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

Tugas Kecil 2 IF2211 Strategi Algoritma
Semester II tahun 2023/2024

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

Batas pengumpulan : Hari Selasa, 7 Mei 2024 pukul 12.59 WIB (sebelum kelas kuliah)
Arsip pengumpulan :

- *Source* program yang dapat dijalankan disertai README
- Laporan (*soft copy*)

Deskripsi Tugas

Word ladder (juga dikenal sebagai *Doublets, word-links, change-the-word puzzles, paragrams, laddergrams*, atau *word golf*) adalah salah satu permainan kata yang terkenal bagi seluruh kalangan. *Word ladder* ditemukan oleh Lewis Carroll, seorang penulis dan matematikawan, pada tahun 1877. Pada permainan ini, pemain diberikan dua kata yang disebut sebagai *start word* dan *end word*. Untuk memenangkan permainan, pemain harus menemukan rantai kata yang dapat menghubungkan antara *start word* dan *end word*. Banyaknya huruf pada *start word* dan *end word* selalu sama. Tiap kata yang berdekatan dalam rantai kata tersebut hanya boleh berbeda satu huruf saja. Pada permainan ini, diharapkan solusi optimal, yaitu solusi yang meminimalkan banyaknya kata yang dimasukkan pada rantai kata. Berikut adalah ilustrasi serta aturan permainan.

Gambar 1. Ilustrasi dan Peraturan Permainan *Word Ladder*

(Sumber: https://wordwormdormdork.com/)

Permainannya cukup sederhana bukan? Jika belum paham dengan peraturan permainannya, cobalah untuk memainkan permainannya pada link sumber di atas. Jika sudah paham dengan permainannya, sekarang adalah waktunya kalian untuk membuat sebuah solver permainan tersebut dengan harapan kita dapat menemukan solusi paling optimal untuk menyelesaikan permainan *Word Ladder* ini.

Spesifikasi Tugas Kecil 3

- Buatlah program dalam bahasa **Java** berbasis **CLI** (*Command Line Interface*) – bonus jika menggunakan GUI – yang dapat menemukan solusi permainan *word ladder* menggunakan algoritma **UCS, *Greedy Best First Search*, dan A***.
- Kata-kata yang dapat dimasukkan harus berbahasa **Inggris**. Cara kalian melakukan validasi sebuah kata dibebaskan, selama kata-kata tersebut benar terdapat pada *dictionary* dan proses validasi tersebut tidak memakan waktu yang terlalu lama. ● Tugas wajib dikerjakan secara individu.
- ***Input*** **:**

  Format masukan **dibebaskan**, dengan catatan dijelaskan pada README dan laporan. Komponen yang perlu menjadi masukan yaitu.

  1. *Start word* dan *end word.* Program harus bisa menangani berbagai panjang kata (tidak hanya kata dengan 4 huruf saja seperti Gambar 1)
  2. Pilihan algoritma yang digunakan (UCS, *Greedy Best First Search,* atau A*)
- ***Output*** **:**

  Berikut adalah luaran dari program yang diekspektasikan.

  1. *Path* yang dihasilkan dari *start word* ke *end word* (cukup 1 *path* saja)
  2. Banyaknya *node* yang dikunjungi
  3. Waktu eksekusi program
- **Bonus :**

  Pastikan sudah mengerjakan spesifikasi wajib sebelum mengerjakan bonus. 1. Program dapat berjalan dengan GUI (*Graphical User Interface*) – silakan berkreasi dalam membuat tampilan GUI untuk tucil ini. Untuk kakas GUI dibebaskan asalkan program algoritma UCS, *Greedy Best First Search*, dan A* dalam bahasa Java.
- **Laporan :**

  Berkas laporan yang dikumpulkan adalah laporan dalam bentuk PDF yang setidaknya berisi:

  1. Analisis dan implementasi dalam algoritma UCS, *Greedy Best First Search*, dan A* (berisi deskripsi langkah-langkahnya, **bukan notasi pseudocode**). Analisis juga perlu memuat jawaban dari pertanyaan-pertanyaan berikut.
     - Definisi dari *f(n)* dan *g(n)*, sesuai dengan salindia kuliah.
     - Apakah heuristik yang digunakan pada algoritma A* *admissible*? Jelaskan sesuai definisi *admissible* dari salindia kuliah.

- Pada kasus *word ladder*, apakah algoritma UCS sama dengan BFS? (dalam artian urutan *node* yang dibangkitkan dan *path* yang dihasilkan sama)
- Secara teoritis, apakah algoritma A* lebih efisien dibandingkan dengan algoritma UCS pada kasus *word ladder*?
- Secara teoritis, apakah algoritma *Greedy Best First Search* menjamin solusi optimal untuk persoalan *word ladder*?

2. *Source code* program implementasi keduanya dalam bahasa pemrograman Java, disertai penjelasan *class* dan *method* yang dibuat.
3. Tangkapan layar yang memperlihatkan *input* dan *output* (minimal 6 contoh untuk masing-masing algoritma).
4. Hasil analisis perbandingan solusi UCS, *Greedy Best First Search*, dan A*. Analisis mencakup **optimalitas, waktu eksekusi, serta memori yang dibutuhkan** disertai dengan bukti pendukung dari hasil tangkapan layar.
5. Penjelasan mengenai implementasi bonus jika mengerjakan.
6. Pranala ke *repository* yang berisi kode program.

- **Struktur Kode :**

  Program disimpan dalam repository yang bernama Tucil3_NIM

  1. Folder **src** berisi *source code*.
  2. Folder **bin** berisi *executable* (jika ada).
  3. Folder **test** berisi data uji yang digunakan pada laporan.
  4. Folder **doc** berisi laporan tugas kecil (dalam bentuk PDF).
  5. **README** yang minimal berisi:
     a. Deskripsi singkat program yang dibuat.
     b. *Requirement* program dan instalasi tertentu bila ada.
     c. Cara mengkompilasi program (bila dikompilasi).
     d. Cara menjalankan dan menggunakan program.
     e. Identitas pembuat program.

  Silahkan gunakan template berikut, sebagai referensi untuk implementasi struktur dalam README.

- Laporan dikumpulkan hari **Selasa, 7 Mei 2024** pada alamat Google Form berikut paling lambat pukul **12.59** (sebelum kelas kuliah):
  https://bit.ly/tucil3stima24
  LInk Pengumpulan Tucil 3 (alternatif jika bit.ly tidak dapat diakses)
- Adapun pertanyaan terkait tugas kecil ini bisa disampaikan melalui QnA berikut:
  https://bit.ly/qnastima24

Perhatikan:

- **Dilarang keras** *copy paste* program dari internet (AI, repository lain, ataupun teman). Program harus dibuat sendiri, kecurangan akan mengakibatkan nilai tugas menjadi nol.

● Pastikan program **dapat setidaknya dikompilasi** pada *windows* dan *linux*.
● Apabila program **tidak bisa dijalankan** maka tidak akan dinilai oleh asisten. ● Pastikan ***repository* program sudah *public*** saat melewati **deadline**. ● Pasti kan penamaan ***repository*** dan **laporan** sesuai dengan **format**. ● Tambahkan *checklist* (centang dengan ✓) berikut pada bagian lampiran laporan Anda.

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		

“Kak, kok gampang banget tucilnya?”

“Biar kalian bisa nyantai di antara 2 tubes :D”