Sistem Prediksi Daftar Produk Penjualan untuk Mendapatkan Keuntungan Maksimum Menggunakan Algoritma Greedy

Hana Fathiyah - 13520047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 123hanafath@gmail.com

***Abstrak*—Strategi algoritma (*algorithm strategies)* merupakan suatu pendekatan umum yang digunakan untuk memecahkan suatu persoalan secara algoritmis. Persoalan merupakan suatu pertanyaan atau tugas yang akan dicari jawabannya. Salah satu contoh persoalan klasik adalah *integer knapsack problem* yang merupakan suatu cara untuk memilih objek-objek untuk dimasukkan ke dalam *knapsack*, sehingga tidak melebihi kapasitas, tetapi memberikan keuntungan maksimal. Persoalan ini dapat diselesaikan dengan algoritma *greedy*. Salah satu implementasi dari persoalan ini adalah bagaimana cara mendapat keuntungan maksimal dari penjualan suatu produk dengan jumlah modal tertentu.**

Kata kunci—Strategi Algoritma, Persoalan, Integer Knapsack Problem, Algoritma Greedy, Keuntungan Maksimal

I. PENDAHULUAN

Menurut Kamus Besar Bahasa Indonesia (KBBI), strategi dapat diartikan sebagai rencana yang cermat mengenai suatu kegiatan untuk mencapai sasaran khusus. Algoritma diartikan sebagai urutan langkah-langkah untuk memecahkan suatu permasalahan. Strategi algoritma merupakan kumpulan metode atau dapat disebut juga sebagai teknik untuk memecahkan masalah guna mencapai suatu tujuan yang telah ditentukan. Dalam hal ini, strategi algoritma berbentuk suatu deskripsi metode atau teknik yang dinyatakan dalam suau urutan langkah-langkah penyelesaian. Strategi algoritma tepat untuk digunakan dalam bermacam-macam persoalan di berbagai bidang komputasi.

Strategi pemecahan masalah dapat dikelompokkan menjadi strategi solusi langsung (*direct solution strategies)*, strategi berbasis pencarian pada ruang status (*state-space base strategies*), strategi solusi atas-bawah (*top-down solution strategies*), dan strategi solusi bawah-atas (*bottom-up solution strategies*). Algoritma *brute force* dan *greedy* termasuk ke dalam algoritma dengan strategi solusi langsung. Algoritma *backtracking* dan *branch and bound* termasuk ke dalam algoritma dengan strategi berbasis pencarian pada ruang status algoritma *divide and conquer* termasuk ke dalam algoritma dengan strategi solusi atas-bawah. Algoritma *dynamic programming* termasuk ke dalam algoritma dengan strategi solusi bawah-atas.

Algoritma *greedy* memiliki kaitan dengan persoalan optimasi. Algoritma *greedy* merupakan metode yang paling popular untuk memecahkan persoalan optimasi. Secara bahasa, *greedy* memiliki arti rakus, tamak, dan sebagainya. Algoritma greedy memiliki prinsip "*take what you can get now!"*. Contoh permasalahan yang menggunakan prinsip algoritma *greedy* di antaranya adalah bagaimana cara memilih beberapa jenis investasi dalam kasus penanaman modal, bagaimana mencari jalur tersingkat dari Bandung ke Surabaya, bagaimana memilih jurusan di perguruan tinggi, dan bagaimana cara bermain kartu remi.

Integer knapsack problem merupakan suatu persoalan yang ramai diperbincangkan dalam ranah ilmu komputasi. Dalam persoalan ini diberikan n buah objek dan sebuah *knapsack* dengan kapasitas bobot K. Setiap objek memiliki properti bobot dan keuntungan. Strategi yang digunakan dalam persoalan ini adalah bagaimana cara memilih objek-objek yang akan dimasukkan ke dalam *knapsack*, sehingga total bobot yang berada di dalamnya tidak melebihi kapasitas *knapsack*, tetapi memberikan keuntungan maksimal.

