TUBES: Desain dan Analisis Algoritma
(CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

Perbandingan Algoritma Greedy dan Algoritma Exhaustive Search Dalam Minimisasi Waktu Pelayanan Dalam Restoran

Ananda Fitri Karimah (NIM: 1301170744)

Ferzi Samal Yerzi (NIM: 1301174642)

Mangaraja Pinayungan (NIM: 1301164015)

Informatics, School of Computing,
Telkom University, Indonesia.

Email: (anandafitri00@gmail.com, ferzisamaly@gmail.com dan mptigor@gmail.com)

Abstract. Pada paper ini mempelajari masalah pada sebuah restoran yang ingin melayani banyak pelanggan, untuk dapat memecahkan menggunakan pendekatan *greedy algorithm* dan *exhaustive search algorithm*. *greedy algorithm* adalah pemecahan masalah dengan langkah demi langkah yang nantinya keputusan yang diambil tidak dapat diubah lagi pada langkah berikutnya, sama dengan *greedy algorithm* pada pemecahan masalah ini menggunakan pendekatan *exhaustive search algorithm*, *exhaustive search algorithm* adalah algoritma dengan mencari semua kemungkinan yang ada dan dengan menggunakan *exhaustive search algorithm* akan menghasilkan problem solving yang lebih optimal namun lebih lama dalam pengoprasiannya. Dengan adanya pendekatan menggunakan *greedy algorithm* dan *exhaustive search algorithm* nantinya akan dibandingkan algoritma yang lebih optimal.

1. Pendahuluan

Masalah yang akan dihadapi di dalam paper ini adalah pada sebuah restoran ingin melayani banyak pelanggan dengan waktu yang sesingkat-singkatnya, sehingga untuk dapat mengetahui pelanggan mana yang akan dilayani terlebih dahulu dapat diketahui melalui minimalisir penjadwalan sistem.

Untuk dapat memecahkan masalah pendekatan yang kami lakukan terhadap permasalahan ini dengan dua cara yaitu dengan menggunakan *exhaustive search algorithm* dan *greedy algorithm* dalam pencarian waktu optimalnya. *Exhaustive search algorithm* adalah algoritma dengan mencari semua kemungkinan yang ada dan dengan menggunakan *exhaustive search algorithm* akan menghasilkan problem solving yang lebih optimal namun lebih lama dalam pengoprasiannya, sedangkan *greedy algorithm* adalah algoritma serakah yang selalu membuat pilihan yang terlihat optimal, artinya ia membuat pilihan yang optimal secara lokal dengan harapan bahwa pilihan ini akan mengarah pada solusi optimal secara global (Thomas, H.C , 2009)

TUBES: Desain dan Analisis Algoritma
(CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

2. Analisis Algoritma

2.1. Exhaustive Search Algorithm

2.1.1. Pengertian Exhaustive Search Algorithm

E*xhaustive Search Algorithm* adalah algoritma dengan mencari semua kemungkinan yang ada, untuk mendapatkan hasil optimasi yang paling terbaik, namun untuk mendapatkan hasil optimasi yang terbaik dengan *exhaustive Search algorithm* membutuhkan waktu yang sangat lama, karena mencari semua kemungkinan yang ada.

2.1.2. Analisis Kompleksitas Waktu Algoritma Exhaustive Search

Best Case : Best Case terjadi jika tidak ada pelanggan yang diinputkan
Worst Case : Worst Case terjadi jika data pelanggan yang diinputkan berjumlah banyak.
Penyelesaian masalah dengan *exhaustive search algorithm* adalah dengan melakukan permutasi ke semua pelanggan yang belum dilayani, jika terdapat *n* buah pelanggan, maka terdapat *n!* urutan pelanggan, dengan kemungkinan banyaknya permutasi yang dilakukan kompleksitas waktu untuk *exhaustive search algorithm* search adalah T(n) = O(nn!).

2.2. Greedy Algorithm

2.2.1. Pengertian Greedy Algorithm

greedy algorithm adalah algoritma yang berusaha memecahkan masalah dengan cara mengambil pilihan terbaik atau solusi optimum yang diperoleh saat itu tanpa mempertimbangkan hasil yang diterimanya kemudian. Dengan *greedy algorithm* pencarian minimisasi waktu pelayanan dikatakan cukup singkat, namun dengan *greedy algorithm* hasil yang dihasilkan tidak selalu dapat di jadikan yang terbaik.

