Penyelesaian Game Wordle dengan Algoritma Greedy dan Decision Tree

Ng Kyle - 13520040
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520040@std.stei.itb.ac.id

***Abstract*—Penyelesaian persoalan menggunakan algoritma dapat diterapkan dalam berbagai aspek kehidupan. Game Wordle yang diciptakan Josh Wordle pada akhir tahun 2021, game yang populer dalam masa pandemi, merupakan game yang memiliki aturan tertentu dengan *goal* dari game agar pemain menebak sebuah kata dengan kesempatan menebak yang terbatas. Kumpulan aturan ini dapat digunakan sebagai dasar dari pengembangan algoritma penyelesaian game Wordle dalam menentukan strategi terbaik yang dapat digunakan dalam menyelesaikan game Wordle. Penggunaan Algoritma berguna dengan mengatasi limitasi kemampuan manusia dalam menampung informasi, sehingga tebakan yang diberikan algoritma memiliki basis dari informasi yang ada sebaik mungkin. Algortima Greedy memanfaatkan aturan-aturan game Wordle sehingga mendapatkan berbagai pilihan kemungkinan tebakan yang baik setiap kesempatannya dengan tujuan menebak dalam kesempatan terbatas yang diberikan**

Keywords—Backtracking; Greedy; Optimization ; Wordle

I. PENDAHULUAN

Game Wordle merupakan game yang diciptakan oleh Josh Wordle pada awal tahun 2021. Game ini terkenal dalam masa pandemi dengan simplisitas model game dan aksesbilitas untuk seluruh usia dari muda ke tua, hingga diakuisisi oleh *The New York Times* sebagai salah satu *game* harian disamping *The Crossword*. Game Wordle memiliki kumpulan aturan yang memberikan informasi kepada pemain mengenai tebakan yang diberikan dan kesesuaian dengan target tebakan. Pemain ditantang untuk menebak kata dengan 5 huruf dalam 6 tebakan atau kurang.

Informasi yang diberikan game Wordle setiap tebakannya menjadi informasi unik dan menjadi tantangan dalam membuat algoritma yang baik dalam memanfaatkan informasi-informasi tersebut sebaik mungkin. Game Wordle yang masih relative baru sehingga masih sangat minim ajuan strategi algoritma yang dapat digunakan dalam menyelesaikan game ini. Menjadi pertimbangan apakah kata yang terbaik yang dapat digunakan sebagai kata pembukan (kata pertama) sebagai tebakan pertama. Selain itu, bagaimana jenis-jenis informasi dapat dimanfaatkan dan masing-masing efektivitas dari penggunaan informasi tersebut.

Algoritma Greedy merupakan salah satu algoritma alternatif yang memanfaatkan *current best move* dalam menyelesaikan persoalan menjadi salah satu pilihan algortima yang dapat dimanfaatkan dalam game ini. Penggunaan algoritma greedy ini memungkinkan pemilihan keputusan tebakan kata yang akan dipilih dari informasi yang telah didapatkan, hal ini berguna mengingat *search space* yang sangat luas ketika melakukan enumerasi seluruh kemungkinan yang ada (mempertimbangkan besar alfabet dan kumpulan kata yang valid). Hal ini mendukung untuk mendapatkan *local minimum* dengan harapan keberhasilan penebakan kata dalam tebakan sedikit mungkin, *global minimum*. Penggunaan Algoritma Decision Tree juga dapat dimanfaatkan untuk melakukan pemilihan keputusan memanfaatkan informasi pengetahuan terhadap dataset (wordlist) dan mempertimbangkan searchspace setiap kata.

