Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Analisis Perbandingan Algoritma Longest Common Subsequence Dengan Simple Hill Climbing Pada Permainan TTS

Yardani*

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: yardani43@gmail.com

Abstrak−Teka-teki silang merupakan sebuah permainan untuk mengasah otak. Teka-teki silang dapat dimanfaatkan dalam bidang pendidikan untuk menguji kemampuan anak didik, sehingga pertanyaan-pertanyaan yang digunakan disesuaikan dengan kebutuhan saja. Tapi akan sulit apabila jumlah pertanyaan yang digunakan banyak, karena untuk membuat sebuah teka-teki silang dari sekumpulan pertanyaan, kombinasi yang mungkin dari kata yang ada sampai terbentuk sebuah TTS. Penelitian dalam optimasi pengisian ruang-ruang kosong dalam papan matriks teka-teki silang tersebut tentunya diperlukan algoritma dengan menggunakan perbandingan algoritma Longest Common Subsequence (LCS) dan algoritma Simple Hill Climbing. Dengan melakukan perbandingan performansi dari algoritma Longest Common Subsequence (LCS) dan algoritma Simple Hill Climbing maka akan dapat diketahui cara kerja dan performansi dalam kecepatan dan ketepatan dari kedua algoritma tersebut. Pertanyaan-pertanyaan yang ditampilkan pada aplikasi teka-teki silang tersebut diperoleh dari database access secara acak oleh program. Teka-teki silang yang dibentuk merupakan kumpulan jawaban dari pertanyaan yang digunakan, jawaban-jawaban pertanyaan disusun sedemikian rupa sehingga saling berpotongan satu sama lain. Pada papan matriks ini nantinya jawaban dari pertanyaan akan diletakkan untuk dibentuk menjadi sebuah teka-teki silang. Setiap huruf dari jawaban akan menempati satu cell pada papan matriks. Jumlah pertanyaan merupakan jumlah dari pertanyaan yang akan digunakan dalam permainan teka-teki silang. Selain menerima masukan dari pengguna, aplikasi juga memberikan keluaran kepada pengguna berupa teka-teki silang dan jawaban yang benar dari TTS tersebut.

Kata Kunci: Teka-Teki Silang; LCS; Simple Hill Climbing

Abstract−Crossword is a game to sharpen the brain. Crossword puzzles can be used in the field of education to test students' abilities, so that the questions used are adapted to their needs. But it will be difficult if a large number of questions are used, because to make a crossword puzzle from a set of questions, the possible combinations of existing words form a crossword puzzle. Research in optimizing the filling of empty spaces in the crossword matrix board certainly requires an algorithm using a comparison of the Longest Common Subsequence (LCS) algorithm and the Simple Hill Climbing algorithm. By comparing the performance of the Longest Common Subsequence (LCS) algorithm and the Simple Hill Climbing algorithm, it will be known how the work and performance in terms of speed and accuracy of the two algorithms work. The questions displayed in the crossword application are obtained from the database access randomly by the program. The crossword puzzle that is formed is a collection of answers to the questions used, the answers to the questions are arranged in such a way that they intersect with one another. On this matrix board, the answers to the questions will be placed to form a crossword puzzle. Each letter of the answer will occupy one cell on the matrix board. The number of questions is the number of questions that will be used in the crossword game. Apart from receiving input from the user, the application also provides output to the user in the form of a crossword puzzle and the correct answer from the TTS.

Keywords: Crossword Puzzles; LCS; Simple Hill Climbing

1. PENDAHULUAN

Permainan teka teki silang ini sering dijumpai pada berbagai media cetak. Namun, permainan ini terkadang sulit untuk dimainkan karena tidak ada bantuan yang dapat digunakan untuk mencari solusi dari permainan. Selain itu, hingga saat ini, masih sangat sedikit permainan teka teki silang yang dapat dimainkan pada komputer dengan soal dari teka teki silang yang dapat diambil dari sebuah sumber data.[1]

Teka-teki silang merupakan sebuah permainan untuk mengasah otak. Teka-teki silang dapat dimanfaatkan dalam bidang pendidikan untuk menguji kemampuan anak didik, sehingga pertanyaan-pertanyaan yang digunakan disesuaikan dengan kebutuhan saja. Tapi akan sulit apabila jumlah pertanyaan yang digunakan banyak, karena untuk membuat sebuah teka-teki silang dari sekumpulan pertanyaan, kombinasi yang mungkin dari kata yang ada sampai terbentuk sebuah TTS. Semakin banyak jumlah pertanyaan, maka semakin banyak kombinasi yang harus dicoba[2]. Masalah optimasi dengan cara memaksimumkan pengisian ruang-ruang kosong, untuk memecahkan masalah optimasi tersebut tentunya diperlukan algoritma. Tujuan teka-teki silang untuk mengasah otak yang dapat dimanfaatkan dalam kalangan masyarakat khususnya dalam bidang pendidikan untuk menguji kemampuan anak didik, sehingga pertanyaan-pertanyaan yang digunakan disesuaikan dengan kebutuhan saja. Tapi akan sulit apabila jumlah pertanyaan yang digunakan banyak, karena untuk membuat sebuah teka-teki silang dari sekumpulan pertanyaan.