2.2.2. Analisis Kompleksitas Waktu Greedy Algorithm

Best Case : Best Case terjadi jika tidak ada pelanggan yang diinputkan
Worst Case : Worst Case terjadi jika data pelanggan yang diinputkan berjumlah banyak.
Penyelesaian masalah dalam *greedy algorithm* adalah dengan setiap langkah, pilihlah pelanggan yang membutuhkan waktu pelayanan terkecil di antara pelanggan lai yang belum dilayani, kemudian lakukan pengulangan kembali sehingga urutan pelanggan berdasarkan waktu pelayanan dalam urutan yang menaik. Jika pelanggan sudah terurut, kompleksitas algoritma menggunakan *greedy algorithm* adalah T(n) = O(n).

3. Experimental Results

3.1 Study Case

Dalam Persoalaan ini terdapat sebuah restoran yang terdapat 10 orang pelangaanm dengan berbeda-beda waktu pelayanan, setiap pelanggan memiliki waktu pelayanan sebagian berikut : t1=5, t2=10, t3=3, t4=11, t5=2, t6=7, t7=15, t8=6 ,t9=1, t10=20.

TUBES: Desain dan Analisis Algoritma
(CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

3.2 Computer specifications

Komponen	Detail
Sistem Operasi	Windows 10 Pro 64-bit
Prosessor	Intel® Core™ i5-3317U CPU @ 1.70Ghz (4CPUs), ~1.7GHz
Kartu Grafis	Intel® HD Graphics 4000 (1792 MB)
Tipe Chip	Intel® HD Graphics Family
Display Mode	1366 x 768 (32 bit) (40Hz)
Monitor	Generic PnP Monitor
Memory	4096MB RAM
Storage	SSD 160GB
Sistem Model	HP EliteBook Folio 9470m
Direct X	DirectX 12

3.3 Experimental results

Kedua algoritma ini menghasilkan waktu optimal yang serupa. Dengan jumlah pelanggan yang sama yaitu sepuluh orang, dan urutan waktu yang sama sepeti pada studi kasus dalam detik, berikut adalah hasil perbandingan *exhaustive search algorithm* dan *greedy algorithm* :

A. *Exhaustive Search Algorithm*

Dalam perhitungan *exhaustive search algorithm* melakukan permutasi sebanyak *n!,* sehinggan menghasilkan urutan pelayanan dengan hasil (1,2,3,5,6,7,10,11,15,20) namun dengan menggunakan *exhaustive search algorithm* memerlukan waktu sekitas 106,9190742969513 detik.

TABLE II

Perhitungan *Exhaustive Search Algorithm*

Urutan	**T**	**Hasil**
1,2,3,4,5,6,7,8,9,10	**5+(5+10)+ (5+10+3)+ (5+10+3+11)+ (5+10+3+11+2)+ (5+10+3+11+2+7)+ (5+10+3+11+2+7+15)+ (5+10+3+11+2+7+15+6)+ (5+10+3+11+2+7+15+6+1)+ (5+10+3+11+2+7+15+6+1+20)**	**388**
1,2,3,4,5,6,7,8,10,9	**5+(5+10)+ (5+10+3)+ (5+10+3+11)+ (5+10+3+11+2)+ (5+10+3+11+2+7)+ (5+10+3+11+2+7+15)+ (5+10+3+11+2+7+15+6)+ (5+10+3+11+2+7+15+6+20)+ (5+10+3+11+2+7+15+6+20+1)**	**407**
1,2,3,4,5,6,7,9,10,8	**5+(5+10)+ (5+10+3)+ (5+10+3+11)+ (5+10+3+11+2)+ (5+10+3+11+2+7)+ (5+10+3+11+2+7+15) +(5+10+3+11+2+7+15+1)+ (5+10+3+11+2+7+15+1+20)+ (5+10+3+11+2+7+15+1+20+6)**	**397**
1,2,3,4,5,6,7,9,8,10	**5+(5+10)+ (5+10+3)+ (5+10+3+11)+ (5+10+3+11+2)+ (5+10+3+11+2+7)+ (5+10+3+11+2+7+15)+ (5+10+3+11+2+7+15+1)+ (5+10+3+11+2+7+15+1+6)+ (5+10+3+11+2+7+15+1+6+20)**	**383**
1,2,3,4,5,6,8,10,7,9	**5+(5+10)+ (5+10+3)+ (5+10+3+11)+ (5+10+3+11+2)+ (5+10+3+11+2+7)+ (5+10+3+11+2+7+6)+ (5+10+3+11+2+7+6+20)+ (5+10+3+11+2+7+6+20+15)+ (5+10+3+11+2+7+6+20+15+20)**	**422**
:	**:**	**:**
9,5,3,1,8,6,2,4,7,10	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)+ (1+2+3+5+6+7)+ (1+2+3+5+6+7+10)+ (1+2+3+5+6+7+10+11)+ (1+2+3+5+6+7+10+11+15)+ (1+2+3+5+6+7+10+11+15+20)**	**281**

