Optimalisasi Penyelesaian Wordle: Perbandingan Brute Force dan Strategi Greedy Heuristik

Anella Utari Gunadi - 13523078
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: anellautari@gmail.com , 13523078@std.stei.itb.ac.id

Abstrak—Wordle merupakan permainan teka-teki kata yang menguji kemampuan pemain dalam menebak kata lima huruf dalam enam langkah atau kurang. Meskipun tampak sederhana, permainan ini menyimpan kompleksitas dalam memilih strategi penebakan yang efisien. Makalah ini membahas dua pendekatan algoritmik untuk menyelesaikan Wordle, yaitu *brute force* dan *greedy* heuristik. Strategi *brute force* menebak setiap kemungkinan secara sistematis tanpa evaluasi tambahan, sementara strategi *greedy* memilih kata berdasarkan nilai heuristik seperti frekuensi dan keunikan huruf. Implementasi dilakukan menggunakan bahasa Python dengan sumber kata dari pustaka NLTK. Berdasarkan hasil eksperimen terhadap lima kata target, strategi *greedy* cenderung lebih efisien dalam jumlah langkah, sedangkan *brute force* memiliki waktu eksekusi yang lebih cepat. Perbandingan ini menunjukkan bahwa pendekatan *greedy* lebih cocok digunakan untuk situasi yang menekankan efisiensi jumlah tebakan, sementara *brute force* tetap relevan untuk pendekatan sederhana dan cepat secara komputasi.

Kata Kunci—*wordle, brute force, greedy, python, nltk*

Abstract—Wordle is a word-guessing puzzle game where players attempt to guess a five-letter word within six tries. Although the rules appear simple, choosing the most efficient guessing strategy involves computational complexity. This paper compares two algorithmic approaches to solving Wordle: brute force and greedy heuristic. The brute force strategy systematically tests all possible words without evaluation, while the greedy strategy selects words based on heuristics such as letter frequency and uniqueness. The algorithms are implemented using Python, with a five-letter word list sourced from the NLTK corpus. Experiments on five target words show that the greedy strategy generally requires fewer guesses, whereas brute force yields faster execution time. These findings suggest that the greedy approach is preferable in scenarios prioritizing guess efficiency, while brute force remains suitable for simple and fast implementations.

Keywords—wordle, brute force, greedy, python, nltk

I. PENDAHULUAN

Wordle merupakan sebuah permainan teka-teki kata yang menjadi fenomena global sejak diluncurkan pada tahun 2021. Dalam permainan ini, pemain diminta menebak sebuah kata yang terdiri dari lima huruf dalam enam percobaan atau kurang. Setiap tebakan akan diberi umpan balik berupa warna, seperti warna hijau jika huruf dan posisinya benar, kuning jika huruf benar tetapi posisi salah, dan abu-abu jika huruf tersebut tidak terdapat dalam kata target. Meskipun aturannya sederhana, permainan ini menantang karena membutuhkan strategi dalam memilih kata yang tepat di setiap langkah.

Di balik kesederhanaan tampilannya, Wordle menyimpan kompleksitas dalam hal strategi penyelesaiannya. Pemain harus mempertimbangkan berbagai kemungkinan kata berdasarkan umpan balik yang diterima, sehingga dapat menyempitkan ruang solusi seefisien mungkin. Permainan ini menjadi menarik untuk ditelaah dari sisi algoritma, khususnya dalam mencari tahu bagaimana sebuah program dapat menyelesaikan Wordle dengan cara yang efisien menggunakan pendekatan tertentu.

Dalam makalah ini, dua pendekatan algoritma dibandingkan, yaitu *brute force* dan *greedy* heuristik. Strategi *brute force* mencoba setiap kemungkinan secara berurutan sambil menyaring berdasarkan umpan balik, tanpa mempertimbangkan seberapa informatif atau bergunanya suatu tebakan. Sebaliknya, strategi *greedy* menggunakan heuristik untuk memilih kata yang dianggap paling informatif pada setiap langkah, dengan harapan dapat mempercepat proses pencarian jawaban. Perbandingan antara kedua strategi ini menjadi menarik karena keduanya relatif sederhana untuk diimplementasikan, namun memiliki performa yang bisa sangat berbeda tergantung kasusnya. Selain itu, studi semacam ini juga memberikan pemahaman yang lebih konkret mengenai konsep eksplorasi ruang solusi, *filtering*, serta pemanfaatan heuristik dalam pengambilan keputusan algoritmik.

Oleh karena itu, makalah ini bertujuan untuk mengimplementasikan kedua strategi tersebut dalam konteks penyelesaian permainan Wordle. Selain itu, makalah ini juga akan melakukan pengujian terhadap beberapa kata target, membandingkan hasil performa dari kedua algoritma, serta menganalisis kelebihan dan kekurangan masing-masing pendekatan.

