Aplikasi String Matching dalam Membantu Diagnosis Penyakit dengan Gejala

Syarifah Aisha Geubrina Yasmin – 13519089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: aishageubrina@gmail.com

***Abstract*—Penyakit merupakan sesuatu yang tidak bisa dihilangkan keberadaannya. Meskipun bumi ditata ulang seluruh isinya, dapat hamper dipastikan, aka nada penyakit yang muncul akibat kondisi lingungan yang baru. Akibatnya, penyakit baru diidentifikasi setiap harinya. Dengan jumlah penyakit yang semakin lama kian membanyak, penulis ingin membantu dokter dengan mempersempit kemungkinan penyakit yang perlu dikaji berdasarkan gejala yang sudah dapat diidentifikasi pada pasien. Program ini dibuat dengan menggunakan algoritma String Matching Boyer-Moore terhadap gejala pasien dengan gejala seluruh penyakit pada database.**

Keywords—Algoritma, Penyakit, Diagnosis, String Matching, Boyer Moore.

I. PENDAHULUAN

Penyakit adalah suatu abnormalitas yang terjadi pada suatu organisme dan berdampak buruk pada fungsionalitas atau struktur tubuhnya. Pada manusia biasanya kata penyakit digunakan untuk mendeskripsikan segala hal yang merujuk pada rasa sakit, disfungsi, distress, dan bahkan kematian. Penyakit secara general dapat dikategorikan menjadi 4 hal, yaitu penyakit menular, penyakit defesiensi, penyakit turunan, dan penyakit fisiologis. Selain itu penyakit juga dapat dikategorikan menjadi penyakit menular dan penyakit tidak menular.

Sampai saat ini sudah sangat banyak penyakit yang berhasil diidentifikasi oleh manusia, bahkan bersamaan dengan ditulisnya makalah ini, besar kemungkinannya penyakit baru berhasil diidentifikasikan. Ditambah lagi, tidak hanya manusia yang berevolusi, ternyata penyakit, atau lebih tepatnya mikroorganisme yang menyebabkan penyakit tersebut juga bisa berevolusi. Secara singkat, kita bisa katakana bahwa setiap perubahan yang signifikan yang telah ditempuh manusia, selain adanya dampak positif yang dihasilkan, pasti juga terdapat dampak negatifnya, tidak terkecuali menimbulkan jenis penyakit baru.

Pertanyaannya sampai saat ini adalah, dari ribuan penyakit yang ada, peran dokter disini adalah mengidentifikasi penyakit tersebut dan menentukan metode penyembuhan yang tepat. Meskipun sekarang sudah ada dokter spesialis, bahkan dokter sub-spesialis, masih bisa kita katakana mustahil untuk mengingat seluruh penyakit yang ada, gejalanya, serta perawatan yang terbaik, dan seterusnya. Oleh karena itu penulis ingin membuat program sederhana yang dapat membantu para dokter dalam mengidentifikasi penyakit pasiennya. Perlu diingat, bahwa program yang dibuat bukan mengidentifikasi penyakit yang dialami oleh pasien, tetapi menggunakan algoritma string matching pada gejala yang dirasakan oleh pasien dengan gejala-gejala setiap penyakit yang terdapat dalam database. Tujuan dari program ini adalah untuk mempersempit kemungkinan penyakit yang dialami pasien dan menampilkannya kepada dokter. Targer pengguna dari program ini adalah seorang dokter, dan bukan pasien untuk menghindari kemungkinan terjadinya diagnosis sendiri oleh pasien.

II. LANDASAN TEORI

A. *Pengertian String/Pattern Matching*

String Matching merupakan sebuah algoritma yang mencari kecocokan string dalam sebuah text atau string yang berukuran lebih besar. Sebagai contoh:

- T: text atau long string dengan panjang karakter n
  Contoh T: “Aku mencari makanan di hutan”
- P: pattern, yaitu string dengan panjang jarakter m
  Contoh P: “makan”

Maka akan dicari pola ‘makan’ pada T, yaitu ditemukan pada index text ke-10 “Aku mencari makanan di hutan”.

Istilah yang perlu dipahami dalam string adalah prefix dan suffix. Prefix adalah kelompok character yang berada pada bagian awal string, atau berawal dari root. Sementara suffix adalah kebalikannya, yaitu kelompok character yang berada pada bagian akhir string.

