Penerapan Metode Heuristik pada Algoritma Brute Force dalam Permainan Wordscapes

Rafi Raihansyah Munandar - 13519154
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519154 @gmail.com

***Abstract*—Permainan Wordscapes merupakan suatu permainan yang memiliki kemiripan dengan teka teki silang. Pemain diharuskan untuk mengisi teka teki hanya dengan beberapa huruf yang disediakan. Algoritma *Brute Force* dapat digunakan untuk mendapatkan semua jawaban dengan tepat. Penulis akan menggunakan metode heuristik untuk optimasi algoritma *brute force*.**

Keywords—Wordscapes; Brute Force; heuristik

I. PENDAHULUAN

Berbagai permasalahan dapat diselesaikan menggunakan komputasi. Penerapan ilmu komputasi untuk menyelesaikan permasalahan ini biasa disebut juga dengan algoritma. Terdapat berbagai algoritma yang tersedia dan dapat digunakan untuk menyelesaikan berbagai macam persoalan, dari yang mudah hingga kompleks. Setiap algoritma dapat memiliki modifikasi untuk menyelesaikan permasalahan-permasalahan yang lebih spesifik.

Algoritma *Brute Force* merupakan salah satu algoritma yang cukup mudah untuk digunakan dan diimplementasikan. Algoritma ini dapat menyelesaikan hampir seluruh permasalahan yang membutuhkan komputasi. Namun, algoritma *brute force* jarang sekali menghasilkan suatu algoritma yang mangkus. Banyak algoritma lain yang merupakan evolusi atau perkembangan dari algoritma *brute force*.

Permainan kata telah banyak berevolusi seiring berkembangnya zaman. Banyak permainan kreatif yang didasari dari permainan-permainan sebelumnya. Salah satu permainan kata yang populer adalah teka teki silang (TTS). Pemain diminta untuk menjawab dan menyambung kata berdasarkan *clue* yang diberikan pada masing-masing nomor. Jawaban dapat ditulis secara horizontal maupun vertikal.

Wordscapes merupakan salah satu permainan yang mirip dengan teka teki silang, tetapi dengan beberapa aturan tambahan yang lebih simpel. Dalam permainan ini, terdapat beberapa kotak kata yang bersambung seperti TTS pada umumnya, namun pemain diberikan beberapa huruf yang harus ditebak kombinasinya untuk menjawab kotak-kotak kata yang ada.

Gambar 1. Wordscapes

Sumber: (https://wordscapesmate.com/wp-content/uploads/2020/11/wordscapes-cheats.jpg)

Algoritma *brute force* dapat digunakan untuk menyelesaikan permasalahan pada wordscapes. Dengan menelusuri kombinasi-kombinasi huruf, maka setidaknya akan menemukan kata yang valid dan merupakan jawabwan.

II. LANDASAN TEORI

A. Algoritma Brute Force

Algoritma *brute force* merupakan salah satu algoritma untuk memecahkan suatu masalah dengan lempang (*straightforward*). Biasanya, algoritma *brute force* didasari oleh beberapa faktor, antara lain adalah pernyataan pada suatu persoalan (*problem statement*) dan juga definisi atau konsep yang dilibatkan dalam permasalahan tersebut. Algoritma *brute force* cenderung sederhana dan memiliki cara pengerjaan yang jelas serta langsung.

Algoritma *brute force* memiliki karakteristik antara lain sebagai berikut:

1. Algoritma *brute force* umumnya tidak “cerdas” dan tidak mangkus, karena langkah yang dibutuhkan untuk menyelesaikan suatu permasalahan cenderung berjumlah besar. Terkadang, algoritma ini juga disebut sebagai algoritma naïf (*naïve algorithm*).
2. Algoritma *brute force* seringkali menjadi pilihan yang tidak banyak disukai atau diminati karena ketidakmangkusannya. Namun, algoritma ini bisa menjadi dasar untuk mencari algoritma-algoritma lain

yang lebih mangkus dengan mencari pola-pola yang mendasar, keteraturan, atau trik-trik khusus.

