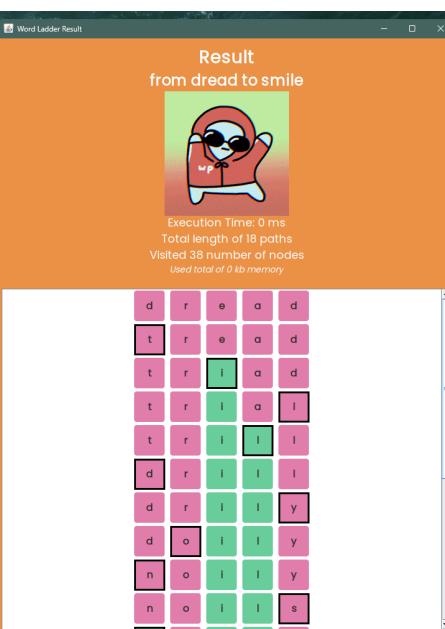
Word Ladder

Start Word: dread

End Word: smile

Algorithm: GBFS

Submit



Result
from dread to smile

Execution Time: 0 ms
Total length of 18 paths
Visited 38 number of nodes
Used total of 0 kb memory

| | | | | |
|---|---|---|---|---|
| d | r | e | a | d |
| t | r | e | a | d |
| t | r | i | a | d |
| t | r | i | i | d |
| d | r | i | i | i |
| d | r | i | i | y |
| d | o | i | i | y |
| n | o | i | i | y |
| n | o | i | i | s |



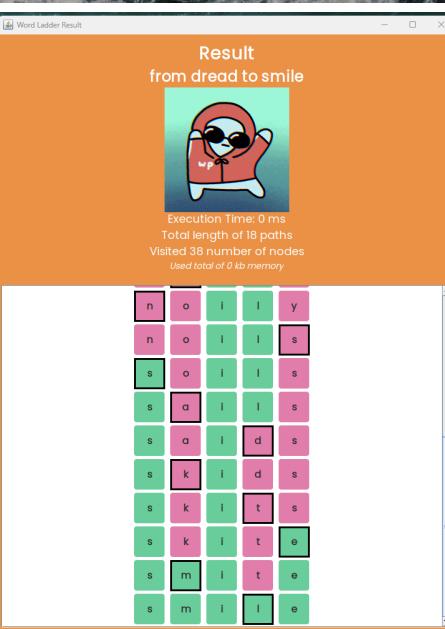
Word Ladder

Start Word: dread

End Word: smile

Algorithm: GBFS

Submit



Result
from dread to smile

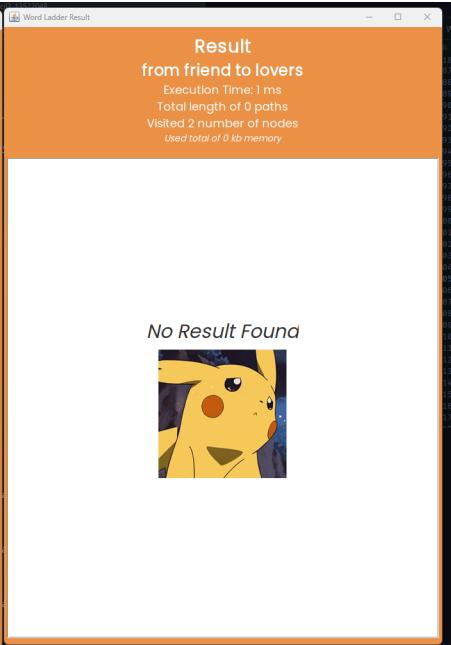
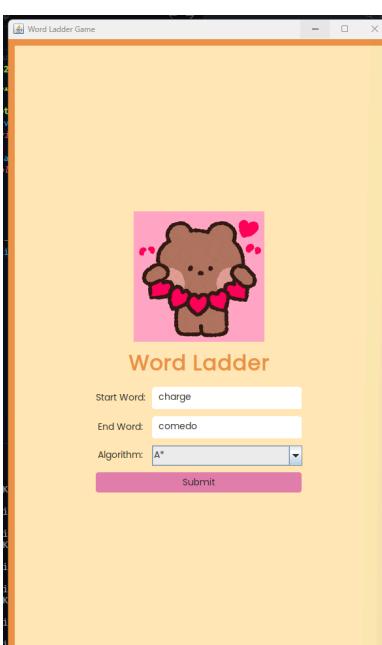
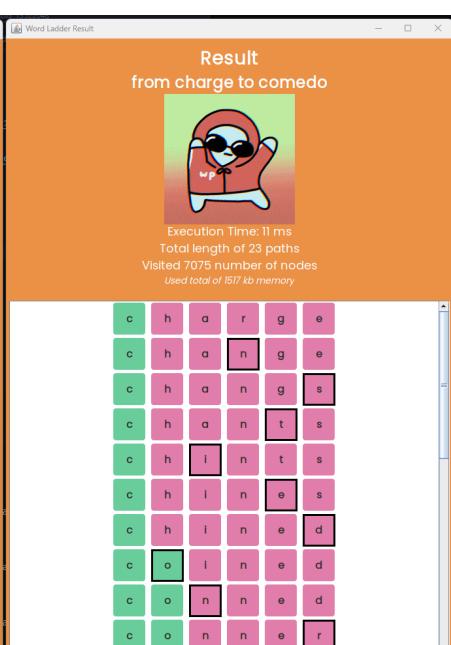
Execution Time: 0 ms
Total length of 18 paths
Visited 38 number of nodes
Used total of 0 kb memory

| | | | | |
|---|---|---|---|---|
| n | o | i | i | y |
| n | o | i | i | s |
| s | o | i | i | s |
| s | a | i | i | s |
| s | a | i | d | s |
| s | k | i | d | s |
| s | k | i | t | s |
| s | m | i | t | e |
| s | m | i | i | e |

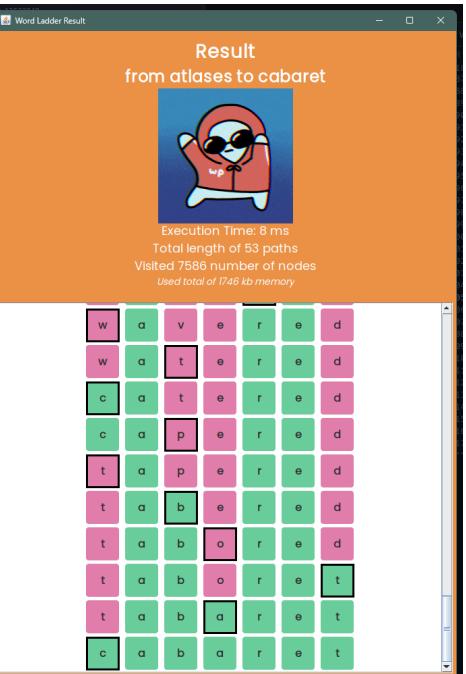
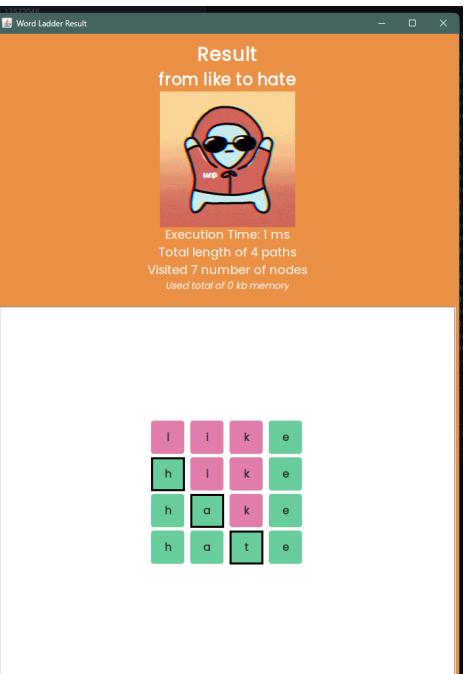
Kata Awal :
“dread”