Di dalam makalah ini, terdapat bentuk lain dari implementasi algoritma *greedy*, yaitu untuk menentukan produk apa saja yang perlu menjadi prioritas untuk dijual bagi para pengusaha muda dinilai dari segi modal yang dimiliki serta keuntungan yang mungkin didapatkan dari penjualan produk tersebut. Persoalan ini menggunakan algoritma *greedy* dengan pendekatan penyelesaian menggunakan *integer knapsack* dengan pengujian menggunakan bahasa Python. Dalam hal ini, kapasitas bobot *knapsack* dianalogikan dengan modal yang dimiliki oleh pengusaha. Properti bobot *(weight)* pada setiap objek dianalogikan dengan harga beli produk tersebut. Keuntungan *(profit)*Algoritma *greedy* dengan pendekatan metode *integer knapsack* dipilih dengan alasan popularitas algoritma *greedy* yang diketahui oleh orang banyak. Walaupun algoritma *greedy* ini tidak pasti menghasilkan solusi yang optimal disebabkan berbagai faktor, popularitas algoritma ini dinilai sangat mudah untuk dipahami untuk orang-orang yang masih awam dengan dunia algoritma dan pemrograman. Dengan demikian, penulis berharap pembaca dapat memahami isi makalah ini dengan baik guna dapat untuk diimplementasikan di berbagai persoalan lainnya.

II. LANDASAN TEORI

A. Persoalan Optimasi

Persoalan optimasi atau disebut juga sebagai *optimization problems* merupakan suatu persoalan yang melakukan pencarian solusi untuk mendapatkan solusi optimum. Terdapat dua jenis persoalan optimasi, yaitu maksimasi (*maximization*) dan minimasi. Solusi optimum atau dapat juga dikatakan sebagai solusi terbaik adalah solusi yang bernilai minimum atau maksimum dan merupakan hasil dari sekumpulan alternatif solusi yang mungkin terjadi. Dalam persoalan optimasi, terdapat dua elemen, yaitu kendala atau yang biasa dikenal dengan *constraints* dan fungsi objektif yang dikenal dengan fungsi optimasi. Solusi yang dinilai memenuhi semua kendala disebut dengan solusi layak (*feasible solution*). Solusi layak yang dapat mengoptimumkan suatu fungsi optimasi disebut solusi optimum.

B. Definisi Algoritma Greedy

Algoritma *greedy* dapat dikatakan sebagai metode penyelesaian yang paling populer dalam kasus pemecahan persoalan optimasi. Dalam praktiknya, algoritma *greedy* membentuk solusi langkah per langkah (*step by step*). Pada saat pengerjaannya, terdapat banyak pilihan solusi yang perlu untuk dieksplorasi di setiap langkah solusi. Maka dari itu, pada setiap langkah perlu dibuat suatu keputusan yang terbaik di setiap penentuan pilihannya. Pada algoritma *greedy*, keputusan yang telah diambil di dalam suatu langkah tidak dapat diubah kembali di dalam langkah berikutnya.

Dalam algoritma *greedy*, pendekatan yang digunakan adalah membuat pilihan yang dinilai merupakan pilihan yang memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (*local optimum*) di dalam setiap langkah dengan harapan bahwa setiap langkah sisa tersebut dapat mengarah ke solusi optimum global *(global optimum*).

C. Skema Umum Algoritma Greedy

Algoritma *greedy* disusun oleh elemen-elemen berikut.

1. Himpunan kandidat

   Himpunan kandidat merupakan himpunan yang berisi elemen-elemen pembentuk solusi.

