# Penerapan Algoritma Greedy dan Pattern Matching dalam Sistem Rekomendasi Menu Rumah Makan

Kevin John Wesley Hutabarat - 13521042
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521042@std.stei.itb.ac.id

***Abstrak*— Makanan merupakan suatu aspek yang krusial dalam hidup. Umat manusia kerap kali merayakan suatu pencapaian dengan makan bersama. Dewasa ini, banyak sekali beredar rumah makan dengan hidangan yang beraneka ragam. Dalam makalah ini, akan dibahas suatu sistem yang dapat memberikan rekomendasi menu di suatu rumah makan dengan memanfaatkan algoritma greedy dan pattern matching'.**

***Keywords—rumah makan, rekomendasi., greedy, pattern matching***

## I. PENDAHULUAN

Makanan adalah satu hal yang tidak dapat dipisahkan dari manusia. Saat ini, umat manusia semakin banyak dan akan terus berkembang biak. Permintaan terhadap makanan juga semakin meningkat, sehingga usaha di bidang kuliner terbilang menjanjikan dan banyak rumah makan yang dibuka saat ini. Rumah makan pun banyak jenisnya, misalnya rumah makan padang, sunda, dan ada juga rumah makan yang menyajikan hidangan dengan bahan dasar tertentu.

Pada zaman sekarang, makanan bukan hanya dijadikan sebagai kebutuhan pokok, tetapi juga sebagai gaya hidup dan simbol dari suatu pencapaian. Orang-orang seringkali merayakan suatu pencapaian ataupun hari besar dengan makan bersama. Menyajikan makanan untuk banyak orang tentunya akan sangat merepotkan bagi beberapa orang, sehingga rumah makan dapat menjadi alternatif yang tepat.

Tetapi, hidangan yang disajikan di rumah makan belum tentu sesuai dengan selera pelanggannya. Biasanya pelanggan akan bertanya kepada pekerja restoran tentang rekomendasi di restoran tersebut. Tetapi hal ini tentu akan sangat merepotkan dan tidak efisien bila banyak pelanggan yang bertanya. Ditambah lagi selera orang berbeda-beda, jadi akan cukup sulit untuk para pekerja merekomendasikan makanan kepada pelanggan. Sebagai tambahan, bahan makanan yang biasanya bersifat rahasia kerap kali menjadi masalah bagi pelanggan karena tidak tahu apakah makanan tersebut dapat dimakan olehnya. Bisa saja makanan tersebut mengandung bahan yang memicu alergi pelanggan, atau dilarang oleh kepercayaan yang dianut pelanggan. Budget yang dimiliki juga menjadi pertimbangan pelanggan dalam memilih menu.

Oleh karena itu, alangkah baiknya bila ada suatu sistem yang dapat merekomendasikan menu kepada pelanggan berdasarkan data yang dimiliki oleh restoran dan syarat-syarat yang diberikan oleh pelanggan. Dengan sistem ini, permintaan dari pelanggan bisa dipenuhi dan rahasia dari rumah makan juga dapat dijaga. Dari sini penulis memiliki ide untuk menyusun makalah yang berjudul Penerapan Algoritma *Greedy* dan *Pattern Matching* dalam Sistem Rekomendasi Menu Rumah Makan.

## II. DASAR TEORI

### A. Algoritma Greedy

Algoritma *Greedy* adalah algoritma yang memecahkan persoalan berdasarkan langkah demi langkah. Pada setiap langkah, algoritma *greedy* berusaha untuk selalu mengambil pilihan terbaik yang dapat diperoleh dari semua kemungkinan pada langkah itu tanpa memikirkan langkah selanjutnya. Algoritma *greedy* memiliki prinsip "*take what you can take now!*". Algoritma ini hanya dapat memperoleh optimum lokal, dan dari memilih optimum lokal tersebut diharapkan akan berakhir dengan optimum global.

Algoritma *greedy* adalah salah satu algoritma yang paling banyak digunakan karena tergolong sederhana untuk memecahkan persoalan optimasi. Persoalan optimasi adalah persoalan untuk mencari solusi yang paling optimal, dapat berupa maksimasi dan minimasi. Algoritma *greedy* dapat digunakan untuk persoalan yang solusi terbaik mutlaknya tidak terlalu diperlukan. Algoritma greedy dapat menghasilkan solusi hampiran yang waktu prosesnya lebih singkat daripada menggunakan algoritma dengan kebutuhan waktu eksponensial. Agar menghasilkan solusi yang optimal, kita harus memilih fungsi seleksi yang tepat.