Mengatasi masalah kesulitan apabila jumlah pertanyaan yang banyak untuk menyelesaikan permasalahan dari optimasi kombinasi kata yang ada sampai terbentuk TTS dari banyaknya sekumpulan pertanyaan dengan menganalisis algoritma yang cukup sederhana dengan perbandingan metode *Longest Common Subsequence* (LCS) dan *Simple Hill Climbing* karena telah banyak diterapkan dalam berbagai aplikasi.

Permainan teka teki silang ini terkadang sulit untuk dimainkan karena tidak ada bantuan yang dapat digunakan untuk mencari solusi dari permainan. Dalam penyelesaian game Teka Teki Silang (TTS) terdapat 3 teknik teknik, seperti *string matching* merupakan pencocokkan antara string yang dimasukkan dengan string yang tersimpan dalam *database* dan *comparison methods* merupakan metode perbandingan dengan menggunakan *If condition*, dan metode *longest common subsequence* (LCS) yaitu merupakan pencarian yang terpanjang pada rangkaian sub kata.[3] Optimasi, penerapan algoritma *simple hill climbing* digunakan untuk mendapatkan suatu pola penyusunan kata dalam ruang-ruang kosong yang paling optimal.

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Untuk mengetahui optimal atau tidaknya pengisian ruang kosong tersebut, dilakukan dengan melihat banyaknya kombinasi kata yang dapat dibuat atau dapat juga dilihat dari banyaknya sisa ruang kosong teka-teki silang tersebut. Jadi, semakin banyak dan baik kombinasi kata yang dibuat untuk mengisi ruang-ruang kosong, maka semakin optimal pengisian tersebut.[1]

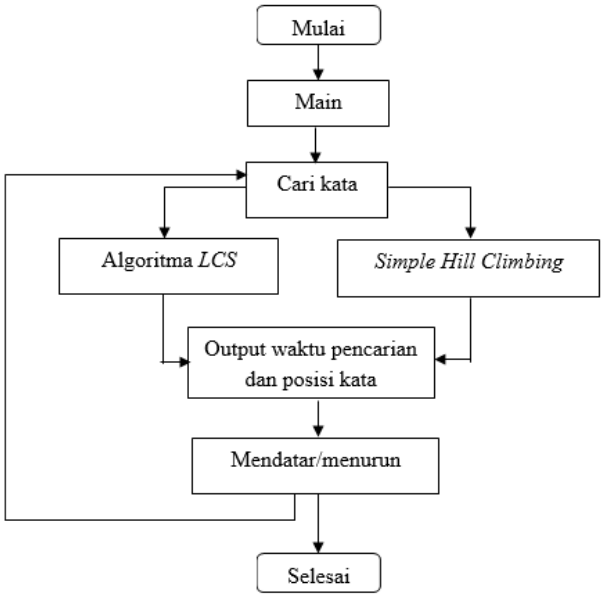
Algoritma pencarian (*searching*) berdasarkan cara kerja melalui mekanisme seleksi alam dan genetik. Tujuannya untuk menentukan struktur-struktur yang disebut individu berkualitas tinggi di dalam suatu domain yang disebut populasi untuk mendapatkan solusi persoalan. Hal ini dikarenakan algoritma genetika tidak berhasil menemukan solusi sesuai dengan soal yang terpilih secara acak. Kemungkinan ini bisa terjadi bila ternyata jawaban soal yang akan digunakan dalam TTS tidak dapat disusun.[4]

Dalam permainan Teka Teki Silang (TTS) terdapat algoritma dalam penyelesaian masalah kombinatorial dengan menganalisis perbandingan algoritma *longest common subsequence* (LCS) dengan Algoritma *simple hill climbing* merupakan algoritma komputasi yang diinspirasi teori evolusi yang kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan membandingkan kedua metode tersebut untuk mencari seperti *string matching* pencocokkan sub kata pada permainan Teka Teki Silan (TTS) sedangkan Algoritma *simple hill climbing* adalah salah satu metode dari sekian banyak metode kecerdasan buatan untuk menyelesaikan permasalahan optimasi. Karena algoritmanya yang cukup sederhana, metode *hill climbing* telah banyak diterapkan dalam berbagai aplikasi. Disamping itu metode *hill climbing* juga mengefisienkan penggunaan memori yang besar. Dengan menggunakan Algoritma *simple hill climbing*, maka aplikasi ini bertujuan untuk mempermudah dalam membuat teka-teki silang dengan pertanyaan-pertanyaan yang dapat ditentukan sendiri oleh pengguna.