2. Himpunan solusi

   Himpunan solusi merupakan himpunan yang berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

3. Fungsi seleksi (*selection function*)

   Fungsi seleksi merupakan fungsi yang berfungsi untuk memilih kandidat yang dinilai paling memungkinkan untuk mencapai solusi optimal. Kandidat yang sudah dipilih di dalam suatu langkah tidak akan pernah dipertimbangkan kembali pada langkah berikutnya

4. Fungsi kelayakan (*feasible*)

   Fungsi kelayakan merupakan fungsi yang memeriksa apakah suatu kandidat yang telah dipilih tersebut dinilai dapat memberikan solusi yang layak, yaitu kandidat dan himpunan solusi yang terbentuk tersebut tidak melanggar kendala yang ada. Kandidat yang dinilai layak selanjutnya dimasukkan ke dalam himpunan solusi. Kandidat yang tidak layak dapat dibuang dan tidak dapat dipertimbangkan kembali kemudian.

5. Fungsi objektif

   Fungsi objektif merupakan fungsi yang memaksimumkan atau meminimumkan suatu nilai solusi.

D. Persoalan Integer Knapsack

Algoritma *greedy* dapat diimplementasikan di dalam kasus penyelesaian permasalahan *integer knapsack.* Kasus ini sudah tidak asing di dalam ranah informatika. Di dalam persoalan *integer knapsack* diberikan n buah objek dan sebuah *knapsack* (karung, tas, tansel, dan sebagainya) dengan kapasitas bobot K. Setiap objek memiliki properti bobot (*weight*) w_i dan keuntungan (*profit*) p_i. Di dalam persoalan ini, akan dicari bagaimana cara memilih objek-objek untuk dimasukkan ke dalam *knapsack* sedemikian sehingga total bobot yang berada di dalamnya tidak melebihi kapasitas *knapsack,* tetapi dapat memberikan keuntungan yang maksimal.

Di dalam persoalan *integer knapsack*, terdapat maksimasi dengan persamaan sebagai berikut.

$$F = \sum_{i=1}^{n} p_i x_i$$

Selain itu, terdapat kendala (*constraint*) dengan persamaan sebagai berikut.

$$\sum_{i=1}^{n} w_i x_i \leq W$$

Dalam hal ini, $x_i = 0$ atau $x_i = 1$ dan $i = 1, 2, 3, \ldots, n$

Penerapan algoritma *greedy* di dalam *integer knapsack* diawali dengan memasukkan objek satu per satu ke dalam *knapsack.* Apabila objek sudah dimasukkan ke dalam *knapsack*, objek tersebut tidak dapat dikeluarkan kembali. Oleh karena itu, algoritma *greedy* tidak menguji seluruh kasus seperti halnya di dalam algoritma *brute force.*

Dalam kasus *integer knapsack*, terdapat beberapa strategi *greedy* yang heuristik yang dapat digunakan untuk memilih objek yang akan dimasukkan ke dalam *knapsack.* Strategi heuristik tersebut adalah sebagai berikut.

1. *Greedy by profit*

   Pada heuristik ini, setiap langkah *knapsack* diisi dengan objek yang memiliki keuntungan terbesar. Strategi ini memiliki taktik untuk memaksimumkan keuntungan dengan memilih objek yang paling menguntungkan terlebih dahulu. Dengan memprioritaskan keuntungan, diharapkan hasil yang diperoleh di dalam solusi akan menghasilkan keuntungan yang besar guna mencapai fungsi objektif yang telah dideklarasikan di awal, yaitu memeroleh keuntungan yang maksimal.

2. *Greedy by weight*

   Pada heuristik ini, setiap langkah *knapsack* diisi dengan objek yang memiliki berat paling ringan. Strategi ini memiliki taktik untuk memaksimumkan keuntungan dengan memasukkan sebanyak mungkin objek ke dalam *knapsack.*

3. *Greedy by density*

   Pada heuristik ini, setiap langkah *knapsack* diisi dengan objek yang memiliki densitas sebesar p_i/w_i terbesar. Strategi ini memilki taktik untuk memaksimumkan keuntungan dengan memilih objek yang mempunyai keuntungan per unit berat terbesar.

