Penerapan Algoritma Greedy dalam Menyelesaikan Permainan Quordle

Thirafi Najwan Kurniatama - 13520157
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): tnajwan26@gmail.com

***Abstract*—Wordle adalah permainan berbasis web yang memiliki cara bermain mirip seperti permainan *Mastermind*. Pemain memiliki maksimal enam percobaan untuk menebak kata dalam Bahasa Inggris yang terdiri dari lima huruf. Tiap tebakan yang tidak berhasil akan memberikan sebuah kombinasi respons tertentu yang memberikan petunjuk ke jawaban akhir. Quordle adalah salah satu permainan yang mirip dengan Wordle, namun harus menebak empat kata sekaligus dengan maksimal sembilan tebakan. Algoritma dengan pendekatan *greedy* akan digunakan demi meminimisasi kemungkinan kata untuk mendapatkan jawaban akhir berdasarkan respons yang diberikan.**

Keywords—quordle; wordle; greedy; permainan

I. PENDAHULUAN

Seiring perkembangan *browser engine* yang mampu me-*render* konten yang semakin kompleks. Aplikasi berbasis web juga turut berkembang menjadi semakin canggih, termasuk permainan berbasis web.

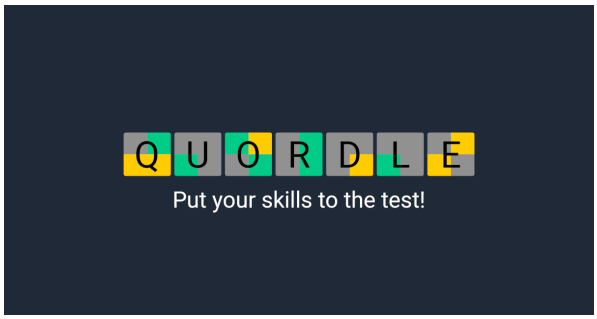
Permainan berbasis web memiliki keuntungan dari permainan konvensional yaitu *platform-independent* sehingga bisa meraih audiens yang lebih luas. Selama sistem punya browser yang mendukung, maka permainan bisa dijalankan. Selain itu, permainan berbasis web biasanya tidak memerlukan *system requirements* yang tinggi.

Gambar 1.1 Wordle Logo
Sumber: https://gamerant.com/wordle-of-the-day-answer-feb-9-humor-humour/

Wordle adalah salah satu permainan kata berbasis web yang dibuat oleh Josh Wardle dan dipublikasikan pada Oktober 2021 [1]. Setiap harinya, Permainan ini menyediakan satu kata berbeda yang terdiri lima huruf dan pemain harus menebaknya dengan maksimal enam kali percobaan.

Wordle mulai mendapatkan popularitas setelah menambahkan fitur yang bisa membagikan hasil harian pemain dalam bentuk *emoji*. Para pengguna Twitter sering membagikan hasil Wordle harian mereka sehingga banyak orang yang tertarik untuk mencoba dan membagikan hasilnya juga. Wordle mencapai tingkat popularitasnya pada bulan Januari 2022 [6] dan di bulan yang sama, The New York Times Company membeli Wordle dan memindahkan permainan tersebut ke situs mereka di https://www.nytimes.com/games/wordle.

Mengikuti kepopuleran Wordle, muncul banyak permainan serupa dari berbagai pengembang yang memvariasikan ide dari Wordle. Sebut saja Katla (https://katla.vercel.app) yang menjadikan kata Bahasa Indonesia sebagai jawabannya. Quordle (https://www.quordle.com) yang memperbanyak kotak tebakan menjadi empat, Octordle (https://octordle.com) yang memperbanyak kotak tebakan menjadi delapan, Nerdle (https://nerdlegame.com) yang menjadikan persamaan matematis sebagai jawaban, dan masih banyak lagi.



Gambar 2.2 Quordle Logo
Sumber: https://www.quordle.com/

Dengan permainan yang sistematis dan memiliki aturan yang cukup sederhana, tentunya penyelesaian permainan tersebut bisa dioptimisasi dengan berbagai pendekatan konsep algoritma. Pada makalah kali ini, akan digunakan algoritma dengan pendekatan konsep *min-max greedy* dan diimplementasikan menggunakan bahasa pemrograman Python untuk menyelesaikan permainan Quordle.