2. METODOLOGI PENELITIAN

2.1 Metodologi Penelitian

Dengan metode penelitian dengan tujuan agar suatu cara atau jalan untuk memperoleh kembali pemecahan terhadap segala permasalahan. Dengan rincian sebagai alat bantu analisis dan bahan, materi dan urutan alur penelitian yang dibuat menjadi secara sistematis. Dalam analisis yang digunakan metode yang dilakukan untuk penelitian adalah untuk tujuan menganalisis cara kerja metode *Longest Common Subsequence (LCS)* dan algoritma *Simple Hill Climbing* terhadap kasus optimasi pada permainan teka-teki silang yang dikhususkan pada proses pencarian kata.



Gambar 1. Pencarian Kata

2.2 Teka Teki Silang

TTS merupakan sebuah teknik media pembelajaran yang menarik dikarenakan mengandung unsur permainan, hiburan dan dapat dilakukan secara santai dengan berbagai variasi. TTS dapat juga dikembangkan intuisis siswa untuk memahami lebih banyak kosa kata karena adanya unsur tantangan yang menimbulkan rasa penasaran. Dengan demikian, selain dapat meningkatkan perbendaharaan kata siswa juga dapat dipahami secara mendalam serta merupakan teknik mengasah ketajaman berfikir.[1]

TTS dibentuk sebuah kumpulan jawaban dari pertanyaan-pertanyaan yang akan digunakan. Jawaban-jawaban dari pertanyaan disusun sedemikian rupa sehingga saling berpotongan satu sama lain. Untuk membentuk sekumpulan jawaban pertanyaan menjadi teka-teki silang digunakan sebuah papan matriks berdimensi dua. Pola papan matriks dianalogikan seperti sebuah papan catur. Setiap kotak (*cell*). Papan matriks ini nantinya jawaban dari pertanyaan akan diletakkan untuk dibentuk menjadi sebuah teka-teki silang. Setiap huruf dari jawaban akan menempati satu *cell* pada papan matriks.[3]

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

2.3 LCS (Longest Common Subsequence)

LCS merupakan sebuah algoritma yang mendasar untuk komputer dalam memproses informasi, karena sebuah program komputer adalah sebuah algoritma yang memberitahukan kepada komputer langkah-langkah spesifik yang akan dijalankan untuk melakukan pekerjaan tertentu. Kesalahan dalam merancang algoritma untuk menyelesaikan suatu problema dapat menyebabkan program gagal dalam implementasinya. Untuk melakukan string comparison, maka dapat diterapkan algoritma LCS. Sequence artinya urutan, deretan atau rangkaian. Problem LCS dapat dideskripsikan sebagai berikut, diberikan dua buah himpunan teks (sequence) dan sasarannya adalah mencari common subsequence yang terpanjang dari kedua himpunan sequence itu tersebut.[5]

LCS mempunyai tahapan diantaranya dengan mencari nilai dari setiap perbandingan karakter *sequence*. Ini dapat dilihat pada algoritma LCS yang memulai pencarian nilai dari kotak kiri paling atas menuju kotak kanan paling bawah, algoritma tidak akan kembali ke kotak yang telah dicari nilainya. Hasil pencarian dari kotak sekarang akan dipakai pada kotak yang berikutnya. Ini mengindikasikan setiap *subproblema* saling berhubungan satu sama lain. Setelah tabel LCS terbentuk, algoritma LCS memulai pencarian solusi optimal dari kotak kanan paling bawah menuju kotak paling atas. Karakter ke-1 dari *sequence* A dan karakter ke-1 dari *sequence* B.

1. Pergerakan kotak kosong kebawah : if x ≠(tidak sama dengan)1,1then (x+1,y)
2. Pergerakan kotak kosong kekiri : if x ≠(tidak sama dengan)1,2then (x+1,y)
3. Pergerakan kotak kosong keatas : if x ≠(tidak sama dengan)1,3then (x-1,y)

Aturan / Oprator : Posisi kotak kosong (x,y)
x = baris kotak kosong
y = kolom kotak kosong

Sebuah *sequence* Z disebut sebagai *common subsequence* dari dua buah *sequence* X dan Y, jika Z adalah *subsequence* dari X dan Y. LCS adalah *common subsequence* terpanjang dari dua buah *sequence* tersebut. Contoh LCS dari *sequence* 'abcdgh' dan *sequence* 'aedfhr' adalah *subsequence* 'adh' sepanjang 3 karakter. (Cormen, et. al, *Introduction to Algorithms*, 2011: 198). Penyelesaian dari masalah ini dapat dicari dengan algoritma *LCS* yang menggunakan konsep *dynamic programming*.[5] Sifat-sifat*dynamic programming* (DP) yang diterapkan pada algoritma *LCS* adalah:

1. DP membagi problema menjadi beberapa *subproblema*.
   Algoritma *LCS* membagi masing-masing *sequence* menjadi karakter-karakter.
2. DP mencari dan menyimpan penyelesaian setiap subproblema dalam tabel.
   Algoritma *LCS* membandingkan karakter *sequence*-1 dan *sequence*-2, mencari nilai dari hasil perbandingan sesuai dengan algoritma *LCS* dan menyimpan hasil perhitungan dalam kotak-kotak pada tabel *LCS*.
3. DP menyelesaikan setiap *subproblema* hanya sekali dan setiap subproblema saling berhubungan satu sama lain.
4. DP berfungsi menyelesaikan setiap *subproblema* hanya sekali dan setiap *subproblema* saling berhubungan satu sama lain.
5. DP mencari solusi optimal dengan pola *bottom-up*.
   Mencari solusi optimal dengan pola *bottom-up*. Apabila tabel *LCS* terbentuk, dengan memulai pencarian solusi optimal dari kotak kanan paling bawah menuju kotak paling atas.

2.4 Simple Hill Climbing

Simple Hill Climbing digunakan jika terdapat suatu fungsi *heuristic* yang untuk mengevaluasi *state*. *Simple Hill Climbing*, secara sederhana, langsung memilih *new state* yang memiliki jalur yang lebih baik ("curam") daripada jalur-jalur sebelumnya tanpa memperhitungkan jalur-jalur lain yang lebih "curam". Sedangkan *Steepest Ascent Hill Climbing*, sesuai dengan namanya, akan mengevaluasi semua *state* yang berada dibawah *current state* dan memilih *state* dengan jalur paling "curam". Dasar penyelesaian permasalahan dengan mengikuti langkah berikut ini[1] :

1. Memulai dari keadaan awal, lakukan pengujian: jika merupakan tujuan, maka berhenti; dan jika tidak, lanjutkan dengan keadaan sekarang sebagai keadaan awal.
2. Selanjutnya lakukan langkah tersebut sampai solusi ditemukan atau sampai tidak ada perator baru yang akan diaplikasikan pada keadaan sekarang
3. Mencari operator yang belum digunakan; untuk mendapatkan keadaan baru.
4. Mengevaluasi keadaan baru :
   a. Apabila keadaan baru merupakan tujuan, keluar.
   b. Apabila bukan tujuan, maka nilai lebih baik daripada keadaan yang sekarang, maka jadikan keadaan baru tersebut menjadi keadaan sekarang.
   c. Dan apabila keadaan baru tidak lebih baik dari pada keadaan sekarang, maka lanjutkan iterasi. Ada 3 masalah yang mungkin, yaitu:
      1) Metode akan berhenti kalau mencapai nilai optimum lokal.
      2) Urutan penggunaan operator akan sangat berpengaruh apabila pada penemuan solusi.
      3) Tidak diizinkan untuk melihat satupun langkah sebelumnya.

Simple Hill Climbing secara sederhana, memilih *new state* yang memiliki jalur yang lebih baik ("curam") dari pada jalur-jalur sebelumnya tanpa memperhitungkan jalur-jalur lain yang lebih "curam". Sedangkan *Steepest Ascent Hill*

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Climbing, sesuai dengan namanya, akan mengevaluasi semua *state* yang berada dibawah *current state* dan memilih *state* dengan jalur paling "curam". Ruang keadaan dimisalkan :
x = baris = [1, 3, 8, 9]
3 Kotak mendatar NIL
8 Kotak mendatar AKLAMASI
9 Kotak mendatar DEMOKRASI
y = kolom = [1, 4, 6]
3 Kotak menurun INDO
4 Kotak menurun LIDI
6 Kotak menurun PREMIS
Aturan / Opreator : Posisi kotak yang kosong (*x,y*)
x = baris pada kotak kosong
y = kolom pada kotak kosong
1. Pergerakan kotak kosong ke atas : *if* $x > 1$ *then* (*x-1, y*)
2. Pergerakan kotak kosong ke bawah : *if* $x < 3$ *then* (*x+1, y*)
3. Pergerakan kotak kosong kekanan *: if* $y < 3$ *then* (*x, y* +1)
4. Gerakkan kotak kosong kekiri : *if* $y > 1$ *then* (*x, y* -1)

3. HASIL DAN PEMBAHASAN

Teka-teki silang memaksimumkan pengisian ruang-ruang kosong, untuk memberikan keluaran kepada pengguna berupa TTS dan jawaban dari sebuah pertanyaan dengan menghasilkan *substring* kosakata dengan jumlah pertanyaan yang ditentukan sendiri oleh pengguna. Pertanyaan-pertanyaan berupa *file* yang akan ditampilkan pada pengguna tersebut diperoleh dari *database access* secara random oleh program. TTS yang akan dibentuk merupakan sebuah kumpulan jawaban dari pertanyaan yang akan digunakan, jawaban-jawaban pertanyaan yang disusun sedemikian rupa sehingga pola kata saling berpotongan satu sama lain. Pada kolom matriks ini nantinya jawaban dari pertanyaan akan diletakkan untuk dibentuk menjadi sebuah TTS. Setiap huruf dari jawaban tersebut akan menempati satu *cell* pada kolom matriks. Selain menerima masukan dari pengguna, aplikasi TTS juga memberikan keluaran kepada pengguna berupa TTS dan jawaban yang benar dari TTS tersebut.

