LAPORAN TUGAS KECIL 4
IF2211 – STRATEGI ALGORITMA
SEMESTER II 2019/2020

Ekstraksi Informasi dari Artikel Berita dengan Algoritma Pencocokan String

Disusun oleh:

Anna Elvira Hartoyo

13518045

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2020

DAFTAR ISI

DAFTAR GAMBAR

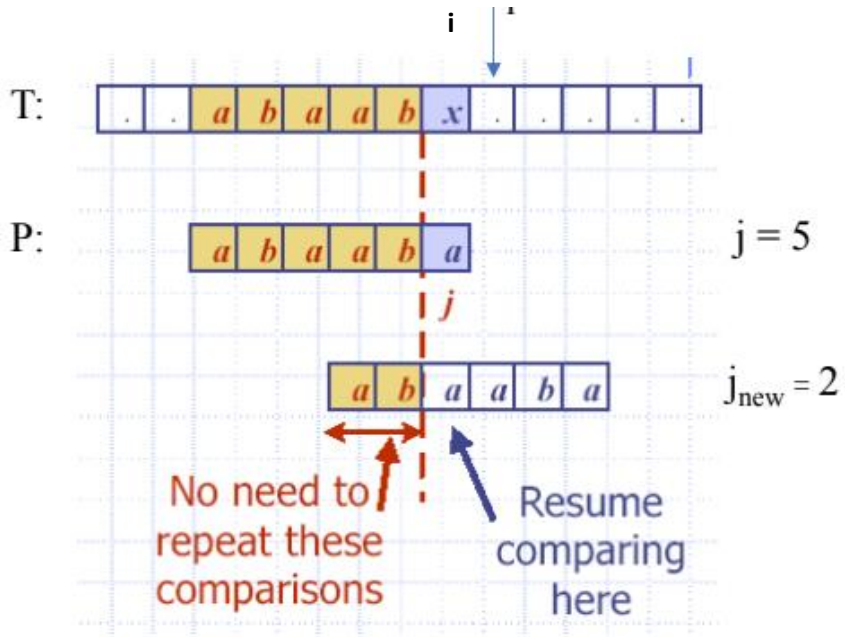
1. **Deskripsi Singkat Algoritma Pencocokan String**

String adalah larik yang setiap elemennya adalah karakter. Pada pencocokan string, terdapat dua istilah, yaitu teks dan pattern. Teks adalah string dengan panjang n karakter, sedangkan pattern adalah string dengan panjang m karakter (asumsi m << n) dan merupakan string yang mau dicari dalam teks.

a) Knuth-Morris-Pratt (KMP)

Algoritma pencocokan string ini dilakukan dengan memcocokan pattern dalam teks dari kiri ke kanan, tetapi pergeseran yang dilakukan lebih cerdas dibandingkan pada algoritma *brute force*. Algoritma ini mempertimbangkan banyak pergeseran dan dengan tidak mengulangi pemeriksaan yang sama.

Mekanisme umumnya sebagai berikut:
Jika terjadi mismatch teks T pada T[i] dan pattern P pada P[j] (T[i] ≠ P[j], pergeseran dilakukan adalah sebanyak nilai prefix P[0..j-1] terbesar yang juga suffix P[1..j-1]. Nilai yang diperoleh inilah yang akan menjadi indeks j yang baru untuk pemeriksaan selanjutnya.



Gambar 1. Pencocokan string dengan KMP

Sebagai contoh, pada gambar 1, mismatch terjadi pada i = 5 dan j = 5 (T[5] ≠ P[5]). Prefix untuk P[0..4] adalah a, ab, aba, abaa, abaab. Suffix untuk P[1..4] adalag b, ab, aab, baab. Dari hasil prefix dan suffix tersebut, nilai terbesar yang sama adalah ab dengan panjang 2, sehingga untuk pemeriksaan selanjutnya dimulai dari indeks j ke-2