II. Dasar Teori

A. *Algoritma Greedy*

Algoritma *greedy* merupakan salah satu metode untuk menyelesaikan persoalan optimasi [2]. *Greedy* sering dipilih karena idenya tidak terlalu sulit sehingga implementasinya kemungkinan tidak terlalu kompleks. Metode ini cocok menjadi langkah awal untuk mencari metode optimisasi lain yang lebih optimal jika ada.

Persoalan optimasi adalah persoalan yang mencari solusi paling optimal dari kasus tertentu. Terdapat dua macam persoalan optimisasi, yaitu *maximization* dan *minimization*. Algoritma *greedy* akan menyelesaikan persoalan ini dengan mencari optimum lokal, yaitu langkah yang ditentukan sekarang hanya berdasarkan kondisi saat itu, kondisi sebelum dan setelahnya tidak diperhitungkan. Harapannya, pemilihan optimum lokal yang baik akan mengarahkan solusi ke optimum global.

Algoritma dengan pendekatan *greedy* tidak menjamin solusi paling optimal, namun biasanya cukup bagus dibanding *random guess* dan lebih cepat dibanding *brute force* atau *exhaustive search*. Maka dari itu, algoritma ini sangat cocok jika kita hanya mengincar hasil yang "*good enough"*.

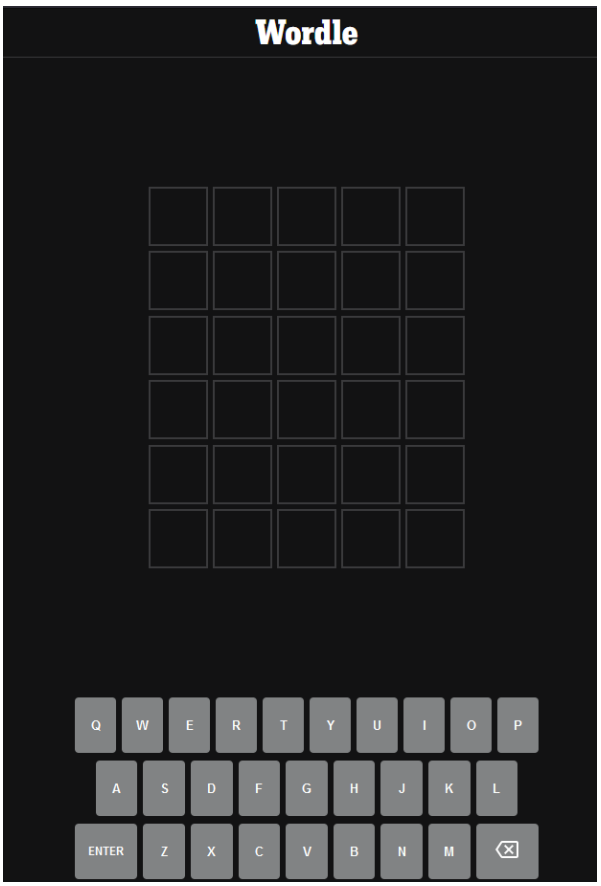
Pada algoritma *greedy*, umumnya terdapat enam buah elemen. yaitu:

1. Himpunan kandidat yang berisi kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi yang berisi kandidat yang sudah dipilih
3. Fungsi solusi yang menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi yang memilih kandidat berdasarkan strategy *greedy* tertentu
5. Fungsi kelayakan yang memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi
6. Fungsi obyektif yang memaksimumkan atau meminimumkan solusi

Beberapa persoalan yang optimal jika menggunakan pendekatan *greedy* adalah *fractional knapsack problem*, *activity selection problem, maximum sum subarray, huffman coding*, dan lain lain.

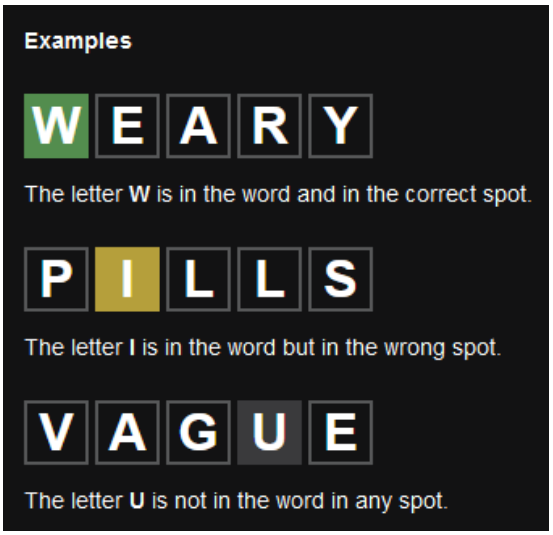
B. *Gameplay Wordle*

Cara bermain Wordle cukup simpel, pemain hanya perlu pergi ke laman https://www.nytimes.com/games/wordle pada browser yang sudah mendukung dan dapat mulai bermain. Kata jawaban dari Wordle akan diganti setiap harinya tepat pukul 00.00 sesuai dengan *timezone* dari perangkat pemain.