Algoritma *greedy* dapat digunakan untuk persoalan optimasi, yaitu memaksimalkan dan meminimalkan suatu hal. Untuk menyelesaikan persoalan dengan algoritma *greedy*, persoalan perlu dipetakan menjadi beberapa elemen agar lebih mudah diimplementasikan. Algoritma *greedy* memiliki beberapa elemen, yaitu:

1. Himpunan kandidat (C): himpunan ini berisi kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi (S): himpunan yang berisi kandidat yang sudah terpilih

3. Fungsi solusi: fungsi yang menentukan himpunan kandidat yang dipilih sudah memberikan solusi atau belum
4. Fungsi seleksi (*selection function*): fungsi yang digunakan untuk memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristic
5. Fungsi kelayakan (*feasible*): fungsi yang digunakan untuk memeriksa kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi atau belum
6. Fungsi obyektif: memaksimumkan atau meminimumkan

Berikut ini adalah skema umum algoritma *greedy*:

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← {}      { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)      { pilih sebuah kandidat dari C}
      C ← C – {x}          { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
          S ← S ∪ {x}      { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then     { solusi sudah lengkap }
     return S
  else
     write('tidak ada solusi')
  endif
```

**Gambar 2.1** Skema Umum Algoritma *Greedy*
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

## B. *Pattern Matching*

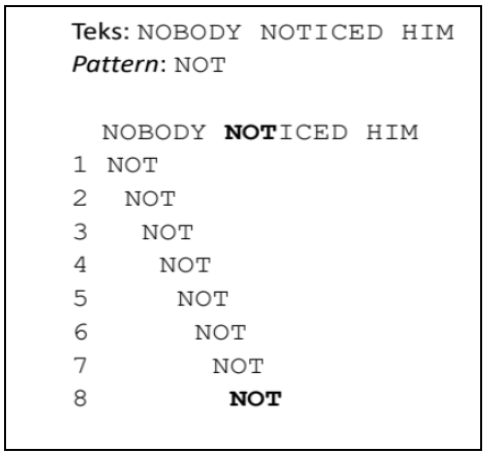
*Pattern matching* adalah sebuah metode untuk mencari kemunculan suatu pola berupa *string* pada suatu teks yang diberikan. Pada persoalan *pattern matching*, diberikan sebuah teks, yaitu *string* yang ukurannya n karakter, serta *pattern*, yaitu *string* dengan panjang m karakter yang akan dicari dalam teks. Dalam persoalan ini, m selalu diasumsikan lebih kecil dari n. Jika tidak, pola tidak akan ditemukan dalam pattern. *Pattern matching* banyak diaplikasikan dalam fungsionalitas-fungsionalitas sehari-hari, seperti pencarian di dalam editor teks, *web search engine*, analisis citra, bahkan di bioinformatics untuk pencocokan rantai asam amino pada rantai DNA.

Dalam *pattern* matching, terdapat istilah *prefix* dan *suffix*. Asumsikan terdapat S yang berupa string dengan panjang m (S = x0x1…xm-1). *Prefix* dari S adalah *substring* dari S[0…k], sedangkan suffix dari S adalah *substring* dari S[k…m-1], dengan k adalah index apapun yang berada di antara 0 dan m-1. Sebagai contoh, jika S adalah string "informatika", maka *prefix* yang memungkinkan dari S adalah 'i', 'in', 'inf', 'info', 'inform', 'informa', 'informat', 'informati', 'informatik', dan 'informatika'. Sementara itu, suffix yang memungkinkan adalah 'a', 'ka', 'ika', 'tika', 'atika', 'matika', 'rmatika', 'ormatika', 'formatika', 'nformatika', dan 'informatika'.

Terdapat beberapa algoritma yang dapat digunakan untuk melakukan *pattern matching*, antara lain:

1. Algoritma *Brute Force*

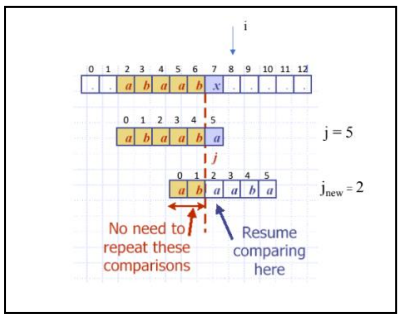
Penyelesaian *pattern matching* dengan *brute force* dilakukan dengan memeriksa karakter per karakter *pattern* terhadap teks. Pencocokan dimulai dari awal teks dan pattern. Jika terjadi *mismatch*, *pattern* akan digeser ke kanan satu karakter, dan pengecekan diulang lagi dari karakter pertama pada *pattern* terhadap karakter pada teks yang bersesuaian.




**Gambar 2.2** Proses *Pattern Matching* dengan Algoritma *Brute Force*
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

2. Algoritma Knuth-Morris-Pratt

KMP adalah algoritma *pattern matching* yang dilakukan dengan cara mencari *pattern* dalam teks dari kanan ke kiri, tetapi dengan pergeseran yang lebih optimal daripada *brute force*. KMP memanfaatkan *prefix* dan *suffix* dari *string*. Jika terjadi *mismatch* antara teks dan *pattern* di P[j], pergeseran dilakukan dengan memerhatikan *prefix* dan *suffix* terbesar yang sama dari pattern yang sudah *match* sebelumnya. Pengecekan dilakukan dari tempat *mismatch*, sehingga pengecekan untuk *string* tersebut tidak perlu dilakukan secara berulang.




**Gambar 2.3** Ilustrasi Pergeseran *Pattern* dalam KMP
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/ 2020-2021/Pencocokan-string-2021.pdf

Terdapat satu lagi algoritma yang digunakan untuk persoalan *pattern matching*, yaitu algoritma Boyer Moore, yang akan dibahas pada poin selanjutnya.

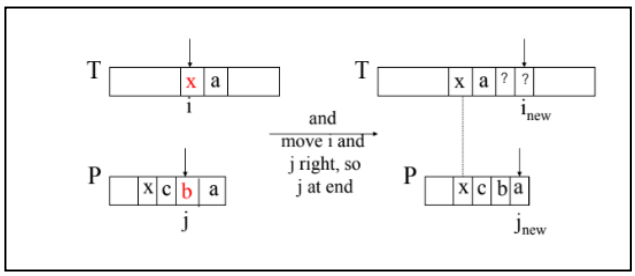
## C. *Algoritma Boyer Moore*

Algoritma Boyer Moore adalah algoritma yang menyelesaikan *pattern matching* dengan berdasar pada dua