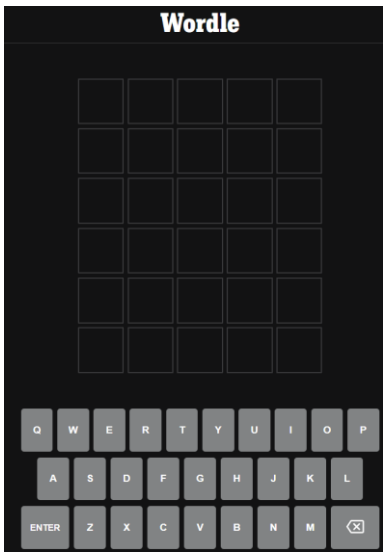

Fig. 1. Tampilan Game Wordle
(https://www.nytimes.com/games/wordle/index.html)

II. DASAR TEORI

A. Algoritma Greedy

Algoritma Greedy merupakan pendekatan algoritma yang menghasilkan solusi berdasarkan sekuens langkah-langkah yang sudah dilakukan sebelumnya dan memilih solusi terbaik setiap langkahnya hingga solusi lengkap dari persoalan didapatkan[1]. Algoritma Greedy dibagi menjadi beberapa elemen penyusun[2]:

- Himpunan Kandidat, yang berisi kandidat yang akan dipilih setiap langkahnya.
- Himpunan solusi, yang berisi kandidat/langkah yang sudah dipilih.
- Fungsi solusi, untuk menentukan himpunan solusi yang telah dimiliki memenuhi kriteria (merupakan solusi dari persoalan).
- Fungsi seleksi, untuk memilih dari himpunan kandidat (merupakan implementasi strategi Greedy) dengan heuristik Greedy,
- Fungsi kelayakan, untuk menentukan kandidat yang dipilih memenuhi batasan (dapat dipilih sebagai solusi).
- Fungsi objektif, untuk memaksimumkan atau meminimumkan solusi yang diinginkan.

Algoritma Greedy tidak selalu menghasilkan solusi optimum global (sub-optimal) sebagai *tradeoff* dari sifatnya yang tidak memeriksa seluruh kemungkinan sehingga algoritma akan berjalan lebih cepat.

B. Decision Tree

Decision Tree merupakan algoritma yang memanfaatkan pohon (Tree) sebagai penentu keputusan berdasarkan inpur yang diberikan (kondisi). Decision Tree memiliki beberapa komponen[3]:

- Decision Nodes
- End Nodes
- Branches

Dalam penentuan keputusan, algoritma akan mencapai End Nodes yang menentukan keputusan yang dilakukan. Untuk mencapai End Nodes ini dilakukan pruning. Pruning ini dilakukan berdasarkan information gain (informasi yang didapat)[4].

C. Wordle Game

Wordle game merupakan game yang menuntut pemain untuk menebak sebuah kata dengan panjang 5 huruf. Untuk memenangkan permainan, pemain harus menebak kata tersebut dalam 6 langkah atau kurang. Wordle game sebagaimana pada lama resminya memiliki aturan sebagai berikut[5]:

1. Menebak kata (WORDLE) dalam 6 kesempatan tebakan.
2. Setiap tebakan merupakan kata yang valid (dalam wordlist / kamus).
3. Setelah setiap tebakan, warna dari tiles akan berubah sesuai dengan kemiripan kata tebakan dengan kata yang harus ditebak. Perubahan warna adalah sebagai berikut:
    a. Tile akan berubah warna hijau ketika huruf pada tebakan bersesuaian dengan kata yang harus ditebak pada tile bersesuaian.
    b. Tile akan berubah warna kuning ketika huruf tebakan pada tile bersesuaian pada kata yang harus ditebak namun tidak berada pada posisi tersebut.
    c. Tile akan tetap jika tidak memenuhi kedua kondisi (a) dan (b).

