# LAPORAN TUGAS KECIL

# IF2211 - STRATEGI ALGORTIMA

# “Penyelesaian Permainan Word Ladder Menggunakan Algoritma UCS, Greedy Best First Search, dan A*”


**Dosen:**

Ir. Rila Mandala, M.Eng., Ph.D.

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

**Nama :**

Axel Santadi Warih (13522155)

## PROGRAM STUDI TEKNIK INFORMATIKA

## INSTITUT TEKNOLOGI BANDUNG

## SEMESTER II TAHUN 2023/2024

# DAFTAR ISI

# BAB I
# Analisis dan Implementasi

## 1.1. Algoritma Uniform Cost Search (UCS)

### Deskripsi Langkah-langkah

1. Inisialisasi sebuah priority queue untuk menyimpan node-node yang akan dieksplorasi.
2. Masukkan node awal ke dalam priority queue dengan cost 0.
3. Selama priority queue tidak kosong:
    a. Ambil node dengan cost terkecil dari priority queue.
    b. Jika node yang diambil merupakan node tujuan, maka konstruksi path dan selesai.
    c. Untuk setiap node yang dapat dicapai dari node saat ini, perbarui cost dan masukkan ke dalam priority queue.
4. Jika priority queue kosong dan node tujuan tidak ditemukan, maka tidak ada solusi.

### Analisis

f(n): f(n) dalam UCS merupakan cost total untuk mencapai node n dari node awal.
g(n): g(n) dalam UCS merupakan cost sejauh ini untuk mencapai node n dari node awal.

## 1.2. Algoritma Greedy Best First Search (GBFS)

### Deskripsi Langkah-langkah

1. Inisialisasi sebuah priority queue untuk menyimpan node-node yang akan dieksplorasi, diurutkan berdasarkan nilai heuristik.
2. Masukkan node awal ke dalam priority queue.
3. Selama priority queue tidak kosong:
    a. Ambil node dengan heuristik terkecil dari priority queue.
    b. Jika node yang diambil merupakan node tujuan, maka konstruksi path dan selesai.
    c. Untuk setiap node yang dapat dicapai dari node saat ini, masukkan ke dalam priority queue.
4. Jika priority queue kosong dan node tujuan tidak ditemukan, maka tidak ada solusi.

### Analisis

f(n): f(n) dalam GBFS tidak memiliki interpretasi langsung seperti pada UCS, namun lebih menekankan pada nilai heuristik yang merupakan estimasi jarak dari node n ke node tujuan.
g(n): g(n) dalam GBFS tidak relevan karena algoritma ini tidak mempertimbangkan cost sejauh ini.

## 1.3. Algoritma A* (A Star)

### Deskripsi Langkah-langkah

1. Inisialisasi sebuah priority queue untuk menyimpan node-node yang akan dieksplorasi, diurutkan berdasarkan nilai f(n).
2. Masukkan node awal ke dalam priority queue dengan nilai f(n) = g(n) + h(n).
3. Selama priority queue tidak kosong:
    a. a. Ambil node dengan nilai f(n) terkecil dari priority queue.
    b. b. Jika node yang diambil merupakan node tujuan, maka konstruksi path dan selesai.
    c. c. Untuk setiap node yang dapat dicapai dari node saat ini, perbarui nilai f(n) dan masukkan ke dalam priority queue.
4. Jika priority queue kosong dan node tujuan tidak ditemukan, maka tidak ada solusi.

### Analisis

f(n): f(n) dalam A* merupakan perkiraan total cost untuk mencapai node tujuan melalui node n.

g(n): g(n) dalam A* merupakan cost sejauh ini untuk mencapai node n dari node awal.

Heuristik yang digunakan pada algoritma A* adalah admissible jika nilainya tidak pernah melebihi nilai sebenarnya dari biaya untuk mencapai tujuan dari node saat ini. Ini memastikan bahwa algoritma A* akan selalu menemukan solusi optimal jika heuristik yang digunakan admissible.

## 1.4. Jawaban Pertanyaan:

1. Definisi dari f(n) dan g(n) telah dijelaskan di atas sesuai dengan konteks algoritma yang digunakan.
2. Heuristik yang digunakan pada algoritma A* adalah admissible jika nilainya tidak pernah melebihi nilai sebenarnya dari biaya untuk mencapai tujuan dari node saat ini. Hal ini memastikan bahwa algoritma A* akan selalu menemukan solusi optimal jika heuristik yang digunakan admissible.
3. Pada kasus Word Ladder, algoritma UCS tidak sama dengan BFS. UCS mempertimbangkan cost untuk mencapai setiap node, sementara BFS hanya mencari jalur terpendek tanpa mempertimbangkan cost.
4. Secara teoritis, algoritma A* lebih efisien dibandingkan dengan algoritma UCS pada kasus Word Ladder karena A* menggunakan heuristik untuk memandu pencarian, yang dapat mengurangi jumlah node yang dieksplorasi.
5. Secara teoritis, algoritma Greedy Best First Search tidak menjamin solusi optimal untuk persoalan Word Ladder karena hanya mempertimbangkan nilai heuristik saat memilih node berikutnya, tanpa memperhatikan cost sejauh ini.

# BAB II
# Implementasi Algoritma

## 1. Algoritma Uniform Cost Search (UCS)

Class: UCS

### Method:

findWordLadderUCS(String startWord, String endWord, Set<String> wordList): Method ini bertanggung jawab untuk menemukan Word Ladder menggunakan algoritma UCS.

### Parameter:

startWord: Kata awal dari Word Ladder.
endWord: Kata tujuan dari Word Ladder.
wordList: Daftar kata yang digunakan untuk pembentukan Word Ladder.

### Return:

List<String>: Path yang ditemukan dari startWord ke endWord.

### Penjelasan:

Class UCS mengimplementasikan algoritma Uniform Cost Search untuk mencari Word Ladder antara dua kata dalam daftar kata yang diberikan. Method findWordLadderUCS menerima kata awal (startWord), kata tujuan (endWord), dan daftar kata (wordList) sebagai input. Kemudian, algoritma UCS diterapkan untuk menemukan jalur dari startWord ke endWord.

## 2. Algoritma Greedy Best First Search (GBFS)

Class: GBFS

### Method:

findWordLadderGreedy(String startWord, String endWord, Set<String> wordList): Method ini bertanggung jawab untuk menemukan Word Ladder menggunakan algoritma Greedy Best First Search.

### Parameter:

startWord: Kata awal dari Word Ladder.
endWord: Kata tujuan dari Word Ladder.
wordList: Daftar kata yang digunakan untuk pembentukan Word Ladder.

### Return:

List<String>: Path yang ditemukan dari startWord ke endWord.

### Penjelasan:

Class GBFS mengimplementasikan algoritma Greedy Best First Search untuk mencari Word Ladder antara dua kata dalam daftar kata yang diberikan. Method findWordLadderGreedy menerima kata awal (startWord), kata tujuan (endWord), dan daftar kata (wordList) sebagai input. Kemudian, algoritma GBFS diterapkan untuk menemukan jalur dari startWord ke endWord.

## 3. Algoritma A*

Class: ABintang

### Method:

findWordLadderABintang(String startWord, String endWord, Set<String> wordList): Method ini bertanggung jawab untuk menemukan Word Ladder menggunakan algoritma A*.