Namun demikian, pemilihan objek berdasarkan satu dari ketiga strategi di atas tidak pasti menjamin bahwa strategi tersebut akan memberikan solusi yang optimal. Ada kemungkinan bahwa ketiga strategi tersebut tidak memberikan solusi optimum. Hal ini disebabkan *greedy* tidak bisa membatalkan pengujian atau melakukan pengujian yang sudah terlewat sebelumnya.

E. Profit dalam Dunia Bisnis

Dilansir dari *Kompas.com*, profit merupakan sesuatu yang banyak dicari pengusaha, pedagang, hingga investor. Menurut Kamus Besar Bahasa Indonesia, profit adalah untung, keuntungan, atau manfaat. Profit juga kerap dikenal dengan istilah laba.

Profit dibagi menjadi tiga jenis, yaitu *gross profit* atau laba kotor, *operating profit* atau laba operasional, dan *net profit* atau laba bersih. Profit merupakan hal yang penting dalam menjalankan bisnis. Mendapatkan profit sebanyak-banyaknya menjadi tujuan utama perusahaan. Profit juga merupakan modal yang dapat digunakan perusahaan untuk berbagai tujuan. Dengan memeroleh profit yang tinggi, perusahaan dapat terus berkembang.

III. Aplikasi Algoritma *Greedy* pada Sistem Prediksi Daftar Produk Penjualan untuk Mendapatkan Keuntungan Maksimum

A. Implementasi Algoritma Greedy pada Sistem Prediksi Daftar Produk Penjualan untuk Mendapatkan Keuntungan Maksimum

Algoritma *greedy* dapat diimplementasikan dalam persoalan mencari daftar produk penjualan agar suatu perusahaan tersebut mendapatkan keuntungan maksimum. Dalam hal ini, digunakan pendekatan persoalan *integer knapsack.* Suatu produk tentunya memiliki harga beli tersendiri atau dikenal dengan istilah *buying price*. Dengan pendekatan *integer knapsack*, *buying price* diibaratkan dengan bobot dari suatu objek. Suatu produk juga tentu memiliki harga jual. Selisih harga jual dengan harga beli disebut dengan keuntungan atau *profit.* Dengan pendekatan *integer knapsack*, *profit* yang merupakan hasil selisih dari harga jual dengan harga beli tersebut diibaratkan dengan *profit* objek pada *integer knapsack.*

B. Skema Umum Algoritma Greedy pada Sistem Prediksi Daftar Produk Penjualan untuk Mendapatkan Keuntungan Maksimum

Berikut merupakan skema umum algoritma *greedy* di dalam persoalan sistem prediksi daftar produk penjualan untuk mendapatkan keuntungan maksimum.

1. Himpunan kandidat

   Di dalam persoalan ini, himpunan kandidat adalah himpunan objek dengan kondisi objek-objek tersebut dapat dimasukkan ke dalam *knapsack* dengan tidak melanggar *constraint*. Pada persoalan ini, kapasitas digambarkan dengan modal atau *capital*, bobot setiap objek digambarkan dengan harga beli atau *buying price*, serta keuntungan setiap objek digambarkan dengan *profit*, yang merupakan hasil dari harga jual dikurang dengan harga beli (*selling price – buying price*).

2. Himpunan solusi

   Di dalam persoalan ini, himpunan solusi adalah total bobot yang terdapat di dalam *knapsack* tidak melebihi kapasitasnya. Pada persoalan ini, kapasitas digambarkan dengan modal atau *capital* yang dimiliki oleh seseorang dan total bobot adalah total pengeluaran yang diakumulasikan pada saat pembelian produk.

3. Fungsi seleksi (*selection function*)

   Di dalam persoalan ini, fungsi seleksi adalah dengan menggunakan prinsip heuristik yang dibagi menjadi tiga hal, yaitu *greedy by profit, greedy by weight,* dan *greedy by density.* Untuk *greedy by profit*, dipilih produk dengan keuntungan yang bernilai tertinggi dari himpunan kandidat yang tersisa. Untuk *greedy by weight*, dipilih produk dengan bobot yang bernilai terendah dari himpunan kandidat yang tersisa. Untuk *greedy by density*, dipilih produk dengan densitas sebesar keuntungan dibagi modal terbesar dari himpunan kandidat yang tersisa.