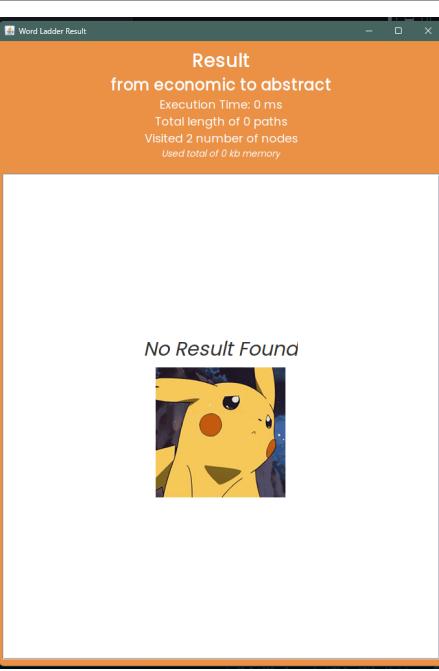
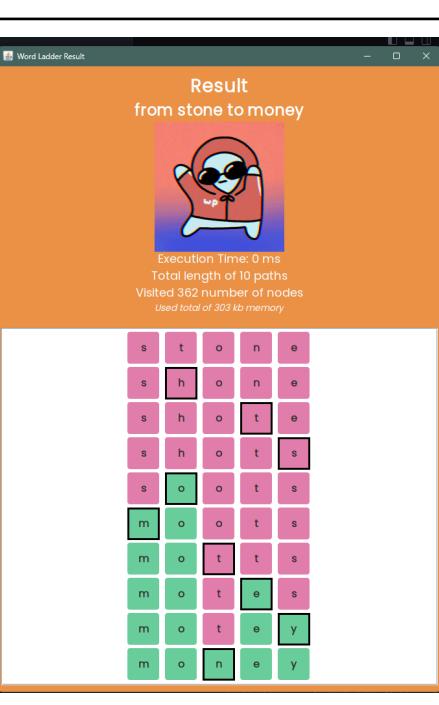
Kata Akhir :
“smile”

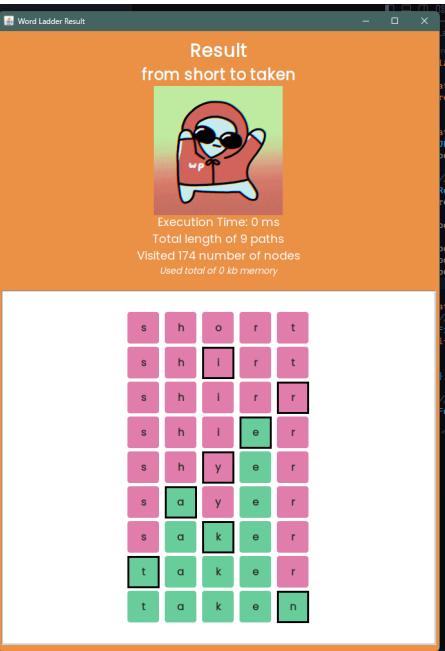
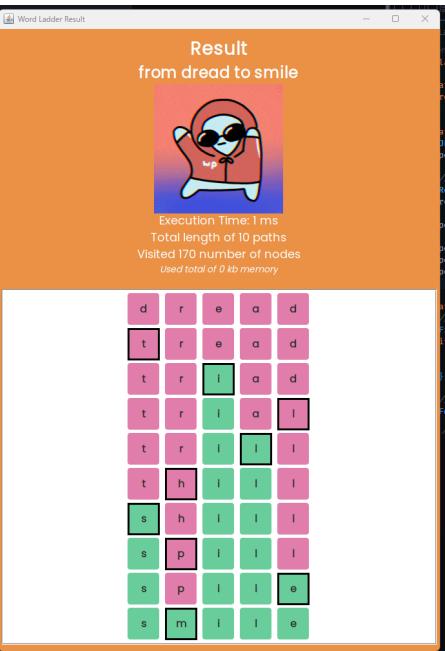
3.3. Tangkapan Layar Algoritma A* (A Star)

| | | |
|--|---|--|
|  |  | <p>Kata Awal : “friend”</p> <p>Kata Akhir : “lovers”</p> |
|  |  | <p>Kata Awal : “charge”</p> <p>Kata Akhir : “comedo”</p> |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| <p>Word Ladder</p> <p>Start Word: charge</p> <p>End Word: comedo</p> <p>Algorithm: A*</p> <p>Submit</p> | <p>Result from charge to comedo</p> <p>Execution Time: 11 ms Total length of 23 paths Visited 7075 number of nodes Used total of 1817 kb memory</p> <table border="1"> <tbody> <tr><td>c</td><td>o</td><td>n</td><td>i</td><td>n</td><td>s</td></tr> <tr><td>c</td><td>o</td><td>n</td><td>i</td><td>n</td><td>g</td></tr> <tr><td>c</td><td>o</td><td>m</td><td>i</td><td>n</td><td>g</td></tr> <tr><td>h</td><td>o</td><td>m</td><td>i</td><td>n</td><td>g</td></tr> <tr><td>h</td><td>o</td><td>m</td><td>i</td><td>l</td><td>y</td></tr> <tr><td>h</td><td>o</td><td>m</td><td>e</td><td>l</td><td>y</td></tr> <tr><td>c</td><td>o</td><td>m</td><td>e</td><td>l</td><td>y</td></tr> <tr><td>c</td><td>o</td><td>m</td><td>e</td><td>d</td><td>y</td></tr> <tr><td>c</td><td>o</td><td>m</td><td>e</td><td>d</td><td>o</td></tr> </tbody> </table> | c | o | n | i | n | s | c | o | n | i | n | g | c | o | m | i | n | g | h | o | m | i | n | g | h | o | m | i | l | y | h | o | m | e | l | y | c | o | m | e | l | y | c | o | m | e | d | y | c | o | m | e | d | o | | | |
| c | o | n | i | n | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | n | i | n | g | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | m | i | n | g | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| h | o | m | i | n | g | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| h | o | m | i | l | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| h | o | m | e | l | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | m | e | l | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | m | e | d | y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | o | m | e | d | o | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Word Ladder</p> <p>Start Word: atlases</p> <p>End Word: cabaret</p> <p>Algorithm: A*</p> <p>Submit</p> | <p>Result from atlases to cabaret</p> <p>Execution Time: 8 ms Total length of 53 paths Visited 7586 number of nodes Used total of 1746 kb memory</p> <table border="1"> <tbody> <tr><td>a</td><td>t</td><td>l</td><td>a</td><td>s</td><td>e</td><td>s</td></tr> <tr><td>a</td><td>n</td><td>l</td><td>a</td><td>s</td><td>e</td><td>s</td></tr> <tr><td>a</td><td>n</td><td>i</td><td>a</td><td>c</td><td>e</td><td>s</td></tr> <tr><td>u</td><td>n</td><td>l</td><td>a</td><td>c</td><td>e</td><td>s</td></tr> <tr><td>u</td><td>n</td><td>i</td><td>a</td><td>d</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>f</td><td>a</td><td>d</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>f</td><td>a</td><td>k</td><td>e</td><td>d</td></tr> <tr><td>u</td><td>n</td><td>c</td><td>a</td><td>k</td><td>e</td><td>s</td></tr> </tbody> </table> | a | t | l | a | s | e | s | a | n | l | a | s | e | s | a | n | i | a | c | e | s | u | n | l | a | c | e | s | u | n | i | a | d | e | d | u | n | f | a | d | e | d | u | n | f | a | k | e | d | u | n | c | a | k | e | s | <p>Kata Awal : “atlases”</p> <p>Kata Akhir : “cabaret”</p> |
| a | t | l | a | s | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | n | l | a | s | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | n | i | a | c | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | l | a | c | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | i | a | d | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | f | a | d | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | f | a | k | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u | n | c | a | k | e | s | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  <p>Word Ladder</p> <p>Start Word: <input type="text" value="atlasses"/></p> <p>End Word: <input type="text" value="cabaret"/></p> <p>Algorithm: A*</p> <p>Submit</p> |  <p>Result from atlases to cabaret</p> <p>Execution Time: 8 ms Total length of 53 paths Visited 7586 number of nodes Used total of 1746 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>w</td><td>a</td><td>v</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>w</td><td>a</td><td>t</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>c</td><td>a</td><td>t</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>c</td><td>a</td><td>p</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>p</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>e</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>o</td><td>r</td><td>e</td><td>d</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>o</td><td>r</td><td>e</td><td>t</td></tr> <tr><td>t</td><td>a</td><td>b</td><td>a</td><td>r</td><td>e</td><td>t</td></tr> <tr><td>e</td><td>a</td><td>b</td><td>a</td><td>r</td><td>e</td><td>t</td></tr> </table> | w | a | v | e | r | e | d | w | a | t | e | r | e | d | c | a | t | e | r | e | d | c | a | p | e | r | e | d | t | a | p | e | r | e | d | t | a | b | e | r | e | d | t | a | b | o | r | e | d | t | a | b | o | r | e | t | t | a | b | a | r | e | t | e | a | b | a | r | e | t |
| w | a | v | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| w | a | t | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | a | t | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| c | a | p | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | p | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | e | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | o | r | e | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | o | r | e | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | b | a | r | e | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| e | a | b | a | r | e | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  <p>Word Ladder</p> <p>Start Word: <input type="text" value="like"/></p> <p>End Word: <input type="text" value="hate"/></p> <p>Algorithm: A*</p> <p>Submit</p> |  <p>Result from like to hate</p> <p>Execution Time: 1 ms Total length of 4 paths Visited 7 number of nodes Used total of 0 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>l</td><td>i</td><td>k</td><td>e</td></tr> <tr><td>h</td><td>i</td><td>k</td><td>e</td></tr> <tr><td>h</td><td>a</td><td>k</td><td>e</td></tr> <tr><td>h</td><td>a</td><td>t</td><td>e</td></tr> </table> | l | i | k | e | h | i | k | e | h | a | k | e | h | a | t | e | <p>Kata Awal : “like”</p> <p>Kata Akhir : “hate”</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| l | i | k | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| h | i | k | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| h | a | k | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| h | a | t | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|--|--|
|   | <p>Kata Awal : “economic”</p> <p>Kata Akhir : “abstract”</p> |
|   | <p>Kata Awal : “stone”</p> <p>Kata Akhir : “money”</p> |