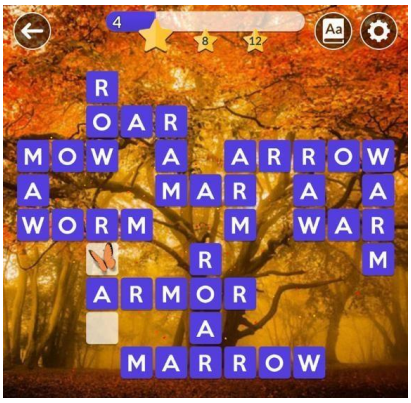
3. Untuk beberapa permasalahan dengan skala kecil, kesederhanaan dari suatu algoritma *brute force* biasanya lebih diperhitungkan dibandingkan dengan ketidakmangkusannya. Algoritma *brute force* biasa digunakan sebagai basis ketika membandingkan beberapa algoritma lain yang mangkus.
4. Teknik *brute force* dapat diterapkan pada sebagian besar permasalahan yang ada. Cukup sulit untuk mencari permasalahan yang tidak bisa diselesaikan dengan *brute force*.
5. Algoritma *brute force* lebih mudah untuk diimplementasikan daripada algoritma yang lebih mangkus/canggih. Terkadang, algoritma *brute force* dapat lebih mangkus karena kesederhanaannya (ditinjau dari segi implementasi).

Teknik heuristik merupakan suatu teknik yang dirancang untuk memecahkan persoalan tanpa perlu membuktikan bahwa teknik tersebut benar secara matematis. Teknik ini menggunakan terkaan, intuisi, dan *common sense*. Dengan menggunakan teknik heuristik, beberapa persoalan yang diselesaikan dengan *brute force* dapat menjadi lebih mangkus dan efisien.

B. *Wordscapes*

Wordscapes adalah permainan *puzzle* yang dibuat oleh studio dari Amerika bernama PeopleFun. Permainan ini dapat dimainkan pada platform Android dan iOS. Wordscapes pernah menduduki 10 besar permainan paling populer di *Google Play Store* dan *Apple App Store*. Dirilis pada 14 Juni 2017, permainan ini telah memiliki 14 juta pemain dari seluruh dunia terhitung pada tahun 2020.

Wordscapes merupakan suatu permainan yang mirip dengan teka teki silang. Terdapat beberapa kotak yang dapat diisi dengan huruf-huruf untuk membentuk suatu kata. Beberapa huruf bisa berpotongan dengan huruf lain pada kata lain seperti pada umumnya.



Gambar 2. Kotak huruf pada permainan wordscapes

Sumber: (https://thenewscrunch.com/wp-content/uploads/2019/08/Wordscapes-Daily-August-24-2019-576x1024.jpeg)

Pada gambar 2, dapat dilihat bahwa kata-kata yang tersedia dapat berbentuk horizontal dan vertikal. Setiap kata dapat memiliki huruf yang terkandun dalam kata lainnya, sehingga bisa menyilang secara horizontal dan vertikal.

Pada teka teki silang, setiap kotak horizontal dan vertikal ditandai dengan nomor, di mana untuk setiap nomor terdapat *clue* atau petunjuk dari kata yang merupakan jawaban tersebut. Namun, pada wordscapes, penomoran dan petunjuk tidak ada. Wordscapes memberikan kumpulan huruf yang kemudian pemain akan menebak kombinasi huruf apa saja yang bisa disambungkan untuk menjadi suatu kata yang valid.