Penelitian dalam optimasi pengisian ruang-ruang kosong dalam papan matriks teka-teki silang tersebut tentunya diperlukan algoritma dengan menggunakan perbandingan algoritma *Longest Common Subsequence (LCS)* dan algoritma *Simple Hill Climbing.* Dengan melakukan perbandingan performansi dari metode *Longest Common Subsequence (LCS)* dan *Simple Hill Climbing* maka akan dapat diketahui cara kerja masing-masing kedua metode tersebut dan performansi dalam kecepatan dan ketepatan dari kedua metode tersebut.

3.1 Penerapan Algoritma Longest Common Subsequence

Dalam permainan teka-teki silang dalam menentukan teks yang akan dicari polanya berdasarkan pencarian berupa jawaban. Pada metode LCS menggunakan konsep *dynamic programming* untuk mencari *longest common subsequence* dari dua buah *sequence*. Sifat-sifat *dynamic programming* yang diterapkan dalam algoritma LCS adalah prosedur LCS membagi *sequence* A (panjang = *m*) dan *sequence* B (panjang = *n*) menjadi karakter-karakter dan membentuk sebuah tabel sebesar *m* baris dan *n* kolom. Algoritma memulai pencarian nilai untuk kotak dengan membandingkan karakter ke-1 dari *sequence* A dan karakter ke-1 dari *sequence* B. Perunutan balik LCS akan dilanjutkan perhitungan *bad-character shift* yang berfungsi untuk mencari akhiran/*suffix* serta *pattern* (Iterasi dari *sequence*). Bila diperhatikan algoritma prosedur LCS, maka dapat diketahui bahwa nilai yang ditempatkan pada kotak-kotak di dalam tabel adalah saling berhubungan. Kotak yang diisi membutuhkan nilai kotak di atasnya, di kiri atau kotak sebelah kiri atas.

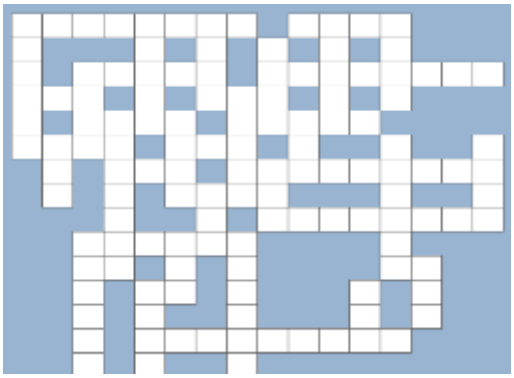
Berdasarkaan sample data pada saat pencarian pola kata pada papan matriks teka-teki silang dengan posisi mendatar dan menurun. Papan permainan dibuat acak yang disusun random 15x16, dimana didalam pertanyaan papan matriks tersebut berisi jawaban dengan menggunakan 6 sampel baik mendatar dan menurun pada teka-teki silang. Pola kata jawaban yang akan di cari dapat dilihat pada tabel sebagai berikut :

Tabel 1. Pola kata jawaban yang akan di cari

Pertanyaan	Papan permainan	Jawaban
Berdarah campuran?	Menurun	Indo
Pemerintahan rakyat?	Mendatar	Demokrasi
Sungai di afrika?	Mendatar	Nil
Dasar pikiran, alasan?	Menurun	Premis
Surat bulat?	Mendatar	Aklamasi
Tulang dari nyiur?	Menurun	Lidi

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Sebagai contoh diberikan sebuah papan matriks teka-teki silang dengan pencarian pola kata berdasarkan dari pertanyaan berisi jawaban pada papan matriks teka-teki silang yang masih kosong. Dapat dilihat pada gambar 2. sebagai berikut :



Gambar 2. Papan matrik TTS 16 x 16

Berikut contoh kasus diatas, pada tahap pertama yaitu akan menentukan teks dengan cara membuat jumlah kotak baik mendatar ataupun menurun yang menempati teks posisi benar pada papan matrik TTS seperti pada tabel 2 yang secara membaca teks secara mendatar dan menurun. Berikut teks tersebut akan dicari jumlah kotak posisi benar yang akan dipilih.

Tabel 2. Pola baris dan kolom

I										
N	I	L					P			L
D		D	E	M	O	K	R	A	S	I
O							E			D
			A	K	L	A	M	A	S	I
							I			
							S			

Mendatar Menurun

10. Nil 24. Indo
12. Demokrasi 36. Premis
14. Aklamasi 38. Lidi

Dari ekstraksi kata tersebut kemudian posisi kotak kosong yang terdiri baris dan kolom matriks yang akan dicari menggunakan perbandingan metode *LCS* dan *Simple Hill Climbing*.