### Parameter:

startWord: Kata awal dari Word Ladder.
endWord: Kata tujuan dari Word Ladder.
wordList: Daftar kata yang digunakan untuk pembentukan Word Ladder.

### Return:

List<String>: Path yang ditemukan dari startWord ke endWord.

### Penjelasan:

Class AStar mengimplementasikan algoritma A* untuk mencari Word Ladder antara dua kata dalam daftar kata yang diberikan. Method findWordLadderAStar menerima kata awal (startWord), kata tujuan (endWord), dan daftar kata (wordList) sebagai input. Kemudian, algoritma A* diterapkan untuk menemukan jalur dari startWord ke endWord.

# BAB III

# Hasil Test-Case

## 1-UCS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): MAKE
Masukkan endWord (Pastikan semua huruf besar): FOOL
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Hasil Dari Uniform Cost Search Algorithm: [MAKE, MALE, MOLE, MOLL, MOOL, FOOL]
Jumlah node yang dikunjungi: 2261
Waktu eksekusi: 52 ms
```

## 1-GBFS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): MAKE
Masukkan endWord (Pastikan semua huruf besar): FOOL
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [MAKE, MAKO, MAYO, MAYS, MAWS, MAWN, MAUN, MAUT, MART, MARL, MALL, MALM, MAIM, MAIR, MAAR, HAAR, HAAF, HALF, HALT, HANT, HANK, HAWK,
 HOWK, HOWS, HOTS, HETS, HETH, HATH, HASH, HASP, HARP, HARM, HAEM, HAES, HAPS, HIPS, HISS, HIST, HEST, HEFT, WEFT, WERT, WERE, WORE, WORN, TORN, TORY, TOWY, COWY, COWL, COOL, FOOL
]
Jumlah node yang dikunjungi: 3087
Waktu eksekusi: 43 ms
```

## 1-AStar

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): MAKE
Masukkan endWord (Pastikan semua huruf besar): FOOL
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [MAKE, FAKE, FACE, FICE, FICO, FINO, FIND, FOND, FOOD, FOOL]
Jumlah node yang dikunjungi: 17
Waktu eksekusi: 8 ms
```

## 2-UCS

```
^Caxelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): GET
Masukkan endWord (Pastikan semua huruf besar): PAL
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Hasil Dari Uniform Cost Search Algorithm: [GET, GEL, GAL, PAL]
Jumlah node yang dikunjungi: 564
Waktu eksekusi: 15 ms
```

# 2-GBFS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): GET
Masukkan endWord (Pastikan semua huruf besar): PAL
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [GET, GEY, GUY, GUV, LUV, LUX, LOX, LOW, LAW, LAY, YAY, YAR, WAR, WAX, ZAX, ZAP, ZIP, ZIN, YIN, YID, YOD, YOU, FOU, FOR, FUR, FUN, FAN, FAT, VAT, VAV, TAV, TAU, SAU, SAL, PAL]
Jumlah node yang dikunjungi: 69
Waktu eksekusi: 9 ms
```

# 2-AStar

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): GET
Masukkan endWord (Pastikan semua huruf besar): PAL
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [GET, PET, PAT, PAL]
Jumlah node yang dikunjungi: 4
Waktu eksekusi: 6 ms
```

# 3-UCS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): LIEGE
Masukkan endWord (Pastikan semua huruf besar): CRUSH
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Hasil Dari Uniform Cost Search Algorithm: [LIEGE, SIEGE, SINGE, SINCE, WINCE, WINCH, WENCH, BENCH, BEACH, BRACH, BRASH, CRASH, CRUSH]
Jumlah node yang dikunjungi: 6554
Waktu eksekusi: 94 ms
```

# 3-GBFS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): libAsisten
Masukkan startWord (Pastikan semua huruf besar): LIEGE
Masukkan endWord (Pastikan semua huruf besar): CRUSH
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [LIEGE, SIEGE, SINGE, SINGS, SINKS, SILKS, SILKY, SILTY, SALTY, SALLY, SADLY, MADLY, MARLY, MARLS, MARKS, MASKS, MUSKS, MUSKY, MUSTY, MISTY, MISTS, MISOS, MILOS, MILES, MILER, MIXER, MIXED, MIRED, MIREX, MUREX, MURES, MURAS, MURAL, MORAL, MORAE, MORSE, MOUSE, LOUSE, LOUSY, LOURY, LOURS, LOUTS, LOOTS, LOOPS, LOOPY, LOONY, MOONY, MOONS, MOORS, DOORS, DORRS, DORPS, GORPS, GORES, GORED, GYRED, GYVED, GYVES, GYBES, GIBES, GIBER, GIVER, GIVEN, RIVEN, RIVET, REVET, REVEL, REFEL, REFER, RIFER, RISER, RISES, RISUS, NISUS, NIDUS, NIDES, NIDED, TIDED, TINED, TINES, TINTS, TILTS, TILTH, FILTH, FILCH, FITCH, WITCH, WATCH, RATCH, RANCH, RANCE, RANGE, RANGY, RANDY, RANDS, RANTS, RAFTS, RAFFS, RUFFS, TUFFS, TUFTS, TUFTY, TUTTY, TUTTI, PUTTI, PUTTO, PUNTO, PINTO, PINTA, PINNA, PENNA, PENNI, BENNI, BENNY, BENDY, BENDS, BENES, BONES, BONEY, BOGEY, DOGEY, DOGES, DOZES, DOZER, DOTER, DOTED, DOPED, DUPED, DUPES, DUKES, DYKES, DYKEY, DIKEY, DIKED, DIVED, WIVED, WITED, WITES, WIZES, WIZEN, WIDEN, WIDER, WADER, WADED, WAXED, WAXEN, WAKEN, WOKEN, WOVEN, DOVEN, DAVEN, DAWEN, DAWED, DAZED, RAZED, RAZES, RAXES, REXES, REGES, LEGES, LEGER, LEVER, LAVER, LAVED, LAYED, PAYED, PAYER, PAYOR, MAYOR, MAYOS, MANOS, MANUS, MAGUS, VAGUS, VAGUE, VALUE, VALVE, VARVE, VERVE, VERSE, PERSE, PEISE, PRISE, PRISS, PRIMS, PRIMP, CRIMP, CRIME, CRIPE, CRAPE, CRAPS, CRAWS, CROWS, CROWN, CLOWN, FLOWN, FLOWS, FLOSS, GLOSS, GLOST, GLOUT, GROUT, GROAT, GREAT, GREET, GREES, GREYS, GRAYS, GRASS, TRASS, TRASH, CRASH, CRUSH]
Jumlah node yang dikunjungi: 474
Waktu eksekusi: 17 ms
```

# 3-AStar

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): a
Masukkan startWord (Pastikan semua huruf besar): LIEGE
Masukkan endWord (Pastikan semua huruf besar): CRUSH
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [LIEGE, SIEGE, SINGE, SINGS, SINKS, KINKS, KINOS, KILOS, KILTS, KISTS, CISTS, CASTS, CARTS, CARDS, CORDS, CORPS, COUPS, COUPE, COUDE, CRUDE, CRUSE, CRUSH]
Jumlah node yang dikunjungi: 95
Waktu eksekusi: 9 ms
```