Gambar 3. Kumpulan huruf yang bisa disambung dan dikombinasikan untuk menjawab kolom kotak gambar 2

Sumber: (https://thenewscrunch.com/wp-content/uploads/2019/08/Wordscapes-Daily-August-24-2019-576x1024.jpeg)

Pada gambar 3, dapat dilihat terdapat 6 huruf yang diberikan oleh permainan. Huruf tersebut adalah M, R, A, W, O, dan R. Jika dilihat pada gambar 2, semua jawaban yang telah terbuka terdiri dari kombinasi elemen-elemen subset dari keenam huruf yang diberikan.

Terdapat 1 kolom kotak vertikal yang belum terisi, terdiri dari 4 huruf dengan huruf pertama yaitu R dan huruf ketiga yaitu A. Maka, pemain dapat mencoba kombinasi-kombinasi huruf untuk mendapatkan jawaban tersebut. “ROAM” merupakan jawaban dari kotak vertikal yang belum terjawab.

Permainan wordscapes ini menggunakan Bahasa Inggris. Namun, telah terdapat berbagai permainan yang mirip dengan Wordscapes menggunakan bahasa lain. Permainan ini ditujukan untuk mengasah otak dan kreativitas pemain, serta memperluas kosa kata. Untuk makalah ini, versi permainan Wordscapes yang digunakan adalah versi *original* yang menggunakan Bahasa Inggris sebagai penentuan setiap katanya.

III. Pembahasan dan Analisis

Algoritma *brute force* akan digunakan untuk memecahkan masalah dan mencari jawaban pada permainan Wordscapes. Konsep *brute force* yang akan digunakan berupa mengecek setiap kolom atau baris kotak jawaban kemudian meninjau panjang kotaknya. Dari panjang kotak tersebut, maka akan dicek seluruh kombinasi dari huruf-huruf yang disediakan permainan dengan panjang tersebut. Setiap kombinasi huruf akan dicek apakah kata yang dibentuk merupakan kata yang valid atau tidak. Jika valid, maka akan ditampilkan ke layar dan

setiap kotak jawaban akan terisi dengan huruf yang berkorespondensi.

Permainan Wordscapes adalah permainan yang dimainkan oleh satu orang. Oleh karena itu, dengan menggunakan algoritma *brute force*, tentu akan berhasil ditemukan setiap jawabannya. Namun, perbedaan bisa bervariasi pada aspek kemangkusan dan waktu atau iterasi yang dibutuhkan untuk memecahkan masalah.

A. Penerapan Brute Force pada Wordscapes

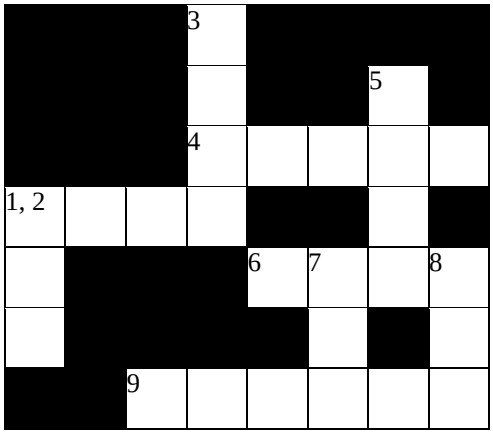
Untuk menyelesaikan dan menemukan jawaban dalam permainan Wordscapes menggunakan algoritma *brute force*, konsep utamanya adalah melakukan permutasi untuk kemungkinan huruf-huruf yang diberikan oleh permainan. Langkah-langkah untuk mendapatkan jawaban pada Wordscapes dengan *brute force* antara lain sebagai berikut.

1. Tinjau panjang kolom atau baris yang akan dicek
2. Lakukan permutasi dengan panjang yang sesuai dengan kolom atau baris yang sedang dicek
3. Untuk setiap kata dalam hasil permutasi, lakukan pengecekan apakah kata tersebut valid (sesuai dengan jawaban pada sistem game) atau tidak.
4. Apabila jawaban telah ditemukan, maka hentikan iterasi atau pengecekan pada kolom/baris tersebut.
5. Simpan data jawaban pada kotak kolom atau baris yang tersedia.
6. Lakukan kembali dari langkah 1 untuk kolom atau baris lain yang belum dicek hingga seluruh jawaban telah didapatkan.