II. LANDASAN TEORI

2.1 Permainan Wordle

Wordle adalah permainan teka-teki kata sederhana yang sempat viral di berbagai platform media sosial. Dalam

permainan ini, pemain diminta untuk menebak sebuah kata yang terdiri dari lima huruf, dan hanya diberikan enam kesempatan untuk menebaknya dengan benar. Setelah setiap tebakan, pemain akan mendapatkan petunjuk berupa warna hijau menandakan bahwa huruf tersebut sudah tepat baik dari sisi huruf maupun posisinya, warna kuning yang berarti huruf tersebut ada dalam kata target tetapi memiliki posisi yang salah, sedangkan warna abu-abu menunjukkan bahwa huruf tersebut tidak ada sama sekali dalam kata target.

(Gambar 2.1 Contoh Permainan Wordle)

Wordle memang tampak seperti permainan santai yang dapat dimainkan untuk mengisi waktu luang. Namun, di balik tampilannya yang sederhana, permainan ini menuntut kemampuan berpikir logis, memilah kemungkinan, dan memilih kata dengan cermat. Setiap umpan balik yang diberikan setelah tebakan sebelumnya harus dimanfaatkan untuk menyempitkan pilihan dan membuat tebakan berikutnya lebih tepat. Inilah yang membuat Wordle menarik untuk dikaji dari sisi algoritmik karena setiap langkah pemain sebenarnya mencerminkan proses penyaringan, strategi, dan kadang bahkan intuisi, yang bisa dimodelkan dalam bentuk algoritma pemecahan masalah.

2.2 Algoritma Brute Force

Algoritma *brute force* adalah salah satu pendekatan paling dasar dalam menyelesaikan suatu permasalahan, di mana semua kemungkinan solusi dicoba satu per satu hingga ditemukan jawaban yang tepat[1]. Algoritma ini memecahkan persoalan dengan cara yang sangat sederhana, langsung, dan mudah dipahami[2]. Pendekatannya tidak melibatkan langkah-langkah rumit, melainkan cukup mengikuti alur logika yang jelas sesuai dengan yang tertulis dalam soal. Bisa dibilang, *brute force* menganut prinsip "*just do it!*" tanpa perlu memikirkan strategi optimasi yang kompleks.

Dalam praktiknya, *brute force* sering digunakan dalam kasus seperti pemecahan sandi atau pencarian kombinasi tertentu, di mana setiap kemungkinan diuji secara berurutan sampai hasil yang diinginkan ditemukan. Meskipun sederhana, keunggulan metode ini terletak pada kepastiannya karena dengan mencoba seluruh kemungkinan, solusi yang benar pasti akan ditemukan. Namun, pendekatan ini juga dikenal boros waktu dan sumber daya, terutama jika ruang solusinya sangat besar.

Dari sisi kompleksitas waktu, algoritma brute force cenderung memiliki performa yang kurang efisien, terutama ketika ruang solusi sangat besar. Karena pendekatannya menguji setiap kemungkinan yang ada, kompleksitas waktunya dapat mencapai $O(n)$, $O(n^2)$, atau bahkan lebih, tergantung pada ukuran dan struktur ruang solusinya.

Cara kerja algoritma *brute force* dapat dibilang cukup langsung dan sederhana. Terdapat beberapa langkah dalam algoritma *brute* force, yaitu inisialisasi, iterasi, pengujian, pemutakhiran, dan penyelesaian[3]. Langkah pertama adalah dengan inisialisasi yaitu menyiapkan segala hal yang dibutuhkan, seperti variabel awal, daftar kandidat solusi, atau parameter lainnya. Selanjutnya, masuk ke langkah iterasi yaitu memulai proses pencarian dengan mencoba setiap kemungkinan satu per satu secara berurutan.

Langkah selanjutnya adalah dengan melakukan pengujian pada setiap iterasi, yaitu mengecek apakah kemungkinan solusi yang sedang dicoba sudah sesuai dengan jawaban yang dicari. Jika solusi tersebut belum tepat, maka masuk ke tahap pemutakhiran, di mana algoritma melanjutkan ke kemungkinan berikutnya dan kembali mengulang proses yang sama. Proses ini terus berlangsung hingga sampai pada tahap penyelesaian, yaitu ketika solusi yang benar berhasil ditemukan dan algoritma dapat menghentikan pencarian.