TUBES: Desain dan Analisis Algoritma
(CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

```
Jumlah pelanggan sebanyak  10  orang, dengan waktu masing-masing pelayanan adalah  [5, 10, 3, 11, 2, 7, 15, 6, 1, 20]

Sistem sedang melakukan perhitungan ...

 --> Waktu paling optimal untuk melayani sebanyak  10  orang adalah  281  detik

Dengan urutan waktu sebagai berikut :  [1, 2, 3, 5, 6, 7, 10, 11, 15, 20]
waktu program =  106.9190742969513 detik
```

Gambar.1. Hasil program Exhaustive Search Algorithm

B. *Greedy Algorithm*

Dalam perhitungan *greedy* langsung mengurutkan waktu pelayanan dari yang terkecil hingga terbesar, sehinggan menghasilkan urutan pelayanan dengan hasil (1,2,3,5,6,7,10,1115,20) namun dengan menggunakan *greedy algorithm* memerlukan waktu sekitas 0,0019989013671875 detik.

TABLE II
Perhitungan *Greedy Algorithm*

Urutan	**T**	**Hasil**
9	**1**	**1**
9,5	**1+(1+2)**	**4**
9,5,3	**1+(1+2)+ (1+2+3)**	**10**
9,5,3,1	**1+(1+2)+ (1+2+3)+ (1+2+3+5)**	**25**
9,5,3,1,8	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)**	**38**
9,5,3,1,8,6	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)+ (1+2+3+5+6+7)**	**62**
9,5,3,1,8,6,2	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)+ (1+2+3+5+6+7)+ (1+2+3+5+6+7+10)**	**96**
9,5,3,1,8,6,2,4	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)+ (1+2+3+5+6+7)+ (1+2+3+5+6+7+10)+ (1+2+3+5+6+7+10+11)**	**141**
9,5,3,1,8,6,2,4,7	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)+ (1+2+3+5+6+7)+ (1+2+3+5+6+7+10)+ (1+2+3+5+6+7+10+11)+ (1+2+3+5+6+7+10+11+15)**	**201**
9,5,3,1,8,6,2,4,7,10	**1+(1+2)+ (1+2+3)+ (1+2+3+5)+ (1+2+3+5+6)+ (1+2+3+5+6+7)+ (1+2+3+5+6+7+10)+ (1+2+3+5+6+7+10+11)+ (1+2+3+5+6+7+10+11+15)+ (1+2+3+5+6+7+10+11+15+20)**	**281**

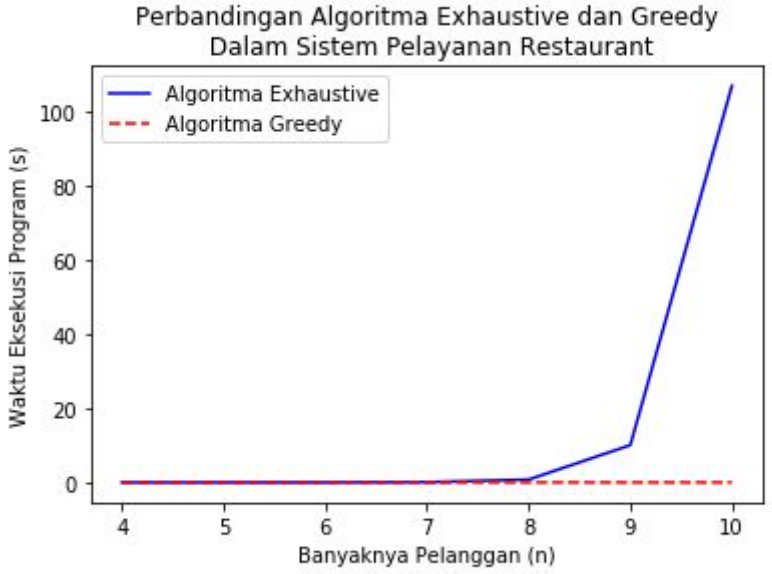
```
jumlah pelanggan sebanyak  10  orang, dengan waktu masing-masing pelayanan adalah  [5, 10, 3, 11, 2, 7, 15, 6, 1, 20]

Setelah di urutkan maka urutan waktu pelanggan akan seperti ini
[1, 2, 3, 5, 6, 7, 10, 11, 15, 20]


 --> Waktu paling optimal untuk melayani sebanyak  10  orang adalah  281  detik

Dengan urutan waktu sebagai berikut :  [1, 2, 3, 5, 6, 7, 10, 11, 15, 20]
Execution time is
0.0019989013671875
```

Gambar.2. Hasil Program Greedy Algorithm

Kedua algoritma di atas memiliki perbandingan hasil optimal yang sama, dan urutan waktu yang sama, tetapi berbeda dalam lamanya program, terlihat pada *exhaustive search algorithm* waktu yang dibutuhkan adalah 106 detik atau 1.766 menit.

