

Penerapan Algoritma Brute Force untuk Menyelesaikan Permainan Nerdle

Dimas Shidqi Parikesit - 13520087
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520087@std.stei.itb.ac.id

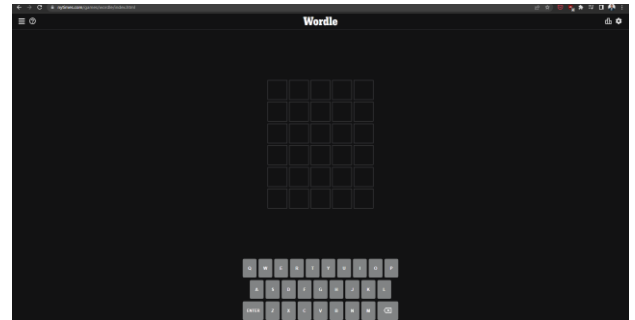
Abstract—Nerdle merupakan permainan tebak persamaan matematika dengan maksimum enam kali percobaan dengan pemberian *feedback* pada tiap percobaan. Strategi *brute force* ditambah dengan pendekatan heuristik adalah salah satu metode yang dapat digunakan. Karena tiap percobaan diberikan *feedback*, ruang kemungkinan solusi akan mengecil sehingga tidak akan membutuhkan waktu lama pada tiap kalkulasi meskipun menggunakan algoritma *brute force*.

Keywords—*brute force, heuristik, nerdle, combination*

I. PENDAHULUAN

Nerdle merupakan permainan yang berasal dari adaptasi permainan lain bernama Wordle. Wordle merupakan permainan yang dibuat oleh Josh Wardle, seorang *software engineer* yang berasal dari Welsh pada tahun 2021. Awalnya Wordle dirilis oleh Wardle pada situs pribadi miliknya secara gratis dan tanpa iklan. Wordle mulai terkenal setelah Wardle menambahkan fitur yang memperbolehkan pemain untuk menyalin hasil mereka sebagai emoji. Setelah Wordle menjadi sangat populer, perusahaan berita The New York Times membeli permainan ini dan memindahkan Wordle ke situs milik mereka. Meskipun begitu, Wordle tetap menjadi permainan yang gratis untuk dimainkan semua orang.

Permainan Wordle merupakan permainan dimana pemain berusaha menebak sebuah kata yang memiliki lima huruf dalam maksimal enam kali mencoba. Pada Wordle versi resmi milik The New York Times, kata untuk tiap percobaan harus merupakan kata berbahasa Inggris yang valid. Pada tiap percobaan menebak, pemain akan mendapatkan *feedback* untuk tiap huruf pada kata, dengan tiga jenis *feedback*, yaitu apabila huruf ada di kata dan berada pada posisi yang tepat, huruf ada di kata tetapi tidak pada posisi yang tepat, dan huruf tidak ada di kata. *Feedback* ini akan menjadi landasan bagi pemain untuk menebak selanjutnya.



Sumber:

<https://www.nytimes.com/games/wordle/index.html>

Permainan Nerdle sebagai adaptasi Wordle memiliki kesamaan pada jumlah percobaan yang diberikan kepada pemain, yaitu enam, dan tiga jenis *feedback* yang diberikan. Akan tetapi, Nerdle memiliki perbedaan dasar yaitu dalam Nerdle, pemain harus menebak sebuah persamaan matematika yang terdiri dari delapan karakter. Dalam persamaan tersebut, karakter yang tergolong legal adalah angka, operator penjumlahan, pengurangan, perkalian, pembagian, dan sama dengan. Tidak terdapat batasan berapa banyak operator yang diperbolehkan. Dalam memasukkan percobaan, persamaan yang dimasukkan haruslah persamaan yang valid secara matematis.



Sumber: <https://nerdlegame.org/>

II. LANDASAN TEORI

A. Persamaan Matematika

Persamaan matematika adalah sebuah ekspresi matematika yang mengandung sebuah simbol sama dengan. Pada sebuah persamaan, sisi kiri dan sisi kanan boleh memiliki satu atau lebih operator. Agar sebuah persamaan matematika disebut valid, maka sisi kiri dan sisi kanan harus memiliki nilai yang sama. Selain itu, tiap sisi nya juga harus valid dengan ketentuan setiap operator harus dikelilingi oleh dua buah operand. Berikut contoh persamaan matematika yang valid