Salah satu kelebihan utama dari algoritma *brute force* adalah kemudahannya dalam diterapkan. Pendekatan ini tidak memerlukan pemahaman mendalam tentang konsep matematika atau logika yang rumit, sehingga cocok digunakan bahkan oleh pemula. Selain itu, karena *brute force* mencoba semua kemungkinan yang ada, hasil yang benar dijamin akan ditemukan pada akhirnya. Metode ini juga cukup fleksibel dan bisa diterapkan untuk berbagai jenis permasalahan, mulai dari pencarian teks dalam string hingga pemecahan kata sandi.

Namun, kelemahan utama dari *brute force* terletak pada efisiensinya. Ketika berhadapan dengan ruang solusi yang sangat besar, algoritma ini dapat menghabiskan waktu dan sumber daya komputasi dalam jumlah besar. Dalam kasus tertentu seperti memecahkan enkripsi yang kompleks, pendekatan brute force bisa memerlukan waktu yang sangat lama sehingga kurang praktis untuk diterapkan secara nyata.

2.3 Algoritma Greedy

Algoritma *greedy* adalah metode penyelesaian masalah yang dilakukan secara bertahap, langkah demi langkah[4]. Pada setiap langkah, algoritma ini akan memilih opsi terbaik yang tersedia saat itu juga, tanpa mempertimbangkan dampak atau konsekuensi terhadap langkah-langkah selanjutnya. Pendekatan ini mengikuti prinsip "ambil yang bisa didapat sekarang" (*take what you can get now!*).

Harapannya, dengan terus memilih solusi terbaik di tiap langkah (optimum lokal), algoritma ini akan mengarah pada solusi terbaik secara keseluruhan (optimum global). Meskipun tidak selalu menghasilkan solusi yang paling optimal, strategi greedy seringkali cukup efektif dan efisien untuk berbagai jenis masalah.

Algoritma greedy memiliki karakteristik khas dalam menyelesaikan masalah dengan mencari solusi yang optimal, baik berupa nilai minimum maupun maksimum[5]. Pemilihan solusi dilakukan berdasarkan opsi terbaik yang tersedia pada setiap langkah. Pendekatan ini bersifat langsung dan fokus pada hasil terbaik saat itu juga, tanpa mempertimbangkan dampak jangka panjang dari keputusan tersebut.

Salah satu keunggulan algoritma greedy adalah kecepatannya. Algoritma ini umumnya memiliki kompleksitas waktu sebesar O(n log n) atau bahkan O(n), sehingga cukup efisien dan cocok digunakan untuk menyelesaikan masalah berskala besar. Selain itu, algoritma greedy bekerja tanpa perlu melakukan pengulangan atau backtracking yang artinya pencarian solusi hanya dilakukan sekali dari awal hingga akhir. Karena alurnya yang sederhana, algoritma ini juga mudah diimplementasikan.

Sebelum menerapkan algoritma greedy pada suatu permasalahan, penting untuk memastikan bahwa sifat dari masalah tersebut memang sesuai. Sebelum menerapkan algoritma greedy, penting untuk mempertimbangkan dua hal: apakah keputusan terbaik perlu diambil pada setiap langkah, dan apakah solusi yang diinginkan bersifat optimal, seperti nilai minimum atau maksimum. Jika kedua kondisi ini terpenuhi, maka strategi greedy merupakan pendekatan yang sesuai.

Algoritma greedy memiliki sejumlah keunggulan yang membuatnya populer digunakan dalam berbagai jenis masalah. Salah satunya adalah kesederhanaannya, algoritma ini cukup mudah dipahami dan diimplementasikan[6]. Selain itu, karena hanya mengambil satu keputusan terbaik di setiap langkah tanpa perlu mempertimbangkan semua kemungkinan, prosesnya cenderung cepat dan efisien. Dalam banyak kasus, solusi yang dihasilkan oleh algoritma greedy cukup mendekati optimal meskipun tidak selalu sempurna.

Di sisi lain, pendekatan *greedy* juga memiliki beberapa kelemahan. Salah satu keterbatasannya adalah bahwa strategi ini tidak menjamin akan selalu menghasilkan solusi terbaik secara keseluruhan. Ada situasi di mana keputusan optimal lokal justru menjauhkan algoritma dari solusi global yang seharusnya. Selain itu, jika pemilihan kriteria "kerakusan" atau penilaian terbaik pada tiap langkah tidak dirancang dengan tepat, hasil akhirnya bisa jauh dari harapan. Karena tidak mempertimbangkan dampak jangka panjang dari setiap keputusan, algoritma ini bisa menghasilkan solusi yang kurang optimal di beberapa kasus.