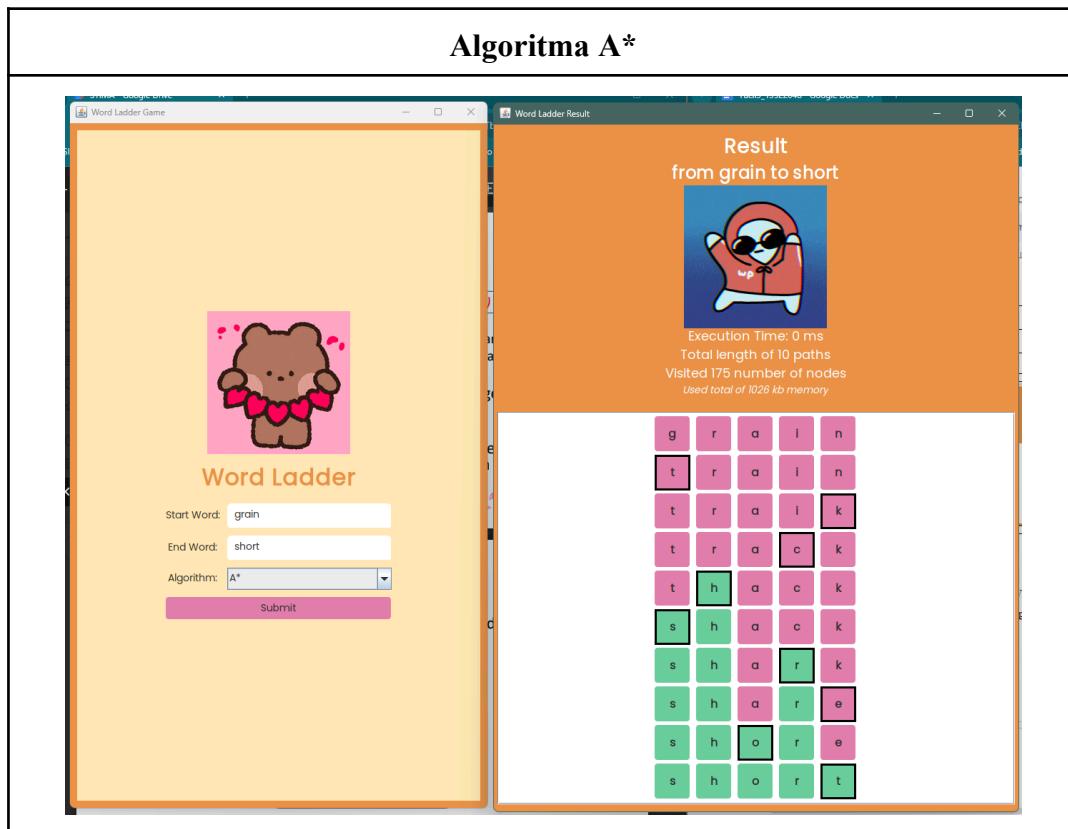
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
|  <p>Word Ladder</p> <p>Start Word: short</p> <p>End Word: taken</p> <p>Algorithm: A*</p> <p>Submit</p> |  <p>Result from short to taken</p> <p>Execution Time: 0 ms Total length of 9 paths Visited 174 number of nodes Used total of 0 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>s</td><td>h</td><td>o</td><td>r</td><td>t</td></tr> <tr><td>s</td><td>h</td><td> </td><td>r</td><td>t</td></tr> <tr><td>s</td><td>h</td><td> </td><td>r</td><td> </td></tr> <tr><td>s</td><td>h</td><td> </td><td>e</td><td>r</td></tr> <tr><td>s</td><td>h</td><td>y</td><td>e</td><td>r</td></tr> <tr><td>s</td><td>a</td><td>y</td><td>e</td><td>r</td></tr> <tr><td>s</td><td>a</td><td>k</td><td>e</td><td>r</td></tr> <tr><td>t</td><td>a</td><td>k</td><td>e</td><td>r</td></tr> <tr><td>t</td><td>a</td><td>k</td><td>e</td><td>n</td></tr> </tbody> </table> | s | h | o | r | t | s | h | | r | t | s | h | | r | | s | h | | e | r | s | h | y | e | r | s | a | y | e | r | s | a | k | e | r | t | a | k | e | r | t | a | k | e | n | <p>Kata Awal : “short”</p> <p>Kata Akhir : “taken”</p> |
| s | h | o | r | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | | r | t | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | y | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | a | y | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | a | k | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | k | e | r | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | a | k | e | n | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  <p>Word Ladder</p> <p>Start Word: dread</p> <p>End Word: smile</p> <p>Algorithm: A*</p> <p>Submit</p> |  <p>Result from dread to smile</p> <p>Execution Time: 1 ms Total length of 10 paths Visited 170 number of nodes Used total of 0 kb memory</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tbody> <tr><td>d</td><td>r</td><td>e</td><td>a</td><td>d</td></tr> <tr><td>t</td><td>r</td><td>e</td><td>a</td><td>d</td></tr> <tr><td>t</td><td>r</td><td> </td><td>a</td><td>d</td></tr> <tr><td>t</td><td>r</td><td> </td><td>a</td><td> </td></tr> <tr><td>t</td><td>h</td><td> </td><td> </td><td> </td></tr> <tr><td>s</td><td>h</td><td> </td><td> </td><td> </td></tr> <tr><td>s</td><td>p</td><td> </td><td> </td><td> </td></tr> <tr><td>s</td><td>p</td><td> </td><td> </td><td>e</td></tr> <tr><td>s</td><td>m</td><td> </td><td> </td><td>e</td></tr> </tbody> </table> | d | r | e | a | d | t | r | e | a | d | t | r | | a | d | t | r | | a | | t | h | | | | s | h | | | | s | p | | | | s | p | | | e | s | m | | | e | <p>Kata Awal : “dread”</p> <p>Kata Akhir : “smile”</p> |
| d | r | e | a | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | r | e | a | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | r | | a | d | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | r | | a | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| t | h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | p | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | p | | | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s | m | | | e | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