Untuk mepermudah proses pencocokan, algorimta KMP melakukan preproses terhadap pattern P dengan mencari semua nilai prefix terbesar pada P[0..j-1] yang sama dengan suffix P[1..j-1] Ketika terjadi mismatch pada P[j]. Nilai *border function* (fungsi pinggiran) b(k) untuk pattern P dengan k sebagai posisi j-1 adalah

b(k) = ukuran prefix terebesar pada P[0..k] yang juga menjadi suffix P[1..k]

Jika mismatch terjadi pada indeks j = 0, yang berarti kita tidak dapat mencari nilai suffix P, lakuakn pergeseran sepeerti pada algoritma brute force, yaitu menggeser i dan j ke kanan sebanyak satu karakter.

Kompleksistas waktu algoritma KMP adalah sebesar O(m+n) yang terdiri dari waktu untuk mencari border function sebesar O(m) dan waktu untuk pencarian string sebesar O(n).

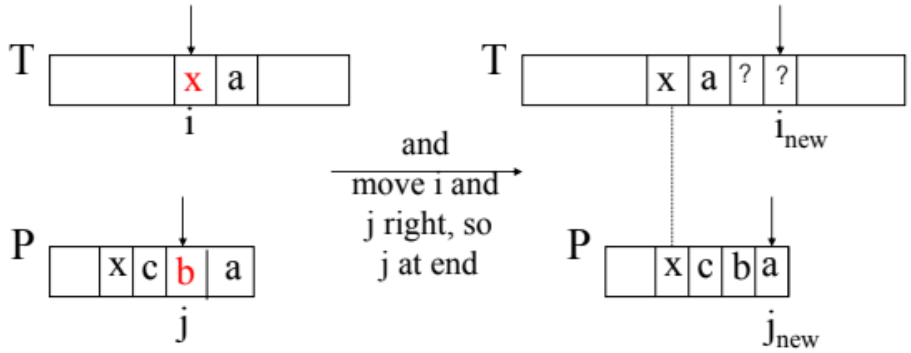
b) Boyer-Moore

Algortima Boyer-Moore melakukan pencocokan string dengan dua teksik:

1) The looking glass technique
   Mencocokan P dalam T dengan pergerakkan mundur (*backward*) yang dimlai dari akhir P
2) The character-jump technique
   Ketika mismatch terjadi pada T[i] == x, karakter pada pattern P[j] tidak sama dengan T[i]

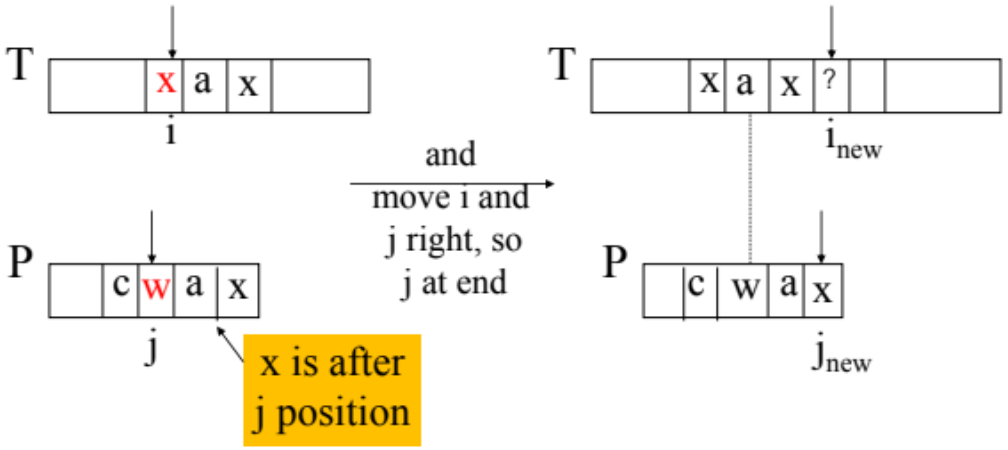
Terdapat tiga kemungkinan kasus:

1) Kasus 1
   Jika pada P terdapat karakter x, lakukan pergeseran ke kanan (shift right) pattern P, sehingga last occurrence dari karakter x pada P sejajar dengan T[i].