# 4-UCS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): a
Masukkan startWord (Pastikan semua huruf besar): MAKAN
Ups, katamu masih belum valid. Pastikan semua hurufnya adalah huruf besar, dan berbahasa inggris ya :3 Silakan coba lagi.
Masukkan startWord (Pastikan semua huruf besar): MAYAN
Masukkan endWord (Pastikan semua huruf besar): MINUM
Ups, katamu masih belum valid. Pastikan semua hurufnya adalah huruf besar, panjangnya sama dengan kata awal, dan berbahasa inggris ya :3 Silakan coba lagi.
Masukkan endWord (Pastikan semua huruf besar): MINUS
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Hasil Dari Uniform Cost Search Algorithm: [MAYAN, MAYAS, MANAS, MANUS, MINUS]
Jumlah node yang dikunjungi: 66
Waktu eksekusi: 11 ms
```

# 4-GBFS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): a
Masukkan startWord (Pastikan semua huruf besar): MAYAN
Masukkan endWord (Pastikan semua huruf besar): MINUS
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [MAYAN, MAYAS, MAYOS, MAYOR, MANOR, MINOR, MINER, MINES, MINUS]
Jumlah node yang dikunjungi: 16
Waktu eksekusi: 9 ms
```

# 4-AStar

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): a
Masukkan startWord (Pastikan semua huruf besar): MAYAN
Masukkan endWord (Pastikan semua huruf besar): MINUS
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [MAYAN, MAYAS, MANAS, MINAS, MINUS]
Jumlah node yang dikunjungi: 5
Waktu eksekusi: 7 ms
```

# 5-UCS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): a
Masukkan startWord (Pastikan semua huruf besar): BASEBALL
Masukkan endWord (Pastikan semua huruf besar): BLIZZARD
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Sayang sekali, tidak ditemukan ladder dari BASEBALL ke BLIZZARD :(
Waktu eksekusi: 12 ms
```

# 5-GBFS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): a
Masukkan startWord (Pastikan semua huruf besar): BASEBALL
Masukkan endWord (Pastikan semua huruf besar): BLIZZARD
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Sayang sekali, tidak ditemukan ladder dari BASEBALL ke BLIZZARD :(
Waktu eksekusi: 11 ms
```

# 5-AStar

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): A
Masukkan startWord (Pastikan semua huruf besar): BASEBALL
Masukkan endWord (Pastikan semua huruf besar): BLIZZARD
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Sayang sekali, tidak ditemukan ladder dari BASEBALL ke BLIZZARD :(
Waktu eksekusi: 10 ms
```