Langkah-langkah tersebut ditujukan untuk melakukan *brute force* pada setiap kemungkinan jawaban. Maka, dapat dipastikan jawaban akan selalu ditemukan. Pseudocode untuk langkah-langkah di atas antara lain sebagai berikut.

```
Type letterbox : <answer: tuple of char,
                  len: int>


Function permutate(Letters: array of char,
length: int) → array of tuple of char
  { Menerima Letters berupa array of
characters yang merupakan daftar kata
untuk jawaban yang diberikan oleh
permainan dan length yaitu panjang hasil
permutasi yang dibutuhkan. Keluaran berupa
array yang berisi kumpulan tuple dari
pasangan karakter hasil permutasi. }

Function getCurrentBox(i: int) →
letterbox
  { menerima masukan integer yang
menyatakan urutan box jawaban yang sedang
dicek. Fungsi akan mengembalikan box
jawaban yang berkorespondensi dengan
urutannya.


Kamus

Letters: array of char { huruf-huruf yang
akan membentuk jawaban }

Word: array of char { Merupakan kata
jawaban dalam permainan }

Words: array [0..nWords] of word {
Kumpulan kata-kata jawaban }

nWords: int { banyak kata yang merupakan
jawaban }

permList : array of tuple of char


Algoritma

for i := 0 to nWords do

  currentBox = getCurrentBox(i) { Mengisi
box untuk dicek sesuai urutan }

  permList = permutate(Letters,
currentBox.len) { melakukan permutasi
sesuai dengan panjang box yang sedang
dicek }


  { melakukan pengecekan untuk setiap
hasil permutasi }

  foreach j := 0 to permList.size do

    { mencari apakah ada kata yang cocok
dengan jawaban }

    if (permList[j] = Words[i]) then

      { mengisi jawaban dan
menghentikan iterasi }

      currentBox.answer ← permList[j]

      break
```

Dengan pseudocode di atas, maka program pertama akan melakukan permutasi sesuai dengan panjang box jawaban yang sedang dicek. Hasil permutasi akan disimpan dalam bentuk *array* yang elemennya adalah *tuple of characters*. Setiap elemen dari *array* tersebut kemudian akan dibandingkan dengan jawaban yang sesuai dengan box jawaban. Apabila ditemukan, maka box jawaban akan diisi dengan karakter-karakter yang berkorespondensi, kemudian iterasi dihentikan.



Tabel 1.Kasus 1 dengan jawaban belum terisi

Pada tabel 1, dapat dilihat terdapat 9 box jawaban untuk diisi. Huruf-huruf yang disediakan untuk menjadi jawaban adalah [S, T, U, O, M, C].

			M				
			O			M	
			S	C	O	U	T
C	O	S	T			S	
U				C	O	T	S
T					U		U
		C	U	S	T	O	M

Tabel 2. Kasus 1 dengan jawaban akhir

Tabel 2 merupakan jawaban yang telah terisi dan benar. Terdapat 9 jawaban, antara lain adalah:

1. CUT (3 huruf)
2. COST (4 huruf)
3. MOST (4 huruf)
4. SCOUT (5 huruf)
5. MUST (4 huruf)
6. COTS (4 huruf)
7. OUT (3 huruf)
8. SUM (3 huruf)
9. CUSTOM (6 huruf)

Untuk nomor 1, box jawaban berupa box jawaban vertikal dengan panjang box yaitu 3. Maka, akan dilakukan permutasi dari 6 huruf yang tersedia sebagai jawaban untuk 3 sampel.

Permutasi akan dilakukan dari 6 huruf jawaban yaitu [S, T, U, O, M, C]. Untuk 3 sampel, maka akan dicek seluruh kemungkinan kombinasi permutasi yang ada, seperti [S, T, U], [S, T, O], [S, T, M], dan seterusnya hingga didapatkan seluruh permutasi yang ada.

