

TUGAS BESAR

STRATEGI ALGORITMA



IF-45-12

Disusun oleh:

JORDAN ADZANI (1301213307)

RAFI AZZAUZI HAIRUN (1301210087)

DIAZ RAMZY (1301210295)

Informatika

Universitas Telkom

2023

DAFTAR ISI

A. ABSTRAK.....	1
B. PENDAHULUAN.....	2
C. DASAR TEORI.....	3
1. Brute Force	3
2. Greedy	3
D. IMPLEMENTASI	4
1. ALGORITMA BRUTE FORCE	4
2. ALGORITMA GREEDY	6
3. Hasil Program dengan Algoritma Brute Force	8
4. Hasil Program dengan Algoritma Greedy	9
E. ANALISIS	10
1 . Running Time	10
2. Hasil Cost	10
3. Kelebihan dan Kekurangan Brute Force dan Greedy	11
A. Brute Force.....	11
B. Greedy.....	11
F. KESIMPULAN	12
DAFTAR PUSTAKA	13
LAMPIRAN	14

A. ABSTRAK

Tugas besar ini bertujuan untuk membandingkan metode Brute Force dan Greedy Algorithm dalam menyelesaikan masalah Assignment Problem. Assignment Problem adalah masalah optimisasi yang melibatkan pemetaan optimal antara sumber daya dan tugas. Metode Brute Force memeriksa semua kemungkinan kombinasi tugas- sumber daya untuk mencari solusi optimal, sementara metode Greedy Algorithm memilih pilihan terbaik pada setiap langkah berdasarkan kriteria yang diberikan.

Dalam penelitian ini, dilakukan implementasi kedua metode tersebut untuk menyelesaikan masalah Assignment Problem dengan menggunakan data pengujian yang berbeda. Hasil program dengan metode Brute Force dan Greedy Algorithm kemudian dianalisis dari segi waktu eksekusi dan cost yang dihasilkan. Selain itu, kelebihan dan kekurangan masing-masing metode juga dievaluasi.

Berdasarkan hasil analisis, ditemukan bahwa metode Brute Force memberikan solusi yang optimal, namun memerlukan waktu yang lebih lama terutama saat jumlah tugas dan sumber daya menjadi besar. Di sisi lain, metode Greedy Algorithm menawarkan waktu eksekusi yang lebih cepat, tetapi solusi yang dihasilkan tidak selalu optimal. Oleh karena itu, pemilihan metode tergantung pada ukuran masalah, sumber daya komputasi yang tersedia, dan kebutuhan optimisasi yang diinginkan.

B. PENDAHULUAN

Assignment Problem adalah masalah optimisasi yang melibatkan pemetaan optimal antara sumber daya dan tugas. Dalam metode Brute Force, semua kemungkinan kombinasi tugas-sumber daya diperiksa untuk mencari solusi optimal. Meskipun metode ini memastikan solusi terbaik, kompleksitas waktunya tinggi karena harus memeriksa banyak kemungkinan, terutama saat ukuran masalah menjadi lebih besar. Oleh karena itu, metode Brute Force sering digunakan untuk masalah dengan jumlah tugas dan sumber daya yang terbatas.

Di sisi lain, metode pencarian Greedy mengadopsi pendekatan yang lebih cepat tetapi tidak menjamin solusi optimal. Metode Greedy memilih pilihan terbaik pada setiap langkah berdasarkan kriteria yang diberikan. Keuntungan utama dari metode ini adalah kecepatan eksekusinya, terutama ketika ukuran masalah menjadi besar. Namun, metode Greedy rentan terhadap kemungkinan menghasilkan solusi yang sub optimal karena tidak mempertimbangkan konsekuensi pilihan saat ini pada langkah-langkah berikutnya.

Memahami kelebihan dan kelemahan masing-masing metode ini penting untuk menyelesaikan Assignment Problem dengan efektif. Dalam praktiknya, pilihan antara metode Brute Force dan Greedy tergantung pada ukuran masalah, sumber daya komputasi yang tersedia, dan kebutuhan optimisasi yang diinginkan. Untuk masalah dengan ukuran masalah yang kecil hingga sedang, metode Brute Force mungkin lebih cocok untuk memastikan solusi optimal. Namun, jika ada batasan waktu yang ketat atau ukuran masalah yang besar, metode Greedy dapat memberikan solusi yang cukup baik dengan waktu komputasi yang lebih singkat.