# 6-UCS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): words_alpha
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Hasil Dari Uniform Cost Search Algorithm: [SAVAGE, PAVAGE, PAVANE, PAVANS, PAVINS, PAVIES, PARIES, PARSES, PASSES, PISSES, HISSES, HISSED]
Jumlah node yang dikunjungi: 5704
Waktu eksekusi: 120 ms
```

# 6-GBFS

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): b
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [SAVAGE, SAVATE, SACATE, VACATE, VOCATE, LOCATE, LOCALE, LOCULE, LUCULE, CUCULE, CUCULI, CUMULI, FUMULI, FAMULI, HAMULI, HAMULE, HAMBLE, HABBLE, HABILE, LABILE, LABITE, LATITE, LATIVE, SATIVE, SATIVA, SALIVA, SALINA, SALINE, SALITE, SALUTE, SOLUTE, SOLATE, SOLACE, SOLACH, KOLACH, KALACH, PALACH, PALACE, PALATE, PARATE, PARALE, PAROLE, PAROLS, PAROUS, PAREUS, PARENS, PARENT, PATENT, LATENT, LATENS, LATINS, LATINO, BATINO, BATING, BAYING, BAYONG, BARONG, BARONY, BACONY, BACONS, BATONS, BETONS, BETONE, BELONE, BELOVE, BELIVE, BELITE, BEBITE, REBITE, REBUTE, REBUTS, RECUTS, RECURS, RECURL, REFURL, REFUEL, REFEEL, REFEED, REFFED, REEFED, REEFER, REEPER, REAPER, REAPED, REAVED, REAVES, REIVES, REIVER, REITER, RETTER, RETTED, RETIED, RELIED, BELIED, BELTED, BELTER, BELTIR, BELTIE, BERTIE, BERTIN, BERAIN, BETAIN, BETAIL, BETOIL, BESOIL, BESOIN, BESPIN, BESPIT, BESUIT, RESUIT, RESULT, RESALT, RESALE, RESAVE, REPAVE, REPAGE, RECAGE, RECASE, RECAST, RECART, REPART, REPARK, REPACK, REPICK, RELICK, RELICT, RELIST, RELISH, REWISH, RAWISH, RAVISH, LAVISH, LAMISH, SAMISH, SALISH, PALISH, PAPISH, PAPIST, TAPIST, TAPISM, TAOISM, MAOISM, MAOIST, MARIST, MARIET, MARIES, MARTES, MARTEN, MARTIN, MARGIN, MARGIE, MAGGIE, MAGGLE, MANGLE, MANGUE, TANGUE, TANGUN, WANGUN, WANGAN, CANGAN, CANMAN, CARMAN, CARMEN, CARMEL, CARVEL, CARVES, CARSES, CAUSES, CAUSER, CAUTER, CAUTEL, CANTEL, CANTED, CANOED, CANOES, CANONS, CAPONS, CAPONE, CAPOTE, CAPITE, CAPITA, CASITA, CASINA, CASINO, CAMINO, COMINO, COMING, COZING, MOZING, MOVING, ROVING, ROSING, ROSINY, RESINY, RESINS, RESIDS, RESIDE, RESIZE, DESIZE, DENIZE, DANIZE, DANITE, TANITE, TAXITE, TAXINE, TAXING, TAWING, RAWING, RAZING, HAZING, HARING, HIRING, HIVING, WIVING, WITING, WYTING, WYLING, WALING, WANING, WONING, WOOING, LOOING, LOXING, POXING, POLING, VOLING, VOTING, TOTING, TOYING, TRYING, FRYING, FLYING, FLEING, FEEING, TEEING, TIEING, TIMING, RIMING, RICING, VICING, VIKING, MIKING, MAKING, MACING, PACING, PAGING, PAGINE, PARINE, PURINE, PURINS, BURINS, BURIES, BURIER, BURRER, BURREL, BURHEL, BUSHEL, BUSHET, BUSKET, BUSKER, BUSTER, BUSTEE, BUSTLE, BUTTLE, BOTTLE, BOOTLE, BOOTIE, BOOTID, BOOTED, BOOTES, BOOTHS, BROTHS, BROTHY, WROTHY, WROTHE, WRITHE, WRITEE, WRITES, WHITES, WHITEY, WHINEY, WHINER, WAINER, WAITER, WAITED, WAILED, WAWLED, YAWLED, YAWLER, YAWPER, YAUPER, YAUPED, YAPPED, YIPPED, ZIPPED, ZIPPER, TIPPER, TOPPER, TOPPED, TUPPED, PUPPED, PUPPET, PUMPET, PUMPER, PULPER, PULPAR, PULPAL, PALPAL, PALEAL, PALEAE, PALLAE, PALLAS, PILLAS, PILLAR, PELLAR, PEDLAR, PEDLER, PEDDER, PODDER, PODDED, PADDED, PARDED, PARDEE, PARDIE, PARTIE, PARTIS, PASTIS, PASTIL, PASTEL, PASTER, PASTOR, CASTOR, CAPTOR, CEPTOR, CEPTER, CENTER, CENSER, CENSED, CESSED, CUSSED, CUSSES, CUSSOS, CUSTOS, CESTOS, CESTON, TESTON, TESTOR, TESTER, TESTES, TASTES, TASTEN, TAUTEN, TAUTED, TARTED, TARTER, TARTAR, TARTAN, TARZAN, SARZAN, SARWAN, SURWAN, DURWAN, DURIAN, DURION, BURION, BURDON, BURDEN, HURDEN, HARDEN, HARDER, HARMER, HARMED, HARPED, HASPED, HASHED, HASHES, HUSHES, HUSHER, TUSHER, TUSHED, TUSKED, MUSKED, MUSKEG, MASKEG, MASKER, MASSER, MASSEL, MISSEL, MISSET, MISSIT, MISSIS, MIOSIS, MYOSIS, PYOSIS, PTOSIS, PTOSES, PROSES, PROSED, PROWED, PLOWED, PLOYED, PLAYED, PLAYER, PLANER, PLANET, PIANET, PINNET, PINNER, PINKER, PINKEY, PINKLY, TINKLY, TINKLE, TINGLE, WINGLE, WINTLE, WENTLE, KENTLE, KETTLE, KETOLE, KETOSE, SETOSE, SEPOSE, SEPONE, REPONE, REPINE, REPILE, REVILE, REVIVE, REGIVE, REGIME, RETIME, RETIRE, RETARE, RETARD, REWARD, REWARN, REDARN, REDAWN, RESAWN, RESAWS, RESAYS, REPAYS, REPASS, REMASS, REMASH, BEMASH, BEMUSH, BEMUCK, BEMOCK, REMOCK, REDOCK, REDECK, REBECK, REBECS, REBELS, REVELS, REVERS, REVERY, RIVERY, RIMERY, RIMERS, RISERS, MISERS, MISERY, MINERY, WINERY, WINERS, WIVERS, WAVERS, WAXERS, TAXERS, TAWERS, TOWERS, TOWELS, ROWELS, ROWENS, ROWANS, SOWANS, SOWINS, SOZINS, SOZINE, BOZINE, BOVINE, COVINE, COVITE, COXITE, FOXITE, FOSITE, NOSITE, NOSINE, NOMINE, NOMINA, NUMINA, LUMINA, LUMINE, LUPINE, LUPINS, LIPINS, LININS, LINENS, LINENE, PINENE, PIRENE, PYRENE, PYRONE, TYRONE, THRONE, THRONG, THRING, TYRING, TYKING, TSKING, ASKING, ASHING, ACHING, ACTING, ANTING, ANDING, ANDINE, ALDINE, ALDANE, ALKANE, ALKENE, ALLENE, ARLENE, ARLINE, ARSINE, ARSINO, ARSONO, ARSONS, ARGONS, ARGANS, ARGAND, ARRAND, ARRANT, ARRENT, ARPENT, ARPENS, ASPENS, ASPERS, ASPERN, ASTERN, ASTERT, ASTART, ASTARE, ASTATE, ASSATE, ANSATE, ENSATE, ENSAFE, UNSAFE, UNSAME, UNSOME, UNSORE, UNSORT, UNSOFT, UNCOFT, UNCOST, UNCAST, UNCASE, UNCAVE, UNPAVE, UNPALE, UNPILE, UNFILE, UNFILM, INFILM, INFILL, INFALL, INWALL, INWALE, INHALE, INHAME, INHUME, INSUME, INSURE, INDURE, INDUCE, INDUCT, INDICT, INVICT, INVECT, INVEST, INNEST, INNESS, INNERS, INTERS, INTERN, INTURN, UNTURN, UNBURN, UNBURY, UNBUSY, UNBUSH, UNLUSH, UNLUST, UNLIST, UNWIST, UNWISE, UNWIVE, UNLIVE, UNLINE, UNLINK, UNWINK, UNWIND, UNRIND, UNRING, UNRUNG, UNHUNG, UNHANG, UNHAND, UNHARD, UNHERD, UNHEED, UNHUED, UNSUED, UNSPED, UNOPED, UNOWED, UNAWED, SNAWED, SNARED, SNARER, SNAPER, SNIPER, SNIPED, SWIPED, SWIVED, SKIVED, SKIVER, SKITER, SKATER, STATER, STATED, STAYED, SWAYED, SWAYER, SWALER, SWALES, SPALES, SPALLS, SPAILS, SPAITS, SPRITS, SPRITE, SHRITE, SHRINE, SHRINK, SHRANK, SPRANK, SPRANG, STRANG, STRANY, STRAWY, STRAWS, STRAYS, STROYS, STROPS, STREPS, STRESS, SIRESS, SIREES, SPREES, SPREED, SCREED, SCREET, STREET, STREEN, STREWN, STROWN, STROWD, STROUD, STROUT, STROOT, SEROOT, SEROON, SERMON, TERMON, TERMOR, TERMER, TERMES, TERNES, TERNED, TURNED, TURNER, TUNNER, TUNNEL, RUNNEL, RUNNET, RUNLET, NUNLET, NUTLET, OUTLET, OUTLER, OUTHER, WUTHER, WITHER, WITHES, WICHES, RICHES, RICHER, RICKER, RICKEY, TICKEY, TICKEN, WICKEN, WICKED, WINKED, WINKEL, WINDEL, WINDER, WINTER, WISTER, WISTED, WISSED, HISSED]
Jumlah node yang dikunjungi: 1833
Waktu eksekusi: 59 ms
```