III. STRATEGI PENYELESAIAN WORDLE

Permainan Wordle, meskipun tampak sederhana, menyimpan tantangan dalam menentukan kata yang tepat dalam jumlah langkah sesedikit mungkin. Oleh karena itu, dibutuhkan pendekatan yang tepat untuk mengeksplorasi ruang solusi dan mengeliminasi kemungkinan secara efisien. Dua strategi utama yang digunakan untuk menyelesaikan permainan ini adalah *brute force* dan *greedy* heuristik. Keduanya memiliki karakteristik yang berbeda dalam cara mereka menebak, menyaring kemungkinan, dan memanfaatkan informasi dari umpan balik (*feedback*) Wordle.

3.1 Strategi Brute Force

Strategi *brute force* yang diimplementasikan pada penyelesaian permainan Wordle adalah dengan cara mencoba seluruh kemungkinan kata secara sistematis. Setiap kata yang terdiri dari lima huruf dianggap sebagai kandidat solusi dan akan diperiksa satu per satu, yaitu dari urutan awal secara alfabetis. Tidak ada pertimbangan khusus dalam memilih tebakan, jadi strategi ini hanya terus menebak berdasarkan urutan yang telah ditentukan sebelumnya hingga menemukan jawaban yang benar.

Pada setiap langkah, hasil tebakan dibandingkan dengan kata target dan diberikan umpan balik berupa tiga kategori warna, yaitu hijau jika huruf dan posisinya tepat, kuning jika huruf berada dalam kata tetapi salah posisi, dan abu-abu jika huruf tidak ada dalam kata sama sekali. Informasi ini digunakan untuk menyaring kemungkinan kata-kata selanjutnya, misalnya dengan memastikan huruf-huruf hijau tetap pada posisinya, huruf kuning tidak muncul di posisi yang sama lagi, dan huruf abu-abu tidak muncul lagi dalam tebakan kecuali sudah terbukti relevan.

Meskipun strategi ini menggunakan umpan balik untuk menyaring kemungkinan, pemilihan tebakan berikutnya tidak dipengaruhi oleh informasi tersebut. Kata selanjutnya tetap diambil dari urutan semula tanpa mempertimbangkan seberapa informatif kata itu. Karena tidak ada proses evaluasi atau penilaian terhadap kualitas suatu kata sebagai tebakan, strategi ini tidak melibatkan unsur heuristik. Dengan demikian, pendekatan ini tetap dapat dikategorikan sebagai *brute force*.

3.2 Strategi Greedy

Berbeda dari pendekatan *brute force* yang berjalan secara kaku, strategi *greedy* berusaha memilih tebakan terbaik di setiap langkah berdasarkan informasi yang tersedia. Tujuannya adalah untuk memperkecil ruang pencarian dengan lebih cepat melalui pemilihan kata yang dianggap paling informatif. Pendekatan ini didasarkan pada prinsip “ambil yang terbaik saat ini”, dan berharap bahwa setiap keputusan lokal yang optimal akan membawa algoritma menuju solusi global yang efisien.

Langkah awal dalam strategi *greedy* dimulai dengan memilih kata yang memiliki huruf-huruf yang paling umum dan berbeda-beda. Kata-kata seperti “adieu” atau “aries” sering dianggap efektif karena mampu menguji banyak vokal atau huruf penting sekaligus (tidak ada huruf yang sama muncul lebih dari satu kali). Setelah umpan balik dari tebakan pertama diperoleh, strategi ini akan menyaring kata-kata yang tidak mungkin menjadi solusi, dengan tetap mempertimbangkan huruf-huruf yang sudah diketahui.

Yang membedakan strategi greedy adalah cara pemilihan tebakan berikutnya. Tidak seperti *brute force* yang berjalan linear, strategi ini menggunakan heuristik untuk menilai seberapa bermanfaat suatu kata dalam proses pencarian. Kata-kata yang mengandung huruf-huruf unik yang belum pernah dicoba, serta huruf dengan frekuensi tinggi di antara kandidat saat ini, akan diprioritaskan. Dengan demikian, tiap tebakan bertujuan tidak hanya untuk menebak jawaban, tetapi juga untuk memperluas informasi yang diperoleh dan menyempitkan ruang solusi secara efisien.

3.3 Eksperimen

Untuk mengetahui performa dari masing-masing strategi algoritmik yang telah dijelaskan sebelumnya, dilakukan serangkaian eksperimen terhadap lima kata target yang dipilih secara acak dari kamus bahasa Inggris lima huruf. Kata-kata tersebut adalah *candy*, *aaron*, *unrun*, *clave*, dan *trove*. Setiap kata diuji menggunakan dua strategi penyelesaian, yaitu *brute force* dan *greedy* heuristik. Eksperimen ini bertujuan untuk mencatat dan membandingkan dua metrik utama, yaitu jumlah langkah yang dibutuhkan untuk menemukan kata target, serta waktu eksekusi yang diperlukan oleh program.