C. DASAR TEORI

Penyelesaian Assignment Problem dengan menggunakan metode Brute Force dan Greedy serta membandingkan kedua metode tersebut.

1. Brute Force

Metode Brute Force adalah pendekatan yang memeriksa semua kemungkinan kombinasi untuk mencari solusi optimal. Pada dasarnya, metode ini melibatkan pengujian seluruh ruang solusi yang memungkinkan. Dalam konteks Assignment Problem, metode Brute Force akan memeriksa semua kemungkinan pemetaan tugas-sumber daya untuk menemukan yang memiliki biaya paling minimal atau nilai maksimal. Keuntungan utama metode ini adalah memastikan solusi optimal, namun kerugiannya adalah kompleksitas waktu yang tinggi karena jumlah kemungkinan yang harus diperiksa tumbuh eksponensial dengan ukuran masalah.

2. Greedy

Metode Greedy adalah pendekatan yang memilih pilihan terbaik pada setiap langkah berdasarkan kriteria yang diberikan. Dalam konteks Assignment Problem, metode Greedy akan memilih kemungkinan data yang paling cocok berdasarkan metrik yang ditentukan pada setiap iterasi. Keuntungan utama metode ini adalah kecepatan eksekusinya, terutama saat menghadapi ukuran masalah yang besar. Namun, metode Greedy tidak menjamin solusi yang optimal karena hanya mempertimbangkan pilihan terbaik pada setiap langkah dan tidak memperhatikan konsekuensi pilihan pada langkah-langkah berikutnya.

Perbandingan antara algoritma Brute Force dan Greedy dalam menyelesaikan Assignment Problem bergantung pada prioritas yang diinginkan. Algoritma Brute Force menawarkan solusi yang optimal namun membutuhkan waktu yang lama untuk mengeksplorasi semua kemungkinan. Di sisi lain, algoritma Greedy menawarkan solusi dengan waktu eksekusi yang lebih cepat, tetapi solusi yang dihasilkan tidak selalu optimal. Dengan mempertimbangkan tujuan dan batasan yang ada, pengguna dapat memilih antara algoritma Brute Force dan Greedy untuk memecahkan Assignment Problem secara efektif.

D. IMPLEMENTASI

1. ALGORITMA BRUTE FORCE

```
1  #fungsi membaca data dari text data
2  def read_assignment_data(filename):
3      data = []
4      with open(filename, 'r') as file:
5          lines = file.readlines()
6          for line in lines:
7              row = list(map(int, line.split()))
8              data.append(row)
9      return data
10
11 #menghitung total cost dari iterasi
12 def calculate_total_cost(assignment, data):
13     total_cost = 0
14     for i in range(len(assignment)):
15         total_cost += data[i][assignment[i]]
16     return total_cost
17
18 #fungsi menghitung permutasi dari kemungkinan yang terjadi
19 def generate_permutations(n, current_permutation, used, permutations, data):
20     if len(current_permutation) == n:
21         total_cost = calculate_total_cost(current_permutation, data)
22         permutations.append((current_permutation[:], total_cost))
23         return
24
25     for i in range(n):
26         if not used[i]:
27             used[i] = True
28             current_permutation.append(i)
29             generate_permutations(n, current_permutation, used, permutations, data)
30             current_permutation.pop()
31             used[i] = False
32
```

```

1  #fungsi mencari hasil minimum yang paling optimal
2  def find_optimal_assignment(data):
3      num_workers = len(data)
4      num_jobs = len(data[0])
5      min_cost = float('inf')
6      optimal_assignment = None
7
8      permutations = []
9      used = [False] * num_jobs
10     generate_permutations(num_jobs, [], used, permutations, data)
11
12     for assignment, cost in permutations:
13         if cost < min_cost:
14             min_cost = cost
15             optimal_assignment = assignment
16
17     return optimal_assignment, min_cost
18
19 #mencari file yang akan dibaca, dan diolah dalam fungsi read data
20 filename = 'C:/Users/diazzr/Documents/tugas pemograman/TUBES SA/data.txt'
21 data = read_assignment_data(filename)
22
23 #mengambil nilai hasil data dengan paling optimal yang didapatkan
24 optimal_assignment, min_cost = find_optimal_assignment(data)
25