# 6-AStar

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): b
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [SAVAGE, RAVAGE, RAMAGE, RAMATE, HAMATE, HAMITE, HALITE, HALIDE, HELIDE, RELIDE, RESIDE, RESIDS, RESINS, RESING, RISING, CISING, COSING, COSINS, COSIES, POSIES, POSSES, PISSES, HISSES, HISSED]
Jumlah node yang dikunjungi: 131
Waktu eksekusi: 13 ms
```

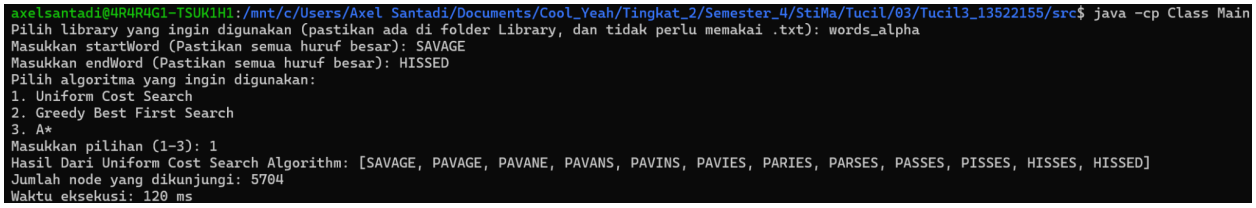
# BAB IV
# Analisis Perbandingan Algoritma

1. Optimalitas

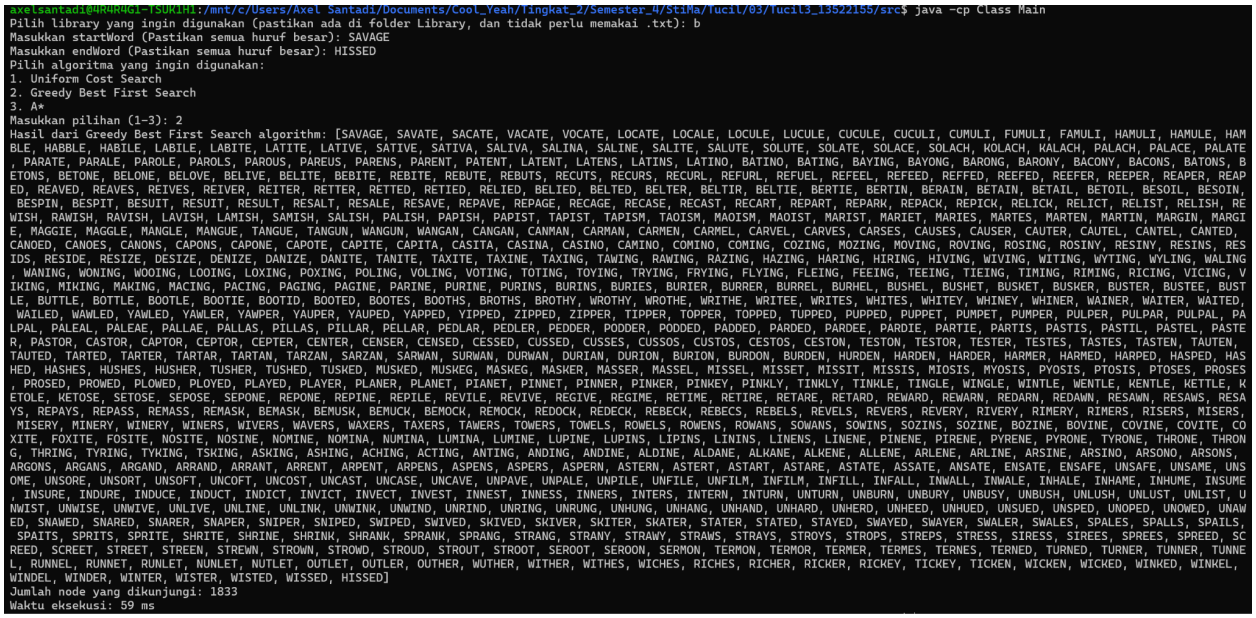
- **UCS (Uniform Cost Search)**: Algoritma UCS menjamin menemukan solusi optimal karena secara iteratif mengeksplorasi node-node dengan biaya terendah terlebih dahulu.
- **Greedy Best First Search**: GBFS tidak menjamin solusi optimal karena hanya memilih node berdasarkan heuristik tanpa mempertimbangkan cost sejauh ini.
- **A* (A Star)**: A menjamin solusi optimal jika heuristik yang digunakan admissible, yaitu tidak pernah melebihi nilai sebenarnya dari biaya untuk mencapai tujuan dari node saat ini.

**Kesimpulan**: UCS dan A* menjamin solusi optimal, sementara GBFS tidak, seperti yang dapat dilihat dari 3 screenshot ini:

UCS:




GBFS:



AStar:

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): b
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [SAVAGE, RAVAGE, RAMAGE, RAMATE, HAMATE, HAMITE, HALITE, HALIDE, HELIDE, RELIDE, RESIDE, RESIDS, RESINS, RESING, RISING, CISING, COSING, COSINS, COSIES, POSIES, POSSES, PISSES, HISSES, HISSED]
Jumlah node yang dikunjungi: 131
Waktu eksekusi: 13 ms
```

## 2. Waktu Eksekusi

- **UCS (Uniform Cost Search)**: Karena harus mengeksplorasi setiap node secara berurutan berdasarkan cost, waktu eksekusi UCS cenderung lebih lama, terutama jika Word Ladder memiliki banyak kemungkinan jalur.
- **Greedy Best First Search**: GBFS dapat lebih cepat daripada UCS karena hanya mempertimbangkan nilai heuristik saat pemilihan node berikutnya.
- **A* (A Star)**: A* dapat lebih cepat daripada UCS karena menggunakan kombinasi cost sejauh ini dan nilai heuristik untuk memandu pencarian.

**Kesimpulan**: A* mungkin menjadi pilihan terbaik dalam hal waktu eksekusi, diikuti oleh GBFS, dan UCS, seperti yang dapat dilihat dari screenshot ini:

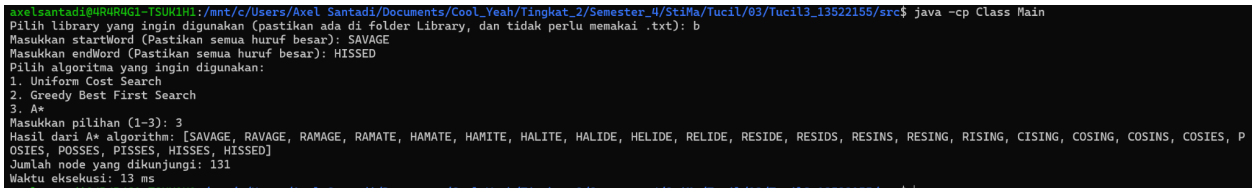
UCS:

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): words_alpha
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 1
Hasil Dari Uniform Cost Search Algorithm: [SAVAGE, PAVAGE, PAVANE, PAVANS, PAVINS, PAVIES, PARIES, PARSES, PASSES, PISSES, HISSES, HISSED]
Jumlah node yang dikunjungi: 5704
Waktu eksekusi: 120 ms
```

GBFS:

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): b
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [SAVAGE, SAVATE, SACATE, VACATE, VOCATE, LOCATE, LOCALE, LOCULE, LUCULE, CUCULE, CUCULI, CUMULI, FUMULI, FAMULI, HAMULI, HAMULE, HAMBLE, HABBLE, HABILE, LABILE, LABITE, LATITE, LATIVE, SATIVE, SATIVA, SALIVA, SALINA, SALINE, SALITE, SALUTE, SOLUTE, SOLATE, SOLACE, SOLACH, KOLACH, KALACH, PALACH, PALACE, PALATE, PARATE, PARALE, PAROLE, PAROLS, PAROUS, PAREUS, PARENS, PARENT, PATENT, LATENT, LATENS, LATINS, LATINO, BATINO, BATING, BAYING, BAYONG, BARONG, BARONY, BACONY, BACONS, BATONS, BETONS, BETONE, BELONE, BELOVE, BELIVE, BELITE, BEBITE, REBITE, REBUTE, REBUTS, RECUTS, RECURS, RECURL, REFURL, REFUEL, REFEEL, REFEED, REFFED, REEFED, REEFER, REEPER, REAPER, REAPED, REAVED, REAVES, REIVES, REIVER, REITER, RETTER, RETTED, RETIED, RELIED, BELIED, BELTED, BELTER, BELTIR, BELTIE, BERTIE, BERTIN, BERAIN, BETAIN, BETAIL, BETOIL, BESOIL, BESOIN, BESPIN, BESPIT, BESUIT, RESUIT, RESULT, RESALT, RESALE, RESAVE, REPAVE, REPAGE, RECAGE, RECASE, RECAST, RECART, REPART, REPARK, REPACK, REPICK, RELICK, RELICT, RELIST, RELISH, REWISH, RAWISH, RAVISH, LAVISH, LAMISH, SAMISH, SALISH, PALISH, PAPISH, PAPIST, TAPIST, TAPISM, TAOISM, MAOISM, MAOIST, MARIST, MARIET, MARIES, MARTES, MARTEN, MARTIN, MARGIN, MARGIE, MAGGIE, MAGGLE, MANGLE, MANGUE, TANGUE, TANGUN, WANGUN, WANGAN, CANGAN, CANMAN, CARMAN, CARMEN, CARMEL, CARVEL, CARVES, CARSES, CAUSES, CAUSER, CAUTER, CAUTEL, CANTEL, CANTED, CANOED, CANOES, CANONS, CAPONS, CAPONE, CAPOTE, CAPITE, CAPITA, CASITA, CASINA, CASINO, CAMINO, COMINO, COMING, COZING, MOZING, MOVING, ROVING, ROSING, ROSINY, RESINY, RESINS, RESIDS, RESIDE, RESIZE, DESIZE, DENIZE, DANIZE, DANITE, TANITE, TAXITE, TAXINE, TAXING, TAWING, RAWING, RAZING, HAZING, HARING, HIRING, HIVING, WIVING, WITING, WYTING, WYLING, WALING, WANING, WONING, WOOING, LOOING, LOXING, POXING, POLING, VOLING, VOTING, TOTING, TOYING, TRYING, FRYING, FLYING, FLEING, FEEING, TEEING, TIEING, TIMING, RIMING, RICING, VICING, VIKING, MIKING, MAKING, MACING, PACING, PAGING, PAGINE, PARINE, PURINE, PURINS, BURINS, BURIES, BURIER, BURRER, BURREL, BURHEL, BUSHEL, BUSHET, BUSKET, BUSKER, BUSTER, BUSTEE, BUSTLE, BUTTLE, BOTTLE, BOOTLE, BOOTIE, BOOTID, BOOTED, BOOTES, BOOTHS, BROTHS, BROTHY, WROTHY, WROTHE, WRITHE, WRITEE, WRITES, WHITES, WHITEY, WHINEY, WHINER, WAINER, WAITER, WAITED, WAILED, WAWLED, YAWLED, YAWLER, YAWPER, YAUPER, YAUPED, YAPPED, YIPPED, ZIPPED, ZIPPER, TIPPER, TOPPER, TOPPED, TUPPED, PUPPED, PUPPET, PUMPET, PUMPER, PULPER, PULPAR, PULPAL, PALPAL, PALEAL, PALEAE, PALLAE, PALLAS, PILLAS, PILLAR, PELLAR, PEDLAR, PEDLER, PEDDER, PODDER, PODDED, PADDED, PARDED, PARDEE, PARDIE, PARTIE, PARTIS, PASTIS, PASTIL, PASTEL, PASTER, PASTOR, CASTOR, CAPTOR, CEPTOR, CEPTER, CENTER, CENSER, CENSED, CESSED, CUSSED, CUSSES, CUSSOS, CUSTOS, CESTOS, CESTON, TESTON, TESTOR, TESTER, TESTES, TASTES, TASTEN, TAUTEN, TAUTED, TARTED, TARTER, TARTAR, TARTAN, TARZAN, SARZAN, SARWAN, SURWAN, DURWAN, DURIAN, DURION, BURION, BURDON, BURDEN, HURDEN, HARDEN, HARDER, HARMER, HARMED, HARPED, HASPED, HASHED, HASHES, HUSHES, HUSHER, TUSHER, TUSHED, TUSKED, MUSKED, MUSKEG, MASKEG, MASKER, MASSER, MASSEL, MISSEL, MISSET, MISSIT, MISSIS, MIOSIS, MYOSIS, PYOSIS, PTOSIS, PTOSES, PROSES, PROSED, PROWED, PLOWED, PLOYED, PLAYED, PLAYER, PLANER, PLANET, PIANET, PINNET, PINNER, PINKER, PINKEY, PINKLY, TINKLY, TINKLE, TINGLE, WINGLE, WINTLE, WENTLE, KENTLE, KETTLE, KETOLE, KETOSE, SETOSE, SEPOSE, SEPONE, REPONE, REPINE, REPILE, REVILE, REVIVE, REGIVE, REGIME, RETIME, RETIRE, RETARE, RETARD, REWARD, REWARN, REDARN, REDAWN, RESAWN, RESAWS, RESAYS, REPAYS, REPASS, REMASS, REMASK, BEMASK, BEMUSK, BEMUCK, BEMOCK, REMOCK, REDOCK, REDECK, REBECK, REBECS, REBELS, REVELS, REVERS, REVERY, RIVERY, RIMERY, RIMERS, RISERS, MISERS, MISERY, MINERY, WINERY, WINERS, WIVERS, WAVERS, WAXERS, TAXERS, TAWERS, TOWERS, TOWELS, ROWELS, ROWENS, ROWANS, SOWANS, SOWINS, SOZINS, SOZINE, BOZINE, BOVINE, COVINE, COVITE, COXITE, FOXITE, FOSITE, NOSITE, NOSINE, NOMINE, NOMINA, NUMINA, LUMINA, LUMINE, LUPINE, LUPINS, LIPINS, LININS, LINENS, LINENE, PINENE, PIRENE, PYRENE, PYRONE, TYRONE, THRONE, THRONG, THRING, TYRING, TYKING, TSKING, ASKING, ASHING, ACHING, ACTING, ANTING, ANDING, ANDINE, ALDINE, ALDANE, ALKANE, ALKENE, ALLENE, ARLENE, ARLINE, ARSINE, ARSINO, ARSONO, ARSONS, ARGONS, ARGANS, ARGAND, ARRAND, ARRANT, ARRENT, ARPENT, ARPENS, ASPENS, ASPERS, ASPERN, ASTERN, ASTERT, ASTART, ASTARE, ASTATE, ASSATE, ANSATE, ENSATE, ENSAFE, UNSAFE, UNSAME, UNSOME, UNSORE, UNSORT, UNSOFT, UNCOFT, UNCOST, UNCAST, UNCASE, UNCAVE, UNPAVE, UNPALE, UNPILE, UNFILE, UNFILM, INFILM, INFILL, INFALL, INWALL, INWALE, INHALE, INHAME, INHUME, INSUME, INSURE, INDURE, INDUCE, INDUCT, INDICT, INVICT, INVECT, INVEST, INNEST, INNESS, INNERS, INTERS, INTERN, INTURN, UNTURN, UNBURN, UNBURY, UNBUSY, UNBUSH, UNLUSH, UNLUST, UNLIST, UNWIST, UNWISE, UNWIVE, UNLIVE, UNLINE, UNLINK, UNWINK, UNWIND, UNRIND, UNRING, UNRUNG, UNHUNG, UNHANG, UNHAND, UNHARD, UNHERD, UNHEED, UNHUED, UNSUED, UNSPED, UNOPED, UNOWED, UNAWED, SNAWED, SNARED, SNARER, SNAPER, SNIPER, SNIPED, SWIPED, SWIVED, SKIVED, SKIVER, SKITER, SKATER, STATER, STATED, STAYED, SWAYED, SWAYER, SWALER, SWALES, SPALES, SPALLS, SPAILS, SPAITS, SPRITS, SPRITE, SHRITE, SHRINE, SHRINK, SHRANK, SPRANK, SPRANG, STRANG, STRANY, STRAWY, STRAWS, STRAYS, STROYS, STROPS, STREPS, STRESS, SIRESS, SIREES, SPREES, SPREED, SCREED, SCREET, STREET, STREEN, STREWN, STROWN, STROWD, STROUD, STROUT, STROOT, SEROOT, SEROON, SERMON, TERMON, TERMOR, TERMER, TERMES, TERNES, TERNED, TURNED, TURNER, TUNNER, TUNNEL, RUNNEL, RUNNET, RUNLET, NUNLET, NUTLET, OUTLET, OUTLER, OUTHER, WUTHER, WITHER, WITHES, WICHES, RICHES, RICHER, RICKER, RICKEY, TICKEY, TICKEN, WICKEN, WICKED, WINKED, WINKEL, WINDEL, WINDER, WINTER, WISTER, WISTED, WISSED, HISSED]
Jumlah node yang dikunjungi: 1833
Waktu eksekusi: 59 ms
```