4. Fungsi kelayakan (*feasible*)

   Di dalam persoalan ini, fungsi kelayakan bertugas untuk memeriksa apakah nilai total dari himpunan harga beli produk yang dipilih tidak melebihi jumlah modal yang tersedia.

5. Fungsi objektif

   Di dalam persoalan ini, fungsi objektif adalah total keuntungan yang dihasilkan selama kegiatan operasional produk dapat bernilai maksimum.

C. Source Code

Berikut dilampirkan implementasi dari algoritma *greedy* pada sistem prediksi daftar produk penjualan untuk mendapatkan keuntungan maksimum dengan menggunakan bahasa pemrograman Python.

```
from operator import itemgetter
import os
```

```
# utilitas untuk mencetak seluruh hasil rekomendasi produk
def printresult(arrayofresult):
    display = []
    cnt = 0
    if(len(arrayofresult) >= 0):
        display.append([1, arrayofresult[0][0], 
arrayofresult[0][1], arrayofresult[0][2]])
        for i in range(1, len(arrayofresult)):
            if(arrayofresult[i] == arrayofresult[i-1]):
                display[len(display)-1][0] += 1
            else:
                display.append([1, arrayofresult[i][0], 
arrayofresult[i][1], arrayofresult[i][2]])

    else:
        print("There's no product you can sell")

    print("Result (format: Qty - product's name - buying 
price - selling price)")
    for i in display:
        print(str(i[0])+"x"+" - "+str(i[1])+" - " +
            str('Rp{:,}'.format(i[2]).replace(",", "."))+" - " +
str('Rp{:,}'.format(i[2]+i[3]).replace(",", ".")))

# utilitas untuk menghitung jumlah profit pada setiap 
strategi yang digunakan
def sum_of_profit(arrayofresult):
    sum = 0
    for i in range(len(arrayofresult)):
        sum += arrayofresult[i][2]
    return sum

# Inisialisasi Array Kosong
os.system("cls")
array_of_product = []
greedy_by_profit = []
greedy_by_weight = []
greedy_by_density = []
knapsack = []

print("Welcome to Find Your Products App")
print("You can input your products below and we'll give 
you some recommendations")
print("")
```

```
# Modal atau capital digunakan sebagai penanda kapasitas 
pada integer knapsack
modal = int(input("Input your capital: "))
print("")

user_input = input("Do you want to continue? (Y/N): ")

while (user_input == 'Y'):
    product_name = input(f"Input your product's name: ")
    # product_buy_price digunakan sebagai penanda weight 
pada integer knapsack
    product_buy_price = int(input(f"Input your product's 
buying price: "))
    product_quantity = int(input(f"Input your product's 
quantity: "))
    product_sell_price = int(input(f"Input your product's 
selling price: "))
    # product_profit digunakan sebagai penanda profit pada 
integer knapsack
    product_profit = product_sell_price - product_buy_price

    array_of_product.append([len(array_of_product)+1, 
product_name, product_buy_price, product_quantity, 
product_sell_price, product_profit])
    print("")

    user_input = input("Do you want to continue? (Y/N): ")

print("")

# i[0]: id, i[1]: name, i[2]: buying price, i[3]: quantity, i[4]: 
selling price, i[5]: profit
for i in array_of_product:
    for j in range(i[3]):
        # append name, buying price (as weight), profit (as 
profit), and i[5]/i[2] as density
        knapsack.append([i[1], i[2], i[5], float(i[5]/i[2])])

# heuristik: greedy berdasarkan weight, profit, dan density

# weight terurut membesar
knapsack_sorted_by_weight = sorted(knapsack, 
key=itemgetter(1))
# profit terururut mengecil
knapsack_sorted_by_profit = sorted(knapsack, 
key=itemgetter(2), reverse=True)
```

```
# density terurut mengecil
knapsack_sorted_by_density = sorted(knapsack,
key=itemgetter(3), reverse=True)

# greedy by profit
loop = 0
total = 0
while (loop < len(knapsack_sorted_by_profit)):
   if(total + knapsack_sorted_by_profit[loop][1] <= modal):
      # append digunakan untuk mengganti nilai 0 dan 1
pada objek
      greedy_by_profit.append(knapsack_sorted_by_profit[l
oop])
      total += knapsack_sorted_by_profit[loop][1]
   loop += 1
# greedy by weight
loop = 0
total = 0
while (loop < len(knapsack_sorted_by_weight)):
   if(total + knapsack_sorted_by_weight[loop][1] <=
modal):
      # append digunakan untuk mengganti nilai 0 dan 1
pada objek
      greedy_by_weight.append(knapsack_sorted_by_weigh
t[loop])
      total += knapsack_sorted_by_weight[loop][1]
   loop += 1

# greedy by density
loop = 0
total = 0
while (loop < len(knapsack_sorted_by_density)):
   if(total + knapsack_sorted_by_density[loop][1] <=
modal):
      # append digunakan untuk mengganti nilai 0 dan 1
pada objek
      greedy_by_density.append(knapsack_sorted_by_densi
ty[loop])
      total += knapsack_sorted_by_density[loop][1]
   loop += 1

print("1. Greedy By Profit")
printresult(greedy_by_profit)
print("Total profit =", str('Rp{:,}'.format(
   sum_of_profit(greedy_by_profit)).replace(",", ".")))
print("")
```

```
print("2. Greedy By Weight")
printresult(greedy_by_weight)
print("Total profit =", str('Rp{:,}'.format(
   sum_of_profit(greedy_by_weight)).replace(",", ".")))
print("")

print("3. Greedy By Density")
printresult(greedy_by_density)
print("Total profit =", str('Rp{:,}'.format(
   sum_of_profit(greedy_by_density)).replace(",", ".")))
print("")
```