```

```

1  #menampilkan setiap permutasi beserta costnya
2  print("Permutasi: ")
3  permutations = []
4  used = [False] * len(data[0])
5  generate_permutations(len(data[0]), [], used, permutations, data)
6  i = 1
7  for assignment, cost in permutations:
8      for j in range(len(assignment)):
9          assignment[j] += 1
10     print("Iterasi ke -", i, "=> Assignment:", assignment, "dengan Cost:", cost)
11     i += 1
12
13 #membuat agar data job yang dipilih lebih jelas
14 for i in range(len(optimal_assignment)):
15     optimal_assignment[i] += 1
16
17 print("Assignment Teroptimal:", optimal_assignment)
18 print("Cost paling sedikit:", min_cost)

```

2. ALGORITMA GREEDY

```
1  #fungsi membaca data dari text data
2  def read_assignment_data(filename):
3      data = []
4      with open(filename, 'r') as file:
5          lines = file.readlines()
6          for line in lines:
7              row = list(map(int, line.split()))
8              data.append(row)
9      return data
10
11 #menghitung total cost dari iterasi
12 def calculate_total_cost(assignment, data):
13     total_cost = 0
14     for i in range(len(assignment)):
15         total_cost += data[i][assignment[i]]
16     return total_cost
17
18 #fungsi menghitung permutasi dari kemungkinan yang terjadi
19 def generate_permutations(n, current_permutation, used, permutations, data):
20     if len(current_permutation) == n:
21         total_cost = calculate_total_cost(current_permutation, data)
22         permutations.append((current_permutation[:], total_cost))
23     return
24
25     for i in range(n):
26         if not used[i]:
27             used[i] = True
28             current_permutation.append(i)
29             generate_permutations(n, current_permutation, used, permutations, data)
30             current_permutation.pop()
31             used[i] = False
32
33 #fungsi mencari hasil minimum yang paling optimal
34 def find_optimal_assignment(data):
35     num_workers = len(data)
36     num_jobs = len(data[0])
37     min_cost = float('inf')
38     optimal_assignment = None
39
40     permutations = []
41     used = [False] * num_jobs
42     generate_permutations(num_jobs, [], used, permutations, data)
43
44     for assignment, cost in permutations:
45         if cost < min_cost:
46             min_cost = cost
47             optimal_assignment = assignment
48
49     return optimal_assignment, min_cost
```