teknik, yaitu teknik *looking-glass*, yaitu mencari *pattern* dalam teks dengan bergerak ke belakang sepanjang *pattern*, mulai dari bagian belakang *pattern*. Teknik yang kedua adalah *character-jump*, yaitu ketika ada *mismatch* terjadi pada T[i] == x, karakter di pattern P[j] tidak sama dengan T[i].

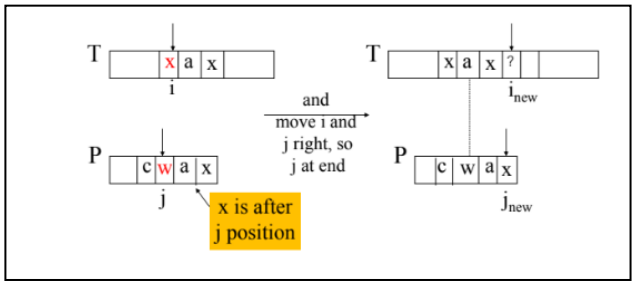
Terdapat tiga kasus yang mungkin dalam algoritma ini, yaitu:

1. Kasus 1 terjadi jika *pattern* mengandung x di dalamnya. Jika kasus ini terjadi, algoritma akan menggeser *pattern* sehingga kemunculan terakhir dari x bisa sejajar dengan T[i].



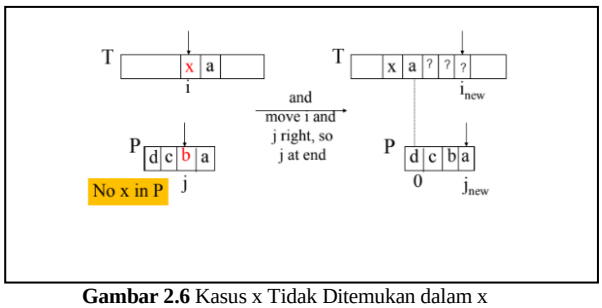

**Gambar 2.4** Kasus *Pattern* Mengandung x
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/ 2020-2021/Pencocokan-string-2021.pdf

2. Kasus 2 terjadi jika *pattern* mengandung x di dalamnya, tetapi pergeseran ke kemunculan terakhir x tidak mungkin (misalkan letak kemunculan x terakhir sudah diperiksa sebelumnya). Dalam kasus ini, *pattern* akan digeser satu karakter ke T[i+1]



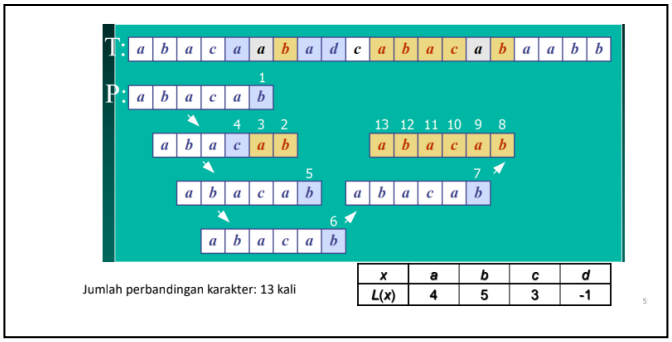

**Gambar 2.5** Kasus Kemunculan x Terakhir Sudah Dilewati
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/ 2020-2021/Pencocokan-string-2021.pdf

3. Kasus 3 terjadi ketika kasus 1 dan 2 tidak terpenuhi. Dalam kasus ini, indeks 0 pada *pattern* akan disejajarkan dengan indeks i+1 pada teks




**Gambar 2.6** Kasus x Tidak Ditemukan dalam x
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/ 2020-2021/Pencocokan-string-2021.pdf

Dalam algoritma ini, terdapat fungsi untuk mencari kemunculan terakhir (*last occurrence function*). Pada awal algoritma ini dijalankan, akan dibentuk sebuah tabel terlebih dahulu, yang mengandung indeks kemunculan terakhir untuk setiap karakter pada *pattern*.




**Gambar 2.7** Contoh Penyelesaian *String Matching* dengan Algoritma Boyer Moore
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/ 2020-2021/Pencocokan-string-2021.pdf

Algoritma Boyer Moore akan cepat apabila variasi dari alfabet dalam teks dan *pattern* cukup banyak, sedangkan akan lambat jika variasinya hanya sedikit. Algoritma Boyer Moore akan baik untuk teks dalam bahasa Inggris, tetapi akan buruk untuk bahasa binary. Kompleksitas waktu Boyer Moore untuk kasus terburuk nya adalah O(nm+A).

## III. Implementasi dan Pembahasan

### A. *Analisis Elemen yang Perlu Diperhatikan*

Dalam memilih menu di rumah makan, setiap orang memiliki pertimbangan tertentu. Berikut ini adalah beberapa aspek yang mungkin untuk dipertimbangkan dalam memilih makanan.

1. Harga

   Sebelum memesan makanan, biasanya orang-orang langsung tertuju kepada harganya. Memilih makanan sesuai budget adalah hal yang penting. Orang-orang akan lebih suka untuk memilih makanan yang murah. Apalagi bila makanan tersebut memiliki potongan harga, tentu saja menjadi hal yang menggiurkan bagi pelanggan

2. Rasa

   Hal yang terpenting dari sebuah makanan adalah rasanya. Rasa dari makanan dapat mengubah perasaan dari seseorang. Makanan yang lezat akan membuat *mood* yang semulanya tidak baik-baik saja menjadi lebih baik. Tetapi, rasa makanan subjektif, yang artinya berbeda-beda untuk setiap orang.

3. Popularitas

   Pada zaman sekarang, orang-orang cenderung mengikuti apa yang banyak dibicarakan oleh orang lain, termasuk dalam pemilihan makanan. Orang-orang beranggapan bahwa makanan yang populer adalah makanan yang enak dan patut untuk dicoba. Oleh

karena itu, dalam memilih makanan orang-orang kerap kali melihat makanan tersebut banyak dibicarakan atau tidak.

4. Kalori

   Dewasa ini, orang-orang memilih makanan bukan hanya untuk sekadar makan. Memilih makan berpengaruh kepada kesehatan. Banyak orang yang memperhatikan bentuk tubuhnya. Untuk menjaga bentuk tubuh yang ideal, biasanya orang-orang memerhatikan kalori dari makanan yang akan dimakannya. Untuk orang yang sedang menjalani program diet, biasanya akan memilih makanan dengan kalori yang rendah.

5. Penampilan

   Penampilan makanan juga merupakan salah satu aspek yang penting dari makanan. Rumah makan biasanya menyertakan foto makanan pada menunya, agar bisa menggambarkan makanan seperti apa yang nantinya akan diperoleh pelanggan. Penampilan yang menarik akan menggugah selera pelanggan. Sebaliknya, penampilan yang buruk membuat pelanggan enggan memilih makanan tersebut. Walaupun penampilan yang baik tidak menjamin makanan tersebut enak, tetapi makanan yang bisa dinikmati adalah makanan dengan penampilan yang baik.

6. Bahan Makanan yang Dihindari

   Semua orang tentu memiliki bahan makanan yang dihindari. Bahan tersebut dihindari bisa karena beberapa penyebab, seperti alergi, larangan dari kepercayaan yang dianut, atau hanya berdasarkan preferensi pribadi yang tidak menyukai bahan tersebut.