Sebagai tambahan, huruf yang sudah dipenuhi oleh kriteria (a) tidak akan dipertimbangkan dalam kriteria (b). Kriteria (b) akan mempertimbangkan sesuai dengan jumlah kemunculan (frekuensi kemunculan) huruf pada kata yang ditebak. Sehingga, jika sebuah kata memiliki 2 huruf A, namun tidak berada pada

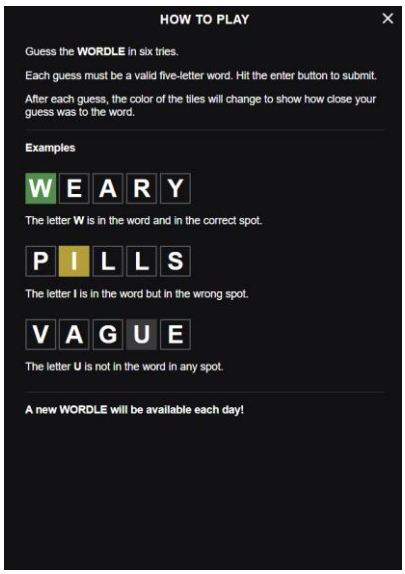

Fig. 2. Aturan WORDLE Official (https://www.nytimes.com/games/wordle/index.html)

III. Implementasi dan Pembahasan

Penyelesaian game Wordle secara algoritmis akan diperlukan beberapa tahap:

1. Pre-processing kamus (wordlist)
2. Penentuan kata pertama (opening word)
3. Penebakan (Penentuan kata yang akan ditebak)

A. Preprocessing

Pada tahap pre-processing, kita akan membaca wordlist yang disediakan. Wordlist dapat apa saja selama alphabet terdefinisi, untuk hal ini menggunakan wordlist yang digunakan game Wordle dengan jumlah kata yang digunakan sebanyak 12947 kata yang termasuk dalam kamus bahasa Inggris[6] (Hal ini dapat diubah menggunakan wordlist lain, namun dalam pembahasan tulisan ini akan menggunakan wordlist tersebut).

Terdapat beberapa konsiderasi processing wordlist, yaitu:

1. Menilai setiap alfabet sesuai dengan kemunculannya pada seluruh kata dalam wordlist yang distinct. (Kemunculan alfabet pada setiap kata hanya dihitung 1 kali). Penilaian huruf ini digunakan untuk menilai

setiap kata menggunakan alfabet pada kata tersebut dengan nilai alfabet yang unik pada kata tersebut.

2. Menilai setiap alfabet sesuai dengan banyaknya kemunculan pada posisi tertentu (posisi 1 hingga 5 mempertimbangkan kata yang ditebak selalu memiliki panjang 5). Menilai setiap kata berdasarkan alfabet dengan posisi bersesuaian.

Hasil dari kedua processing wordlist (nilai alfabet dan kata pada wordlist) adalah sebagai berikut. Untuk opsi pertama (Preprocessing 1), kita dapatkan statisitik:

TABLE I. KEMUNCULAN ALFABET PADA WORDLIST DISTINCT

A	5324	H	1702	O	3904	V	673
B	1516	I	3582	P	1880	W	1026
C	1913	J	289	Q	111	X	287
D	2294	K	1435	R	3905	Y	2024
E	5696	L	3109	S	5924	Z	391
F	987	M	1867	T	3030		
G	1539	N	2783	U	2433		

TABLE II. NILAI KATA BERDASARKAN KEMUNCULAN ALFABET

5 Terbaik		**5 Terburuk**	
AEROS	24753	XYLYL	5420
AROSE	24753	GYPPY	5443
SOARE	24753	FUFFY	5444
AESIR	24431	HYPHY	5606
ARISE	24431	QAJAQ	5724

Menggunakan cara Preprocessing 1, didapat bahwa banyak kata yang memiliki komposisi alfabet penyusun yang sama akan memiliki nilai yang sama (AEROS, AROSE, dan SOARE memiliki alfabet A, E, O, R, S). Hal ini dapat menimbulkan efek kurang baik ketika diperlukan menentukan kata tertentu (perlu memilih secara acak untuk nilai yang sama).