Berikut ini adalah hasil eksperimen untuk masing-masing kata:

- Pada kata “candy”, algoritma *brute force* membutuhkan 5 langkah dengan waktu eksekusi sebesar 0.0313 detik. Sementara itu, strategi *greedy* juga menemukan solusi dalam 5 langkah, namun dengan waktu eksekusi yang sedikit lebih tinggi yaitu 0.0790 detik.

```
Tebakan 1: AALII
Tebakan 2: AARON
Tebakan 3: BANAK
Tebakan 4: CANCH
Tebakan 5: CANDY

[Brute Force + Filter] Jawaban ditemukan: CANDY dalam 5 langkah (0.0313 detik)
Tebakan 1: ARIES
Tebakan 2: ONLAY
Tebakan 3: TANDY
Tebakan 4: BANDY
Tebakan 5: CANDY

[Greedy] Jawaban ditemukan: CANDY dalam 5 langkah (0.0790 detik)
```

(Gambar 3.1: Hasil program untuk kata “candy” dengan brute force dan greedy)

- Untuk kata “aaron”, algoritma *brute force* hanya memerlukan 2 langkah dengan waktu eksekusi 0.0284 detik, karena kata tersebut muncul sangat awal dalam urutan kandidat. Di sisi lain, algoritma *greedy* membutuhkan 5 langkah dengan waktu 0.0694 detik.

```
Tebakan 1: AALII
Tebakan 2: AARON

[Brute Force + Filter] Jawaban ditemukan: AARON dalam 2 langkah (0.0284 detik)
Tebakan 1: ARIES
Tebakan 2: ATOUR
Tebakan 3: ACRON
Tebakan 4: APRON
Tebakan 5: AARON

[Greedy] Jawaban ditemukan: AARON dalam 5 langkah (0.0694 detik)
```

(Gambar 3.2: Hasil program untuk kata “aaron” dengan brute force dan greedy)

- Pada kata “unrun”, algoritma *brute force* menyelesaikan dalam 7 langkah dengan waktu 0.0392 detik, sedangkan *greedy* lebih efisien secara langkah, hanya membutuhkan 4 langkah walaupun waktu eksekusinya mencapai 0.0706 detik.

```
Tebakan 1: AALII
Tebakan 2: BEBED
Tebakan 3: CHOCK
Tebakan 4: FRUMP
Tebakan 5: GURRY
Tebakan 6: STRUT
Tebakan 7: UNRUN

[Brute Force + Filter] Jawaban ditemukan: UNRUN dalam 7 langkah (0.0392 detik)
Tebakan 1: ARIES
Tebakan 2: YOURN
Tebakan 3: UPRUN
Tebakan 4: UNRUN

[Greedy] Jawaban ditemukan: UNRUN dalam 4 langkah (0.0706 detik)
```

(Gambar 3.3: Hasil program untuk kata “unrun” dengan brute force dan greedy)

- Untuk kata “clave”, algoritma *brute force* memerlukan 6 langkah dengan waktu 0.0359 detik, sedangkan *greedy* berhasil menyelesaikan dengan 5 langkah dalam waktu 0.0803 detik.

```
Tebakan 1: AALII
Tebakan 2: BABEL
Tebakan 3: CHELA
Tebakan 4: CLAME
Tebakan 5: CLARE
Tebakan 6: CLAVE

[Brute Force + Filter] Jawaban ditemukan: CLAVE dalam 6 langkah (0.0359 detik)
Tebakan 1: ARIES
Tebakan 2: ENTAL
Tebakan 3: CABLE
Tebakan 4: CLAME
Tebakan 5: CLAVE

[Greedy] Jawaban ditemukan: CLAVE dalam 5 langkah (0.0803 detik)
```

(Gambar 3.4: Hasil program untuk kata “clave” dengan brute force dan greedy)

- Terakhir, kata “trove” diselesaikan oleh *brute force* dalam 9 langkah dengan waktu 0.0421 detik, sementara *greedy* lebih efisien dengan hanya 6 langkah dan waktu 0.0698 detik.

```
Tebakan 1: AALII
Tebakan 2: BEBED
Tebakan 3: CHECK
Tebakan 4: EGGER
Tebakan 5: FORME
Tebakan 6: OUTRE
Tebakan 7: PROTE
Tebakan 8: TRONE
Tebakan 9: TROVE

[Brute Force + Filter] Jawaban ditemukan: TROVE dalam 9 langkah (0.0421 detik)

Tebakan 1: ARIES
Tebakan 2: PROTE
Tebakan 3: TRODE
Tebakan 4: TROKE
Tebakan 5: TRONE
Tebakan 6: TROVE

[Greedy] Jawaban ditemukan: TROVE dalam 6 langkah (0.0698 detik)
```

(Gambar 3.5: Hasil program untuk kata “trove” dengan brute force dan greedy)