Jika ada string (S) dengan panjang m,
S: $x_0x_1x_2\ldots x_{m-1}$

maka,

- Prefix dari S adalah substring dari S[0..K];
- Suffix dari S adalah substring dari S[K..m-1], dengan K adalah index apapun diantara 0–m-1

Contohnya,

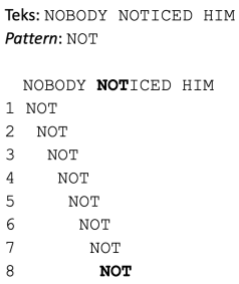
H	A	S	S	A	N

Maka yang termasuk dalam prefix adalah
1. “H”
2. “HA”
3. “HAS”, dst.

Sementara itu, yang termasuk suffix adalah
1. “N”
2. “AN”
3. “SAN”, dst.

B. *Algoritma String/Pattern Matching*

1. Algoritma Brute Force
   Algoritma ini, seperti namanya, akan memeriksa setiap index dari text-nya. Apabila terdapat T[0..n] dan ingin dilakukan pencocokan substring P[0..m], maka akan ditelusuri pencocokan sebagai berikut.



Gambar 2.2.1.1. String Matching Brute Force
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Tahapan dari algortima ini adalah sebagai berikut.
1. Cocokkan pattern P[0..m] pada bagian awal text
2. Apabila tidak terjadi mismatch sampai akhir pattenr, maka pencarian selesai
3. Jika terjadi mismatch, maka dilakukan pencocokan pattern P[0] kembali pada $text_{i+1}$
4. Ulangi Langkah 2 atau 3, apabila semua text suka dicocokan, berarti tidak P bukan merupakan substring dari T.

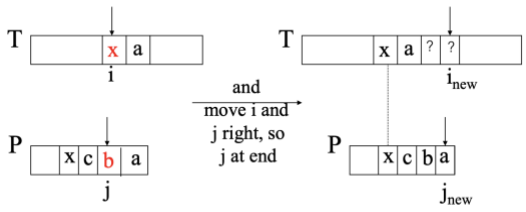
Dari gambar tersebut, terlihat bahwa pergerakan pencocokan–apabila terdapat mismatch–hanya satu index. Oleh karena itu apabila terdapat T: “aaaaaaaaaaaaaaah” dan P: “aaah”, akan dilakukan pencocokan berulang kali sebelum sampai ke tujuan.

Algoritma ini lebih baik digunakan saat variasi character-nya cenderung luas, dan tidak monoton seperti pada contoh di paragraf sebelumnya. Umumnya, kompleksitas algoritma brute force untuk string matching adalah O(m+n)

2. Algoritma Boyer-Moore
   Algoritma ini didasarkan pada dua (2) teknik, yaitu
   1. *Looking glass technique*
      Pencocokan P pada T dengan meninjaunya dari index terakhir (bergerak mundur)
   2. *The character jump technique*
      Ketika terjadi mismatch pada T[i] == x, dan karakter di P[j] != T[i]

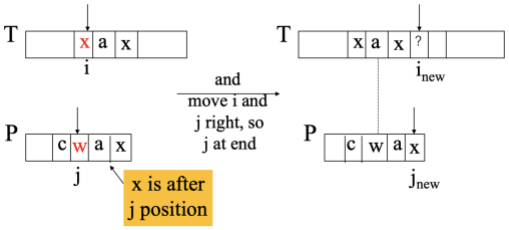
Terdapat tiga (3) kemungkinan apabila terjadi mismatch, yaitu

- Case 1 : Jika x merupakan substring dari P dan index x pada P kurang dari j, maka sejajarkan I dengan index x pada P (new j)



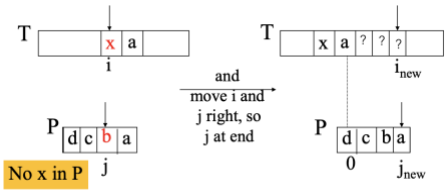
Gambar 2.2.2.1. Case 1 Algoritma Boyer Moore apabila terdapat mismatch
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

- Case 2 : Jika x merupakan substring dari P dan index x pada P lebih dari j, maka geser P 1 karakter ke T[i+1]



Gambar 2.2.1.2. Case 2 Algoritma Boyer Moore apabila terdapat mismatch
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

- Case 3 : Jika x bukan merupakan substring dari P, maka cocokan P[0] dengan T[i+1]