Untuk opsi kedua (Preprocessing 2), kita dapatkan statistik:

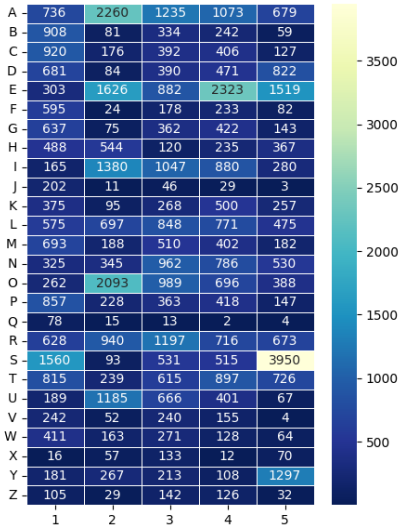

Fig. 3. Heatmap Kemunculan Alfabet tiap Posisi

TABLE III. NILAI KATA BERDASARKAN POSISI ALFABET

5 Terbaik		**5 Terburuk**	
SORES	11123	ENZYM	1080
SANES	11055	ETHYL	1245
SALES	10941	EWHOW	1346
SONES	10888	IMSHI	1399
SATES	10708	OXBOW	1413

Menggunakan opsi Preprocessing 2 didapatkan penilaian yang lebih menyebar. Hal ini mempertimbangkan kemungkinan munculnya alfabet pada posisi tertentu. Sehingga, kata yang memiliki komposisi alfabet yang sama akan memiliki nilai yang berbeda. Selain itu, penilaian per kata yang berbeda akan menghasilkan tebakan yang berbeda setiap tebakannya.

B. Greedy on Rules

Menggunakan hasil dari tahap preprocessing, kita mendapatkan informasi mengenai nilai dari setiap kata dalam wordlist yang telah dimiliki. Terdapat 2 kemungkinan tahap preprocessing yang masing-masing memiliki kata awal terbaik yang terbaik, yaitu AEROS/AROSE/SOARE menggunakan penilaian preprocessing 1 atau SORES menggunakan tahap preprocessing 2. Selanjutnya diperlukan algoritma yang memanfaatkan informasi dari game. Dalam tahapan ini, dapat dibagi menjadi 3, yaitu:

1. Informasi Hijau – (alfabet pada kata tebakan dan target pada posisi yang bersesuaian sama).
2. Informasi Kuning – (alfabet pada kata tebakan pada posisi tertentu terdapat pada kata target namun pada posisi berbeda)

3. Informasi Putih – (alfabet pada kata tebakan tidak terdapat pada kata target)

Algoritma Greedy yang digunakan dalam memanfaatkan informasi adalah menggunakan informasi bahwa terdapat sebuah alfabet yang sesuai posisinya pada kata target, maka tetap gunakan pada tebakan selanjutnya.