Sayangnya, aspek di atas tidak semuanya bisa diukur (objektif). Untuk mengimplementasi algoritma *greedy*, semua aspek diatas haruslah diubah ke bentuk yang dapat diukur. Berikut ini adalah pendekatan yang digunakan untuk masing-masing aspek.

1. Harga

   Harga merupakan sesuatu yang dapat diukur, jadi dalam algoritma *greedy* dapat langsung menggunakan harga dari tiap makanan.

2. Rasa dan Penampilan

   Rasa dan penampilan merupakan sesuatu yang sulit diukur karena akan berbeda untuk setiap orang. Pendekatan yang digunakan disini adalah *rating*. *Rating* adalah penilaian dari pelanggan yang sudah mencoba makanan tersebut. *Rating* yang diberikan pelanggan biasanya berupa *integer* yang berada di rentang angka 1-5. Semakin besar ratingnya menandakan bahwa makanan tersebut enak menurut sebagian besar orang.

3. Popularitas

   Popularitas dapat diukur dengan pendekatan seberapa sering makanan tersebut dipesan. Semakin sering pesanan tersebut dipesan menandakan bahwa makanan tersebut populer.

4. Kalori

   Kalori adalah sesuatu yang dapat diukur, jadi dapat langsung digunakan dalam algoritma *greedy*.

5. Bahan yang dihindari

   Akan sulit menerapkan bahan yang dihindari sebagai parameter dalam algoritma *greedy*. Jadi, bahan yang dihindari digunakan sebagai fungsi kelayakan dalam program ini. Jika makanan mengandung bahan yang dihindari, program tidak akan memasukkan makanan tersebut ke dalam rekomendasi.

### *B. Dekomposisi Persoalan*

Persoalan rekomendasi makanan ini dapat dipetakan menjadi persoalan optimasi. Persoalan dapat dibagi menjadi dua, yaitu persoalan memilih makanan sebanyak-banyaknya dengan *budget* yang terbatas, atau memilih sejumlah makanan yang sudah ditetapkan sebelumnya.

1. Himpunan Kandidat (C)

   Himpunan kandidat berisi menu-menu yang disediakan oleh rumah makan. Menu-menu ini berisikan nama makanan, bahan yang digunakan untuk membuat makanan tersebut, harga makanan, *rating*, kalori dari makanan dan jumlah makanan yang sudah terjual.

2. Himpunan Solusi (S)

   Himpunan solusi berisi makanan yang disarankan. Makanan ini diperoleh dari algoritma *greedy*, yang pemilihannya berdasarkan kriteria tertentu, sesuai dengan masukan yang diterima program.

3. Fungsi solusi

   Algoritma greedy dapat dikatakan sudah mencapai solusi apabila sudah memenuhi fungsi solusi. Fungsi solusi dalam makalah ini ada dua, tergantung masukan pengguna. Apabila pengguna memilih opsi untuk memilih makanan sebanyak-banyaknya sesuai *budget*, fungsi solusinya adalah total harga tidak lebih besar dari *budget.* Jika pengguna memilih opsi untuk membatasi jumlah makanan, maka fungsi solusinya adalah total makanan yang sudah dimasukkan ke himpunan solusi sama dengan angka pembatas masukan pengguna.

4. Fungsi seleksi

   Fungsi seleksi digunakan sebagai penentuan prioritas makanan apa yang akan dimasukkan ke dalam himpunan solusi. Di dalam penulisan ini, digunakan beberapa versi dari algoritma *greedy*. Algoritma *greedy* yang digunakan adalah *greedy by price, greedy by rating, greedy by popularity*, dan *greedy by calories*. Terdapat juga strategi *greedy by all* yang memadukan analisis harga, *rating*, dan jumlah

makanan yang sudah terjual untuk mencapai solusi yang terbaik.

5. Fungsi kelayakan

   Fungsi kelayakan adalah fungsi yang menentukan makanan yang akan dimasukkan ke dalam himpunan solusi layak atau tidak. Pada kasus ini, digunakan fungsi kelayakan berupa apakah makanan memiliki bahan yang dihindari oleh pelanggan. Pelanggan dapat memasukkan satu atau lebih bahan yang dihindari. Implementasi dari fungsi kelayakan ini menggunakan *pattern matching* dengan algoritma Boyer Moore.

6. Fungsi Objektif

   Fungsi objektif dalam program ini adalah memaksimumkan kepuasan pelanggan terhadap makanan yang dipilih dari rumah makan.

## C. *Penerapan Strategi Greedy*

1. *Greedy by price, rating, popularity*, dan *calories*

   Untuk algoritma greedy yang berdasarkan satu parameter saja, strategi *greedy* yang diterapkan adalah dengan mengambil makanan yang paling optimal berdasarkan parameter tersebut, Untuk *price* dan *calories*, setiap pengambilan dimulai dari *price* atau *calories* yang lebih kecil terlebih dahulu. Sedangkan untuk *rating* dan *popularity*, setiap pengambilan dimulai dari yang paling besar. Untuk mempermudah, menu diurutkan terlebih dahulu berdasarkan parameter yang dipilih. Kemudian, pengambilan dilakukan dengan iterasi dari baris menu paling atas.

   Secara umum, implementasi *greedy* untuk keempat parameter tersebut adalah sama. Perbedaannya hanya terletak pada mekanisme sorting yang berdasarkan masukan pengguna. Berikut ini adalah contoh implementasi *greedy by price* di bahasa pemrograman Python.

TABEL I. IMPLEMENTASI *GREEDY BY PRICE* DI PYTHON