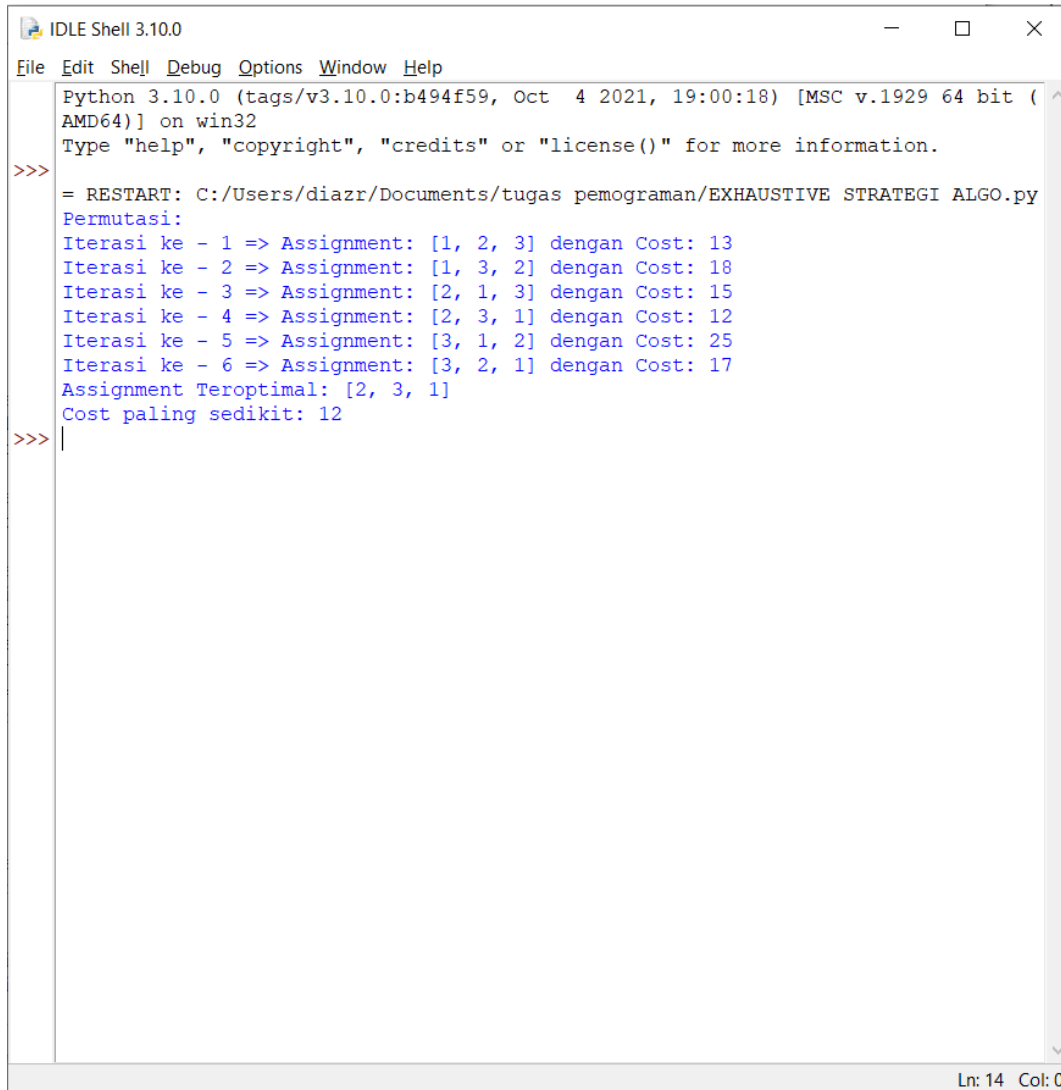
```

1  #mencari file yang akan dibaca, dan diolah dalam fungsi read data
2  filename = 'C:/Users/diazr/Documents/tugas pemograman/TUBES SA/data.txt'
3  data = read_assignment_data(filename)
4
5  #mengambil nilai hasil data dengan paling optimal yang didapatkan
6  optimal_assignment, min_cost = find_optimal_assignment(data)
7
8  #menampilkan setiap permutasi beserta costnya
9  print("Permutasi: ")
10 permutations = []
11 used = [False] * len(data[0])
12 generate_permutations(len(data[0]), [], used, permutations, data)
13 i = 1
14 for assignment, cost in permutations:
15     for j in range(len(assignment)):
16         assignment[j] += 1
17     print("Iterasi ke -", i, "=> Assignment:", assignment, "dengan Cost:", cost)
18     i += 1
19
20 #membuat agar data job yang dipilih lebih jelas
21 for i in range(len(optimal_assignment)):
22     optimal_assignment[i] += 1
23
24 print("Assignment Teroptimal:", optimal_assignment)
25 print("Cost paling sedikit:", min_cost)
26

```

3. Hasil Program dengan Algoritma Brute Force

Dengan menggunakan data dengan 3 orang memilih 3 job, maka hasilnya seperti dibawah ini