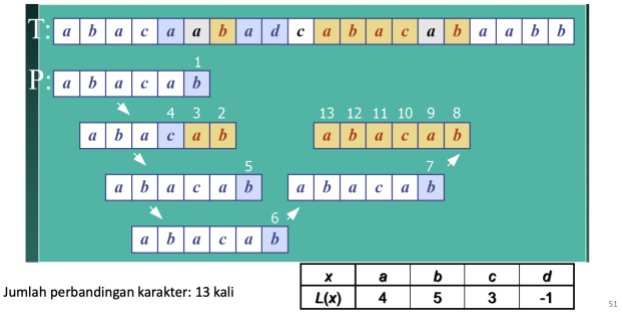


Gambar 2.2.2.3. Case 3 Algoritma Boyer Moore apabila terdapat mismatch
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Selain itu, pada algoritma Boyer-Moore juga dibutuhkan fungsi *Last Occurrence*, yaitu sebuah fungsi yang memetakan sebuah karakter pada alfabet A menjadi int sebagai index pada P. L(x) dapat didefinisikan menjadi:

- index i terbesar sehingga P[i]==x atau -1 jika tidak ada (x adalah character pada A)

Fungsi ini digunakan apabila terdapat mismatch sesuai dengan Case 1, dan terdapat lebih dari satu character x pada P, maka akan dipilih character x yang paling terakhir muncul. Contoh pengerjaan Boyer-Moore adalah sebagai berikut.



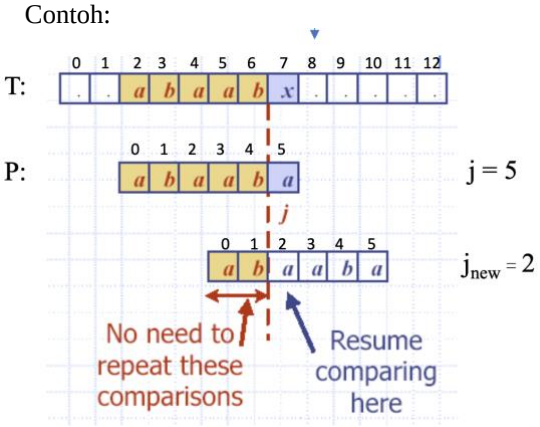
x	a	b	c	d
L(x)	4	5	3	-1

Gambar 2.2.2.4. Contoh Penyelesaian dengan Algoritma Boyer Moore
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Algoritma ini lebih cocok digunakan apabila variasi alfabetnya luas dan tidak monoton juga dan kompleksitas untuk kasus terburuknya adalah O(mn+A), dengan A adalah jumlah alphabetnya. Selain itu, algoritma ini juga lebih cepat daripada brute force.

3. Algoritma Knuth-Moris-Pratt (KMP)
KMP, mirip dengan brute force, akan meninjau kecocokan string dari kiri ke kanan. Namun memiliki meotde yang lebih efektif apabila terdapat mismatch, yaitu jika terjadi mismatch pada P di P[j], maka peninjauan selanjutnya akan melewati prefix pada P[0..j-1] yang memiliki nilai yang sama dengan suffixdari P[1..j]. Hal ini dilakukan untuk menghindari pencocokan yang sebenarnya tidak perlu.

Contoh:



Gambar 2.2.3.1. Ilustrasi Algoritma KMP
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Agar lebih mudah, kita dapat menggunakan *border function*, yaitu fungsi yang menentukan nilai dari j_{new}.
Misal:

j = posisi mismatch pada P
k = j-1, maka
border function/b(k) = ukuran prefix terbesar dari P[0..k] yang bernilai sama dengan suffix P[1..k].

(k = j-1)

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a
k	-	0	1	2	3	4
b(k)	-	0	0	1	1	2

b(k) is the size of the largest border.

Gambar 2.2.3.2. Contoh Tabel Border Function
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Contoh permasalahannya adalah sebagai berikut.

Gambar 2.2.3.3. Contoh Penyelesaian dengan Algoritma KMP
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Sebenarnya, algoritma KMP ini hanyalah modifikasi dari brute force, namun jauh lebih cepat dan efesien dengan kompleksitas waktu O(m+n). Namun berbeda dari algoritma pada sub poin sebelumnya, algoritma ini kurang efesien apabila variasi alphabetnya semakin luas karena meningkatkan kemungkinan untuk terjadinya mismatch pada awal pattern P.

4. Diagnosis
Menurut KBBI, diagnosis adalah penentuan jenis penyakit dengan cara meneliti atau memeriksa gejala gejalanya.