Gambar 2.1 Tampilan Utama Wordle
Sumber: Dokumen Pribadi

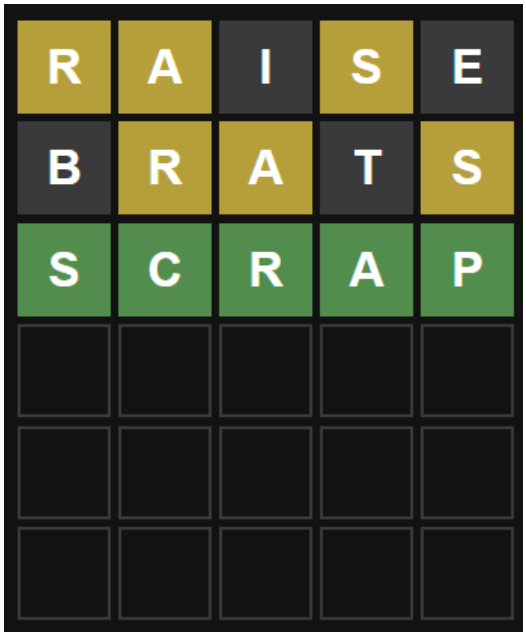
Wordle menyediakan *on-screen keyboard* untuk memasukkan tebakan pemain. Setelah setiap tebakan, Wordle akan memberikan tiga macam response untuk setiap huruf.



Gambar 2.2 Respons Wordle
Sumber: https://www.nytimes.com/games/wordle/

Abu-abu menandakan bahwa huruf tersebut tidak ada di jawaban akhir, Kuning menandakan huruf tersebut ada di jawaban akhir, namun berada di posisi yang salah. Hijau

menandakan huruf tersebut ada di jawaban akhir dan posisinya sudah benar. Jika tebakan pemain memiliki dua karakter yang sama dan jawaban akhir hanya memiliki satu, maka satu akan berwarna kuning atau hijau, dan sisanya berwarna abu-abu.

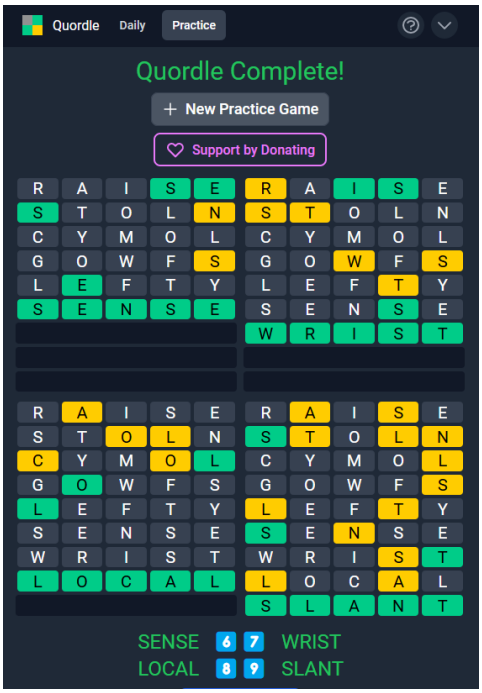


Gambar 2.3 Simulasi Permainan Wordle
Sumber: Dokumen Pribadi

Setiap hari, pemain akan diberikan enam kesempatan untuk menebak kata akhir pada hari itu. Jika berhasil, akan muncul *pop-up* yang menyediakan tombol untuk membagikan hasil Wordle pemain ke platform manapun yang diinginkan.

Wordle juga menyediakan mode sulit (*Hard Mode*) yang mengharuskan petunjuk dari respons sebelumnya digunakan pada tebakan berikutnya. Misal, jika R berwarna kuning pada tebakan pertama, maka tebakan kedua dan seterusnya harus mengandung huruf R.

C. *Gameplay Quordle*



Gambar 2.4 Simulasi Permainan Quordle
Sumber: Dokumen Pribadi

Quordle memiliki cara bermain yang identik dengan Wordle, namun dengan tingkat kesulitan yang lebih tinggi. Pada Quordle, tersedia empat boks yang harus ditebak secara simultan dalam maksimum sembilan kali percobaan.