Dari kumpulan permutasi yang telah dijalankan, maka akan ada iterasi tambahan untuk mengecek setiap elemen hasil permutasi tersebut. Untuk kasus nomor 1, pengecekan akan berlangsung hingga ditemukan pasangan hasil permutasi yaitu (C, U, T).

Permutasi akan dilakukan untuk setiap huruf pada seluruh kata dan jawabwan yang tersedia, sehingga banyaknya pengecekan yang dilakukan untuk masing-masing permutasi adalah

$$ {}_6P_3 + {}_6P_4 + {}_6P_4 + {}_6P_5 + {}_6P_4 + {}_6P_4 + {}_6P_3 + {}_6P_3 + {}_6P_6 $$

$$ \Leftrightarrow 120 + 360 + 360 + 720 + 360 + 360 + 120 + 120 + 720 $$

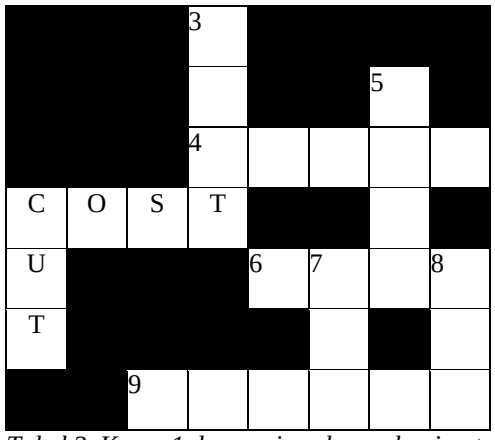
$$ = 3240 $$

Terdapat 3240 kali operasi untuk mendapatkan hasil permutasi. Namun, setelah dilakukan masing-masing permutasi untuk satu box jawaban, akan dilakukan iterasi sebanyak jumlah hasil permutasi. Maka, jumlah iterasi untuk keadaan terburuk (*worst case*) adalah 3240. Sehingga, terdapat 3240 iterasi untuk menghasilkan permutasi dan 3240 iterasi untuk membandingakan hasil permutasi dengan jawaban.

B. Penerapan Teknik Heuristik pada Brute Force

Teknik Heuristik digunakan untuk memangkuskan algoritma berdasarkan terkaan atau ide yang masuk akal tanpa perlu pembuktian matematisnya. Untuk itu, dalam menerapkan *brute force* pada permainan Wordscapes, dapat digunakan juga teknik heuristik yang bertujuan untuk mengurangi iterasi atau operasi yang tidak diperlukan.

Ide dari teknik heuristik yang digunakan untuk *brute force* pada Wordscapes ini adalah beberapa kata bisa memiliki bagian huruf yang sudah terisi, sehingga ketika melakukan pengecekan pada box jawabwan tersebut, tidak perlu dilakukan permutasi secara keseluruhan.

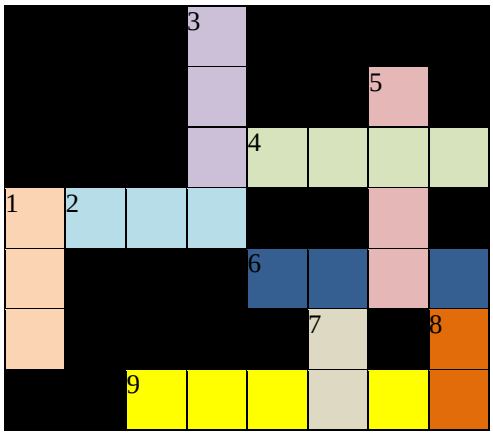


Tabel 3. Kasus 1 dengan jawaban sebagian terisi

Pada tabel 3, dapat dilihat bahwa box jawaban nomor 1 dan 2 telah diisi. Ketika ingin melakukan pengecekan pada box jawaban nomor 3, umumnya akan dilakukan permutasi dengan panjang setiap elemen hasil permutasi (sampel) sebesar 4 karena panjang box jawaban tersebut adalah 4. Namun, ketika

beberapa jawaban telah terisi (pada kasus di tabel 3 adalah box jawaban nomor 2 telah diisi), ada kemungkinan box jawaban yang sedang dicek telah memiliki suatu huruf di tempat tertentu.