BAB IV

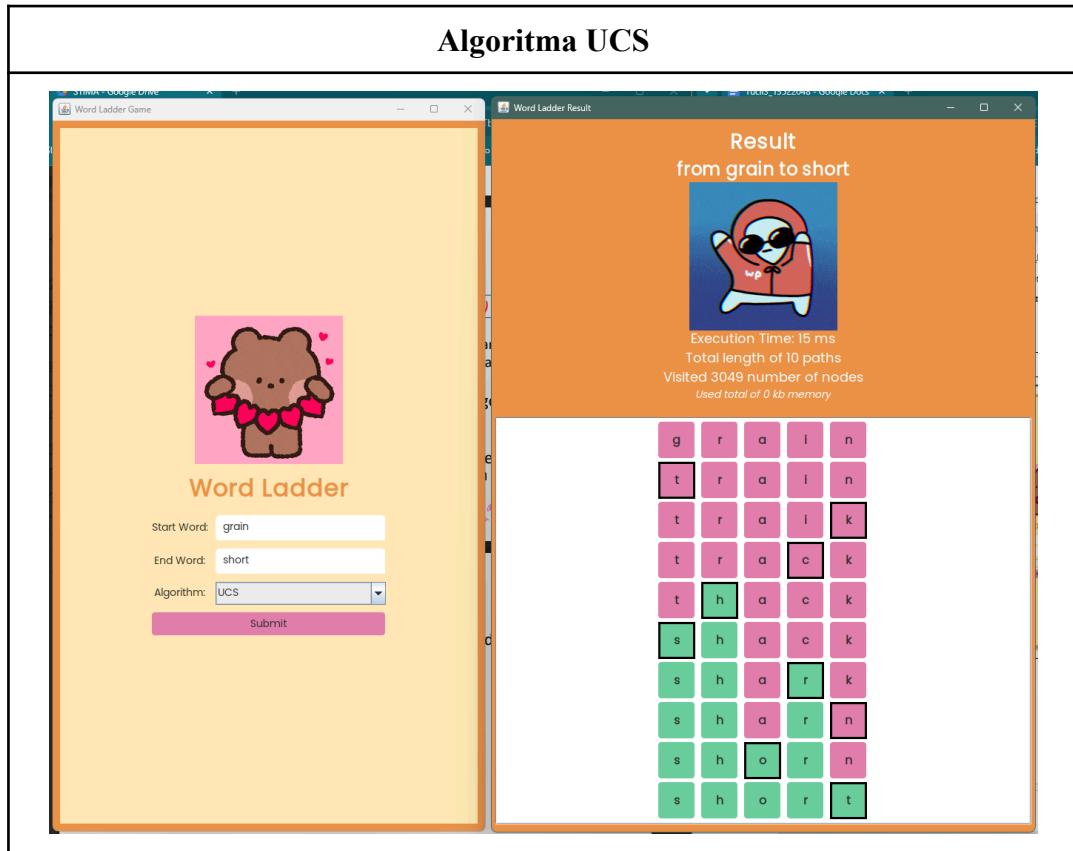
Analisis Perbandingan Algoritma

4.1. Perbandingan Optimalitas

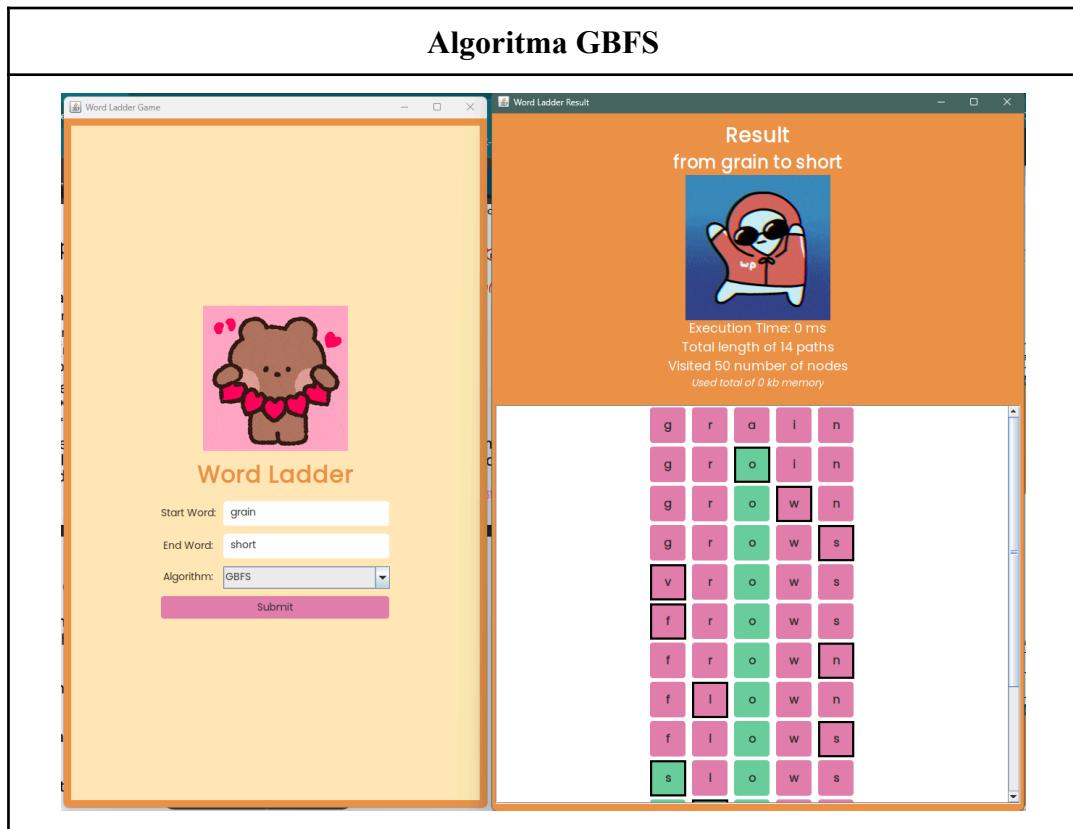
Berdasarkan percobaan yang telah dilakukan A* dan UCS akan selalu menghasilkan hasil yang optimal. Hal ini terjadi karena UCS akan mencari menggunakan *actual cost*, dan memastikan bahwa simpul yang diekspansi merupakan simpul yang dengan *cost* paling minimum. A*, selain mempertimbangkan *actual cost* simpul, juga mempertimbangkan perkiraan biaya *cost* menuju simpul tujuan, karena sifat heuristik A* yang *admissible* maka solusi dari A* merupakan optimal dan konsisten. Berbeda dari UCS dan A*, GBFS hanya mempertimbangkan prediksi *cost* dari simpul menuju *goal*, tanpa melihat *cost* yang diperlukan untuk menuju ke simpul tersebut. Alhasil, solusi tidak akan optimal karena heuristik tidak selalu *admissible*, sehingga solusi tidak konsisten dan menjadi tidak optimal



Algoritma UCS

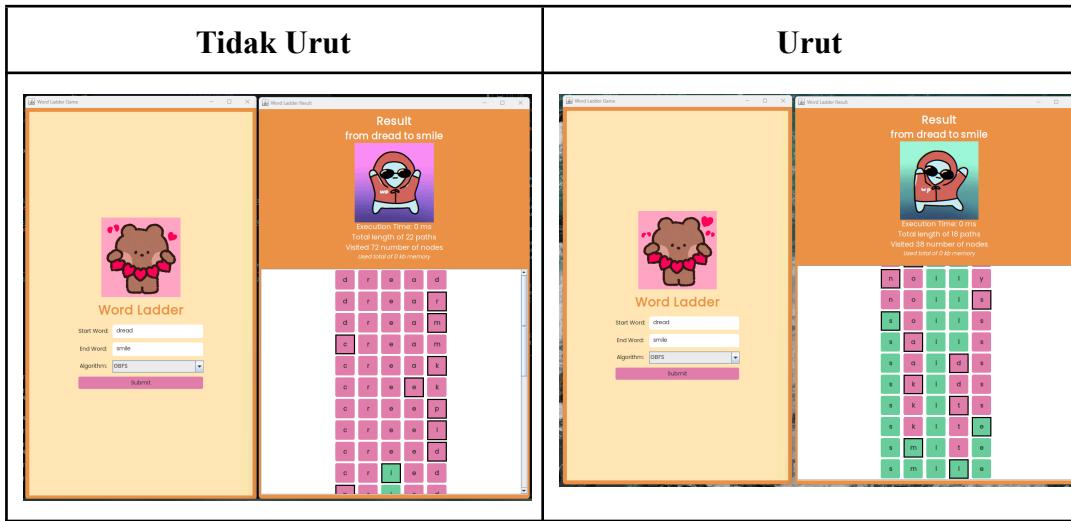


Algoritma GBFS



Sebagai perbandingan, merujuk ke tabel yang ada di atas, untuk kasus dari “grain” ke “short”, UCS dan A* memberikan hasil 10 *paths*, sedangkan GBFS memberikan 14 *paths*. Hal ini menunjukkan GBFS tidak memberikan solusi yang optimal.

Sebagai pembanding lain, GBFS akan sangat terpengaruhi dengan urutan masukkan simpul ke dalam queue. Berikut merupakan contoh jika, simpul dimasukkan berdasarkan urutan leksikografis dan jika tidak terurut.