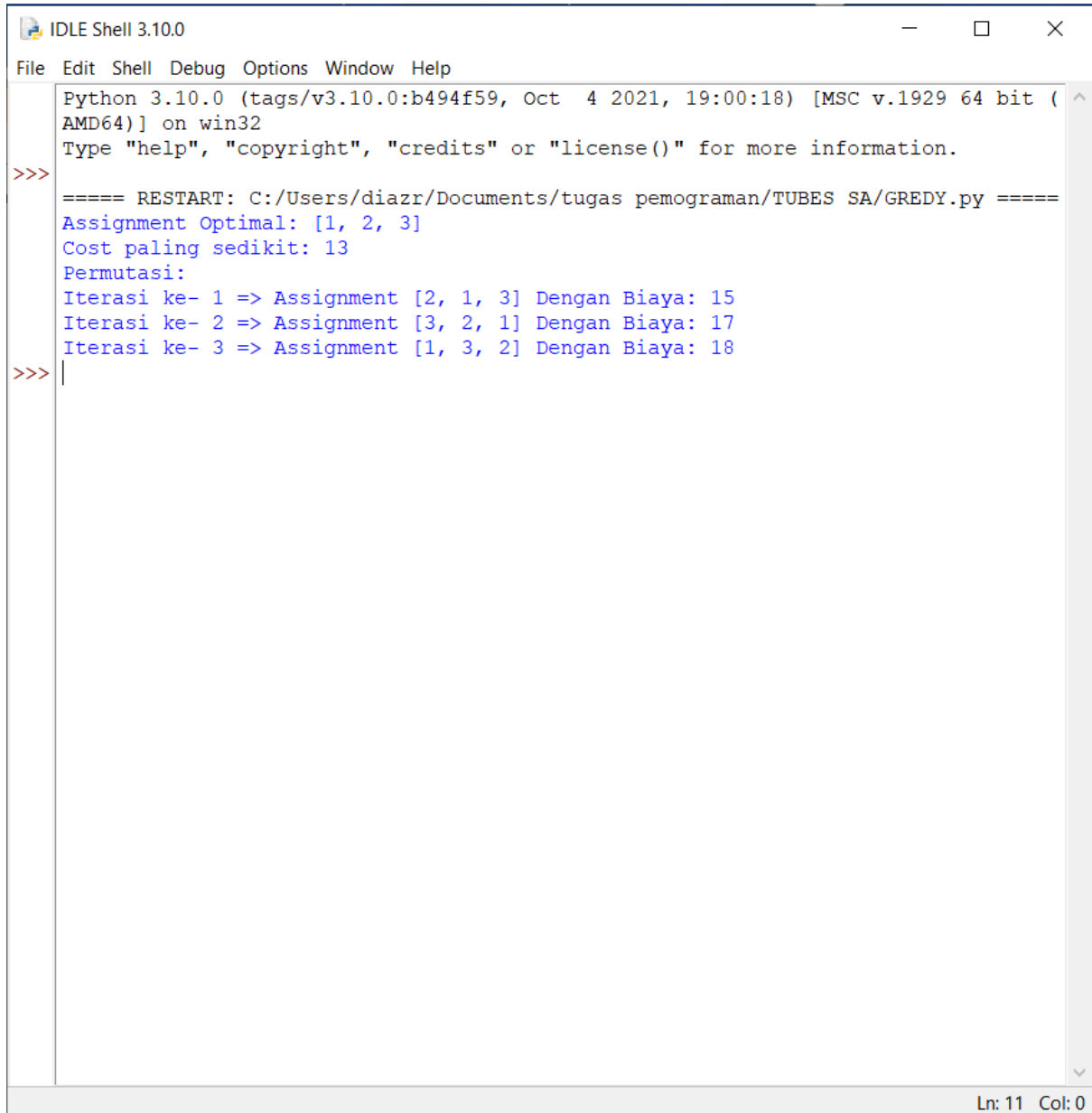


```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/diazzr/Documents/tugas pemograman/EXHAUSTIVE STRATEGI ALGO.py
Permutasi:
Iterasi ke - 1 => Assignment: [1, 2, 3] dengan Cost: 13
Iterasi ke - 2 => Assignment: [1, 3, 2] dengan Cost: 18
Iterasi ke - 3 => Assignment: [2, 1, 3] dengan Cost: 15
Iterasi ke - 4 => Assignment: [2, 3, 1] dengan Cost: 12
Iterasi ke - 5 => Assignment: [3, 1, 2] dengan Cost: 25
Iterasi ke - 6 => Assignment: [3, 2, 1] dengan Cost: 17
Assignment Teroptimal: [2, 3, 1]
Cost paling sedikit: 12
>>> |
```

Ln: 14 Col: 0

4. Hasil Program dengan Algoritma Greedy

Sama menggunakan data dengan 3 orang memilih 3 job, maka hasilnya seperti dibawah ini