III. Implementasi Algoritma

Algoritma yang digunakan pada percobaan ini adalah algoritma Boyer-Moore, karena Bahasa yang digunakan adalah Bahasa Indonesia (khusus pada kasus uji coba ini, namun penerapan ke depannya dapat digunakan *medical ter*m yang tepat). Berikut saya tampilkan algoritma Boyer Moore yang dibuat dalam Bahasa Python.

```
def lastOcc(arr, c):
    n = len(arr) - 1
    while(n != 0):
        if(arr[n] != c):
            n -= 1
        else:
            return n
    return -1


def boyerMoore(line, pattern):
    if(len(pattern) > len(line)):
        return -1
    else:
        n = len(line)
        m = len(pattern)
        i = j = m - 1
        while(i != n):
            if line[i].lower() ==
pattern[j].lower():
                if j == 0:
                    return i
                else:
                    i -= 1
                    j -= 1
            else:
                c = line[i]
                l = lastOcc(line, c)
                i = i + m - min(j, 1+l)
                j = m - 1

        return -1
```

Seperti yang sudah dijelaskan sebelumnya, algoritma ini akan mencari kecocokan pola dari sebuah pattern dan sebuah text. Fungsi tersebut akan mengembalikan index pertama pada text yang sama dengan character awal pada pattern, T[i] == P[0]. Apabila tidak terdapat kecocokan pola, maka akan dikembalikan -1.

Data uji yang digunakan pada pengimplementasian ini merupakan sampel data yang diambil dari internet dan dengan jumlah yang terbatas. Data uji yang dimaksudn adalah 10 nama-nama penyakit dan gejala-gejalanya. Apabila dimaksudkan untuk dikembangkan dapat dibuat database tersendiri untuk menyimpan datanya. Berikut adalah contoh sample data yang digunakan pada pengimplementasiannya.

Nama Penyakit	Gejala
Alergi	iritasi mata, pilek, hidung mampet, mata berair, bersin, radang tenggorokan, tenggorokan gatal
Flu	demam, sakit kepala, badan lemas dan sakit, lelah, batuk-batuk parah
Conjunctivitis	mata memerah, mata gatal, mata berair, kelopak mata mengeras
Diare	tinja encer, sering BAB, sakit perut, mual, kembung, demam, tinja berdarah
Migraine	kepala berdenyut, sakit kepala sebelah, sakit kepala, berlangsung berjam-jam, mengganggu aktivitas sehari-hari, mual dan muntah
Mononucleosis	kelelahan ekstrim, sering tidur, sakit tenggorokan, demam, tidak nafsu makan, otot sakit
Alzheimer	Kebingungan, disorientasi, tersesat di tempat yang sudah kenal, sulit membuat keputusan, kesusahan berbicara, personalitas berubah, halusinasi, mood buruk, gelisah
Addison	lelah, lemas, otot lemah, mood buruk, nafsu makan hilang, berat badan turun, mudah halusinasi
Sinuitis	Muncul cairan hijau di hidung, cairan kuning dihidung, hidung mampet, demam, sakit gigi, bau mulut, indra penciuman berkurang
Pneunomia	batuk, susah nafas, detak jantung cepat, deg degan, demam, berkeringat, menggigil, nafsu makan hilang, sakit dada, batuk berdarah, sakit kepala, lemas, mual, muntah, otot sakit, sandi sakit

Tabel 3.1. Contoh sample data penyakit dan gejalanya (Sumber: Penulis)

Alur Program yang telah dibuat adalah sebagai berikut.

1. Pembacaan data dari file eksternal, pada kasus ini, data akan berasal dari file txt yang berisi seperti pada table diatas yang sudah diformat sedemikian rupa.
2. Penyimpanan data dalam bentuk matriks. Sama seperti table diatas, akan dibuat dua kolom, dengan kolom pertama menyimpan nama penyakit dan kolom kedua menyimpan gejala-gejala dari penyakit tersebut.
3. Input gejala yang ingin dicari oleh user, perlu diingat pada program yang telah dibuat, setiap jumlah gejala dapat dipisahkan dengan “, ”.

```
Tools to Help Narrowing Down All Possible Diseases
Symptons:
```

Gambar 3.1. Tampilan Program 1
(Sumber: Penulis)

4. Pencocokan string secara traversal pada seluruh penyakit yang terdapat pada matriks. Pada tahap ini, setiap gejala penyakit akan ditinjau apakah memiliki semua gejala yang sama pada input (boleh lebih, tetapi tidak boleh kurang). Pencarian kecocokan string disini akan memanggil fungsi Boyer Moore.

```
Tools to Help Narrowing Down All Possible Diseases
Symptons: sakit kepala, mual, muntah
```

Gambar 3.2. Tampilan Program 2
(Sumber: Penulis)

5. Semua Penyakit yang memiliki gejala yang sama dengan input user akan ditampilkan pada layer

```
Possible disease with that sympton is:
1. Migraine
2. Pneunomia
```

Gambar 3.3. Tampilan Program 3
(Sumber: Penulis)

IV. HASIL EKSPERIMEN

1) Uji Coba 1 : Masukan string kosong

```
Tools to Help Narrowing Down All Possible Diseases
Symptons:

Input symptons failed
```

Gambar 4.1. Uji Coba 1
(Sumber: Penulis)

Pada program kasus string kosong ditangani dengan metode try and exception.