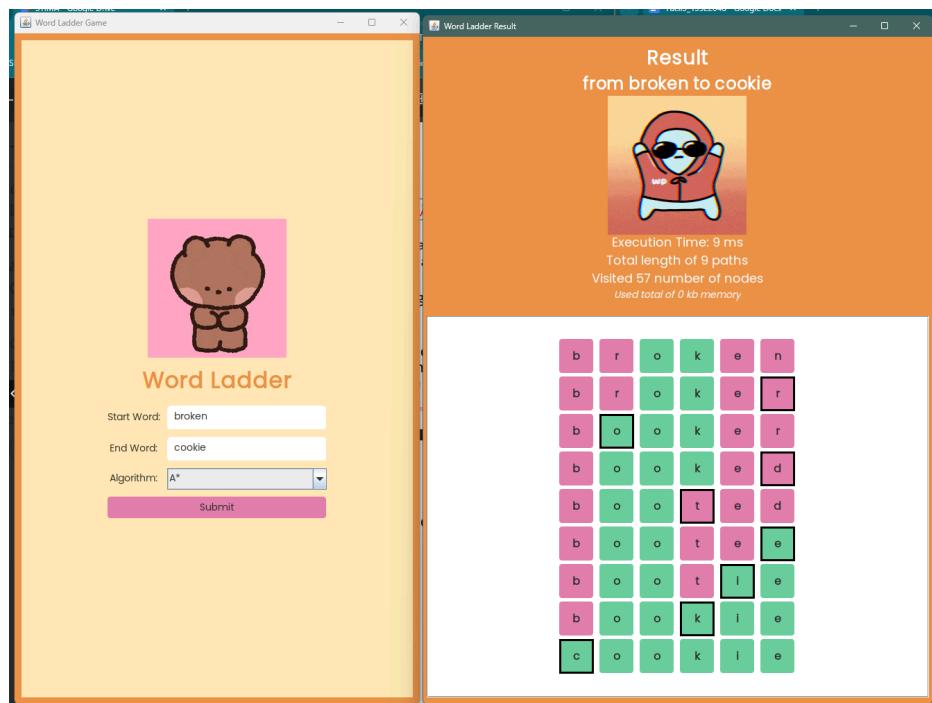


Perbedaan panjang solusi yang diberikan oleh GBFS menunjukkan bahwa, solusi GBFS tidak konsisten sehingga GBFS tidak selalu memberikan solusi yang optimal.

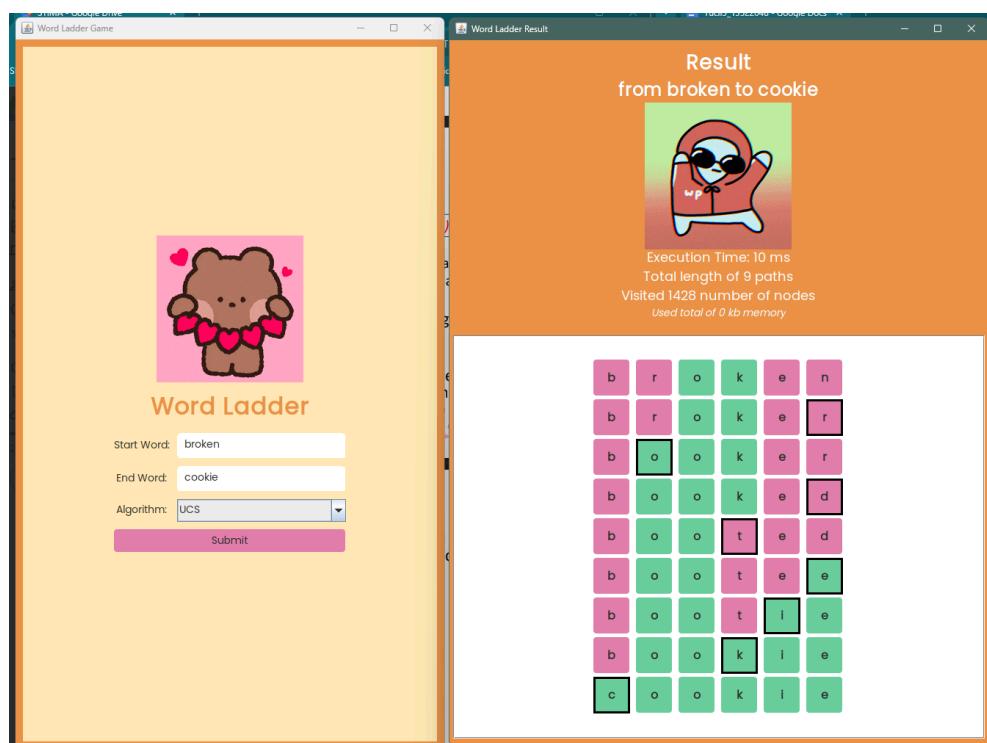
4.2. Perbandingan Waktu Eksekusi

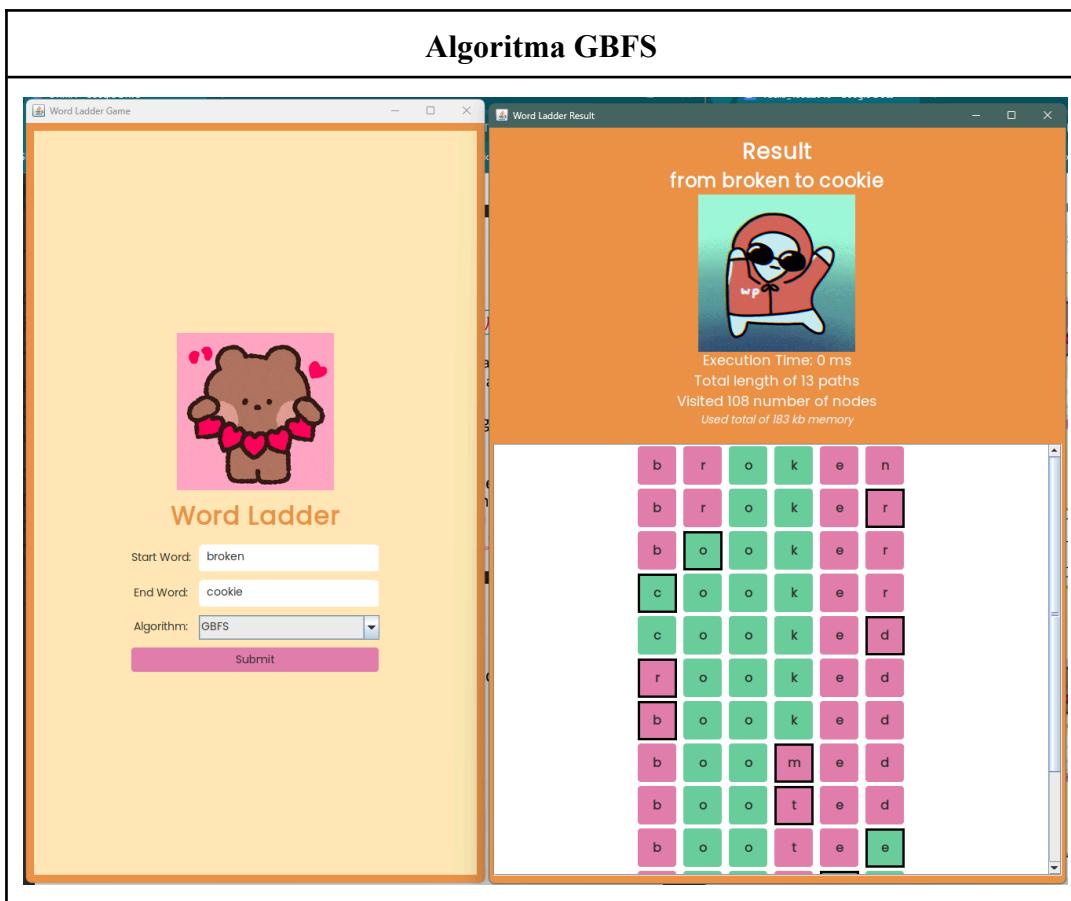
GBFS akan memiliki waktu eksekusi yang paling cepat, karena hanya memperhitungkan *heuristic cost*. *Cost* untuk tiap simpul tidak akan berubah dalam kasus GBFS, sedangkan A* dan UCS perlu menghitung *cost* jarak suatu simpul. Untuk tiap simpul yang ditemukan A* dan UCS perlu melakukan *update* terhadap *cost*-nya yang tidak bernilai tetap. Namun, A* akan berjalan lebih cepat dibandingkan dengan UCS. Hal ini terjadi karena UCS memeriksa lebih banyak simpul dibandingkan dengan A*. A* juga mempertimbangkan *heuristic cost* sehingga pencarian dilakukan secara lebih efisien. Untuk melihat perbedaannya akan disajikan dalam bentuk tabel sebagai berikut:

Algoritma A*



Algoritma UCS



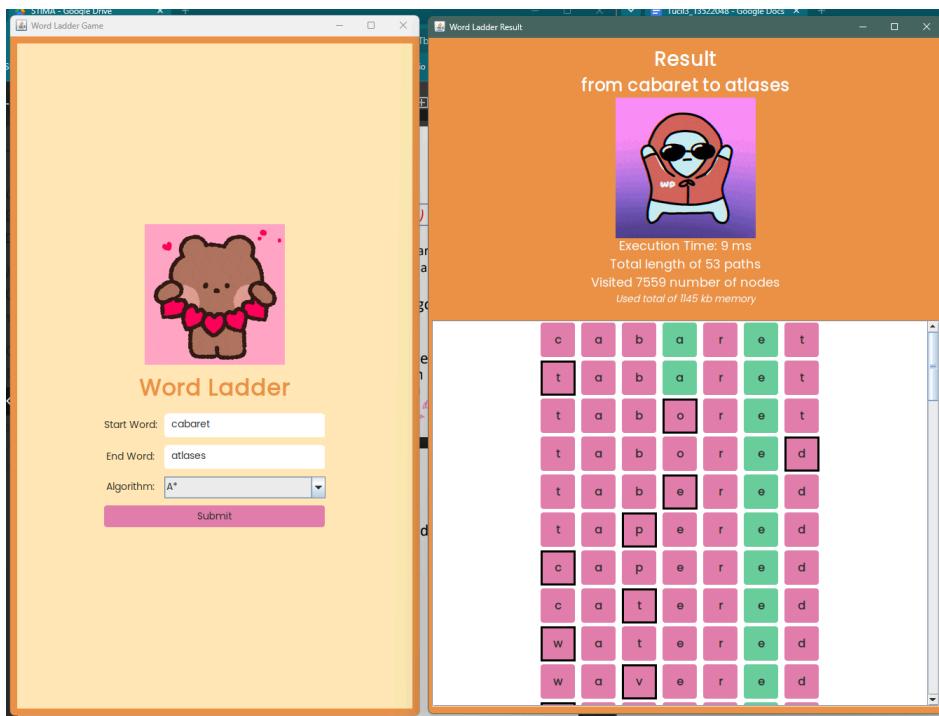


Dapat dilihat bahwa untuk “broken” ke “cookie”, A* membutuhkan waktu 9 ms, UCS membutuhkan 10 ms, dan GBFS membutuhkan 0 ms. Jumlah nodes yang dikunjungi UCS juga jauh lebih banyak. Hasil menunjukkan bahwa GBFS memiliki waktu eksekusi terpendek, dilanjutkan dengan A*, lalu UCS.

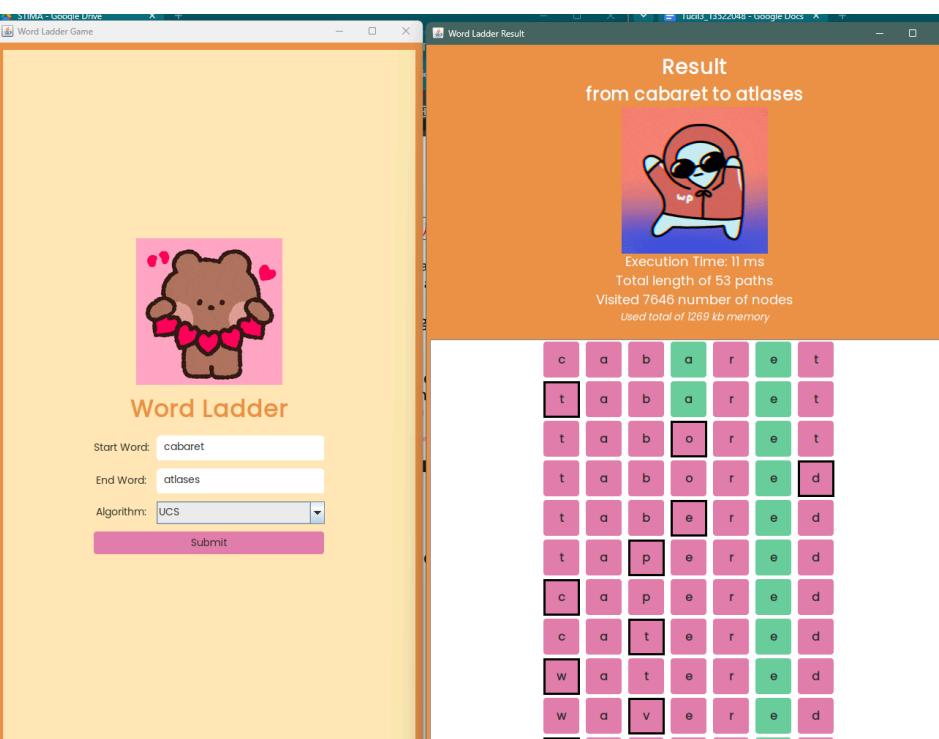
4.3. Perbandingan Memori yang Dibutuhkan

GBFS menggunakan paling sedikit memori karena GBFS hanya perlu menyimpan data simpul yang sedang diekspansi dan juga *heuristic cost*-nya. GBFS tidak melihat rute-rute lain yang ada, dan hanya melihat simpul mana yang paling dekat dengan tujuan. UCS membutuhkan paling banyak memori, karena mengunjungi lebih banyak simpul, serta harus menyimpan dan memperbarui tiap *cost* simpul. A* berada di antara UCS dan GBFS, karena A* menyimpan nilai *heuristic cost* sehingga simpul yang dikunjungi tidak sebanyak UCS. Berikut merupakan perbedaan penggunaan memori antar ketiga algoritma:

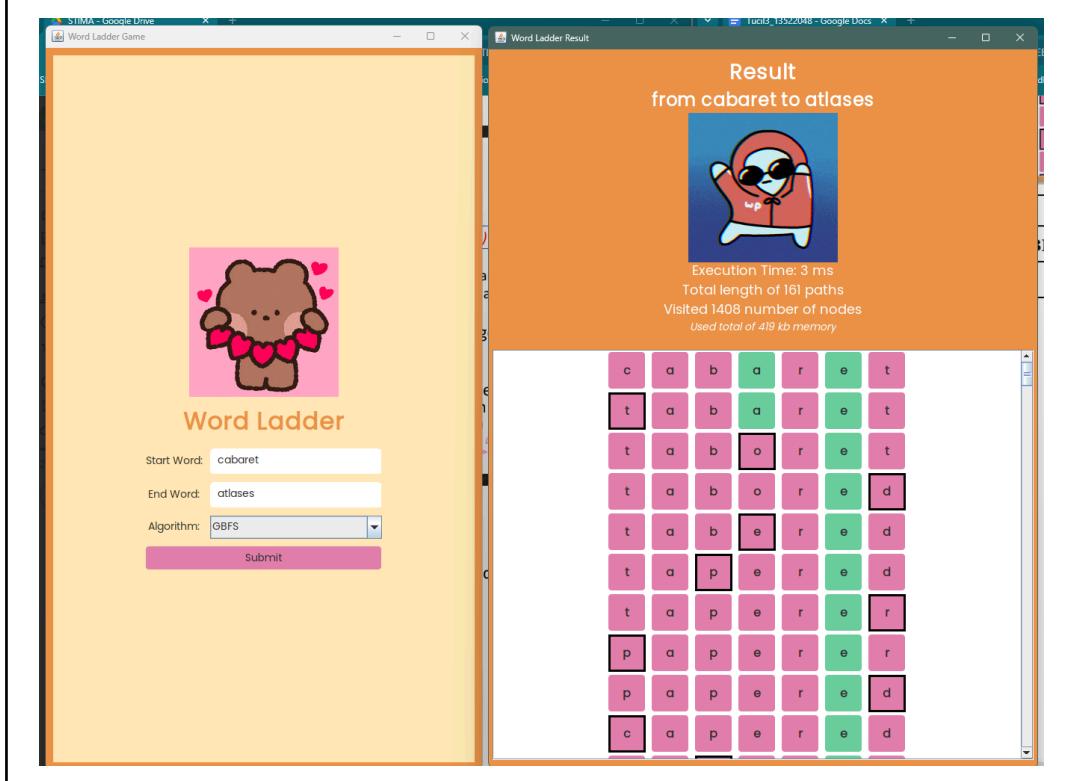
Algoritma A*



Algoritma UCS



Algoritma GBFS



Untuk kasus “cabaret” ke “atlases” , GBFS membutuhkan memori sebanyak 419 kb , A* sebanyak 1145 kb , UCS sebanyak 1269 kb. Hal ini menunjukkan GBFS menggunakan memori paling sedikit dibanding dengan A* dan UCS. Sedangkan UCS membutuhkan memori yang paling banyak di antara ketiga algoritma.