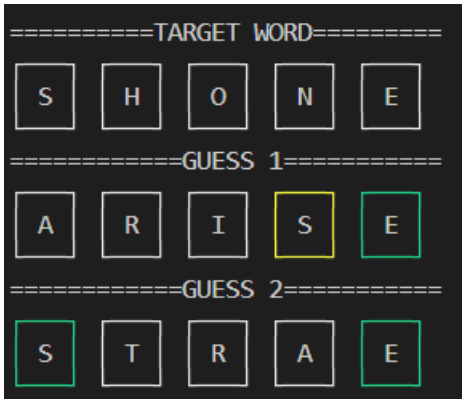

Fig. 4. Contoh Penerapan Informasi Hijau

- Himpunan solusi: wordlist tersisa
- Fungsi solusi: fungsi checkGuess; Memeriksa apakah tebakan sudah benar (hijau) semua.
- Fungsi seleksi: Memilih kata yang dengan nilai terbesar dan memenuhi kriteria (terdapat alfabet bersesuaian pada lokasi hijau)
- Fungsi kelayakan: Menentukan apakah sebuah kata dapat dipilih (setiap alfabet pada kata termasuk pada toEvaluate)
- Fungsi objektif: Menebak kata dalam 6 tebakan atau kurang

```
def solveGreen(wordscores, guessWord, toEvaluate, guessResult):
    #Fungsi Seleksi
    wordscores.pop(guessWord,None)
    key_list= list(wordscores)
    indices = []
    for i in range(5):
        if(guessResult[i] == Fore.GREEN):
            toEvaluate[i] = guessWord[i]
            indices.append(i)
    for wordsLeft in key_list:
        for i in indices:
            #Fungsi Kelayakan
            if (wordsLeft[i] not in toEvaluate[i]):
                wordscores.pop(guessWord,None)
                break
    return wordscores, toEvaluate
```

Algoritma lainnya selain memanfaatkan masing-masing

Selain dari informasi hijau, perlu digunakan informasi kuning (alfabet berada pada kata target namun tidak pada posisi yang benar). Dalam menggunakan informasi ini, kita dapat melakukan filter terhadap wordlist yang kita miliki sehingga wordlist tersisa ketika memiliki alfabet yang bersesuaian. Sehingga word pada wordlist dihapus jika memenuhi kriteria: word tidak memiliki alfabet kuning atau pada posisi bersesuaian word dan kuning memiliki alfabet yang sama.

- Himpunan solusi: wordlist tersisa dari solveGreen
- Fungsi solusi: fungsi checkGuess; Memeriksa apakah tebakan sudah benar (hijau) semua.
- Fungsi seleksi: Memilih kata yang dengan nilai terbesar dan memenuhi kriteria, yaitu terdapat alfabet bersesuaian pada kata (kecuali yang sudah ditemukan hijau)
- Fungsi kelayakan: Menentukan apakah sebuah kata dapat dipilih (setiap alfabet pada kata termasuk pada toEvaluate)
- Fungsi objektif: Menebak kata dalam 6 tebakan atau kurang

```
def solveYellow(wordscores,guessWord, toEvaluate, guessResult):
    #Fungsi Seleksi
    wordscores.pop(guessWord,None)
    key_list= list(wordscores)
    indices = []
    for i in range(5):
        if(guessResult[i] == Fore.YELLOW):
            toEvaluate[i]= toEvaluate[i].replace(guessWord[i], '')
            indices.append(i)
    for wordsLeft in key_list:
        for i in indices:
            #Fungsi Kelayakan
            if (guessWord[i] not in wordsLeft or wordsLeft[i] not in toEvaluate[i]):
                wordscores.pop(wordsLeft,None)
                break
    return wordscores, toEvaluate
```

Terakhir, memanfaatkan informasi dari alfabet yang tidak terdapat pada kata target, program dapat melakukan filter terhadap huruf yang valid dengan menghapus kata pada wordlist yang memiliki alfabet yang tidak terdapat pada kata tersebut. Sehingga sebuah word pada wordlist akan dihapus ketika memiliki alfabet yang bersesuaian pada posisi yang belum ditemui informasi (kuning maupun hijau).

- Himpunan solusi: wordlist tersisa dari solveGreen
- Fungsi solusi: fungsi checkGuess; Memeriksa apakah tebakan sudah benar (hijau) semua.
- Fungsi seleksi: Memilih kata yang dengan nilai terbesar dan memenuhi kriteria, yaitu tidak terdapat alfabet bersesuaian pada posisi yang tidak memiliki informasi hijau maupun kuning.
- Fungsi kelayakan: Menentukan apakah sebuah kata dapat dipilih (setiap alfabet pada kata termasuk pada toEvaluate)
- Fungsi objektif: Menebak kata dalam 6 tebakan atau kurang

```
def solveGray(wordscores,guessWord, toEvaluate, guessResult):
    #Fungsi Seleksi
    wordscores.pop(guessWord,None)
    key_list= list(wordscores)
    indices = []
    for i in range(5):
        if(guessResult[i] == Fore.WHITE):
            indices.append(i)
     for i in indices:
        toEvaluate[i] = toEvaluate[i].replace(guessWord[i], '')
    for wordsLeft in key_list:
        for i in indices:
            #Fungsi Kelayakan
            if(wordsLeft[i] not in toEvaluate[i]):
                wordscores.pop(wordsLeft,None)
                break
  return wordscores, toEvaluate
```