2) Uji Coba 2 : Masukan "mual"

```
Tools to Help Narrowing Down All Possible Diseases
Symptons: mual

Possible disease with that sympton is:
1. Diare
2. Migraine
3. Pneunomia
```

Gambar 4.2. Uji Coba 2
(Sumber: Penulis)

Jika dicocokan dengan tabel data diatas, dapat dilihat bahwa penyakit yang memiliki gejala mual juga terdapat 3, yaitu Diare, Migraine, dan Pneunomia sehingga input dan output sudah sesuai dengan harapan.

3) Uji Coba 3: Masukan "batuk, demam, sakit kepala"

```
Tools to Help Narrowing Down All Possible Diseases
Symptons: batuk, demam, sakit kepala

Possible disease with that sympton is:
1. Flu
2. Pneunomia
```

Gambar 4.3. Uji Coba 3
(Sumber: Penulis)

Jika dicocokan dengan tabel diatas, dapat dilihat juga bahwa penyakit yang memiliki gejala setidaknya "batuk, demam, sakit kepala" adalah flu dan pneumonia.

4) Uji Coba 4: Masukan "d"

```
Tools to Help Narrowing Down All Possible Diseases
Symptons: d

Possible disease with that sympton is:
1. Alergi
2. Flu
3. Diare
4. Migraine
5. Mononucleosis
6. Alzheimer
7. Addison
8. Sinuitis
9. Pneunomia
```

Gambar 4.4. Uji Coba 4
(Sumber: Penulis)

Pada kasus ini belum ada penanganan khusus untuk masukan yang valid, namun tidak membentuk kata yang dikenali, seperti satu character ("d","a","c"), sementara seluruh penyakit kemungkinan besar memiliki huruf-huruf tersebut pada gejalanya.

5) Uji Coba: Tidak ada penyakit yang sesuai

```
Tools to Help Narrowing Down All Possible Diseases
Symptons: sakit mata, batuk, mual

No diease have tha symptons you want
```

Gambar 451. Uji Coba 5
(Sumber: Penulis)

Apabila gejala yang dimasukan tidak dikenali atau tidak ditemukan kecocokan dari database, maka akan diberikan pesan error.

V. SIMPULAN

Kesimpulan yang dapat diambil dari hasil uji coba program yang sudah dibuat adalah algoritma string matching Boyer Moore berhasil diimplementasikan dengan cukup baik untuk membantu mempersempit kemungkinan penyakit dari gejala pasien. Namun apabila program ini ingin dilanjutkan oleh penulis ataupun pihak lainnya, masih perlu dilakukan penyesuaian dan penanganan kasus special lainnya. Selain itu perlu dicatat juga bahwa program ini sangat bergantung terhadap kelengkapan database penyakit dan gejalanya, akan lebih baik untuk digunakan database yang lebih konkrit dari yang sudah dibuat saat ini.

LINK

Github https://github.com/syarifahaisha/Tugas-Makalah-IF2211---String-Matching-Boyer-Moore

Youtube https://youtu.be/_tW5Q7SFXtg

UCAPAN TERIMA KASIH

Sebelumnya saya ingin mengucapkan puji syukur atas Allah Swt.. Oleh karena Rahmat-Nya lah saya dapat mengerjakan tugas ini dengan usaha terbaik saya dan sampai selesai. Selain itu, saya ingin mengucapkan terima kasih yang sebesar-besarnya kepada seluruh pihak yang terlibat dalam mata kuliah IF 2211 Strategi Algoritma, khususnya dosen pengajar kelas 02, Dr. Nur Ulfa Maulidevi. Penugasan dari mata kuliah ini sangat membantu saya berkembang sebagai mahasiswa dan juga sebagai manuisa.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

[2] Munir, Rinaldi, Pencocokan-String-2020

[3] https://uhs.princeton.edu/health-resources/common-illnesses

[4] https://www.nhsinform.scot/illnesses-and-conditions/a-to-z#R

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.

Jakarta, 11 Mei 2021

Syarifah Aisha 13519089