```
def greedyByPrice(filename, budget,
totalparticipant, listAvoidIngredient,max):
    result = np.empty((0,2))
    listprice = getPrice(filename)
    listbahan = getBahan(filename)
    totalprice = 0
    maxnow = max
    listpricesort = sortarrayasc(listprice)
    i = 0
    while (totalparticipant > 0 and totalprice <
budget):
        menu = listpricesort[i][0]
        bahan = getBahanByMenu(listbahan,menu)
        if
(fungsikelayakan(listAvoidIngredient,bahan) ==
False):
            result = addMenu(result,menu)
            totalparticipant -= 1
            totalprice += listpricesort[i][1]
            maxnow -=1
            if (maxnow == 0):
                maxnow = max
                i += 1
        else:
            i += 1
    if (i>len(listpricesort)-1):
        i = 0
if (totalprice > budget):
    totalprice -= listpricesort[i][1]
    result = removeLastElement(result)
return result, totalprice
```

2. *Greedy by ratio*

   Untuk algoritma *greedy* yang berdasarkan rasio, digunakan parameter harga, *rating*, dan popularitas secara bersamaan. Kalori tidak dimasukkan karena aspek ini terpisah dari yang ketiga lainnya. Secara umum, algoritma untuk rasio sama seperti algoritma pada poin sebelumnya. Perbedaannya terdapat pada list menu, yang memiliki kolom berupa rasio. Rasio dihitung berdasarkan perhitungan sebagai berikut.

   $$Ratio = rating*popularity/price$$

   Perhitungan ini didasarkan pada sifat dari masing-masing aspek. *Rating* dan *popularity* berbanding lurus dengan kepuasan pelanggan, sedangkan *price* berbanding terbalik dengan kepuasan pelanggan. Berikut ini adalah algoritma untuk mendapatkan rasio dari elemen-elemen tersebut.

TABEL II. IMPLEMENTASI PERHITUNGAN RASIO