Tabel 3 memperlihatkan bahwa ketika sedang mengecek box jawaban nomor 3, telah terdapat huruf 'T' yang telah terbuka dari jawaban box nomor 2. Untuk itu, tidak perlu lagi dilakukan pengecekan atau permutasi yang memiliki huruf akhir selain 'T' di box jawaban nomor 3. Hal ini membuat jumlah operasi permutasi yang sebelumnya diambil 4 sampel menjadi 3 sampel.



Tabel 4. Urutan pengecekan dan permutasi untuk kasus 1

Pada tabel 4, dapat dilihatketerangan warna yang berbeda. Warna tersebut merepresentasikan box huruf yang akan dicek saat melakukan permutasi.

Untuk box jawaban nomor 1, permutasi masih dilakukan dengan 3 sampel. Hal ini disebabkan belum ada jawaban huruf yang terbuka atau didapatkan dari nomor lain.

Untuk box jawaban nomor 2, permutasi dilakukan dengan mengurangi 1 huruf. Huruf pertama pada box nomor 2 akan selalu sama dengan huruf pertama pada box nomor 1 (untuk kasus 1). Hal ini membuat pengecekan permutasi tidak perlu lagi dilakukan dengan 4 sampel, sehingga bisa dikurangi menjadi 3 sampel.

Begitupun juga untuk nomor-nomor selanjutnya. Seperti pada nomor 5, karena nomor 4 akan dijawab terlebih dahulu, maka huruf kedua dari box jawaban nomor 5 sudah terdefinisi. Untuk itu, hanya perlu melakukan permutasi pada huruf-huruf lainnya yang belum ditemukan. Maka, panjang huruf untuk masing-masing jawaban adalah sebagai berikut.

1. CUT (3 huruf)
2. OST (3 huruf)
3. MOS (3 huruf)
4. COUT (4 huruf)
5. MST (3 huruf)
6. COS (3 huruf)
7. UT (2 huruf)
8. UM (2 huruf)
9. CUSO (4 huruf)

Jumlah operasi permutasi untuk masing-masing kata di atas adalah

$$ {}_6P_3 + {}_6P_3 + {}_6P_3 + {}_6P_4 + {}_6P_3 + {}_6P_3 + {}_6P_2 + {}_6P_2 + {}_6P_4 $$

$$ \Leftrightarrow 120 + 120 + 120 + 360 + 120 + 120 + 30 + 30 + 360 $$

$$ = 1380 $$

Untuk setiap operasi pada permutasi yang dihasilkan, akan dilakukan iterasi sebanyak hasil permutasi yang didapatkan. Jumlah iterasi untuk membandingkan hasilnya adalah 1380, sehingga total iterasi dari permutasi dan perbandingan jawaban adalah

C. Analisis Waktu Eksekusi Program

Salah satu cara untuk membandingkan kemangkusan dari suatu algoritma adalah dengan menghitung waktu yang dibutuhkan untuk mengeksekusi algoritma. Algoritma yang lebih mangkus cenderung memiliki *runtime* yang lebih kecil. Perbandingan akan dilakukan pada algoritma yang normal dan yang menggunakan teknik heuristik.

```
-----NORMAL BRUTE FORCE-----
Result found - CUT
Result found - COST
Result found - MOST
Result found - SCOUT
Result found - MUST
Result found - COTS
Result found - OUT
Result found - SUM
Result found - CUSTOM
--- 1.9962787628173828 milidetik ---
```