$$1 + 2 = 3$$

B. Kombinasi

Kombinasi merupakan salah satu bagian dari kombinatorika, dimana kombinatorika itu sendiri adalah bidang matematika yang berkaitan dengan permasalahan pemilihan, dan pengaturan dalam sebuah sistem diskrit. Kombinasi digunakan untuk jenis persoalan dimana ingin ditentukan banyaknya cara untuk memilih objek berjumlah k dari sekumpulan objek berjumlah n tanpa memikirkan urutan pemilihan. Sementara itu, ada dua jenis kombinasi, yaitu kombinasi tanpa pengulangan dan kombinasi dengan pengulangan. Pada kombinasi dengan pengulangan, sebuah objek dapat dipilih lebih dari sekali pada himpunan solusi nya. Pada kombinasi tanpa pengulangan, sebuah objek hanya dapat dipilih sekali. Kombinasi tanpa pengulangan memiliki rumus sebagai berikut

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Sedangkan kombinasi dengan pengulangan memiliki rumus yang merupakan modifikasi dari rumus kombinasi tanpa pengulangan

$$\binom{n+k-1}{k}$$

Pada makalah ini, kombinasi akan digunakan untuk membangkitkan himpunan solusi yang mungkin.

C. Algoritma Brute Force

Algoritma brute force merupakan algoritma yang paling sederhana dan paling mudah diimplementasikan karena algoritma ini menggunakan pendekatan yang mencari semua kemungkinan yang ada. Akan tetapi, pendekatan tersebut memiliki kekurangan yaitu kompleksitas waktu yang buruk.

Metode ini dipilih karena beberapa alasan, yaitu adalah karena mempermudah pembangkitan solusi dan karena sederhana dalam implementasi.

D. Teknik Heuristik

Kelemahan dari algoritma *brute force* adalah waktu yang dibutuhkan untuk mengeksplorasi semua kemungkinan solusi. Agar program dapat menghasilkan hanya satu solusi pada tiap percobaan maka diperlukan fungsi pemilihan solusi. Fungsi pemilihan solusi ini didasari pada pendekatan heuristik.

Fungsi heuristik itu sendiri memiliki sifat dapat mempercepat pencarian solusi tanpa perlu bukti bahwa fungsi tersebut benar secara matematis.

III. IMPLEMENTASI DAN ANALISIS PENGUJIAN

Algoritma *brute force* diterapkan untuk mencari semua solusi yang mungkin dengan mencari semua kemungkinan solusi menggunakan kombinasi kemudian mengetes solusi yang dihasilkan tersebut pada fungsi yang mengecek apakah hasil kombinasi tersebut valid. Kemudian dari seluruh kombinasi yang valid akan dimasukkan kedalam fungsi heuristik untuk menghasilkan rekomendasi solusi.

Dalam menyelesaikan puzzle Nerdle, program akan memberikan rekomendasi solusi kemudian menerima input mengenai *feedback* yang diberikan oleh Nerdle. Kemudian program akan memberikan rekomendasi baru sampai hasil ditemukan.

A. Data Struktur

Terdapat x data struktur utama yang digunakan. Yang pertama adalah larik yang menyimpan karakter final. Kemudian terdapat larik lain untuk menyimpan semua percobaan yang pernah dilakukan. Kemudian ada sebuah larik yang berisi *sets* yang menyimpan karakter apa saja yang pernah digunakan pada posisi tertentu. Digunakan juga larik yang menyimpan karakter yang masih mungkin digunakan, yaitu karakter yang belum ditentukan tidak ada oleh Nerdle. Terakhir adalah larik yang berisi karakter yang harus ada, yaitu karakter yang ditentukan oleh Nerdle ada, tetapi posisi nya belum ditemukan.

```
6 final = ['' for i in range(8)]
7 history = []
8 possible = ['0','1','2','3','4','5','6','7','8','9','+','-','*','/','=','']
9 tried = [set() for i in range(8)]
10 must = []
```

Sumber: Penulis

B. Input valid

Persamaan matematika yang valid adalah persamaan yang sisi kiri dan sisi kanan menghasilkan hasil yang sama. Hal tersebut memungkinkan adanya operand di sisi kanan. Pada implementasi ini, tidak diperbolehkan kombinasi yang menyebabkan munculnya operand di sebelah kanan untuk mengurangi kemungkinan dua solusi sama yang akibat sifat komutatif matematika.

Sementara itu, input yang memungkinkan adalah angka, yaitu 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 dan empat operator dasar yaitu kali ($\times$), bagi ($\div$), tambah (+), dan kurang ($-$).