Pada akhir program akan diperoleh tiga buah larik, yaitu larik greedy_by_weight yang berisi himpunan solusi yang diperoleh dari tiga strategi heuristik, yaitu *greedy by weight*, *greedy by profit*, dan *greedy by density*. Dari ketiga strategi tersebut, dipilih strategi yang menghasilkan keuntungan paling maksimum. Meskipun demikian, hasil yang diperoleh tidak pasti menentukan bahwa solusi yang didapatkan merupakan solusi yang paling optimum.

D. Implementasi Input dan Output Program

Berikut merupakan salah satu contoh implementasi dari *input* dan *output* program. Cara menjalankannya adalah dengan memindahkan direktori menuju direktori yang berisi program dengan nama *file* main.py. Setelah itu, dapat dilakukan *running* dengan mengetikkan python main.py pada *command line.*

```
Welcome to Find Your Products App
You can input your products below and we'll give you some recommendations

Input your capital: 30000

Do you want to continue? (Y/N): Y
Input your product's name: marina UV
Input your product's buying price: 12000
Input your product's quantity: 2
Input your product's selling price: 24000

Do you want to continue? (Y/N): Y
Input your product's name: wardah
Input your product's buying price: 18000
Input your product's quantity: 2
Input your product's selling price: 36000

Do you want to continue? (Y/N): N
```

1. Greedy By Profit

Result (format: Qty - product's name - buying price - selling price)

1x - wardah - Rp18.000 - Rp36.000

1x - marina UV - Rp12.000 - Rp24.000

Total profit = Rp30.000

2. Greedy By Weight

Result (format: Qty - product's name - buying price - selling price)

2x - marina UV - Rp12.000 - Rp24.000

Total profit = Rp24.000

3. Greedy By Density

Result (format: Qty - product's name - buying price - selling price)

2x - marina UV - Rp12.000 - Rp24.000

Total profit = Rp24.000

E. Analisis Kompleksitas Algoritma

Dengan mengabaikan waktu pengurutan objek kompleksitas waktu asimptotik untuk algoritma *greedy* dalam persoalan ini adalah O(n). Oleh karena itu, dapat dikatakan bahwa algoritma ini efisien di luar perhitungan cara pengurutannya. Dengan demikian, jumlah persoalan berbanding lurus dengan kompleksitasnya.