Tahapan LCS dengan mencari nilai dari setiap perbandingan karakter *sequence*. LCS yang memulai pencarian nilai dari kotak kiri paling atas serta selanjutnya menuju kotak kanan paling bawah, metode tidak akan kembali ke kotak yang telah dicari nilainya. Hasil dari pencarian dari kotak yang sekarang akan digunakan pada kotak yang berikutnya. Untuk mengindikasikan pada setiap *subproblema* saling berhubungan satu sama lain. Apabila tabel LCS sudah terbentuk, LCS memulai melakukan pencarian solusi optimal dari kotak kanan paling bawah menuju ke kotak paling atas.

Misalkan :DEMOKRASI blok (jumlah kotak) = 9
PREMISblok (jumlah kotak) = 6
m = baris dan*n* = kolom
Karakter ke-1 dari *sequence* A dan karakter ke-1 dari *sequence* B
Pergerakan kotak kosong kebawah : *if* x ≠(tidak sama dengan)1,1 *then* (x+1,y)
Pergerakan kotak kosong kekiri : *if* x ≠(tidak sama dengan)1,2 *then* (x+1,y)
Pergerakan kotak kosong keatas : *if* x ≠(tidak sama dengan)1,3 *then* (x-1,y)
Aturan / Oprator : (x,y)
x = baris kotak kosong
y = kolom kotak kosong

Tabel 3. Keadaan Goal

I	O	O	O	O	O	O	O	O	O	O
N	**I**	**L**	O	O	O	O	**P**	O	O	**L**
D	O	**D**	**E**	**M**	**O**	**K**	**R**	**A**	**S**	**I**
O	O	O	O	O	O	O	**E**	O	O	**D**
O	O	O	**A**	**K**	**L**	**A**	**M**	**A**	**S**	**I**
O	O	O	O	O	O	O	**I**	O	O	O
O	O	O	O	O	O	O	**S**	O	O	O

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Berikut Contoh dalam penerapan pada prosedur *LCS-Length.* Dimisalkan diambil contoh *sequence* yang akan dibandingkan adalah *Sequence*-1= "DEMOKRASI' dan *Sequence*-2='PREMIS'. Berikut tabel LCS :

Tabel 4. Panjang karakter input string

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,1	1,11
2	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,1	2,11
3	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9	3,1	3,11
4	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9	4,1	4,11
5	5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9	5,1	5,11
6	6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9	6,1	6,11
7	7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9	7,1	7,11

Aturan :
Nilai = 0
Mendatar

1. If ≠(1,1) = "I" And (1,2) = "O" And (1,3) = "O" And (1,4) = "O" And (1,5) = "O" And (1,6) = "O" And (1,7) = "O" And (1,8) = "O" And (1,9) = "O" And (1,10) = "O" And (1,11) = "O" Then Nilai = +1
2. If ≠(2,1) = "N" And (2,2) = "I" And (2,3) = "L" And (2,4) = "O" And (2,5) = "O" And (2,6) = "O" And (2,7) = "O" And (2,8) = "P" And (2,9) = "O" And (2,10) = "O" And (2,11) = "O" Then Nilai = +1
3. If ≠(3,1) = "D" And (3,2) = "G" And (3,3) = "**D**" And (3,4) = "**E**" And (3,5) = "**M**" And (3,6) = "**O**" And (3,7) = "**K**" And (3,8) = "**R**" And (3,9) = "**A**" And (3,10) = "**S**" And (3,11) = "**I**" Then Nilai = +1

Menurun

1. If ≠(1,1) = "I" And (1,2) = "O" And (1,3) = "O" And (1,4) = "O" And (1,5) = "O" And (1,6) = "O" And (1,7) = "O" And (1,8) = "O" And (1,9) = "O" And (1,10) = "O" And (1,11) = "O" Then Nilai = +1
2. If ≠(2,1) = "N" And (2,2) = "I" And (2,3) = "L" And (2,4) = "O" And (2,5) = "O" And (2,6) = "O" And (2,7) = "O" And (2,8) = "**P**" And (2,9) = "O" And (2,10) = "O" And (2,11) = "O" Then Nilai = +1
3. If ≠(3,1) = "D" And (3,2) = "G" And (3,3) = "D" And (3,4) = "E" And (3,5) = "M" And (3,6) = "O" And (3,7) = "K" And (3,8) = "**R**" And (3,9) = "A" And (3,10) = "S" And (3,11) = "I" Then Nilai = +1
4. If ≠(4,1) = "O" And (4,2) = "O" And (4,3) = "O" And (4,4) = "O" And (4,5) = "M" And (4,6) = "O" And (4,7) = "O" And (4,8) = "**E**" And (4,9) = "O" And (4,10) = "O" And (4,11) = "D" Then Nilai = +1
5. If ≠(5,1) = "O" And (5,2) = "O" And (5,3) = "O" And (5,4) = "A" And (5,5) = "K" And (5,6) = "L" And (5,7) = "A" And (5,8) = "**M**" And (5,9) = "A" And (5,10) = "S" And (5,11) = "I" Then Nilai = +1
6. If ≠(6,1) = "O" And (6,2) = "O" And (6,3) = "O" And (6,4) = "O" And (6,5) = "O" And (6,6) = "O" And (6,7) = "O" And (6,8) = "**I**" And (6,9) = "O" And (6,10) = "O" And (6,11) = "O" Then Nilai = +1
7. If ≠(7,1) = "O" And (7,2) = "O" And (7,3) = "O" And (7,4) = "O" And (7,5) = "O" And (7,6) = "O" And (7,7) = "O" And (7,8) = "**S**" And (7,9) = "O" And (7,10) = "O" And (7,11) = "O" Then Nilai = +1