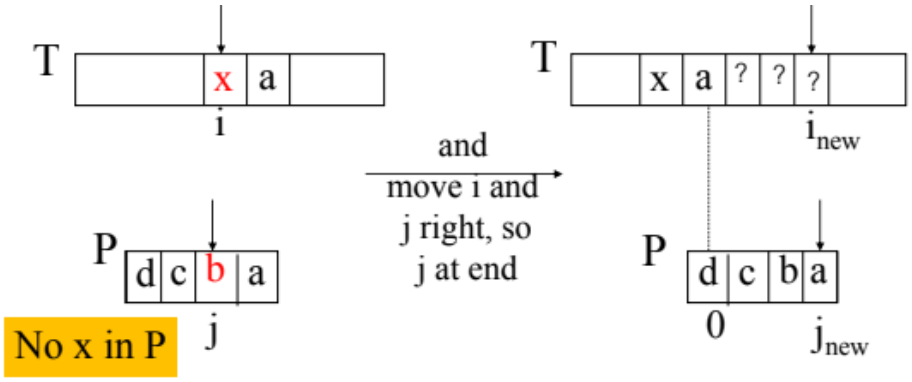
Gambar 2. Kasus 1 saat pencocokan string dengan BM

2) Kasus 2
   Jika pada P terdapat karakter x (setelah j), tetapi pergeseran ke kanan (shift right) tidak mungkin dapat menemukan last occurrence dari x pada P, lakuakn pergeseran ke kanan untuk P sebanyak satu karakter ke letak T[i+i].



Gambar 3. Kasus 2 saat pencocokan string dengan BM

3) Kasus 3
Jika kasus 1 dan kasus 2 tidak terpenuhi karena tidak terdapat x pada P, lakukan pergeseran P sehingga P[0] sejajar denan T[i+1].



Gambar 4. Kasus 3 saat pencocokan string dengan BM

Untuk mempermudah proses pencocokan string, algortima BM ini melakukan preproses terhadap pattern P dan alfabet A untuk menentukan last occurrence function atau L(). Fungsi L() ini memetakan seluruh huruf pada A menjadi integer. L(x) dengan x adalah huruf pada A akan menghasilkan nilai indeks I terbesar sehingga P[i] == x dan akan menghasilkan nilai -1 jika indeks tidak ditemukan atau dengan kata lain x tidak ada pada P.

Dalam kasus terburuk, kompleksitas waktu algoritma ini adalah O(nm + A)

c) Regular Expression

Regular expression (regex) adalah notasi standar yang mendeskripsikan suatu pola (pattern) berupa urutan karakter atau string. Regex digunakan untuk pencocokan string (string matching) dengan efisien. Regex memudahkan kita mencari pattern tertentu dari teks yang Panjang. Proses dialawi dengan membuat pola terlebih dahulu. Kemudian pola ini akan dicocokkan dengan teks atau tulisan yang panjang. Jika dijumpai string yang cocok dengan pola, maka string tersebut pun bisa diekstrak atau diambil.

Pada tugas ini, pencocokan dengan regex menggunakan library re dari python dengan memanfaatkan fungsi search() dan findall()

Pola atau pattern yang dimaksud dapat kita definisikan sesuai kebutuhan, misalnya untuk mencari angka dapat digunakan pattern r'\d'. Terdapat banyak keyword dan symbol yang dapat digunakan untuk membangun pattern yang diinginkan agar diperoleh hasil pencarian yang diinginkan.