```
def getAll(filename):
    directory = os.getcwd() + "\\test\\" +
filename + ".xlsx"
    column = ['Menu','Harga','Rating','Terjual']
    data_frame =
pd.read_excel(directory,usecols=column)
    listorigin = data_frame.values

    listresult= np.empty((0,2))
    for i in range(0,len(listorigin)):
        # Kolom ketiga: harga, kolom keempat:
```

```
rating, kolom kelima: Jumlah terjual
        ratio =
float(listorigin[i][2])*float(listorigin[i][3])/fl
oat(listorigin[i][1])
        array = np.array([listorigin[i][0],ratio])
        listresult =
np.append(listresult,[array],axis=0)
    return listresult
```

*D. Penerapan Fungsi Kelayakan*

Fungsi kelayakan dalam program ini didasari pada input pengguna saat ditanya apakah ada bahan makanan yang dihindari. Implementasi dari fungsi kelayakan ini adalah dengan algoritma Boyer Moore. Algoritma ini digunakan karena menu tergolong ke dalam bahasa Indonesia, sehingga algoritma Boyer Moore lebih efektif digunakan daripada algoritma yang lain. Berikut ini adalah implementasi fungsi untuk mencari input pengguna di dalam bahan makanan.

TABEL III. PENERAPAN FUNGSI KELAYAKAN

```
def getLastIdxOfCharacter(pattern):
    lastIdx = [0 for i in range(128)]
    for i in range (0,128):
        lastIdx[i]=0
    for i in range (0,len(pattern)):
        lastIdx[ord(pattern[i])] = i
    return lastIdx

def boyermoore(text,pattern):
    lastIdx = getLastIdxOfCharacter(pattern)
    textlen = len(text)
    patternlen = len(pattern)

    positionInText = patternlen - 1
    if (positionInText > textlen-1):
        return -1
    else:
        positionInPattern = positionInText
        while (positionInText <= textlen-1):
            if (ord(pattern[positionInPattern]) ==
ord(text[positionInText])):
                if (positionInPattern==0):
                    return positionInText
                else:
                    positionInText -= 1
                    positionInPattern -= 1
            else:
                lo =
lastIdx[ord(text[positionInText])]
                if (positionInPattern < lo+1):
                    positionInText += patternlen -
positionInPattern
                else:
                    positionInText += patternlen -
(lo+1)
                positionInPattern = patternlen-1
    return -1 #tidak ditemukan