IV. ANALISIS DAN PERBANDINGAN

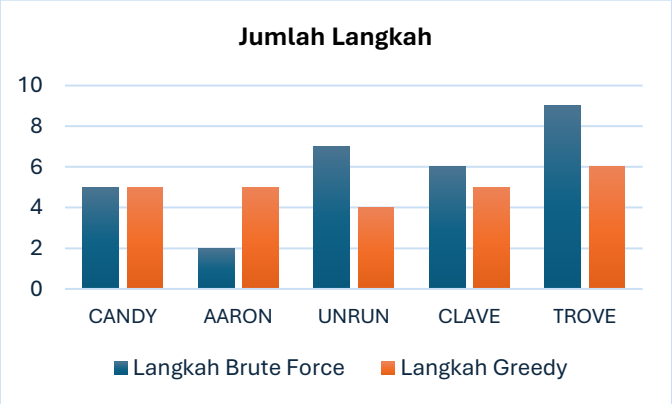
4.1 Hasil Eksperimen

Bagian ini menyajikan hasil dari penerapan dua strategi algoritma dalam menyelesaikan permainan Wordle. Fokus utama analisis adalah pada dua aspek, yaitu seberapa banyak langkah yang diperlukan oleh masing-masing algoritma untuk menemukan jawaban, serta berapa lama waktu eksekusi yang dibutuhkan. Hasil dari eksperimen tersebut ditampilkan pada Tabel 4.1 berikut.

Kata Target	Algoritma	Jumlah Langkah	Waktu Eksekusi (detik)
candy	Brute force	5	0.0313
	Greedy	5	0.0790
aaron	Brute force	2	0.0284
	Greedy	5	0.0694
unrun	Brute force	7	0.0392
	Greedy	4	0.0706
clave	Brute force	6	0.0359
	Greedy	5	0.0803
trove	Brute force	9	0.0421
	Greedy	6	0.0698

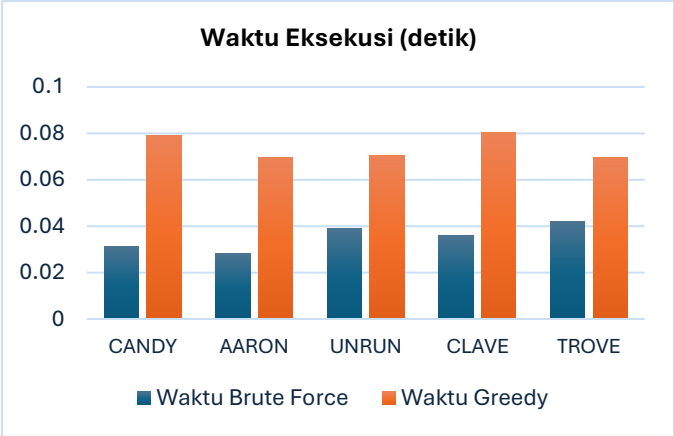
(Tabel 4.1 Hasil Eksperimen dengan 5 Kata Target)

Selain disajikan dalam bentuk tabel, hasil eksperimen juga divisualisasikan dalam bentuk grafik untuk mempermudah perbandingan performa kedua algoritma. Gambar 4.1 menunjukkan perbandingan jumlah langkah yang dibutuhkan oleh masing-masing strategi dalam menyelesaikan Wordle pada lima kata target. Terlihat bahwa strategi *greedy* cenderung membutuhkan lebih sedikit langkah dibandingkan *brute force* pada sebagian besar kasus.



(Gambar 4.1 Grafik Jumlah Langkah dari Kedua Algoritma)

Sementara itu, Gambar 4.2 menunjukkan perbandingan waktu eksekusi antara kedua algoritma. Meskipun *greedy* lebih efisien dalam jumlah langkah, waktu komputasinya relatif sedikit lebih tinggi dibandingkan *brute force* karena adanya proses tambahan dalam pemilihan kata berdasarkan heuristik. Perbedaan waktu ini berada dalam skala milidetik dan secara praktis tidak terlalu signifikan.



(Gambar 4.2 Grafik Waktu Eksekusi dari Kedua Algoritma)

4.2 Pembahasan Hasil

Dari hasil eksperimen, terlihat bahwa strategi *greedy* cenderung menghasilkan jumlah langkah yang lebih sedikit atau setara dibanding *brute force*. Pada tiga dari lima kata (unrun, clave, trove), strategi *greedy* membutuhkan langkah yang lebih sedikit. Pada kata "candy", keduanya membutuhkan lima langkah, dan hanya pada kata "aaron" algoritma *brute force* lebih unggul karena kata tersebut berada pada awal daftar kamus.