Gambar 4. Hasil eksekusi algoritma brute force

Sumber: Penulis

```
-----HEURISTIC BRUTE FORCE-----
Result found - CUT
Result found - OST
Result found - MOS
Result found - COUT
Result found - MST
Result found - COS
Result found - UT
Result found - UM
Result found - CUSO
--- 0.9946823120117188 milidetik ---
```

Gambar 5. Hasil eksekusi algoritma brute force dengan teknik heuristik

Sumber: Penulis

Gambar 4 menunjukkan eskekusi algoritma *brute force* untuk menyelesaikan permasalahan permainan Wordscapes. Pada gambar 5, eksekusi algoritma *brute force* mengimplementasikan teknik heuristik seperti yang telah dijelaskan pada bagian sebelumnya. Terdapat ± 1 milidetik perbedaan pada kedua eksekusi program.

IV. KESIMPULAN

Algoritma *brute force* dapat menyelesaikan hampir semua masalah yang ada. Salah satu permasalahan yang dapat diselesaikan menggunakan *brute force* adalah untuk menyelesaikan permasalahan permainan Wordscapes. Dengan menulusuri seluruh hasil permutasi setiap kumpulan huruf yang menjadi kandidat jawaban, akan selalu dipastikan jawaban yang benar berhasil ditemukan.

Teknik heuristik digunakan pada algoritma *brute force* untuk memangkuskan algoritma yang sebelumnya. Pada permasalahan Wordscapes, terdapat beberapa box jawaban yang sebagian hurufnya telah terisi. Oleh karena itu, hanya perlu dilakukan permutasi untuk kotak-kotak huruf yang masih belum terisi saja. Terdapat perbedaan kecepatan eksekusi program antara *brute force* tanpa dan dengan teknik heuristic sebesar 1 milidetik.

VIDEO LINK AT YOUTUBE

Penjelasan lebih lanjut terkait isi dari makalah penulis dapat dilihat dan diakses melalui pranala *Youtube* berikut: (https://youtu.be/BCGiJ4fe8KM).

ACKNOWLEDGMENT *(Heading 5)*

Penulis memanjatkan puji dan syukur kepada Tuhan Yang Maha Esa karena berkat rahmat serta karunia-Nya penulis dapat menulis dan menyelesaikan makalah "Penerapan Metode Heuristik pada Algoritma Brute Force dalam Permainan Wordscapes". Penulis berterima kasih kepada Bapak Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. selaku dosen mata kuliah IF2211 Strategi Algoritma yang telah membimbing saya melalui penjelasan materi di kelas serta pemberian tugas selama kurang lebih satu semester. Terima kasih kepada bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah IF2211 Strategi Algoritma yang telah mengarsipkan materi dan *slide* perkuliahan pada website yang telah terintegrasi. Penulis berterima kasih juga kepada teman-teman Teknik Informatika Institut Teknologi Bandung angkatan 2019, khususnya untuk kelas K-03 pada mata kuliah Strategi Algoritma. Tanpa lupa, penulis berterima kasih kepada sumber-sumber untuk referensi yang telah dicantumkan pada makalah ini untuk digunakan sebenar-benarnya.

REFERENCES

[1] Munir, Rinaldi. Algoritma Brute Force. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)-Bag1.pdf diakses 10 Mei 2021 pukul 15.12.

[2] Karakteristik Algoritma Brute Force [Internet] https://www.berwirausaha.net/2016/02/karakteristik-algoritma-brute-force.html/ diakses 10 Mei 2021 pukul 15.12.

[3] 'Wordscapes' Sea of Words Event Answers [Internet] https://www.newsweek.com/wordscapes-sea-words-event-answers-cheats-solutions-february-12-2019-daily-1328034 diakses 11 Mei 2021 pukul 14.26

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Tangerang Selatan, 11 Mei 2021

Rafi Raihansyah Munandar - 13519154