2. Kode Program

Program terbagi atas algoritma pencocokan string (menggunakan python), tampilan web (menggunakan HTML dan CSS), dan back-end web (menggunakan flask)

Program untuk algoritma pencocokan string terdiri atas empat file:

KMP.py

```
'''
Fungsi untuk menghitung nilai fungsi fail (border function)
untuk setiap karakter pada string pattern dan mengembalikan
tabel fail
'''
def computeFail(pattern):
    fail = [0] * len(pattern)

    m = len(pattern)
    j = 0;
    k = 1;

    while (k < m):
        if (pattern[j] == pattern[k]):
            fail[k] = j + 1
            k += 1
            j += 1
        elif (j > 0):
            j = fail[j-1]
        else: # tidak match sama sekali
            fail[k] = 0
            k += 1

    return fail

'''
Fungsi utama dalam pencocokan string menggunakan metode KMP
Mencari patternInput dalam textInput
Jika ditemukan, fungsi mengembalikan indeks pertama letak kemunculan
pattern dalam teks, jika tidak ditemukan, fungsi mengembalikan -1
'''
```

```
def kmpAlgorithm(textInput, patternInput):

    text = textInput.upper()
    pattern = patternInput.upper()

    n = len(text)
    m = len(pattern)

    fail = computeFail(pattern)

    i = 0
    j = 0

    while (i < n):
        if (pattern[j] == text[i]):
            if (j == m-1): # posisi j sudah di akhir pattern
                return i-m + 1
            i+=1
            j+=1
        elif (j > 0):
            j = fail[j-1] # mulai mencocokan dari nilai fail[j-1]
        else:
            i+=1

    return -1; #tidak match
```

BM.py

```
'''
Fungsi untuk mencari nilai last occurence bagi setiap
karakter pada string pattern. Fungsi mengembalikan tabel
last
'''
def lastOccurrence(pattern):
    last = [-1] * 128
    for i in range (len(pattern)):
        last[ord(pattern[i])] = i

    return last;

'''
Fungsi utama dari algoritma pencocokan string dengan metode BM
Mencari patternInput dalam textInput. Jika ditemukan, fungsi
Mengembalikan indeks pertama kemunculan pattern, jika tidak ditemukan,
Fungsi mengembalikan -1
'''
def bmAlgorithm(textInput, patternInput) :
```

```
    pattern = patternInput.upper()
    text = textInput.upper()

    last = lastOccurrence(pattern)
    n = len(text)
    m = len(pattern)
    i = m-1

    if (i > n-1):#pattern lebih panjang dari text
        return -1

    j = m-1
    while True:
        if (pattern[j] == text[i]): #jika match
            if (j == 0): #pattern sudah sampai awal
                return i
            else: #pattern belum habis, cocokkan secara mundur
                  (looking glass)
                i-=1
                j-=1
        else: # tidak match, lakukan character jump
            lastOcc = last[ord(text[i])]
            i = i + m - min(j, 1 + lastOcc)
            j = m -1

        if(i > n-1):
            break #lakukan selama text belum habis

    return -1 #text dan pattern tidak match
```

regularExpression.py

```
import re

'''
Pencocokkan string dengan menggunakan library re
Mencari string patternInput pada textInput
Jika pattern terdapat pada text, fungsi mengembalikan inndeks
pertama letak kemunculan pattern
Jika tidak ada pattern dalam text, fungsi mengembalikan -1
'''
def regex(textInput, patternInput):
    pattern = patternInput.upper()
    text = textInput.upper()

    if(re.search(pattern, text)):
        return ((re.search(pattern, text)).span()[0])
```

```
    else:
        return -1
```