Selain itu, terdapat beberapa aturan yang harus dicek untuk memperkecil kemungkinan solusi, yaitu satu operator pada tiap urutan. Hal ini berarti operator yang mungkin valid pada bahasa pemrograman seperti ++, --, **, // akan dianggap tidak valid. Tetapi sesbelum pengecekan tersebut, akan dilakukan sanitasi input dengan menggunakan regex yang hanya memperbolehkan angka dan operator.

```

16 def checkMathFromString(sentence):
17     try:
18         sentence = sentence.split()[0]
19         if len(sentence) != 8:
20             return False
21
22         if len(re.findall(r"([0-9])|([+])|([-])|([/])|([-+])", sentence)) != len(sentence):
23             return False
24
25         if sentence[0] != '+' or sentence[0] != '-' or sentence[0] != '*' or sentence[0] != '/' or sentence[0] != '=':
26             return False
27
28         for dop in doubleOperator:
29             if dop in sentence:
30                 return False
31
32         idx = sentence.index('=')
33         for bop in basicOperator:
34             if bop in sentence[idx:]:
35                 return False
36
37         sentence = sentence[idx:] + '=' + sentence[idx:]
38         # return eval(sentence)
39         return simple_eval(sentence)

```

Sumber: Penulis

C. Feedback

Dalam menyelesaikan puzzle, diperlukan *feedback* setelah tiap percobaan untuk mengurangi kemungkinan solusi yang ada. *Feedback* ini dimasukkan kedalam program untuk ditambahkan sebagai pertimbangan kemungkinan. *Feedback* dimasukkan dengan aturan sebagai berikut. Karena terdapat delapan karakter pada sebuah persamaan yang valid, maka diperlukan masukan berupa delapan karakter *feedback*, dengan karakter 0 yang menandakan huruf tersebut tidak ada di solusi, 1 yang menandakan huruf tersebut terdapat di solusi tetapi pada posisi yang lain, 2 yang menandakan huruf tersebut ada di solusi pada posisi tersebut.

D. Teknik Heuristik

Teknik heuristik digunakan untuk mengeliminasi kemungkinan solusi yang ada hingga menyelesaikan satu solusi untuk direkomendasikan.

Pada program ini, digunakan pendekatan heuristik dengan cara memilih solusi yang memiliki karakter unik paling banyak atau paling bervariasi yang terakhir dibangkitkan.

```

77 def uniquecount(arr):
78     opr = set()
79     opd = set()
80     for i in range(len(arr)):
81         if arr[i] == '+' or arr[i] == '-' or arr[i] == '*' or arr[i] == '/' or arr[i] == '=':
82             opr.add(arr[i])
83         else:
84             opd.add(arr[i])
85     return len(opr), len(opd)
86

```

Sumber: Penulis

E. Alur program

1) Rekomendasi pertama

Pada percobaan pertama belum didapatkan informasi apapun yang dapat mengurangi kemungkinan solusi, sehingga banyaknya kombinasi adalah

$$14^8 = 1475789056$$

Tentunya kombinasi tersebut tidak semuanya merupakan kombinasi yang valid, akan tetapi dalam membuat kemungkinan solusi, 1475789056 kemungkinan tersebut harus dicek apakah valid atau tidak yang akan mengakibatkan waktu yang sangat lama.

Oleh karena itu, untuk percobaan pertama ditentukan rekomendasi persamaan

$$7 \times 9 - 3 = 60$$

Rekomendasi: 7*9-3=60
Masukkan hasil:

Sumber: Penulis

2) Rekomendasi kedua

Setelah dilakukan percobaan pertama akan didapatkan hasil *feedback* mengenai percobaan pertama. Rekomendasi kedua ditentukan dengan cara yang sama seperti rekomendasi pertama, yaitu rekomendasi tetap dengan persamaan

$$12/4 + 5 = 8$$

Rekomendasi: 12/4+5=8
Masukkan hasil:

Sumber: Penulis

Rekomendasi kedua ditentukan secara tetap karena masih ada tujuh karakter yang belum dicoba, sehingga kemungkinan solusi masih terlalu banyak. Rekomendasi kedua tersebut melibatkan tujuh karakter sisa yang mungkin. Dengan begitu, setelah percobaan kedua semua karakter sudah diketahui apakah ada di dalam persamaan atau tidak.

3) Rekomendasi selanjutnya

Pada percobaan ketiga, semua karakter sudah diketahui posisinya. Kemudian dapat dilakukan pembangkitan semua kombinasi karakter yang masih ada. Pada tiap kombinasi yang dibangkitkan akan dicek apakah kombinasi tersebut merupakan persamaan matematika yang valid. Apabila valid maka akan disimpan sebagai kemungkinan persamaan.