IV. KESIMPULAN

Algoritma *greedy* merupakan algoritma yang populer di dalam menyelesaikan persoalan optimasi. Salah satu implementasi dari algoritma *greedy* adalah menyelesaikan persoalan *integer knapsack* menggunakan tiga jenis heuristik, yaitu *greedy by profit, greedy by weight,* dan *greedy by density.* Di dalam makalah ini, diimplementasikan suatu penyelesaian persoalan berupa sistem prediksi daftar produk penjualan untuk mendapatkan keuntungan maksimum menggunakan algoritma *greedy.* Meskipun algoritma *greedy* ini sangat populer, kita tidak dapat memastikan apakah himpunan solusi yang merupakan hasil dari penyelesaian algoritma ini dapat dikatakan optimum atau tidak. Hal ini disebabkan *greedy* tidak dapat melakukan pengujian yang sudah terlewat dan *greedy* juga tidak dapat menghapus pengujian yang sudah diuji. Untuk solusinya, dapat digunakan algoritma lain yang bersifat mangkus dan melibatkan pengujian seluruh kasus, seperti halnya *branch and bound.* Namun demikian, tidak ada salahnya kita menggunakan algoritma *greedy* karena perincian dari algoritma ini sangat detail, sehingga mudah untuk dipahami.

V. UCAPAN TERIMA KASIH

Puji dan syukur mari kita panjatkan kepada Allah SWT, Tuhan semesta alam karena atas berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan makalah ini dengan tepat waktu guna memenuhi salah satu tugas mata kuliah Strategi Algoritma tahun ajaran 2021/2022. Selain itu, penulis juga mengucapkan terima kasih sebesar-besarnya kepada Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc., Ibu Dr. Masayu Leylia Khodra, S.T., M.T., dan Bapak Dr. Ir. Rinaldi Munir, M.T., selaku dosen pengajar mata kuliah Strategi Algoritma, yang telah membimbing penulis dalam mengerjakan makalah ini. Semoga kebaikan Bapak dan Ibu dapat dibalas oleh Allah SWT dengan sebaik-baiknya balasan. Terima kasih banyak juga kepada orang tua, keluarga, dan sahabat penulis, yaitu Bayu Samudra atas dukungan dan motivasi yang selalu diberikan kepada penulis. Penulis juga meminta maaf apabila dalam kepenulisan makalah ini, penulis memiliki banyak kesalahan karena sejatinya penulis adalah pembelajar yang masih harus terus belajar. Semoga makalah ini dapat menjadi manfaat untuk orang banyak, tidak hanya untuk mahasiswa informatika saja, tetapi juga untuk masyarakat umum. Harapan dari penulis, makalah ini dapat meningkatkan kecintaan banyak orang terhadap algoritma karena sejatinya algoritma adalah ilmu yang sangat penting dalam hidup ini.

TAUTAN VIDEO PADA YOUTUBE

https://youtu.be/P3I8JaMtRK8

TAUTAN REPOSITORY PADA GITHUB

https://github.com/hanafathiyah/Find-Your-Products.git

REFERENSI

[1] R. Munir. 2004. *Algoritma Greedy.* Bandung.

[2] Pratama, A. M. (2022, January 17). *Apa Itu Profit dan kenapa profit penting untuk bisnis? Halaman all*. KOMPAS.com. Diakses pada 22 Mei, 2022, dari https://money.kompas.com/read/2022/01/17/151000726/apa-itu-profit-dan-kenapa-profit-penting-untuk-bisnis-?page=all

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, 22 Mei 2022

Hana Fathiyah 13520047