3.2 Penerapan Metode Simple Hill Climbing

Dalam Algortima *Simple Hill Climbing* merepresentasi jumlah kotak yang menempati posisi ketika jawaban benar dari pertanyaan tersebut. *Crossover* atau pindah bersilang dalam proses pembentukan *kromosom* (*offspring*). *Crossover* yang bertujuan menambah *string* dalam satu populasi dengan penyilangan *string* yang akan diperoleh dari reproduksi sebelumnya.

Tahapan pencarian memilih *new state* yang memiliki jalur ("curam") dari pada jalur-jalur sebelumnya tanpa memperhitungkan jalur-jalur lain yang lebih "curam". Sedangkan *Steepest Ascent Hill Climbing* akan mengevaluasi semua *state* yang berada dibawah *current state* dan memilih *state* dengan jalur paling "curam".
Berikut ruang keadaan di misalkan :
x = baris = [1, 3, 8, 9]
3 Kotak mendatar NIL
8 Kotak mendatar AKLAMASI
9 Kotak mendatar DEMOKRASI
y = kolom = [1, 4, 6]
4 Kotak menurun INDO
4 Kotak menurun LIDI
7 Kotak menurun PREMIS
Aturan / Opreator : (x,y)
x = adalah baris kotak yang kosong
y = adalah kolom kotak yang kosong
Pergerakan kotak yang kosong ke atas : *if* x > 1 *then* (x-1, y)
Pergerakan kotak yang kosong kebawah : *if* x < 3 *then* (x+1, y)
Pergerakan kotak yang kosong kekanan : *if* y < 3 *then* (x, y +1)
Pergerakan kotak yang kosong kekiri : *if* y >1 *then* (x, y -1)

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Tabel 5. Keadaan Goal

I	O	O	O	O	O	O	O	O	O	O
N	**I**	**L**	O	O	O	O	**P**	O	O	**L**
D	O	**D**	**E**	**M**	**O**	**K**	**R**	**A**	**S**	**I**
O	O	O	O	O	O	O	**E**	O	O	**D**
O	O	O	**A**	**K**	**L**	**A**	**M**	**A**	**S**	**I**
O	O	O	O	O	O	O	**I**	O	O	O
O	O	O	O	O	O	O	**S**	O	O	O

Tabel 6. Karakter panjang yang di input string

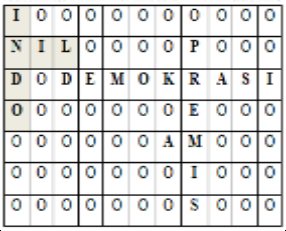
	1	2	3	4	5	6	7	8	9	10	11
1	**1,1**	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,1	1,11
	I	O	O	O	O	O	O	O	O	O	O
2	**2,1**	**2,2**	**2,3**	2,4	2,5	2,6	2,7	**2,8**	2,9	2,1	**2,11**
	N	**I**	**L**	O	O	O	O	**P**	O	O	**L**
3	**3,1**	3,2	**3,3**	**3,4**	**3,5**	**3,6**	**3,7**	**3,8**	**3,9**	**3,1**	**3,11**
	D	O	**D**	**E**	**M**	**O**	**K**	**R**	**A**	**S**	**I**
4	**4,1**	4,2	4,3	4,4	4,5	4,6	4,7	**4,8**	4,9	4,1	**4,11**
	O	O	O	O	O	O	O	**E**	O	O	**D**
5	5,1	5,2	5,3	**5,4**	**5,5**	**5,6**	**5,7**	**5,8**	**5,9**	**5,1**	**5,11**
	O	O	O	**A**	**K**	**L**	**A**	**M**	**A**	**S**	**I**
6	6,1	6,2	6,3	6,4	6,5	6,6	6,7	**6,8**	6,9	6,1	6,11
	O	O	O	O	O	O	O	**I**	O	O	O
7	7,1	7,2	7,3	7,4	7,5	7,6	7,7	**7,8**	7,9	7,1	7,11
	O	O	O	O	O	O	O	**S**	O	O	O