C. Building Decision Tree

Pengetahuan terhadap wordlist dapat digunakan untuk membentuk Decision Tree berdasarkan hasil evaluasi antara satu kata dengan kata lainnya lalu mengurangi search space menggunakan filter hasil evaluasi kata yang ditebak. Lalu menggunakan informasi tersebut membentuk sebuah decision tree dengan komponen:

- Nodes: Kata
- Branch: Evaluasi

Karena pada setiap posisi alfabet pada kata dapat menghasilkan salah satu dari 3 evaluasi yaitu Green (G), Yellow (Y), White (W) sehingga setiap branchnya memiliki maksimum 3^5 branch. Namun, tidak semua kata memiliki branch maksimum.

TABLE IV. ILUSTRASI LOOKUP TABLE UNTUK DECISION TREE

AAHED	WWWWW	[BEACH, …]
	…	…
	GGWWW	[AALII, AARTI]
…	…	…
ZYMID	WWWWW	[AAHED, AARGH, ABAFT, ABAKA, …]
	…	
	GWGWW	[ZAMAN, ZAMBO]

```
#Fungsi Pembangkit Decision Tree
def buildTree(wordlist):
    tree = {}
    for word in wordlist:
        tree[word] = {}
        for referWord in wordlist:
            if(word != referWord):
                eval = str(evalGuess(word, referWord))
                tree[word].setdefault(eval, [])
                tree[word][eval].append(referWord)
  return tree
#Fungsi Penyelesaian dengan Decision Tree
def solveTree(wordscores,guessWord,tree, guessResult):
    filter = tree[guessWord][str(guessResult)]
    for word in list(wordscores):
        if word not in filter:
            wordscores.pop(word,None)
  return wordscores
```

IV. Eksperimen dan Hasil

Menggunakan pembahasan sebelumnya, didapatkan komponen penyelesaian persoalan sebagai berikut:

1. Algoritma Preprocess (Penilaian Kata):
    - Preprocess 1: Menilai berdasarkan kemunculan huruf pada kata distinct.
    - Preprocess 2: Menilai berdasarkan kemunculan huruf berdasarkan posisi.
2. Algoritma Penyelesaian:
    - Greedy by Informations
    - Decision Tree

Menggunakan kombinasi antara algoritma preprocess dan penyelesaian akan didapat 4 buah kombinasi. Dalam mengevaluasi setiap algoritma dalam menyelesaikan game Wordle ini dilakukan simulasi game untuk setiap kata pada kamus yang digunakan. Berikut merupakan statistik hasil masing-masing kombinasi penyelesaian persoalan.