Respons yang diberikan Quordle adalah secara simultan untuk keempat boks tersebut, sehingga jika tidak dapat menebak *in advance* keempat jawaban tersebut pada tebakan kelima (karena empat turn terakhir untuk memasukkan jawaban ke masing-masing boks). Maka besar kemungkinan tidak dapat menebak keseluruhan jawaban.

Quordle menyediakan mode Daily dan Practice. Mode Daily diubah jawabannya tiap hari sama seperti Wordle, sedangkan mode Practice dapat dicoba dan diulang berkali kali.

III. Pembahasan

Pada bab ini, akan dijelaskan mengenai strategi *greedy* yang akan digunakan untuk menentukan tebakan terbaik di setiap *turn* berdasarkan petunjuk sebelumnya. Akan dibahas juga asumsi-asumsi apa saja yang digunakan serta sedikit implementasinya dalam Bahasa Pemrograman Python

A. *Asumsi*

Pada algoritma ini, diasumsikan bahwa tebakan pertama (*word opener*) terbaik untuk algoritma ini adalah anagram dari kata RAISE [4]. Maka dari itu, RAISE akan selalu menjadi tebakan pertama dari setiap permainan. Algoritma ini juga mengasumsikan bahwa seluruh kemungkinan kata jawaban dari Quordle sama dengan Wordle karena penulis tidak dapat menemukan sumber untuk *word pool* dari Quordle.

B. Algoritma Penyelesaian

Algoritma ini akan menggunakan pendekatan *min-max greedy*. Ide utamanya adalah dengan meminimalisir jumlah rata-rata maksimum *search space* dari keempat kotak pada setiap tebakan. Nantinya, algoritma ini akan melakukan proses eliminasi dari *word pool* Quordle yang berjumlah 12.947 kata [3] berdasarkan petunjuk dan prinsip *greedy*.

Elemen *greedy* dari algoritma ini adalah sebagai berikut:

1. Himpunan Kandidat: seluruh kata yang mungkin pada Quordle.

2. Himpunan Solusi: kata yang memenuhi petunjuk yang sudah diberikan pada tebakan sebelumnya

3. Fungsi Solusi: fungsi yang menentukan kata mana saja yang sesuai dengan petunjuk (*hint*)

4. Fungsi Seleksi: fungsi yang memilih kata yang memberikan rata-rata maksimal search space terkecil

5. Fungsi Kelayakan: fungsi yang memeriksa apakah kata tersebut sudah sesuai dengan peraturan Quordle

6. Fungsi Objekti: menebak seluruh jawaban dengan tebakan seminimal mungkin

Secara singkat, cara kerja algoritma ini adalah sebagai berikut:

1. Program akan mulai menebak dengan kata *default* "RAISE" terlebih dahulu.

2. Pengguna akan memasukkan respons dari Quordle berdasarkan tebakan tadi untuk masing-masing kotak pada program sesuai dengan urutan yang diminta. 0 untuk kotak berwarna abu-abu, 1 untuk kotak berwarna kuning, dan 2 untuk kotak berwarna hijau, dipisah dengan koma tanpa spasi. Misal: jika kotak berwarna HIJAU ABU-ABU HIJAU KUNING ABU-ABU, maka pemain akan menuliskan "2,0,2,1,0".

3. Program akan mengurangi himpunan solusi berdasarkan respons tersebut dan akan dicari dari himpunan solusi, anggota mana yang memberikan rata-rata maksimum himpunan solusi yang paling minimum.

4. Anggota himpunan pada langkah 3 akan dijadikan tebakan pada *turn* tersebut.

5. Ulangi langkah 2 hingga 5 terus sampai mendapatkan seluruh jawaban untuk setiap kotak

Dengan meminimalisir *search space* berdasarkan petunjuk, diharapkan terjadi konvergensi yang cukup baik sehingga proses eliminasi kandidat dapat dilakukan dengan cepat dan efisien.

Dari langkah-langkah yang disebutkan di atas, struktur program akan dibagi menjadi tiga bagian yaitu: program utama; fungsi response untuk mensimulasikan petunjuk yang akan diberikan oleh Quordle; dan fungsi predict yang akan memberikan kata tebakan terbaik berdasarkan himpunan solusi dan pendekatan strategi *greedy*.