Aturan : Nilai = 0
Mendatar
1. If (2,1) = “N” And (2,2) = “I” And (5,6) = “L” Then Nilai = +1
2. If (3,1) = “D” And (3,4) = “E” And (3,5) = “M” And (3,6) = “O” And (3,7) = “K” And (3,8) = “R” And (5,9) = “A” And (5,10) = “S” And (5,11) = “I” Then Nilai = +1
3. If (5,4) = “A” And (5,5) = “K” And (5,6) = “L” And (5,7) = “A” And (5,8) = “M” And (5,9) = “A” And (5,10) = “S” And (5,11) = “I” Then Nilai = +1

Menurun
1. If (1,1) = “I” And (2,1) = “N” And (3,3) = “D” And (3,6) = “O” Then Nilai = +1
2. If (2,11) = “L” And (3,11) = “I” And (4,11) = “D” And (5,11) = “I” Then Nilai = +1
3. If (2,8) = “P” And (3,8) = “R” And (4,8) = “E” And (5,8) = “M” And (6,8) = “I” And (7,8) = “S” Then Nilai = +1

Hasil perbandingan dari *Longest Common Subsequence (LCS)* menunjukan pencarian *string* pola kata jawaban denganprosedur *LCS-Length*yang diambil *sequence*-1dan*sequence*-2pola kata teka-teki silang baik jawaban mendatar ataupun menurun seperti *Sequence*-1=“DEMOKRASI’, *Sequence*-2=’PREMIS’jikapola susunan kata keadaan sekarang pada jawaban menjadi keadaan baru.Sedangkan algoritma *Simple Hill Climbing*menunjukkan pencarian *string* pola kata jawaban menggunakan pola kata *string* untuk mendapatkan keadaan yang baru.Jika bukan tujuan pola kata jawaban, namun apabila nilainya lebih baik dari pada keadaan yang sekarang, maka dijadikan keadaan yang baru tersebut menjadi keadaan yang sekarang. Seperti persentase pada gambar 7.

Tabel 7. Hasil Perbandingan

Metode	Perbandingan	Persentasi keadaan
Longest Common Subsequence (LCS)		Jika pola susunan kata keadaan sekarang pada jawaban TTS menjadi keadaan baru.
Simple Hill Climbing		Jika bukan tujuan pola kata jawaban TTS, jika nilainya lebih baik dari pada keadaan yang sekarang, maka dijadikan keadaan yang baru tersebut menjadi keadaan yang sekarang

Bulletin of Artificial Intelligence
Volume 1, No 2, October 2022 Page: 62−69
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

4. KESIMPULAN

Berdasarkan kesimpulan yang di dapatkan dalam penelitian dan penyusunan skripsi tersebut serta disesuaikan dengan tujuan, yakni sebagai berikut Hasil perbandingan dari *Longest Common Subsequence (LCS)* menunjukan pencarian *string* pola kata jawaban denganprosedur *LCS-Length* yang diambil *sequence*-1dan *sequence*-2.Sedangkan algoritma *Simple Hill Climbing* menunjukkan pencarian *string* pola kata jawaban menggunakan pola kata *string* untuk mendapatkan keadaan yang baru. Menghasilkan penerapan dari permainana Teka-Teki dalam proses optimasi teka-teki silang menggunakan papan matriks yang dibentuk menjadi sebuah TTS, pada setiap huruf dari jawaban yang benar menempati satu *cell* pada papan matriks TTS.

REFERENCES

[1] J. M. Penusa and I. Pendahuluan, "IMPLEMENTASI ALGORITMA SIMPLE HILL CLIMBING PADA APLIKASI GAME TTS (TEKA – TEKI SILANG) Paska Marto Hasugian Program Studi Teknik Informatika STMIK Pelita Nusantara , Jl Iskandar Muda No 1 Medan , Sumatera Utara , Indonesia Abstrak," vol. 19, no. 1, pp. 138–141, 2016.

[2] P. N. Saleh, "Implementasi Algoritma Longest Common Subsequence Dengan Algoritma Genetika Pada Permainan Word Search Puzzle," vol. 8, pp. 361–366, 2020.

[3] H. Simalango, "RANCANG BANGUN GAME TEKA TEKI SILANG (TTS) DENGAN METODE LONGEST COMMON SUBSEQUENCE," 2012.

[4] J. M. Informasi, V. No, and D. M. Hutagalung, "Konsep algoritma genetika sebagai dasar pembuatan teka teki silang berbasis komputer," vol. 3, no. 1, pp. 1–7, 2018.

[5] A. Minandar, A. Tanoto, and D. Tanadi, "Aplikasi Algoritma Pencarian String Boyer-Moore pada Pencocokan DNA," *Inst. Teknol. Bandung*, pp. 1–3,