A. Preprocess 1 + Greedy by Rules

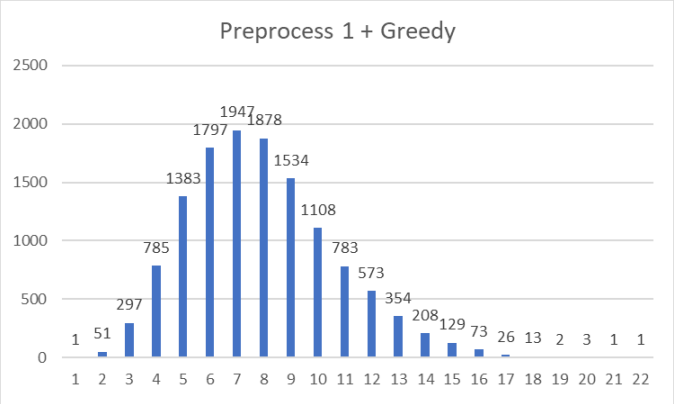

Fig. 5. Histogram Frekuensi Tebakan Strategi Preprocess 1 + Greedy by Rules

B. Preprocess 2 + Greedy by Rules

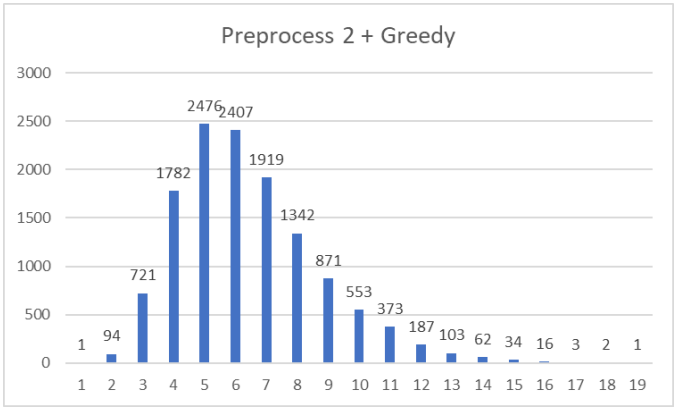

Fig. 6. Histogram Frekuensi Tebakan Strategi Preprocess 2 + Greedy by Rules

C. Preprocess 1 + Decision Tree

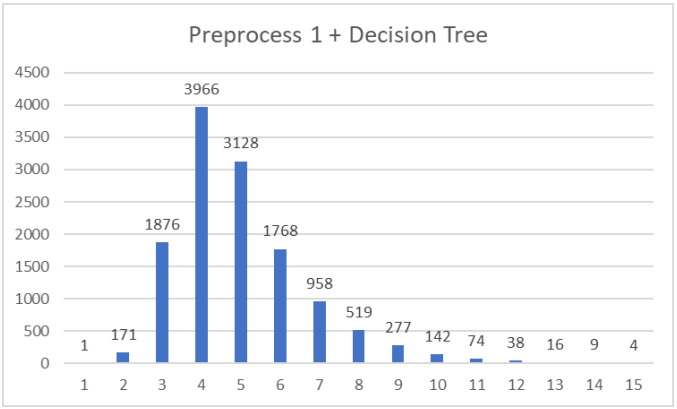

Fig. 7. Histogram Frekuensi Tebakan Strategi Preprocess 1 + Decision Tree

D. Preprocess 2 + Decision Tree

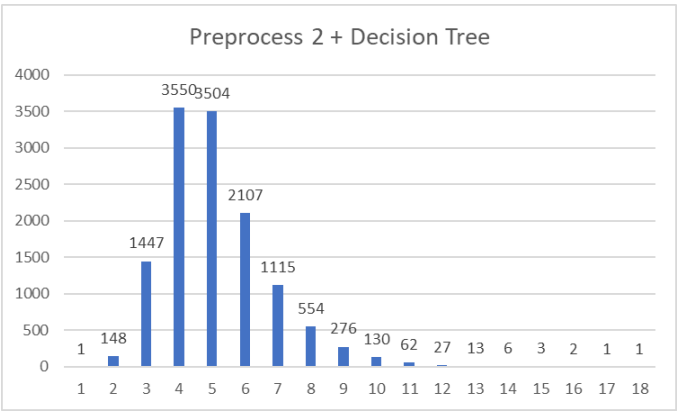

Fig. 8. Histogram Frekuensi Tebakan Strategi Preprocess 2 + Decision Tree

Statistik keberhasilan setiap kombinasi strategi menggunakan wordlist dengan banyak kata pada wordlist 12947 kata:

TABLE V. Combinations of Stragies Success (Success Rate)

	Greedy by Rules	**Decision Tree**
Preprocess 1	4314 (0.333204603)	10910 (0.842666255)
Preprocess 2	7481 (0.577817255)	10757 (0.830848845)