util.py

```
import re
import nltk
nltk.download('punkt')


'''
Fungsi untuk memisahkan text panjang yang terdiri dari banyak kalimat
Menjadi list of kalimat
'''
def splitText(text):
    sentences = nltk.sent_tokenize(text)
    return sentences

'''
Fungsi untuk mencari seluruh angka dengan fomrat tertentu
pada kalimat sentence
'''
def findNum(sentence):
    nums =  re.findall(r'(?:\d{1,}[,.]?\d{1,}%?)[\s]?(?:juta|ribu)?[\s]?(?:
    orang|jiwa|penduduk|kasus|pasien)?', sentence)
    return nums

'''
Fungsi untuk mencari seluruh tanggal dengan fomrat tertentu
pada kalimat sentence
'''
def findDate(sentence):
    dates = re.findall(r'(?:Senin|Selasa|Rabu|Kamis|Jumat|Sabtu|Minggu)?[,]?[
    \s]?[(]?(?:\d{1,2})[-
    /\s](?:Jan(?:uari)?|Feb(?:ruari)?|Mar(?:et)?|Apr(?:il)?|Mei|Jun(?:i)?|Jul
    (?:i)?|Agustus|Sep(?:tember)?|Okt(?:ober)?|Nov(?:ember)?|Des(?:ember)?)?[
    \s]?(?:\d{1,2})?[-
    /)\s,]?(?:\d{2,4})?[)\s]?(?:pukul\s)?(?:\d{1,2}[:.\s])?(?:\d{1,2})?(?:\sW
    IB)?', sentence)
    return dates

'''
Fungsi untuk mencari angka pada list nums yang terdekat jaraknya dengan kata
berindkes awal idx
pada kalimat sentence
'''
```

```
def searchClosestNum(nums, sentence, idx):
    min = len(sentence)
    for num in nums:
        occ = [m.end() for m in re.finditer(num, sentence)]
        if(abs(int(occ[0]) - idx) < min):
            min = int(occ[0])
            closestNum = num

    return closestNum
```

Program untuk tampilan web terdiri atas dua file, yaitu:

mainPage.html

```
<!DOCTYPE html>
<html>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">

    <title>
        Tugas Kecil Stima 4
    </title>
    <style>
        body {font-family: "Lato", sans-serif}
    </style>

    <body>

        <!-- Navigation Bar -->
        <ul>
            <li><a href="#home">Home</a></li>
            <li><a href="#perihal">Perihal</a></li>
            <li><a href="#credit">Credit</a></li>
        </ul>

        <a name = "home"></a>
        <h1 class = "center">

            MY INFO-EXTRACTION <br/> APPLICATION

        </h1>

        <!-- form untuk input -->
        <form action="#results" method="post" id = "boxshadow">
            <p>Select files  : 
```

```
            <input type="file" name="filename" multiple placeholder="Enter a
            keyword">
            </p>

            <p>Keyword   :  
            <input type="text" name="keyword" placeholder="Enter a keyword">
            <br/>
            </p>

            <p>Select Algorithm :</p>
            <label class = "container">Boyer Moore
                <input type="radio" id="bm" name = "algorithm" value="bm">
                <span class="checkmark"></span>
            </label>

            <label class = "container">Knuth-Morris-Pratt
                <input type="radio" id="kmp" name = "algorithm" value="kmp">
                <span class="checkmark"></span>
            </label>

            <label class = "container">Regular Expression
                <input type="radio" id="regex" name = "algorithm"
                value="regex">
                <span class="checkmark"></span>
            </label>

            <button class = "button1" type="submit" value="submit">EXTRACT!
            </button>
        </form>

        <!-- Container untuk hasil output -->
        <div class = "generalCont" id = "boxshadow">
            <a name = "results"></a>
        <h1>RESULTS</h1>
        {% for i in range(kalimat|length): %}
            <p>{{i+1}}. Jumlah  : {{jumlah[i]}}</p>
            <p>  Waktu : {{tanggal[i]}}</p>
            <p>  {{kalimat[i]}}</p>
            <p>  Sumber : ({{sumber[i]}})</p>
        <br/>
        {% endfor %}
        </div>

        <!-- container untuk perihal -->
        <div class= "generalCont" id = "boxshadow" name = "perihal">
            <a name = "perihal"></a>
            <div class="row">
```

```
<h1>PERIHAL</h1>
<p>
   String adalah larik yang setiap elemennya adalah karakter.
   Pada pencocokan string, terdapat dua istilah, yaitu teks
   dan pattern. Teks adalah string dengan panjang n karakter,

    sedangkan pattern adalah string dengan panjang m karakter
    (asumsi m << n) dan merupakan string yang mau dicari
    dalam teks.
</p>
<div class="column">
    <div class="card">
        <p>
            <b>
                KNUTH-MORRIS PRATT (KMP)<br/>
                ALGORITHM
            </b>
        </p>
            <p> Algoritma pencocokan string ini dilakukan
                dengan memcocokan pattern dalam teks dari
                kiri ke kanan, tetapi pergeseran yang
                dilakukan lebih cerdas dibandingkan pada
                algoritma brute force.
                Algoritma ini mempertimbangkan banyak
                pergeseran dan dengan tidak
                mengulangi pemeriksaan yang sama.
            </p>
    </div>
</div>

<div class="column">
    <div class="card">
        <p>
            <b>
                BOYER-MOORE(BM)<br/>
                ALGORITHM
            </b>
        </p>
            <p> Algortima Boyer-Moore melakukan pencocokan
                string dengan dua teknik:
                The looking glass technique, yaitu
                mencocokan P dalam T dengan pergerakkan
                mundur (backward) yang dimlai dari akhir P
                dan the character-jump technique, yaitu
                Ketika mismatch terjadi pada T[i] == x,
                karakter pada pattern P[j] tidak sama dengan
                T[i]
```

```
                            </p>
                        </div>
                    </div>
                    <div class="column">
                        <div class="card">
                            <p>
                                <b>
                                    REGULAR<br/>
                                    EXPRESSION
                                </b>
                            </p>
                                <p>
                                    Regular expression (regex) adalah notasi
                                    standar yang mendeskripsikan suatu pola
                                    (pattern) berupa urutan karakter atau string.
                                    Regex digunakan untuk pencocokan string
                                    (string matching) dengan efisien.Pada tugas
                                    ini, pencocokan dengan regex menggunakan libr
                                    ary re dari python dan digunakan beberapa
                                    fungsi dari library
                                </p>
                        </div>
                    </div>
                </div>
                <p>
                    Program dibuat dalam bahan python untuk penerapan algoritma
                    pencocokan string.Untuk front-end website, digunakan HTML
                    dan CSS dan untuk back-end digunakan flask
                </p>
            </div>


            <!-- bagian credits -->
            <div class = "creditsCont">
                <a name = "credit"></a>
                <h2 class = 'center'>CREDITS BY</h2>
                <h3 class="center">Anna Elvira Hartoyo
                    <br/>
                    13518045
                    <br/>
                    APRIL 2020
                </h3>
            </div>
        </body>
    </html>
```