Kemudian dari seluruh kemungkinan persamaan akan dipilih satu solusi untuk dijadikan rekomendasi dengan menggunakan fungsi heuristik.

```

87 def generate():
88     needed = notFinal()
89     if len(possible) > 8:
90         return "12/4+5=8"
91     comb = product(possible, repeat=needed)
92     best = ""
93     opr = 0
94     opd = 0
95
96     for kombinasi in comb:
97         tempFinal = copy.copy(final)
98         idx = 0
99         cont = True
100         for i in range(needed):
101             while(tempFinal[idx] != ''):
102                 idx += 1
103             if kombinasi[i] in tried[idx]:
104                 cont = False
105             tempFinal[idx] = kombinasi[i]
106             idx += 1
107         if cont:
108             kombString = ''.join(tempFinal)
109             if kombString not in history:

```

```

110     for i in range(len(must)):
111         if must[i] not in kombString:
112             cont=False
113         if checkMathFromString(kombString) and cont:
114             # print(kombString)
115             tempOpr, tempOpd = uniqueCount(kombString)
116             if tempOpd>opd or (tempOpd==opd and tempOpr>opr):
117                 best = kombString
118                 opr = tempOpr
119                 opd = tempOpd
120     print(best)
121     return best

```

Sumber: Penulis

F. Hasil Pengujian

Berikut adalah hasil salah satu pengujian yang dilakukan. Pada pengujian ini, kotak abu-abu menandakan karakter tidak ada di hasil (karakter 0), kotak kuning menandakan karakter ada di hasil tetapi posisi salah (karakter 1), kotak hijau menandakan karakter ada di hasil dengan posisi yang benar (karakter 2).

1) Percobaan pertama



Sumber: <https://nerdlegame.org/>

Rekomendasi: $7*9-3=60$
Masukkan hasil: 0 2 1 0 1 1 1 0

Sumber: Penulis

Berdasarkan gambar dapat diketahui bahwa karakter 7, 0, dan - tidak ada di hasil akhir. Sementara itu karakter 9, 3, =, 6 ada di hasil akhir tetapi posisi salah. Didapatkan pula informasi bahwa karakter * berada pada posisi yang tepat.

2) Percobaan kedua



Sumber: <https://nerdlegame.org/>

Rekomendasi: $12/4+5=8$
Masukkan hasil: 2 0 1 1 0 0 2 0

Sumber: Penulis

Berdasarkan gambar tersebut dapat diketahui bahwa karakter 1 dan = berada pada posisi yang tepat, karakter / dan 4 ada di solusi akhir tetapi salah posisi, sementara karakter 2, +, 5, 8 tidak ada di solusi akhir.

Pada akhir percobaan kedua ini, semua karakter sudah diketahui apakah ada di solusi akhir atau tidak. Selanjutnya hanya perlu mengikuti rekomendasi dari program.

3) Percobaan ketiga



Sumber: <https://nerdlegame.org/>

Rekomendasi: $1*36/4=9$
Masukkan hasil:

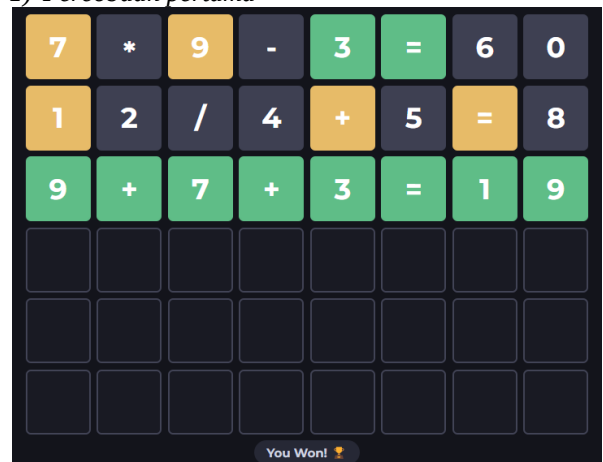
Sumber: Penulis

Rekomendasi pada percobaan ketiga telah memberikan hasil yang benar.

G. Pengujian lain

Pada bagian sebelumnya telah didapatkan bahwa program dapat menyelesaikan Nerdle hanya dalam tiga kesempatan dari total enam kesempatan yang diperbolehkan. Pada bagian ini akan dilakukan lima kali percobaan untuk mengetahui rata-rata percobaan yang diperlukan untuk menyelesaikan Nerdle menggunakan algoritma *bruteforce* ini.