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/diazzr/Documents/tugas pemograman/TUBES SA/GREDY.py =====
Assignment Optimal: [1, 2, 3]
Cost paling sedikit: 13
Permutasi:
Iterasi ke- 1 => Assignment [2, 1, 3] Dengan Biaya: 15
Iterasi ke- 2 => Assignment [3, 2, 1] Dengan Biaya: 17
Iterasi ke- 3 => Assignment [1, 3, 2] Dengan Biaya: 18
>>> |
```

Ln: 11 Col: 0

E. ANALISIS

1 . Running Time

Kami membuat uji coba running time, dengan data dimulai dari 2 orang memilih 2 pekerjaan sampai dengan 6 orang memilih 6 pekerjaan, Sehingga didapatkan data sebagai berikut

HASIL PERHITUNGAN RUNNING TIME (DETIK)						
WORKER X JOB	METODE					
	PERCOBAAN KE - 1		PERCOBAAN KE - 2		PERCOBAAN KE - 3	
	BRUTE FORCE	GREEDY	BRUTE FORCE	GREEDY	BRUTE FORCE	GREEDY
2 x 2	0.000999	0.001236	0.007926	0.001005	0.000993	0.000973
3 x 3	0.001416	0.000997	0.007849	0.001551	0.001603	0.00203
4 x 4	0.001999	0.002234	0.004001	0.00115	0.002188	0.001624
5 x 5	0.001585	0.002062	0.005337	0.001087	0.002114	0.001569
6 x 6	0.001002	0.002515	0.005543	0.001185	0.002022	0.001543

2. Hasil Cost

Berdasarkan hasil cost yang didapatkan dengan mengambil data 3 orang memilih 3 pekerjaan, 6 orang memilih 6 pekerjaan sebagai berikut:

WORKER X JOB	OPTIMAL COST	
	BRUTE FORCE	GREEDY
3 x 3	12	13
6 x 6	13	16

3. Kelebihan dan Kekurangan Brute Force dan Greedy

Berdasarkan data yang telah didapatkan didapatkan beberapa hasil analisis kelebihan dan kekurangan yang sangat terlihat dari dua metode, yaitu :

A. Brute Force

Kelebihan:

1. Hasil yang didapatkan sangat optimal dengan mencari semua kemungkinan permutasi.
2. Cocok untuk dengan jumlah pekerjaan dan pekerja yang relatif kecil.

Kekurangan:

1. Tidak efisien untuk jumlah pekerjaan dan pekerja yang besar.
2. Memiliki kompleksitas waktu yang tinggi, terutama saat jumlah pekerjaan dan pekerja yang besar.

B. Greedy

Kelebihan:

1. Lebih efisien karena tidak memeriksa semua kemungkinan permutasi.
2. Cocok untuk dengan jumlah pekerjaan dan pekerja yang besar.

Kekurangan:

1. Tidak menjamin solusi yang optimal.
2. Tidak menggambarkan seluruh ruang solusi atau permutasi yang mungkin.

F. KESIMPULAN

Berdasarkan perbandingan dari data yang kami dapatkan, disimpulkan bahwa metode Brute Force lebih cocok untuk masalah dengan skala kecil dan solusi yang optimal diperlukan. Namun, metode ini cenderung tidak efisien saat masalah atau data besar.

Sementara itu, metode greedy lebih cocok untuk masalah dengan skala yang lebih besar dan solusi optimal bukanlah prioritas utama. Metode ini menawarkan solusi yang cukup baik dalam waktu yang relatif singkat, meskipun tidak menjamin solusi optimal.

DAFTAR PUSTAKA

BrainKart.com (Enhanced by Google). (n.d.). Introduction to the Design and Analysis of Algorithms : Brute Force and Exhaustive Search. Diambil dari BrainKart.com: https://www.brainkart.com/article/Brute-Force-Search_8013/

Parewa Labs Pvt. Ltd. (n.d.). Greedy Algorithm. Diambil dari Programiz: <https://www.programiz.com/dsa/greedy-algorithm>

Wikipedia. (2017, May 26). Artificial Intelligence/Search/Brute Force search/Breadth-firstsearch. Diambil dari WIKIBOOKS: https://en.wikibooks.org/wiki/Artificial_Intelligence/Search/Brute_Force_search/Breadth-first_search

LAMPIRAN

https://drive.google.com/drive/folders/1ePsqCOj2iQeWvz2n7z_895HOfmP5W2W1?usp=sharing