Berikut adalah *pseudocode* untuk fungsi response dan predict:

```
function response(guess: string, answer:
string) -> tuple
{ simulasi respons oleh Quordle }

KAMUS LOKAL:
   resp: list
ALGORITMA:
   resp <- {0,0,0,0,0}
   for i = 0 to 4
      if guess[i] = answer[i] then
         resp[i] <- 2
         answer[i] <- '-'
   for i = 0 to 4
      if guess[i] in answer and resp[i] = 0
then
         resp[i] <- 1
         answer[answer.index(guess[i])] <- '-'
   return tuple(resp)
```

```
function predict(candidate: list, sol1, sol2,
sol3, sol4: list) -> string, dict, dict, dict,
dict
{ penentuan tebakan terbaik beserta search
space nya }

KAMUS LOKAL:
   min_avg: float
   guess: string
   posans: string
   avg_max: float
   bestguess: string
   t1: dict
   … {untuk seluruh kotak}
   t4: dict
   tsol1: dict
   …
   tsol4: dict
   maxsol1: int
   …
   maxsol4: int

ALGORITMA:
   min_avg <- 100000.0
   for each guess in candidate
      for each posans in sol1
         resp = response(guess, posans)
         if resp not in tsol1 then
            tsol1[resp] <- posans
         else
            tsol[resp] <- tsol[resp] + posans
      … {untuk keseluruhan kotak}
      maxsol1 <- jumlah search space tertinggi
tsol1
      …
      maxsol4 <- jumlah search space tertinggi
tsol4
      avg_max <- avg(maxsol1, …, maxsol4) {
rata-rata search space }
      if avg_max < min_avg then
         min_avg <- avg_max
         bestguess <- guess
         t1 <- tsol1 { tabel search space }
         …
         t4 <- tsol4
   return bestguess, t1, t2, t3, t4
```

Lalu, berikut adalah *pseudocode* untuk program utama:

```
PROGRAM UTAMA:

KAMUS:
   wl: list
   turn: integer
   alrg: list
   doneg: list
   tans: list
   candidate: list
   sol1: list
   …
   sol4: list
   resp1: tuple
   …
   resp4: tuple
   inp: string
   t1: dict
   …
   t4: dict
   bestguess: string
   first_word: string

ALGORITMA:
   wl <- open('wordlist.txt')
   first_word <- 'RAISE'
   for turn = 0 to 8
      output("Turn " + turn)
      if turn = 0 then
         alrg <- {false, false, false, false}
         doneg <- {false, false, false, false}
         candidate <- first_word
         sol1 <- wl
         …
         sol4 <- wl
      else
         candidate <- sol1 + sol2 + sol3 + sol4
      for i = 0 to 3
         if doneg[i] and tans[i] in candidate
then
            candidate.remove(tans[i])
      bestguess, t1, …, t4 <-
predict(candidate, sol1, …, sol4)
      output("BEST GUESS:", bestguess)
      inp = input("Enter Response for Box 1: ")
      if inp is empty then
         resp1 <- (2,2,2,2,2)
      else
         resp1 <- inp
      sol1 <- empty if alrg[0] else t1[resp1]
      if len(sol1) = 1 and not alrg[0] then
         alrg[0] <- true
         tans[0] <- sol1[0]
      … {lakukan untuk seluruh kotak}
      if any(alrg) then
         for i = 0 to 3
            if alrg[i] then
               output("Box" + int(i+1) + "
answer is" + tempans[i])
```

C. Implementasi dalam Python

Pada bagian ini, akan diberikan sedikit *snippet* implementasi algoritma tersebut dalam bahasa pemrograman Python. Program ini nantinya akan digunakan untuk simulasi pada eksperimen di bab berikutnya. Kode lengkap beserta instruksi untuk menjalankan program terdapat pada laman https://github.com/reverseon/quordle-solver.