1) Percobaan pertama



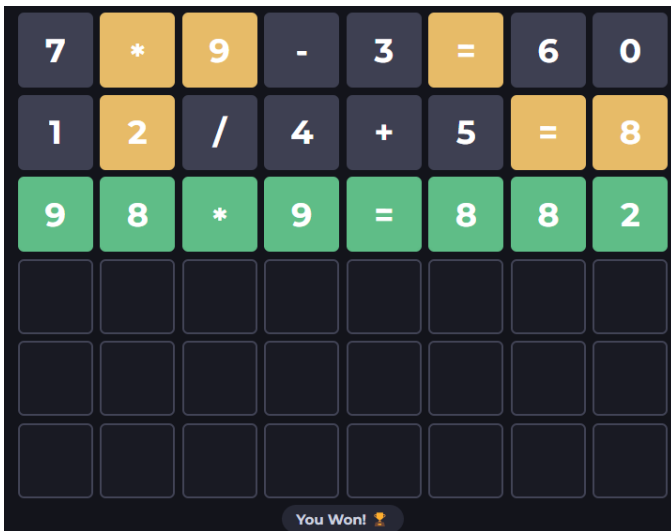
Sumber: <https://nerdlegame.org/>

Rekomendasi: $7*9-3=60$
Masukkan hasil: 1 0 1 0 2 2 0 0
Rekomendasi: $12/4+5=8$
Masukkan hasil: 1 0 0 0 1 0 1 0
Rekomendasi: $9+7+3=19$
Masukkan hasil: 2 2 2 2 2 2 2 2

Sumber: Penulis

Pada percobaan pertama, Nerdle berhasil terpecahkan dalam tiga percobaan.

2) Percobaan kedua



Sumber: <https://nerdlegame.org/>

```

Rekomendasi: 7*9-3=60
Masukkan hasil: 0 1 1 0 0 1 0 0
Rekomendasi: 12/4+5=8
Masukkan hasil: 0 1 0 0 0 0 1 1
Rekomendasi: 98*9=882
Masukkan hasil: 2 2 2 2 2 2 2 2
  
```

Sumber: Penulis

Pada percobaan kedua, Nerdle berhasil terpecahkan dalam tiga percobaan.

3) Percobaan ketiga



Sumber: <https://nerdlegame.org/>

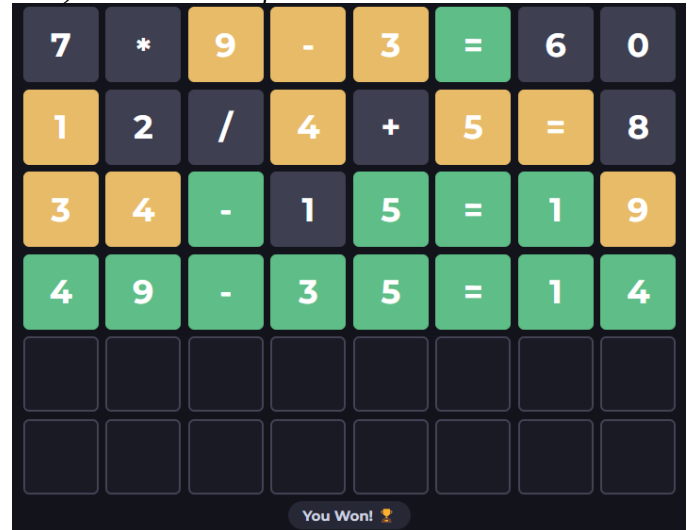
```

Rekomendasi: 7*9-3=60
Masukkan hasil: 2 0 0 1 1 1 1 0
Rekomendasi: 12/4+5=8
Masukkan hasil: 1 0 1 0 0 0 2 0
Rekomendasi: 7-36/6=1
Masukkan hasil: 2 2 2 2 2 2 2 2
  
```

Sumber: Penulis

Pada percobaan ketiga, Nerdle berhasil terpecahkan dalam tiga percobaan.