def fungsikelayakan(avoidIngredients,bahan):
    arrayAvoid = avoidIngredients.split()
    avoid = False
    for avoid in arrayAvoid:
        if (boyermoore(bahan,avoid) == -1):
            avoid = False
        else:
            avoid = True
            break
    return avoid
```

## IV. EKSPERIMEN DAN PENGUJIAN

Untuk pengujian, digunakan sebuah file restoran.xlsx yang berisi menu sebagai berikut.

TABEL IV. DAFTAR MENU DALAM RESTORAN.XLSX

| Menu | Bahan | Harga | Kalori | Rating | Terjual |
|---|---|---|---|---|---|
| Mie ayam | ayam mie saos tiram jamur kecap asin | 12,000 | 421 | 4.70 | 432 |
| Nasi ayam bakar | ayam nasi kecap | 17,000 | 194 | 4.60 | 324 |
| Nasi goreng | nasi kecap bawang merah bawang putih bakso sosis | 16,000 | 267 | 3.80 | 567 |
| Spageti | mie tomat | 15,000 | 157 | 5 | 230 |

| | | | | | |
|---|---|---|---|---|---|
| | bawang | | | | |
| Nasi udang | udang nasi cabai | 18,000 | 321 | 4.90 | 512 |
| Nasi ikan nila | ikan nila nasi saos tiram cabai | 18,000 | 128 | 4.90 | 100 |
| Nasi rendang | daging sapi nasi santan | 20,000 | 450 | 4.80 | 598 |
| Soto ayam | mie ayam kunyit serai bawang putih bawang merah merica jahe daun salam | 13,000 | 250 | 3.40 | 245 |
| Pecel lele | lele cabai nasi | 16,000 | 167 | 4.50 | 312 |
| Ketoprak | tahu ketupat taoge bihun kacang tanah bawang putih bawang merah | 11,000 | 306 | 4.60 | 216 |

Pengujian dilakukan untuk *greedy by rating*. Jika kita ingin memesan sebanyak-banyaknya dengan *budget* Rp 70.000, - yang diperoleh adalah sebagai berikut.

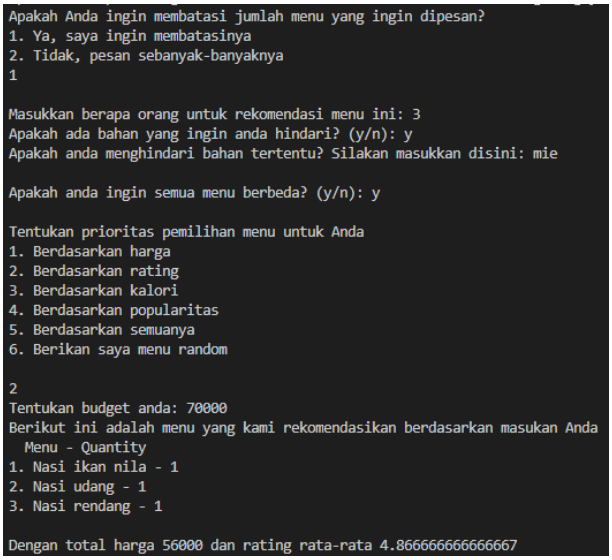
```
----------SELAMAT DATANG----------
Apakah Anda ingin membatasi jumlah menu yang ingin dipesan?
1. Ya, saya ingin membatasinya
2. Tidak, pesan sebanyak-banyaknya
2

Apakah ada bahan yang ingin anda hindari? (y/n): y
Apakah anda menghindari bahan tertentu? Silakan masukkan disini: udang

Apakah anda ingin semua menu berbeda? (y/n): y

Tentukan prioritas pemilihan menu untuk Anda
1. Berdasarkan harga
2. Berdasarkan rating
3. Berdasarkan kalori
4. Berdasarkan popularitas
5. Berdasarkan semuanya
6. Berikan saya menu random

2
Tentukan budget anda: 70000
Berikut ini adalah menu yang kami rekomendasikan berdasarkan masukan Anda
  Menu - Quantity
1. Spageti - 1
2. Nasi ikan nila - 1
3. Nasi rendang - 1
4. Mie ayam - 1
5. Ketoprak - 1

Dengan total harga 64999 dan rating rata-rata 4.766666666666667
```

**Gambar 4.1** Hasil Pengujian untuk Budget 70 ribu rupiah dan Menghindari Udang
Sumber: Dokumentasi Pribadi

```
Apakah Anda ingin membatasi jumlah menu yang ingin dipesan?
1. Ya, saya ingin membatasinya
2. Tidak, pesan sebanyak-banyaknya
2

Apakah ada bahan yang ingin anda hindari? (y/n): n

Apakah anda ingin semua menu berbeda? (y/n): y

Tentukan prioritas pemilihan menu untuk Anda
1. Berdasarkan harga
2. Berdasarkan rating
3. Berdasarkan kalori
4. Berdasarkan popularitas
5. Berdasarkan semuanya
6. Berikan saya menu random

2
Tentukan budget anda: 70000
Berikut ini adalah menu yang kami rekomendasikan berdasarkan masukan Anda
  Menu - Quantity
1. Spageti - 1
2. Nasi ikan nila - 1
3. Nasi udang - 1