Namun, jika dilihat dari sisi waktu eksekusi, *brute force* justru lebih cepat pada seluruh kata. Hal ini terjadi karena algoritma *brute force* hanya menyaring kandidat berdasarkan umpan balik, tanpa melakukan proses evaluasi tambahan. Sementara itu, greedy membutuhkan waktu ekstra untuk menghitung skor heuristik dari setiap kandidat sebelum memilih tebakan terbaik. Proses ini meningkatkan beban komputasi, sehingga meskipun lebih efisien dari segi langkah, waktu eksekusinya menjadi sedikit lebih tinggi.

Perbedaan waktu ini memang relatif kecil karena hanya dalam skala milidetik dan tidak signifikan secara praktis untuk permainan Wordle biasa. Namun, ini tetap menunjukkan bahwa strategi yang lebih "pintar" belum tentu lebih cepat dalam eksekusi, terutama jika ruang solusinya masih relatif kecil.

4.3 Kelebihan dan Kekurangan Algoritma

Strategi *brute force* memiliki keunggulan utama dalam hal kesederhanaan dan kemudahan implementasi. Pendekatan ini tidak memerlukan pemikiran algoritmik yang rumit atau evaluasi tambahan, karena hanya menjalankan pencarian solusi secara sistematis dari daftar kata yang tersedia. Selama kata target berada dalam ruang pencarian, *brute force* dijamin akan menemukan solusi yang benar. Selain itu, waktu eksekusinya cenderung cepat karena proses per langkahnya ringan, tidak ada proses penilaian atau perhitungan skor yang membebani kinerja algoritma.

Namun, kelemahan dari *brute force* terletak pada efisiensi jumlah langkah yang dibutuhkan. Karena tidak mempertimbangkan informasi yang diperoleh dari tebakan sebelumnya dalam pemilihan kata selanjutnya, performanya sangat bergantung pada posisi kata target dalam daftar. Jika kata target berada di akhir, jumlah langkah bisa sangat besar. Dengan kata lain, strategi ini tidak beradaptasi terhadap situasi, sehingga tidak efisien untuk kasus dengan batas langkah terbatas seperti Wordle.

Dari sisi kompleksitas, strategi brute force dalam implementasi ini memiliki kompleksitas waktu sebesar $O(n^2)$ pada kasus terburuk, di mana n adalah jumlah total kata lima huruf dalam kamus. Hal ini disebabkan karena pada setiap langkah, program menyaring seluruh kandidat berdasarkan *feedback* yang didapat, dan proses ini diulang hingga jawaban ditemukan. Meskipun setiap langkahnya cukup ringan secara komputasi, jumlah total langkah dan proses *filtering* yang dilakukan berulang kali dapat menyebabkan waktu eksekusi membesar, terutama jika kata target berada di akhir daftar.

Sementara itu, strategi *greedy* heuristik menonjol dalam hal efisiensi langkah. Dengan memanfaatkan heuristik seperti frekuensi huruf dan keberagaman karakter, strategi ini mampu memilih kata yang memberikan informasi maksimal di setiap langkah. Hasilnya, jumlah tebakan yang diperlukan untuk menemukan kata target menjadi jauh lebih sedikit dan konsisten antar kasus. Pendekatan ini sangat cocok digunakan dalam permainan seperti Wordle, yang membatasi jumlah tebakan hingga enam kali.

Kekurangan dari strategi *greedy* adalah adanya beban komputasi tambahan di setiap langkah, karena algoritma harus menghitung skor atau nilai informatif dari setiap kata kandidat. Proses ini membuat waktu eksekusi lebih tinggi dibanding *brute force*, terutama jika jumlah kandidat cukup besar. Selain itu, karena keputusan dibuat berdasarkan kondisi lokal terbaik tanpa mempertimbangkan hasil jangka panjang, strategi ini tidak selalu menjamin solusi paling optimal di semua situasi.

Dalam implementasi yang digunakan, strategi greedy memiliki kompleksitas waktu sekitar $O(n \times k)$, dengan n adalah jumlah kandidat kata dan k adalah jumlah langkah tebakan. Pada setiap langkah, algoritma harus menghitung skor dari semua kata kandidat berdasarkan frekuensi huruf dan huruf-huruf unik yang belum digunakan. Proses ini memerlukan waktu $O(n)$, dan dilanjutkan dengan *filtering* kandidat juga dalam $O(n)$. Meskipun proses pada tiap langkah lebih mahal dibanding *brute force*, strategi *greedy* cenderung membutuhkan lebih sedikit langkah untuk menemukan solusi, sehingga secara keseluruhan tetap efisien.