4) Percobaan keempat



Sumber: <https://nerdlegame.org/>

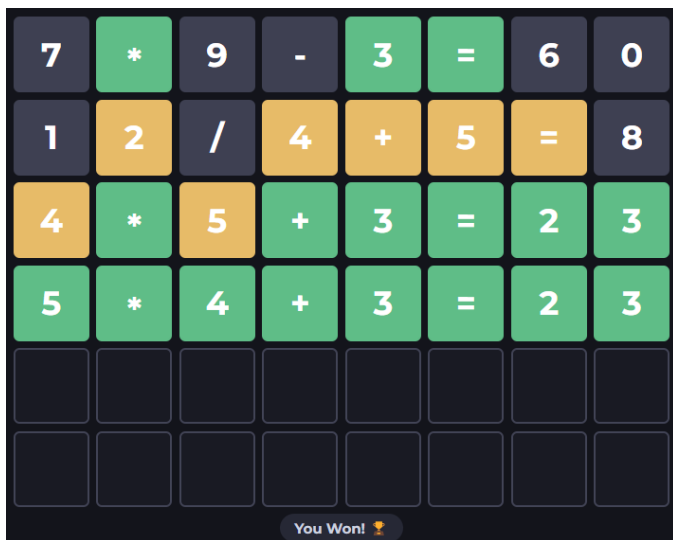
```

Rekomendasi: 7*9-3=60
Masukkan hasil: 0 0 1 1 1 2 0 0
Rekomendasi: 12/4+5=8
Masukkan hasil: 1 0 0 1 0 1 1 0
Rekomendasi: 34-15=19
Masukkan hasil: 1 1 2 0 2 2 2 1
Rekomendasi: 49-35=14
Masukkan hasil: 2 2 2 2 2 2 2 2
  
```

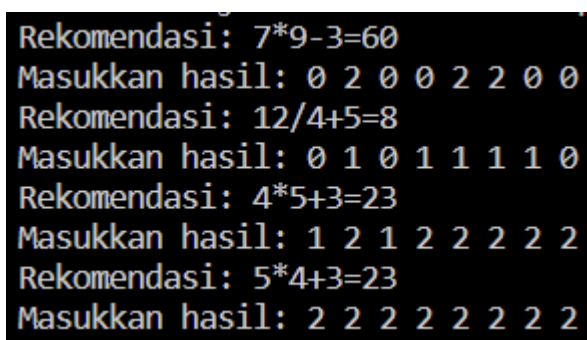
Sumber: Penulis

Pada percobaan keempat, Nerdle berhasil terpecahkan dalam empat percobaan.

5) Percobaan kelima



Sumber: <https://nerdlegame.org/>



Sumber: Penulis

Pada percobaan kelima, Nerdle berhasil terpecahkan dalam empat percobaan.

IV. KESIMPULAN

Berdasarkan percobaan yang dilakukan, dapat disimpulkan bahwa algoritma *brute force* dan teknik heuristik dapat digunakan untuk menyelesaikan permainan Nerdle dengan cukup efektif. Dari lima percobaan, tiga percobaan terselesaikan dalam tiga langkah, dua percobaan terselesaikan dalam empat langkah. Akan tetapi, terdapat pengembangan yang dapat dilakukan, yaitu optimisasi fungsi heuristik yang digunakan, serta pengubahan rekomendasi pada percobaan kedua agar lebih adaptif.

VIDEO LINK DI YOUTUBE

Video penjelasan mengenai makalah ini berada pada pranala berikut: <https://youtu.be/f-DIVF4wZr4>

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu pengerjaan makalah ini sehingga dapat selesai dengan maksimal. Puji syukur kepada Tuhan Yang Maha Esa karena telah melancarkan pengerjaan makalah ini. Terima kasih juga kepada dosen mata kuliah IF2210 Strategi Algoritma, Dr. Ir. Rinaldi Munir, M.T., dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc, dan Dr. Masayu Leylia Khodra, S.T, M.T. atas bimbingan selama ini dalam mengajar sehingga penulis dapat membuat makalah ini. Penulis juga berterima kasih kepada keluarga serta rekan-rekan yang telah memberikan semangat kepada penulis.

REFERENSI

- [1] R. Munir, "Algoritma Brute Force (Bagian 1)." Program Studi Teknik Informatika STEI ITB, pp. 1–70, 2022.
- [2] R. Munir, "Algoritma Brute Force (Bagian 2)." Program Studi Teknik Informatika STEI ITB, pp. 1–29, 2022.
- [3] BBC, "What is an equation?." <https://www.bbc.co.uk/bitesize/topics/z83rkqt/articles/zv2ck2p>, diakses 20 Mei 2022.
- [4] A. Hossein, "Combinatorial Mathematics: How to Count Without Counting" <https://home.ubalt.edu/ntsbarsh/business-stat/otherapplets/comcount.htm>, diakses pada 20 Mei 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

Dimas Shidqi Parikesit - 13520087