3.3.1 Grafik Komparasi Waktu Algoritma

Berikut adalah perbandingan waktu *exhaustive search algorithm* dan *greedy algorithm* dengan melakukan input jumlah pelanggan dari sebanyak empat orang pelanggan hingga sepuluh orang pelanggan. Urutan waktu yang diterapkan sama yaitu (5,10,3,11,2,7,15,6,1,20) dalam detik :



Gambar. 3 Perbandingan Algoritma *Exhaustive Search* dan *Greedy*.

4. Kesimpulan

Dengan melakukan perlakuan inputan angka yang sama di dalam *exhaustive search algorithm* dan *greedy algorithm*. Dapat disimpulkan bahwa dalam menentukan hasil yang optimal, kedua algoritma ini memiliki kesamaan hasil, tetapi dari sisi waktu yang dibutuhkan untuk melakukan perhitungan, *greedy algorithm* sangat baik, sementara *exhaustive search algorithm* membutuhkan waktu yang sangat lama dikarenakan mencari kombinasi-kombinasi waktu dari seorang pelanggan, dimana memerlukan waktu sebanyak (n!).

Maka dari itu penggunaan *greedy algorithm* lebih dianjurkan karena waktu yang dibutuhkan jauh lebih cepat dibandingkan *exhaustive search algorithm* , walaupun *greedy algorithm* terkadang memiliki hasil yang kurang optimal tetapi untuk memprediksi hasil cukup baik digunakan mengingat kembali waktu yang dibutuhkan sangat singkat.

References

Thomas, H.C(2009). Introduction to Algorithms.Retrieved from https://books.google.co.id/books?isbn=0262032937.

TUBES: Desain dan Analisis Algoritma
(CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

LAMPIRAN

Lampiran 1. Program Code

Program *Code Greedy*

```
import time
i = 1
#Array dibawah adalah array waktu pelanggan
l= [5,10,3,11,2]
hasil = []
print("")
print("jumlah pelanggan sebanyak ", len(l) , " orang, dengan
waktu masing-masing pelayanan adalah ", l)
print("")
l.sort(reverse=False)
print("Setelah di urutkan maka urutan waktu pelanggan akan
seperti ini")
print(l)
print ("\n")
start = time.time()
ix = 0
jum = 0
while ix < len(l) :
      jx = 0
      while jx <= ix :
            jum = jum + l[jx]
            jx = jx + 1
      ix = ix + 1
print(" --> Waktu paling optimal untuk melayani sebanyak ",
len(l)," orang adalah ", jum , " detik")
print()
print("Dengan urutan waktu sebagai berikut : ", l)
end = time.time()
print ("Execution time is ")
print(end - start)
```

TUBES: Desain dan Analisis Algoritma
(CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

Program *Code Exhaustive Search*

```
import time
from itertools import permutations
from os import system
#Array dibawah adalah array waktu pelanggan
ArrayWaktu = [5,10,3,11,2]
i = 0
JumWaktu = 0
baru = []
print("")
print("Jumlah pelanggan sebanyak ", len(ArrayWaktu) , " orang,
dengan waktu masing-masing pelayanan adalah ", ArrayWaktu)
print("")
start = time.time()
perm = permutations(ArrayWaktu)
print("Sistem sedang melakukan perhitungan ...")
print("")
for j in perm:
      i = 0
      JumWaktu = 0
      while i < len(j):
            k = 0
            while k <= i:
                  JumWaktu = JumWaktu + j[k]
                  k = k + 1
            i = i + 1
      baru.append(JumWaktu)
baru.sort()
ArrayWaktu.sort()
# system('cls')
print(" --> Waktu paling optimal untuk melayani sebanyak ",
len(ArrayWaktu)," orang adalah ", baru[0] , " detik")
print()
print("Dengan urutan waktu sebagai berikut : ", ArrayWaktu)
end = time.time()
print("waktu program = ", end-start, "detik")
if(end - start > 120) :
      menit = (end - start) / 60
      print("Waktu dalam menit : ", menit , "menit")
```

TUBES: Desain dan Analisis Algoritma (CSH2G3) 2019
DOSEN: PHN
KELAS: IFIK 41-01

Lampiran 2. Dokumentasi