AStar:

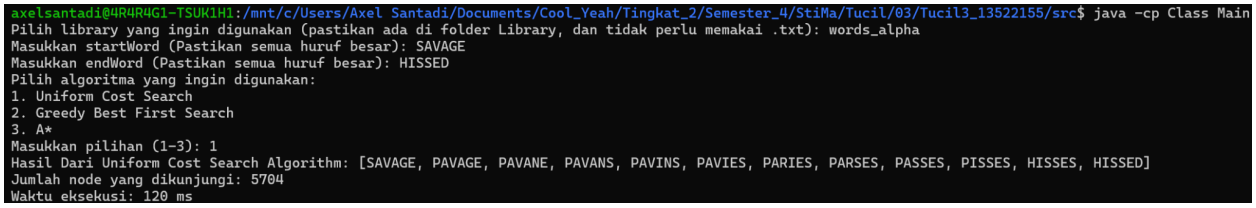



## 3. Memori yang Dibutuhkan

- **UCS (Uniform Cost Search)**: Meskipun UCS hanya perlu menyimpan node-node yang dieksplorasi, namun karena mungkin harus menyimpan banyak node yang memiliki cost rendah, terutama jika Word Ladder memiliki banyak kemungkinan jalur, maka memori yang dibutuhkan bisa menjadi signifikan.
- **Greedy Best First Search**: GBFS memerlukan memori yang cukup besar karena harus menyimpan semua node yang mungkin, terutama jika heuristik yang digunakan tidak efisien dan menyebabkan penumpukan node yang banyak.
- **A* (A Star)**: Meskipun A* menggunakan heuristik untuk memandu pencarian, namun memori yang dibutuhkan bisa menjadi besar karena harus menyimpan informasi tentang semua node yang telah dieksplorasi, terutama jika Word Ladder memiliki banyak kemungkinan jalur.

**Kesimpulan**: Meskipun A* menggunakan heuristik untuk memandu pencarian, namun UCS cenderung membutuhkan memori yang paling sedikit, diikuti oleh GBFS, dan UCS, seperti yang dapat dilihat dari screenshot berikut:

UCS:




GBFS:

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): b
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 2
Hasil dari Greedy Best First Search algorithm: [SAVAGE, SAVATE, SACATE, VACATE, VOCATE, LOCATE, LOCALE, LOCULE, LUCULE, CUCULE, CUCULI, CUMULI, FUMULI, FAMULI, HAMULI, HAMULE, HAMBLE, HABBLE, HABILE, LABILE, LABITE, LATITE, LATIVE, SATIVE, SATIVA, SALIVA, SALINA, SALINE, SALITE, SALUTE, SOLUTE, SOLATE, SOLACE, SOLACH, KOLACH, KALACH, PALACH, PALACE, PALATE, PARATE, PARALE, PAROLE, PAROLS, PAROUS, PAREUS, PARENS, PARENT, PATENT, LATENT, LATENS, LATINS, LATINO, BATINO, BATING, BAYING, BAYONG, BARONG, BARONY, BACONY, BACONS, BATONS, BETONS, BETONE, BELONE, BELOVE, BELIVE, BELITE, BEBITE, REBITE, REBUTE, REBUTS, RECUTS, RECURS, RECURL, REFURL, REFUEL, REFEEL, REFEED, REFFED, REEFED, REEFER, REEPER, REAPER, REAPED, REAVED, REAVES, REIVES, REIVER, REITER, RETTER, RETTED, RETIED, RELIED, BELIED, BELTED, BELTER, BELTIR, BELTIE, BERTIE, BERTIN, BERAIN, BETAIN, BETAIL, BETOIL, BESOIL, BESOIN, BESPIN, BESPIT, BESUIT, RESUIT, RESULT, RESALT, RESALE, RESAVE, REPAVE, REPAGE, RECAGE, RECASE, RECAST, RECART, REPART, REPARK, REPACK, REPICK, RELICK, RELICT, RELIST, RELISH, REWISH, RAWISH, RAVISH, LAVISH, LAMISH, SAMISH, SALISH, PALISH, PAPISH, PAPIST, TAPIST, TAPISM, TAOISM, MAOISM, MAOIST, MARIST, MARIET, MARIES, MARTES, MARTEN, MARTIN, MARGIN, MARGIE, MAGGIE, MAGGLE, MANGLE, MANGUE, TANGUE, TANGUN, WANGUN, WANGAN, CANGAN, CANMAN, CARMAN, CARMEN, CARMEL, CARVEL, CARVES, CARSES, CAUSES, CAUSER, CAUTER, CAUTEL, CANTEL, CANTED, CANOED, CANOES, CANONS, CAPONS, CAPONE, CAPOTE, CAPITE, CAPITA, CASITA, CASINA, CASINO, CAMINO, COMINO, COMING, COZING, MOZING, MOVING, ROVING, ROSING, ROSINY, RESINY, RESINS, RESIDS, RESIDE, RESIZE, DESIZE, DENIZE, DANIZE, DANITE, TANITE, TAXITE, TAXINE, TAXING, TAWING, RAWING, RAZING, HAZING, HARING, HIRING, HIVING, WIVING, WITING, WYTING, WYLING, WALING, WANING, WONING, WOOING, LOOING, LOXING, POXING, POLING, VOLING, VOTING, TOTING, TOYING, TRYING, FRYING, FLYING, FLEING, FEEING, TEEING, TIEING, TIMING, RIMING, RICING, VICING, VIKING, MIKING, MAKING, MACING, PACING, PAGING, PAGINE, PARINE, PURINE, PURINS, BURINS, BURIES, BURIER, BURRER, BURREL, BURHEL, BUSHEL, BUSHET, BUSKET, BUSKER, BUSTER, BUSTEE, BUSTLE, BUTTLE, BOTTLE, BOOTLE, BOOTIE, BOOTID, BOOTED, BOOTES, BOOTHS, BROTHS, BROTHY, WROTHY, WROTHE, WRITHE, WRITEE, WRITES, WHITES, WHITEY, WHINEY, WHINER, WAINER, WAITER, WAITED, WAILED, WAWLED, YAWLED, YAWLER, YAWPER, YAUPER, YAUPED, YAPPED, YIPPED, ZIPPED, ZIPPER, TIPPER, TOPPER, TOPPED, TUPPED, PUPPED, PUPPET, PUMPET, PUMPER, PULPER, PULPAR, PULPAL, PALPAL, PALEAL, PALEAE, PALLAE, PALLAS, PILLAS, PILLAR, PELLAR, PEDLAR, PEDLER, PEDDER, PODDER, PODDED, PADDED, PARDED, PARDEE, PARDIE, PARTIE, PARTIS, PASTIS, PASTIL, PASTEL, PASTER, PASTOR, CASTOR, CAPTOR, CEPTOR, CEPTER, CENTER, CENSER, CENSED, CESSED, CUSSED, CUSSES, CUSSOS, CUSTOS, CESTOS, CESTON, TESTON, TESTOR, TESTER, TESTES, TASTES, TASTEN, TAUTEN, TAUTED, TARTED, TARTER, TARTAR, TARTAN, TARZAN, SARZAN, SARWAN, SURWAN, DURWAN, DURIAN, DURION, BURION, BURDON, BURDEN, HURDEN, HARDEN, HARDER, HARMER, HARMED, HARPED, HASPED, HASHED, HASHES, HUSHES, HUSHER, TUSHER, TUSHED, TUSKED, MUSKED, MUSKEG, MASKEG, MASKER, MASSER, MASSEL, MISSEL, MISSET, MISSIT, MISSIS, MIOSIS, MYOSIS, PYOSIS, PTOSIS, PTOSES, PROSES, PROSED, PROWED, PLOWED, PLOYED, PLAYED, PLAYER, PLANER, PLANET, PIANET, PINNET, PINNER, PINKER, PINKEY, PINKLY, TINKLY, TINKLE, TINGLE, WINGLE, WINTLE, WENTLE, KENTLE, KETTLE, KETOLE, KETOSE, SETOSE, SEPOSE, SEPONE, REPONE, REPINE, REPILE, REVILE, REVIVE, REGIVE, REGIME, RETIME, RETIRE, RETARE, RETARD, REWARD, REWARN, REDARN, REDAWN, RESAWN, RESAWS, RESAYS, REPAYS, REPASS, REMASS, REMASK, BEMASK, BEMUSK, BEMUCK, BEMOCK, REMOCK, REDOCK, REDECK, REBECK, REBECS, REBELS, REVELS, REVERS, REVERY, RIVERY, RIMERY, RIMERS, RISERS, MISERS, MISERY, MINERY, WINERY, WINERS, WIVERS, WAVERS, WAXERS, TAXERS, TAWERS, TOWERS, TOWELS, ROWELS, ROWENS, ROWANS, SOWANS, SOWINS, SOZINS, SOZINE, BOZINE, BOVINE, COVINE, COVITE, COXITE, FOXITE, FOSITE, NOSITE, NOSINE, NOMINE, NOMINA, NUMINA, LUMINA, LUMINE, LUPINE, LUPINS, LIPINS, LININS, LINENS, LINENE, PINENE, PIRENE, PYRENE, PYRONE, TYRONE, THRONE, THRONG, THRING, TYRING, TYWING, TSKING, ASKING, ASHING, ACHING, ACTING, ANTING, ANDING, ANDINE, ALDINE, ALDANE, ALKANE, ALKENE, ALLENE, ARLENE, ARLINE, ARSINE, ARSINO, ARSONO, ARSONS, ARGONS, ARGANS, ARGAND, ARRAND, ARRANT, ARRENT, ARPENT, ARPENS, ASPENS, ASPERS, ASPERN, ASTERN, ASTERT, ASTART, ASTARE, ASSATE, ANSATE, ENSATE, ENSAFE, UNSAFE, UNSAME, UNSOME, UNSORE, UNSORT, UNSOFT, UNCOFT, UNCOST, UNCAST, UNCASE, UNCAVE, UNPAVE, UNPALE, UNPILE, UNFILE, UNFILM, INFILM, INFILL, INFALL, INWALL, INWALE, INHALE, INHAME, INHUME, INSUME, INSURE, INDURE, INDUCE, INDUCT, INDICT, INVICT, INVECT, INVEST, INNEST, INNESS, INNERS, INTERS, INTERN, INTURN, UNTURN, UNBURN, UNBURY, UNBUSY, UNBUSH, UNLUSH, UNLUST, UNLIST, UNWIST, UNWISE, UNWIVE, UNLIVE, UNLINE, UNLINK, UNWINK, UNWIND, UNRIND, UNRING, UNRUNG, UNHUNG, UNHANG, UNHAND, UNHARD, UNHERD, UNHEED, UNHUED, UNSUED, UNSPED, UNOPED, UNOWED, UNAWED, SNAWED, SNARED, SNARER, SNAPER, SNIPER, SNIPED, SWIPED, SWIVED, SKIVED, SKIVER, SKITER, SKATER, STATER, STATED, STAYED, SWAYED, SWAYER, SWALER, SWALES, SPALES, SPALLS, SPAILS, SPAITS, SPRITS, SPRITE, SHRITE, SHRINE, SHRINK, SHRANK, SPRANK, SPRANG, STRANG, STRANY, STRAWY, STRAWS, STRAYS, STROYS, STROPS, STREPS, STRESS, SIRESS, SIREES, SPREES, SPREED, SCREED, SCREET, STREET, STREEN, STREWN, STROWN, STROWD, STROUD, STROUT, STROOT, SEROOT, SEROON, SERMON, TERMON, TERMOR, TERMER, TERMES, TERNES, TERNED, TURNED, TURNER, TUNNER, TUNNEL, RUNNEL, RUNNET, RUNLET, NUNLET, NUTLET, OUTLET, OUTLER, OUTHER, WUTHER, WITHER, WITHES, WICHES, RICHES, RICHER, RICKER, RICKEY, TICKEY, TICKEN, WICKEN, WICKED, WINKED, WINKEL, WINDEL, WINDER, WINTER, WISTER, WISTED, WISSED, HISSED]
Jumlah node yang dikunjungi: 1833
Waktu eksekusi: 59 ms
```

AStar:

```
axelsantadi@4R4R4G1-TSUK1H1:/mnt/c/Users/Axel Santadi/Documents/Cool_Yeah/Tingkat_2/Semester_4/StiMa/Tucil/03/Tucil3_13522155/src$ java -cp Class Main
Pilih library yang ingin digunakan (pastikan ada di folder Library, dan tidak perlu memakai .txt): b
Masukkan startWord (Pastikan semua huruf besar): SAVAGE
Masukkan endWord (Pastikan semua huruf besar): HISSED
Pilih algoritma yang ingin digunakan:
1. Uniform Cost Search
2. Greedy Best First Search
3. A*
Masukkan pilihan (1-3): 3
Hasil dari A* algorithm: [SAVAGE, RAVAGE, RAMAGE, RAMATE, HAMATE, HAMITE, HALITE, HALIDE, HELIDE, RELIDE, RESIDE, RESIDS, RESINS, RESING, RISING, CISING, COSING, COSINS, COSIES, POSIES, POSSES, PISSES, HISSES, HISSED]
Jumlah node yang dikunjungi: 131
Waktu eksekusi: 13 ms
```

# BAB V

# Lampiran

Repository :

https://github.com/AxelSantadi/Tucil3_13522155

Tabel Pengerjaan:

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