interactive.py

```
from solver import predict

FILEOPENER = open("wordlist.txt","r")
WORD_LIST = FILEOPENER.readlines()
FIRST_WORD = "raise"

print("Quordle Solver")
print("--------------")
print("Instruction: ")
print("Use 0 for grey response, 1 for yellow response, and 2 for green response
separated with a comma. If the box is already guessed, just click enter.")
print("Ex: GREEN GREEN GREY YELLOW GREY")
print("In: 2,2,0,1,0")

for turn in range(9):
    print(f"Turn {turn+1}")
    if turn == 0:
        alrguessed = [False for _ in range(4)]
        doneguess = [False for _ in range(4)]
        tempans = ["" for _ in range(4)]
        candidate = [FIRST_WORD]
        sol1 = WORD_LIST[:]
        sol2 = WORD_LIST[:]
        sol3 = WORD_LIST[:]
        sol4 = WORD_LIST[:]
    else:
        candidate = list(dict.fromkeys(sol1 + sol2 + sol3 + sol4))
    for i in range(4):
        if doneguess[i] and tempans[i] in candidate:
            candidate.remove(tempans[i])
    bestguess, table1, table2, table3, table4 = predict(candidate, sol1, sol2,
sol3, sol4)
    print(f"BEST GUESS: {bestguess.upper()}")
    inpbuf = input("Enter response for Box 1: ")
    resp1 = (2,2,2,2,2) if inpbuf == "" else tuple([int(x) for x in
inpbuf.split(",")])
    inpbuf = input("Enter response for Box 2: ")
    resp2 = (2,2,2,2,2) if inpbuf == "" else tuple([int(x) for x in
inpbuf.split(",")])
    inpbuf = input("Enter response for Box 3: ")
    resp3 = (2,2,2,2,2) if inpbuf == "" else tuple([int(x) for x in
inpbuf.split(",")])
    inpbuf = input("Enter response for Box 4: ")
    resp4 = (2,2,2,2,2) if inpbuf == "" else tuple([int(x) for x in
inpbuf.split(",")])
    sol1 = [] if doneguess[0] else table1[resp1]
    sol2 = [] if doneguess[1] else table2[resp2]
    sol3 = [] if doneguess[2] else table3[resp3]
    sol4 = [] if doneguess[3] else table4[resp4]
    if len(sol1) == 1 and alrguessed[0] == False:
        alrguessed[0] = True
        tempans[0] = sol1[0]
    if len(sol2) == 1 and alrguessed[1] == False:
        alrguessed[1] = True
        tempans[1] = sol2[0]
    if len(sol3) == 1 and alrguessed[2] == False:
        alrguessed[2] = True
        tempans[2] = sol3[0]
    if len(sol4) == 1 and alrguessed[3] == False:
        alrguessed[3] = True
        tempans[3] = sol4[0]
    if any(alrguessed):
        for i in range(4):
            if alrguessed[i]:
                print(f"Box {i+1} answer is {tempans[i].upper()}")
    if bestguess in tempans:
        doneguess[tempans.index(bestguess)] = True
    if all(doneguess):
        print("All boxes are answered, exiting...")
        exit(0)
print("Failed to guess after 9 turns, exiting...")
```

Gambar 3.1 *Snippet* Program Utama
Sumber: Dokumen Pribadi

predict.py

```
def predict(candidate, solution1, solution2, solution3, solution4):
    min_avg = 1e5 # HIGH ENOUGH
    table1 = {}
    table2 = {}
    table3 = {}
    table4 = {}
    bestguess = ""
    for guess in candidate:
        guess = guess.strip()
        tsol1 = {}
        tsol2 = {}
        tsol3 = {}
        tsol4 = {}
        for posans in solution1:
            posans = posans.strip()
            resp = response(guess, posans)
            if resp not in tsol1:
                tsol1[resp] = [posans]
            else:
                tsol1[resp].append(posans)
        for posans in solution2:
            posans = posans.strip()
            resp = response(guess, posans)
            if resp not in tsol2:
                tsol2[resp] = [posans]
            else:
                tsol2[resp].append(posans)
        for posans in solution3:
            posans = posans.strip()
            resp = response(guess, posans)
            if resp not in tsol3:
                tsol3[resp] = [posans]
            else:
                tsol3[resp].append(posans)
        for posans in solution4:
            posans = posans.strip()
            resp = response(guess, posans)
            if resp not in tsol4:
                tsol4[resp] = [posans]
            else:
                tsol4[resp].append(posans)
        maxsol1 = max([len(v) for v in tsol1.values()] + [0])
        maxsol2 = max([len(v) for v in tsol2.values()] + [0])
        maxsol3 = max([len(v) for v in tsol3.values()] + [0])
        maxsol4 = max([len(v) for v in tsol4.values()] + [0])
        avg_max = (maxsol1 + maxsol2 + maxsol3 + maxsol4) / 4
        if avg_max < min_avg:
            min_avg = avg_max
            bestguess = guess
            table1 = tsol1
            table2 = tsol2
            table3 = tsol3
            table4 = tsol4
    return bestguess, table1, table2, table3, table4
```