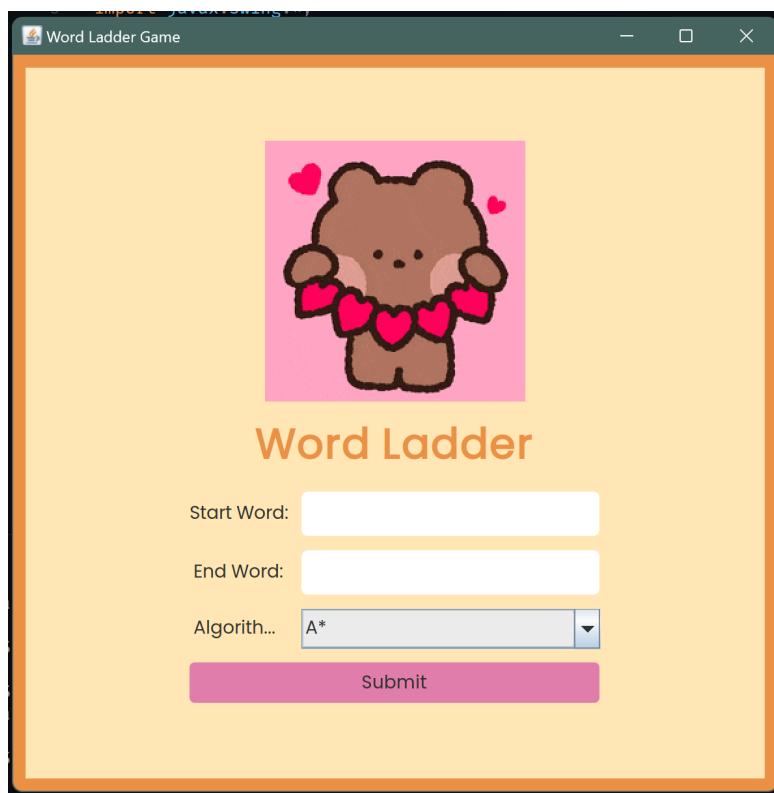
Namun, penggunaan memori akan selalu variatif, tergantung dengan simpul-simpul yang dikunjungi. Sehingga terdapat kemungkinan jika penggunaan memori tidak selalu seperti UCS > A* > GBFS.

BAB V

Implementasi Bonus

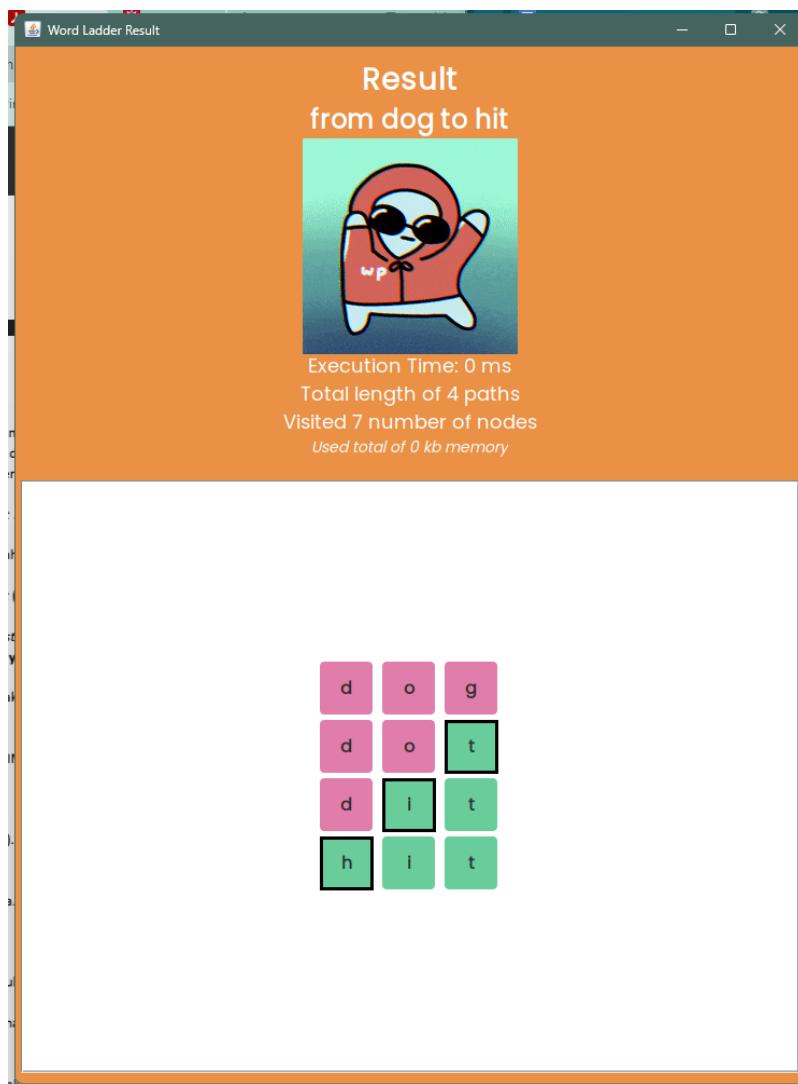
5.1.Tampilan GUI

Tampilan GUI dikembangkan menggunakan Java Swing dan juga Java AWT. Terdapat dua tampilan utama , yaitu tampilan awal untuk menerima masukkan dan tampilan untuk menunjukkan hasil.



Gambar 5.1.1. Tampilan input

Tampilan ini menggunakan komponen JPanel sebagai dasar. Terdapat komponen JTextField yang di *custom* sehingga memiliki ujung yang *rounded*. Terdapat JComboBox<String> yang digunakan untuk memilih algoritma yang akan dipakai. Terakhir ada komponen JButton yang *custom* sehingga memiliki ujung yang *rounded*. Terdapat *event listener* sehingga, ketika tombol “submit” ditekan, proses pencarian akan tereksekusi. Terdapat gif sebagai hiasan pada panel masukkan.



Gambar 5.1.2. Tampilan hasil

Tampilan ini menggunakan JFrame untuk melakukan pop up dan juga JPanel sebagai panel utama. Terdapat dua panel tambahan yaitu untuk menunjukkan informasi, dan untuk menunjukkan rute. Pada panel untuk informasi terdapat JLabel yang digunakan untuk menampilkan tulisan, dan juga terdapat GIF yang akan ditunjukkan jika terdapat solusi. Untuk panel yang menunjukkan hasil, akan menunjukkan hasil rute, dan menandakan karakter mana yang berubah dan karakter mana yang sudah sama dengan *endword*. Tiap karakter dikemas oleh sebuah komponen JLabel, dimana tiap karakter akan diiterasi untuk membuat JLabel sendiri.

5.2. Kode Utama GUI

GUI berada dalam kelas WordLadderGUI.java yang melakukan *inheritance* terhadap JFrame.



```
1 private JLabel startlabel, endlabel, algorithmLabel, titleLabel;
2 private JTextField startField, endField;
3 private JComboBox<String> algorithmComboBox;
4 private JButton submitButton;
5 private src.utils.Dictionary dictionary;
6 public static Font customFont;
7 public static String[] colpal = {"#6ad9d9", "#e27eab", "#f9d9e0", "#eb954a", "#ffeaeb"}; // TURQOISE, DARK PINK, LIGHT PINK, ORANGE, LIGHT ORANGE
```

Gambar 5.2.1. Attribut kelas GUI



```
1 // Method for executing the program
2 private class handleButtonClick implements ActionListener {
3     @Override
4     public void actionPerformed(ActionEvent e) {
5         String startWord = startField.getText().trim().toLowerCase();
6         String endWord = endField.getText().trim().toLowerCase();
7
8         // Validate the word entries
9         if (!isValidInput(startWord) || !isValidInput(endWord)) {
10             JOptionPane.showMessageDialog(WordLadderGUI.this, "Please enter a valid input.", "Input Error", JOptionPane.ERROR_MESSAGE);
11             return;
12         }
13
14         // If word's length are not the same
15         if (startWord.length() != endWord.length()) {
16             JOptionPane.showMessageDialog(WordLadderGUI.this, "Your input words have different lengths. No solution found", "Input Error", JOptionPane.ERROR_MESSAGE);
17             return;
18         }
19
20         // If both of the words are valid
21         if (dictionary.validWord(startWord) && dictionary.validWord(endWord)) {
22
23             String selectedAlgorithm = (String) algorithmComboBox.getSelectedItem();
24
25             // Creating algorithm objects
26             Search search;
27             if (selectedAlgorithm.equals("A*")) {
28
29                 search = new AStar();
30             } else if (selectedAlgorithm.equals("UCS")) {
31
32                 search = new UCS();
33             } else {
34
35                 search = new GBFS();
36             }
37
38             // Getting memory before the search method
39             long startMemory = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();
40
41             // Searching the solution & getting execution time
42             long start = System.currentTimeMillis();
43             Result result = search.findSolution(startWord, endWord, dictionary);
44             long end = System.currentTimeMillis();
45
46             long executionTime = end - start;
47
48             // Extracting Result object
49             List<String> solutions = result.getSolution();
50             int nodesVisited = result.getNumberOfVisitedNodes();
51
52             // Getting memory after search
53             long endMemory = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();
54             long memoryUsed = endMemory - startMemory;
55
56             // Converting to KB
57             long memoryUsedKB = memoryUsed / (1024);
58
59             // Calling Result Display Frame
60             showResultPopup(solutions, executionTime, startWord, endWord, nodesVisited, memoryUsedKB);
61
62         } else {
63             JOptionPane.showMessageDialog(WordLadderGUI.this, "Your startword or endword don't exist in the dictionary. Please input a valid word", "Input Error", JOptionPane.ERROR_MESSAGE);
64         }
65     }
66 }
```