Dengan total harga 51000 dan rating rata-rata 4.9
```

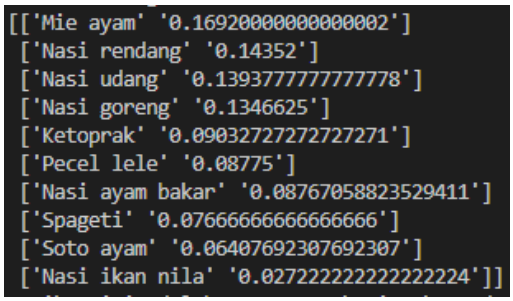
**Gambar 4.2** Hasil Pengujian untuk Budget 70 ribu rupiah tanpa adanya bahan yang dihindari
Sumber: Dokumentasi Pribadi

Dari dua kasus pengujian ini, yang berbeda hanyalah pada bahan yang dihindari. Bisa dilihat ketika pengguna memasukkan udang sebagai bahan yang dihindari, menu yang ditampilkan adalah menu yang sama sekali tidak mengandung udang. Sebaliknya, ketika pelanggan tidak mengisi bahan yang dihindari, program akan menampilkan makanan sesuai dengan algoritma *greedy by rating* tanpa menghiraukan bahannya.

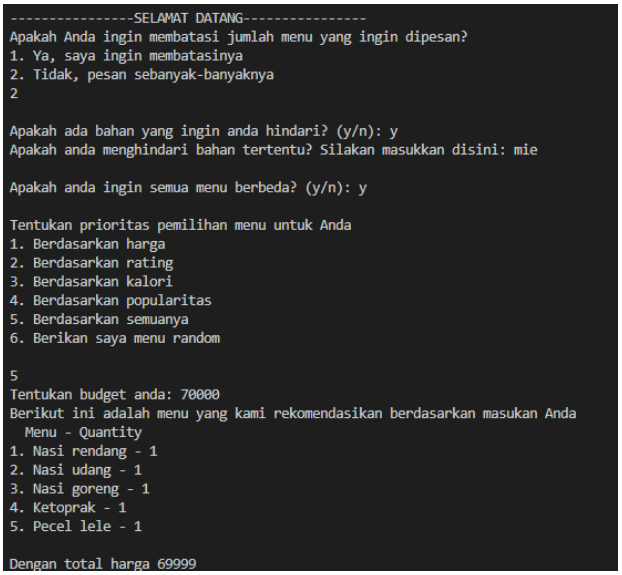



**Gambar 4.3** Pengujian untuk Membatasi Jumlah Makanan yang Dipilih
Sumber: Dokumentasi Pribadi

Dari pengujian ini, dapat dilihat bahwa pelanggan hanya ingin memesan sebanyak tiga makanan. Proses iterasi dalam penambahan *list* rekomendasi akan berhenti saat *list* solusi sudah mengandung tiga makanan. Bisa dilihat pada gambar bahwa semua makanan yang direkomendasikan adalah makanan yang tidak mengandung mie, sesuai dengan masukan pelanggan.




**Gambar 4.4** List Menu dengan Rasionya yang Dihasilkan oleh Fungsi pada Tabel 3.2
Sumber: Dokumentasi Pribadi




**Gambar 4.5** Pengujian untuk *Greedy by Ratio*
Sumber: Dokumentasi Pribadi

Dalam kasus ini, program menampilkan makanan yang teratas dari menu yang sudah diurutkan berdasarkan rasionya. Bisa dilihat bahwa hasil yang ditampilkan sudah sesuai dengan *list* yang terdapat pada Gambar 4.4. Bisa dilihat juga bahwa semua makanan yang ditampilkan tidak mengandung mie, sesuai dengan keinginan pelanggan.

## V. KESIMPULAN

Kepuasan seseorang terhadap makanan adalah sesuatu yang subjektif. Namun, apabila dikumpulkan beberapa data tentang makanan tersebut dan dilakukan analisis statistik pada makanan, data yang diperoleh dapat dijadikan sebagai acuan untuk pelanggan yang akan datang di masa depan. Dari data tersebut juga dapat diimplementasikan algoritma *greedy* untuk mendapatkan menu dengan nilai optimal berdasarkan parameter harga, rating, popularitas, dan kalori

Setiap orang juga mempunyai preferensi tentang bahan yang disukai dan tidak disukai untuk makanan. Permasalahan ini dapat diselesaikan dengan *Pattern matching*, yang pada makalah ini menggunakan algoritma Boyer Moore. Dengan semua ini, Dengan semua ini, dapat dibuat sistem rekomendasi yang dapat memaksimalkan kepuasan pelanggan terhadap makanan di suatu rumah makan.

## LINK REPOSITORY GITHUB

Pengujian lebih lanjut dapat dilakukan dengan kode program dalam repository pada link berikut https://github.com/kevinjohn01/MakalahStima


## UCAPAN TERIMA KASIH

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa, atas berkat-Nya penulis dapat menyelesaikan makalah yang berjudul "Penerapan Algoritma *Greedy* dan *Pattern Matching* dalam Sistem Rekomendasi Menu Rumah Makan" ini dengan baik dan tepat waktu. Penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulva Maulidevi, S.T., M.Sc. selaku

dosen pengampu mata kuliah IF2211 Strategi Algoritma K02 beserta segenap Tim Pengajar IF2211 Strategi Algoritma yang telah menyampaikan materi kuliah selama ini sehingga penulis dapat menyelesaikan makalah ini. Besar harapan penulis bahwa makalah ini dapat bermanfaat bagi semua orang yang seringkali merasa bingung dalam memilih makanan di rumah makan. Penulis meminta maaf bila ada kekurangan ataupun kesalahan penulisan dalam makalah ini. Akhir kata, penulis berterima kasih kepada siapapun yang sudah membaca makalah ini

## REFERENCES


[1] Munir, Rinaldi. Algoritma Greedy (Bagian 1). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[2] Munir, Rinaldi. Algoritma Greedy (Bagian 2). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

[3] Munir, Rinaldi. Algoritma Greedy (Bagian 3). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf

[4] Munir, Rinaldi. Pencocokan String (*String*/*Pattern Matching*). Institut Teknologi Bandung: Bandung, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

[5] https://id.quora.com/Hal-apa-saja-yang-jadi-pertimbanganmu-dalam-memilih-tempat-makan-atau-restoran, 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, 22 Mei 2023

Kevin John Wesley Hutabarat