Gambar 3.2 Implementasi fungsi predict
Sumber: Dokumen Pribadi

response.py

```python
from functools import lru_cache

@lru_cache(maxsize=None) # CACHING
def response(guess, answer):
    lguess = list(guess)
    lanswer = list(answer)
    resp = [0 for _ in range(5)]
    for i in range(5):
        if lguess[i] == lanswer[i]:
            resp[i] = 2
            lanswer[i] = '-'
    for i in range(5):
        if lguess[i] in lanswer and resp[i] == 0:
            resp[i] = 1
            lanswer[lanswer.index(lguess[i])] = '-'
    return tuple(resp)
```

Gambar 3.3 Implementasi fungsi response
Sumber: Dokumen Pribadi

Pada fungsi response tersebut, akan digunakan decorator Python berupa lru_cache() yang membuat hasil komputasi yang sering digunakan disimpan. Ini bertujuan agar hanya tebakan pertama saja yang memakan waktu lama, dan pencarian tebakan berikutnya hanya tinggal mengambil hasil yang sudah dikomputasi tadi.

IV. Eksperimen

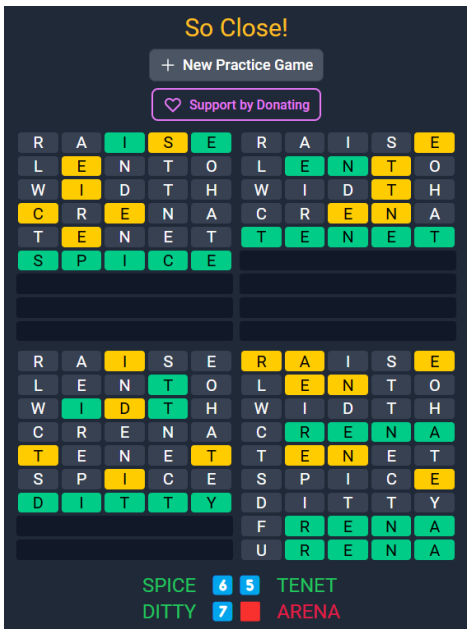
Pada bagian ini, akan diperlihatkan program yang diimplementasikan pada bagian III.C untuk menyelesaikan tiga permainan Quordle di practice mode. Tebakan pertama selalu diawali dengan kata “RAISE”.

A. Permainan Pertama

Tebakan Program:

1. RAISE
2. LENTO
3. WIDTH
4. CRENA
5. TENET
6. SPICE
7. DITTY
8. FRENA
9. URENA

Cuplikan permainan:



Gambar 4.1 Cuplikan permainan pertama
Sumber: Dokumen Pribadi

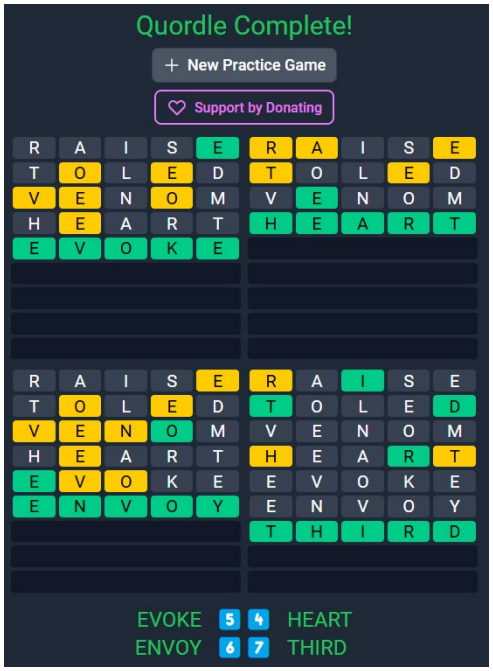
Pada permainan pertama, terlihat program berhasil untuk menebak ketiga jawaban dalam sembilan tebakan, namun gagal untuk menebak jawaban di kotak terakhir. Terlihat bahwa program terjebak pada pemilihan kata yang berakhiran RENA. FRENA dan URENA dipilih terlebih dahulu murni karena kata tersebut berada lebih awal di urutan kata daripada ARENA.