4.4 Rangkuman Perbandingan

Secara keseluruhan, kedua strategi memiliki keunggulan dalam konteks yang berbeda. Strategi *brute force* unggul dalam kesederhanaan dan kecepatan eksekusi, namun kurang efisien dalam jumlah langkah dan sangat bergantung pada urutan data. Di sisi lain, strategi *greedy* mampu menyelesaikan permainan dengan langkah lebih sedikit secara konsisten, meskipun mengorbankan sedikit waktu eksekusi karena penggunaan heuristik.

Dalam konteks permainan seperti Wordle, di mana jumlah langkah sangat terbatas, strategi *greedy* lebih cocok karena mengutamakan efisiensi langkah. Namun, strategi *brute force* tetap menjadi pilihan yang valid jika dibutuhkan solusi yang pasti dan tidak terlalu memedulikan efisiensi di setiap langkah.

V. KESIMPULAN

Dalam makalah ini, telah dilakukan implementasi dan analisis terhadap dua strategi algoritmik untuk menyelesaikan permainan Wordle, yaitu *brute force* dan *greedy* heuristik. Keduanya diuji pada lima kata target berbeda, dengan pengukuran jumlah langkah dan waktu eksekusi sebagai indikator performa.

Hasil eksperimen menunjukkan bahwa strategi *greedy* secara umum lebih unggul dalam hal efisiensi jumlah langkah. Hal ini terjadi karena pemilihan kata dalam *greedy* didasarkan pada heuristik frekuensi dan keunikan huruf, yang memungkinkan eliminasi kandidat secara lebih cepat dan konsisten. Sementara itu, meskipun strategi *brute force* lebih sederhana dan cepat dalam waktu eksekusi, strategi ini menunjukkan performa yang sangat bergantung pada posisi kata target dalam daftar kamus.

Secara garis besar, kelebihan dari algoritma *brute force* adalah kesederhanaannya serta waktu komputasi yang lebih ringan karena tidak memerlukan proses penilaian heuristik. Namun, kekurangannya adalah tidak stabil dan dapat memerlukan banyak langkah jika kata target berada di akhir urutan. Di sisi lain, strategi *greedy* memiliki keunggulan dalam efisiensi langkah dan konsistensi performa, tetapi membutuhkan perhitungan tambahan yang membuat waktu eksekusinya sedikit lebih tinggi.

Berdasarkan hasil yang diperoleh, strategi *greedy* heuristik lebih cocok digunakan dalam sistem yang mengutamakan efisiensi jumlah tebakan dan kualitas solusi. Namun demikian, algoritma *brute force* masih relevan digunakan pada kasus-kasus tertentu yang memprioritaskan kemudahan implementasi dan kecepatan komputasi.

VI. LAMPIRAN

Tautan repository:
https://github.com/anellautari/Makalah-Stima

Tautan video youtube:
https://youtu.be/CXyxkVjKyOs

UCAPAN TERIMA KASIH

Penulis menyampaikan rasa syukur yang mendalam kepada Tuhan Yang Maha Esa atas segala karunia dan kemudahan yang telah diberikan selama proses penyusunan makalah ini. Penulis juga berterima kasih kepada keluarga tercinta atas dukungan moral dan doa yang tak henti-hentinya.

Ucapan terima kasih khusus ditujukan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah Strategi Algoritma, atas ilmu, bimbingan, serta arahannya yang sangat berharga selama perkuliahan. Penulis berharap makalah ini dapat memberikan manfaat dan menjadi referensi yang berguna bagi para pembaca.

REFERENSI

[1] *Algoritma Brute Force,"* Lawencon, [Online]. Tersedia: https://www.lawencon.com/algoritma-brute-force/. [Diakses: 24 Juni 2025].

[2] R. Munir, *"Algoritma Brute Force (Bagian 1),"* Institut Teknologi Bandung, [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf. [Diakses: 24 Juni 2025].

[3] *"Cara Kerja Algoritma Brute Force,"* Cloudeka, [Online]. Tersedia: https://www.cloudeka.id/id/berita/web-sec/cara-kerja-algoritma-brute-force/. [Diakses: 24 Juni 2025].

[4] R. Munir, *"Algoritma Greedy (Bagian 1),"* Institut Teknologi Bandung, [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf. [Diakses: 24 Juni 2025].

[5] *"Greedy Algorithms Explained,"* freeCodeCamp, [Online]. Tersedia: https://www.freecodecamp.org/news/greedy-algorithms/. [Diakses: 24 Juni 2025].

[6] *"Algoritma Greedy: Pengertian, Jenis, dan Contoh Program,"* FIKTI UMSU, [Online]. Tersedia: https://fikti.umsu.ac.id/algoritma-greedy-pengertian-jenis-dan-contoh-program/. [Diakses: 24 Juni 2025].

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, 24 Juni 2025

Anella Utari Gunadi - 13523078