TABLE VI. Combinations of Stragies Average Guess Attempts

	Greedy by Rules	**Decision Tree**
Preprocess 1	7.900903684	4.98323936
Preprocess 2	6.431296826	5.127983317

Dari statistik yang didapat, terlihat bahwa secara umum Decision Tree memiliki performansi yang jauh lebih baik dengan persentase keberhasilan lebih dari 83% menggunakan

preprocess 1 maupun preprocess 2. Sedangkan Preprocess 1 maupun Preprocess 2 efektif pada stratefi penyelesaian tertentu yaitu strategi Greedy by Rules memiliki performansi yang lebih baik menggunakan Preprocess 1 sedangkan strategi Decision Tree memiliki performasi yang lebih baik menggunakan Preprocess 2.

Namun, Decision Tree memiliki kekurangan yaitu memiliki waktu overhead dalam pembuatan Tree karena perlu melakukan evaluasi setiap kata dengan kata lainnya pada wordlist. Dalam hal ini, Tree yang dibuat perlu melakukan evaluasi sebanyak 12947 x 12947. Selain itu selama melakukan simulasi terhadap seluruh wordlist menggunakan 100 thread, strategi yang menggunakan Decision Tree akan memakan waktu lebih dari 10 menit (tanpa melakukan pembangkitan Tree), sedangkan Greedy by Rules hanya memerlukan waktu kurang dari 3 menit.

V. KESIMPULAN

Penyelesaian game Wordle dapat dilakukan dengan algoritma Greedy maupun Decision Tree. Penggunaan Decision Tree akan memberikan performansi yang lebih baik daripada algoritma Greedy karena mempertimbangkan seluruh search space dari informasi evaluasi antara setiap kata. Decision Tree memiliki kekurangan dalam aspek waktu yang digunakan akan lebih lambat dari algoritma Greedy, khsusunya dalam pembangkitan Tree dan pruning Tree dalam mencapai keputusan.

Hasil dari eksperimen yang dilakukan mendapatkan hasil terbaik dalam menggunakan algoritma Preprocess 2 yang mempertimbangkan kemunculan huruf setiap posisi pada seluruh kata pada wordlist sebagai penilaian untuk setiap kata pada wordlist bersamaan dengan algoritma Decision Tree dalam melakukan pencarian solusi (penebakan terhadap kata). Hasil terbaik memiliki rata-rata tebakan hingga menemukan solusi sebanyak 4.98323936 dengan tingkat sukses 0.842666255 (84%).

VI. VIDEO LINK AT YOUTUBE

https://youtu.be/D9vt6mN4Oks

VII. ACKNOWLEDGMENT

Saya ingin berterima kasih kepada tim pengajar IF2211 yang memberikan kesempatan dalam pembuatan makalah ini. Selain itu saya berterima kasih kepada pembuat game Wordle,Josh Wordle,sehingga saya dapat membuat makalah ini berdasarkan game Wordle.

VIII. REFERENCES

[1] Levitin, A., 2014. Introduction to the Design and Analysis of Algorithms. Pearson Education UK, pp. 315-316.

[2] Munir, R., 2022. [online] Informatika.stei.itb.ac.id. Available at: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf> [Accessed 5 May 2022].

[3] Nytimes.com. 2022. Wordle - A daily word game. [online] Available at: <https://www.nytimes.com/games/wordle/index.html> [Accessed 5 May 2022].

[4] Kamiński B, Jakubczyk M, Szufel P. A framework for sensitivity analysis of decision trees. Cent Eur J Oper Res. 2018;26(1):135-159. doi:10.1007/s10100-017-0479-6

[5] Cs.cmu.edu. 2022. Decision Trees. [online] Available at: <https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/> [Accessed 8 May 2022].

[6] https://github.com/tabatkins/wordle-list

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.

Bandung, 20 Mei 2022

Ng Kyle 13520040