B. Permainan Kedua

Tebakan program:

1. RAISE
2. TOLED
3. VENOM
4. HEART
5. EVOKE
6. ENVOY
7. THIRD

Cuplikan permainan:



Gambar 4.2 Cuplikan permainan kedua
Sumber: Dokumen Pribadi

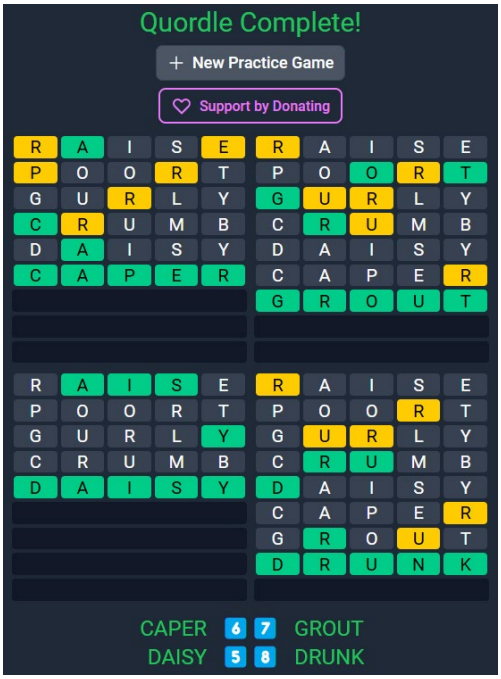
Pada permainan kedua, program berhasil menebak seluruh jawaban hanya dalam tujuh tebakan saja.

C. Permainan Ketiga

Tebakan program:

1. RAISE
2. POORT
3. GURLY
4. CRUMB
5. DAISY
6. CAPER
7. GROUT
8. DRUNK

Cuplikan permainan:



Gambar 4.3 Cuplikan permainan ketiga
Sumber: Dokumen Pribadi

Pada permainan ketiga, program berhasil menebak seluruh jawaban dalam delapan tebakan.

V. KESIMPULAN

Dari percobaan yang dilakukan pada bab IV, terlihat bahwa algoritma ini masih belum optimal pada beberapa kasus. Hal ini dikarenakan algoritma ini hanya mencoba untuk meminimasi *search space* dari kemungkinan jawaban saja. Jika ada tebakan yang memberikan konvergensi *search space* yang setara, maka akan dipilih tebakan yang pertama kali muncul di urutan kata seperti yang terlihat pada kasus IV.A. Algoritma ini ini bisa diperbaiki kedepannya dengan menambahkan elemen *Information Theory* pada proses pemilihan tebakan di strategi *greedy* seperti informasi persebaran huruf dari *word pool* hingga *common occurences* kata tersebut pada Bahasa Inggris.

TAUTAN VIDEO YOUTUBE

https://www.youtube.com/watch?v=oxA7MovLTQ4

UCAPAN TERIMA KASIH

Saya mengucapkan Terima Kasih kepada para dosen strategi algoritma yang telah membimbing saya dengan semangat pada mata kuliah ini sehingga saya bisa menyelesaikan makalah ini. Saya juga mengucapkan terima kasih kepada teman-teman saya yang selalu mendukung saya dan membantu saya sehingga makalah ini dapat terselesaikan dengan baik.

DAFTAR PUSTAKA

[1] D. Victor, "Wordle is a Love story," *The New York Times*, 03-Jan-2022. [Online]. Available: https://www.nytimes.com/2022/01/03/technology/wordle-word-game-creator.html. [Accessed: 21-May-2022].

[2] "Bahan Kuliah IF2211 strategi algoritma algoritma greedy." [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf. [Accessed: 21-May-2022].

[3] Tabatkins, "Tabatkins/wordle-list: List of possible wordle words," *GitHub*. [Online]. Available: https://github.com/tabatkins/wordle-list. [Accessed: 21-May-2022].

[4] B. Steenhuisen, "Raising the WORDLE first-guess bar!," *Medium*, 31-Dec-2021. [Online]. Available: https://noxville.medium.com/raising-the-wordle-first-guess-bar-8770e0ef47d4. [Accessed: 21-May-2022].

[5] I. Frizler, "The science behind Wordle," *Medium*, 12-Jan-2022. [Online]. Available: https://ido-frizler.medium.com/the-science-behind-wordle-67c8112ed0d1. [Accessed: 21-May-2022].

[6] @WordleStats, "#Wordle 226 2022-01-31," *Twitter*, 01-Feb-2022. [Online]. Available: https://twitter.com/WordleStats/status/1488557815224365059. [Accessed: 21-May-2022].

[7] Y. Gafni, "Automatic Wordle Solving," *Medium*, 29-Jan-2022. [Online]. Available: https://towardsdatascience.com/automatic-wordle-solving-a305954b746e. [Accessed: 21-May-2022].

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

Thirafi Najwan Kurniatama
13520157