style.css

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: teal;
  }

li {
    float: left;
}

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 25px;
    text-decoration: none;
}

li a:hover {
    background-color: darkslategray;
}

h1{
    color : teal;
    font-weight:bold;
    margin-top: 30px;
}

h2{
    color : white;
    font-weight:bold;
    margin-top: 20px;
}

h3{
    color : white;
    font-size: 14px;
}


form {
    width: 40%;
    padding: 20px 40px 40px 60px;
    border-radius: 8px;
```

```
    margin: 30px auto;
    float : center;
}

input[type=text] {
    width: 70%;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    font-size: 14px;
    background-color: white;
    padding: 12px 12px 12px 12px;
    margin-bottom:8px ;
    transition-duration: 0.2s;
}

input[type=text]:focus {
    border: 2px solid teal;
    box-shadow: 1px 2px 3px teal;
}

input[type=text]:hover {
    border: 2px solid teal;
}


.generalCont {
    width: 80%;
    padding: 20px 40px 40px 40px;
    border-radius: 8px;
    border: 1px solid teal;
    margin: 50px auto;
    float : center;
}

.creditsCont {
    width: 100%;
    padding: 5px 0px 20px 0px;
    border: 1px solid teal;
    margin-top: 60px;
    left : 0px;
    bottom : 0px;
    float : center;
    background-color: teal;
}
```

```css
.button1{
    background-color: teal;
    padding: 12px;
    width : 40%;
    color : white;
    border-width: 0;
    border-radius: 4px;
    margin : auto;
    display: block;
    transition-duration: 0.2s;
    cursor: pointer;
}