Gambar 5.2.2. Method untuk mengeksekusi

Method ini akan berjalan ketika tombol submit ditekan, lalu program akan memeriksa masukkan *startWord* dan *endWord*, jika sudah valid program akan memeriksa masukkan pilihan algoritma dan membuat objek pencarian yang sesuai. Ketika hasil sudah didapatkan akan memanggil method untuk menampilkan *frame* hasil



```
1 public WordLadderGUI() {
2     // Load mapped dictionary
3     dictionary = new src.utils.Dictionary('src/mapped_dictionary.txt');
4
5     // Load custom font
6     try {
7         customFont = loadCustomFont("src/gui/font.ttf");
8     } catch (IOException | FontFormatException e) {
9         e.printStackTrace();
10    }
11
12     // Setting up main frame
13     setTitle("Word Ladder Game");
14     setSize(600, 600);
15     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17     // Setting main panel
18     JPanel mainPanel = new JPanel(new GridBagLayout());
19     mainPanel.setBackground(Color.decode(colpal[4]));
20     mainPanel.setFocusable(true);
21
22     // Set border around panel
23     Border border = BorderFactory.createLineBorder(Color.decode(colpal[3]), 10);
24     mainPanel.setBorder(border);
25
26     // Add gridbag layouting
27     GridBagConstraints gbc = new GridBagConstraints();
28     gbc.insets = new Insets(5,5,5,5);
29     gbc.anchor = GridBagConstraints.CENTER;
30
31     int yInit = 0;
32
33     // Add gif
34     Icon startImg = new ImageIcon("src/gui/img/start.gif");
35     JLabel imgLabel = new JLabel(startImg);
36     gbc.gridx = 0;
37     gbc.gridy = yInit;
38     gbc.gridwidth = 2;
39     mainPanel.add(imgLabel, gbc);
40
41     yInit++;
42
43     // Add title
44     titleLabel = new JLabel("Word Ladder");
45     titleLabel.setFont(customFont.deriveFont(Font.BOLD, 32));
46     titleLabel.setForeground(Color.decode(colpal[3]));
47     gbc.gridx = 0;
48     gbc.gridy = yInit;
49     gbc.gridwidth = 2;
50     mainPanel.add(titleLabel, gbc);
51
52     yInit++;
53     gbc.gridwidth = 1;
54
55     // Add start input label
56     startLabel = new JLabel("Start Word:");
57     startLabel.setFont(customFont.deriveFont(Font.PLAIN, 14));
58     gbc.gridx = 0;
59     gbc.gridy = yInit;
60     mainPanel.add(startLabel, gbc);
61
62     // Add input field
63     startField = new RoundedTextField(15,8);
64     startField.setFont(customFont.deriveFont(Font.PLAIN,14));
65     gbc.gridx = 1;
66     gbc.gridy = yInit;
67     mainPanel.add(startField, gbc);
68
69     yInit++;
70
71     // Add end input label
72     endLabel = new JLabel("End Word:");
73     endLabel.setFont(customFont.deriveFont(Font.PLAIN,14));
74     gbc.gridx = 0;
75     gbc.gridy = yInit;
76     mainPanel.add(endLabel, gbc);
77
78     // Add input field
79     endField = new RoundedTextField(15,8);
80     endField.setFont(customFont.deriveFont(Font.PLAIN,14));
81     gbc.gridx = 1;
82     gbc.gridy = yInit;
83     mainPanel.add(endField, gbc);
84
85     yInit++;
86
87     // Add algorithm label
88     algorithmLabel = new JLabel("Algorithm:");
89     algorithmLabel.setFont(customFont.deriveFont(Font.PLAIN,14));
90     gbc.gridx = 0;
91     gbc.gridy = yInit;
92     mainPanel.add(algorithmLabel, gbc);
93
94     // Add algorithm picker
95     String[] algorithms = {"A*", "UCS", "GBFS"};
96     algorithmComboBox = new JComboBox(algorithms);
97     algorithmComboBox.setFont(customFont.deriveFont(Font.PLAIN,14));
98     gbc.gridx = 1;
99     gbc.gridy = yInit;
100    gbc.fill = GridBagConstraints.HORIZONTAL;
101    mainPanel.add(algorithmComboBox, gbc);
102
103    yInit++;
104
105    // Add submit button
106    submitButton = new RoundedButton("Submit", Color.decode(colpal[1]), Color.decode(colpal[2]), 8);
107    submitButton.setFont(customFont.deriveFont(Font.PLAIN,14));
108    gbc.gridx = 0;
109    gbc.gridy = yInit;
110    gbc.gridwidth = 2;
111    mainPanel.add(submitButton, gbc);
112
113    submitButton.addActionListener(new handleButtonClick());
114
115    add(mainPanel);
116    setLocationRelativeTo(null);
117    setVisible(true);
118}
119
```

Gambar 5.2.3. Program tampilan utama

```
1 // Validating input method
2     private boolean isValidInput(String input) {
3         return !input.isEmpty() && !input.contains(" ");
4     }
5
6 // Method for displaying Result Frame
7     private void showResultPopUp(List<String> solutions, long executionTime, String startWord, String endWord, int nodesVisited, long memory) {
8         JFrame popupFrame = new JFrame("Word Ladder Result");
9         popupFrame.setLayout(new BorderLayout());
10
11        // Create and add the result display panel
12        ResultDisplayPanel resultPanel = new ResultDisplayPanel(solutions, executionTime, startWord, endWord, nodesVisited, memory);
13        resultPanel.setFont(customFont);
14
15        popupFrame.add(resultPanel, BorderLayout.CENTER);
16
17        popupFrame.setSize(750, 1000);
18        popupFrame.setLocationRelativeTo(this);
19        popupFrame.setVisible(true);
20    }
21
22    private Font loadCustomFont(String fontFileName) throws IOException, FontFormatException {
23        // Load custom font file
24        File fontFile = new File(fontFileName);
25        if (!fontFile.exists()) {
26            System.err.println("Font file not found: " + fontFileName);
27            return null;
28        }
29
30        // Create font object from file
31        Font loadedFont = Font.createFont(Font.TRUETYPE_FONT, fontFile);
32
33        // Register font with GraphicsEnvironment
34        GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
35        ge.registerFont(loadedFont);
36
37        return loadedFont;
38    }
```

Gambar 5.2.4. Method-method pembantu

Method `isValidInput` berguna untuk memvalidasi masukkan harus 1 kata dan bukan alphanumerik. Method `showResultPopUp` akan memanggil *frame* yang menunjukkan hasil pencarian. Method `loadCustomFont` akan membaca file “.ttf” untuk menampilkan *custom font* pada GUI.

BAB VII

Lampiran

| Poin | Ya | Tidak |
|--|----|-------|
| 1. Program berhasil dijalankan. | ✓ | |
| 2. Program dapat menemukan rangkaian kata dari <i>start word</i> ke <i>end word</i> sesuai aturan permainan dengan algoritma UCS. | ✓ | |
| 3. Solusi yang diberikan pada algoritma UCS optimal, | ✓ | |
| 4. Program dapat menemukan rangkaian kata dari <i>start word</i> ke <i>end word</i> sesuai aturan permainan dengan algoritma <i>Greedy Best First Search</i> . | ✓ | |
| 5. Program dapat menemukan rangkaian kata dari <i>start word</i> ke <i>end word</i> sesuai aturan permainan dengan algoritma A*. | ✓ | |
| 6. Solusi yang diberikan pada algoritma A* optimal. | ✓ | |
| 7. [Bonus] Program memiliki tampilan GUI | ✓ | |

Link Repository GitHub : https://github.com/angiekriera/Tucil3_13522048