.button1:hover {
    background-color: darkslategray;
    box-shadow: 1px 4px 4px darkgray;
}

.container {
    display: block;
    position: relative;
    padding-left:30px;
    margin-bottom: 12px;
    width: 40%;
    cursor: pointer;
    font-size: 14px;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}


.container input {
    position: absolute;
    opacity: 0;
    cursor: pointer;
}

.checkmark {
    position: absolute;
    top: 0;
    left: 0;
    height: 20px;
    width: 20px;
    background-color: #eee;
    border-radius: 50%;
```

```
}


.container:hover input ~ .checkmark {
    background-color: #ccc;
}

.container input:checked ~ .checkmark {
    background-color: teal;
}

.checkmark:after {
    content: "";
    position: absolute;
    display: none;
}

.container input:checked ~ .checkmark:after {
    display: block;
}

.container .checkmark:after {
      top: 6px;
      left: 6px;
      width: 8px;
      height: 8px;
      border-radius: 50%;
      background: white;
}


.column {
    float: left;
    width: 31%;
    padding: 0 10px;
}

.row {
    margin: 0 -5px;
}


.row:after {
    content: "";
    display: table;
    clear: both;
}
```

```
.card {
    border:1px solid teal;
    border-radius: 4px;
    padding: 16px;
    text-align: center;
    background-color: white;
}


#boxshadow {
    box-shadow: 1px 10px 18px darkgrey;
    background: white;
}

.active {
    background-color: #4CAF50;
}

.center {
    text-align: center;

}
```

Program untuk back-end web

web.py

```
from flask import Flask, redirect, url_for, render_template, request
import webFunctions
app = Flask(__name__)

'''
File ini berisi fungsi untuk back-end dari website
'''

@app.route("/", methods=["POST", "GET"])
def home():
  if request.method == "POST":
    inputFiles = request.form.getlist("filename")
    inputKeyword = request.form["keyword"]
    #print(isikey == "sedang")
    method = request.form["algorithm"]
    webFunctions.extractKalimat(inputFiles,inputKeyword,method)

    return render_template("homePage.html", algoritma=method,
```

```
    folder=inputFiles, key=inputKeyword, jumlah=webFunctions.numsResult , tan
    ggal=webFunctions.datesResult, kalimat=webFunctions.sntcResult, sumber=we
    bFunctions.source)
  else:
    return render_template("homePage.html", algoritma="", folder="",
    key = "", number=[] , tanggal=[], result=[], source=[])

if __name__ == "__main__":
    app.static_folder = 'static'
    app.run(debug=True)
```

webFunctions.py

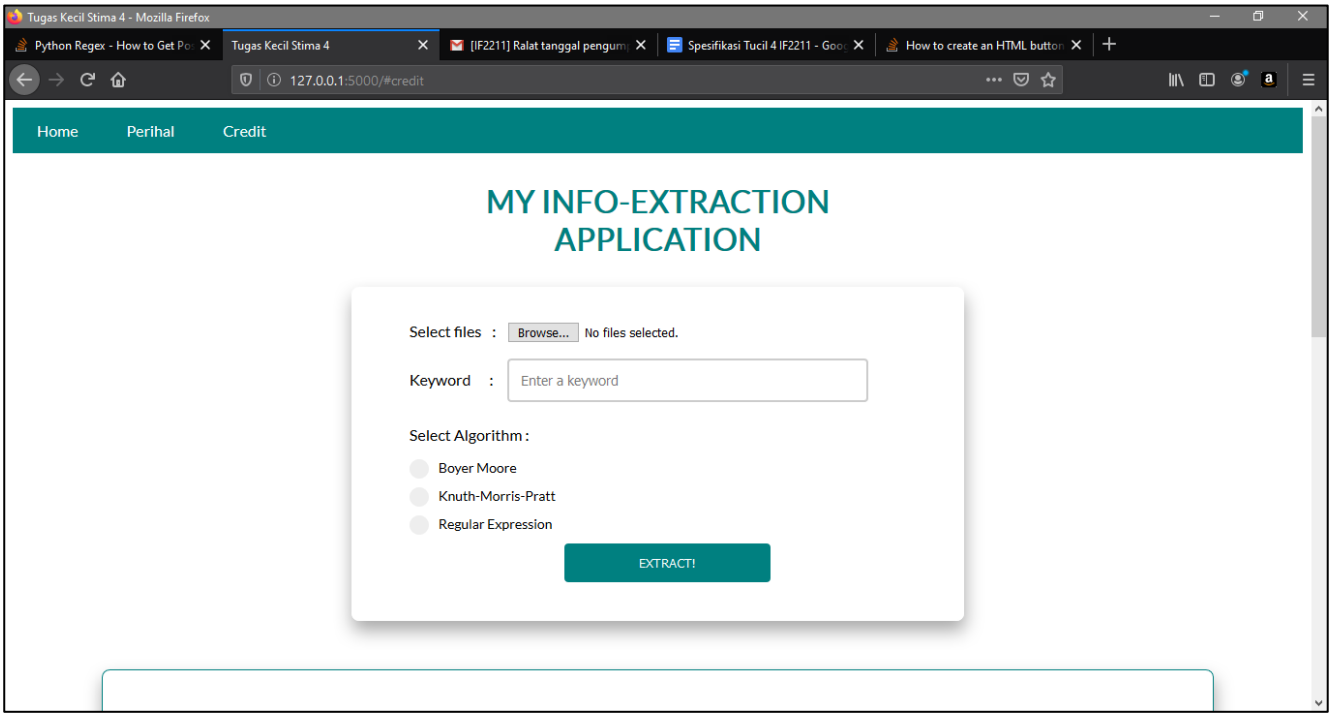
```
import util
import KMP
import BM
import regularExpression as regexp

'''
File berisi kumpulan fungsi untuk back-end website
'''

tglBerita = ""
source = []
datesResult = []
numsResult = []
sntcResult = []

'''
Fungsi untuk mengekstraksi berita dengan mengambil tanggal berita
dan melakukan dekomposisi text berita dari source menjadi kalimat-kalimat
'''
def extractBerita(source) :
    global tglBerita

    f = open(source)
    text = f.read()

    sentences = util.splitText(text)
    headerBerita = sentences[0].split("\n")
    tglBerita = headerBerita[2]
    kalimatBerita = []
    for sentence in headerBerita:
        kalimatBerita.append(sentence)
    for i in range (1, len(sentences)):
        kalimatBerita.append(sentences[i])
```

```
    return kalimatBerita

'''
Prosedur untuk memproses sebuah kalimat pada berita dengan memeriksa
apakah pada kalimat terdapat angka dan tanggal lalu memsaukkan angka
dan tanggal yang ditemukan serta kalimat tersebut dan sumber berikta
ke array hasil yang sesuai
'''
def prosesKalimat(kalimat, berita, foundIndex):
    global source
    global datesResult
    global numsResult
    global sntcResult
    global tglBerita


    numsTemp = util.findNum(kalimat)
    if(len(numsTemp)>0):
        print(numsTemp)
        numsResult.append(util.searchClosestNum(numsTemp, kalimat, foundIndex
))
        source.append(berita)
        sntcResult.append(kalimat)

        datesTemp = util.findDate(kalimat)
        lenAwal = len(datesResult)
        if(len(datesTemp) > 0):
            for i in range (len(datesTemp)):
                if(len(datesTemp[i])>=8):
                    datesResult.append(datesTemp[i])

            if(len(datesResult) == lenAwal):
                datesResult.append(tglBerita)
        else:
            datesResult.append(tglBerita)

'''
Prosedur untuk melakukan pencocokan string text pada setiap
filesOfBerita dengan pattern keyword menggunakan method algoritma
tertentu
'''
def extractKalimat(filesOfBerita, keyword, method):
    global source
    global datesResult
    global numsResult
    global sntcResult
```

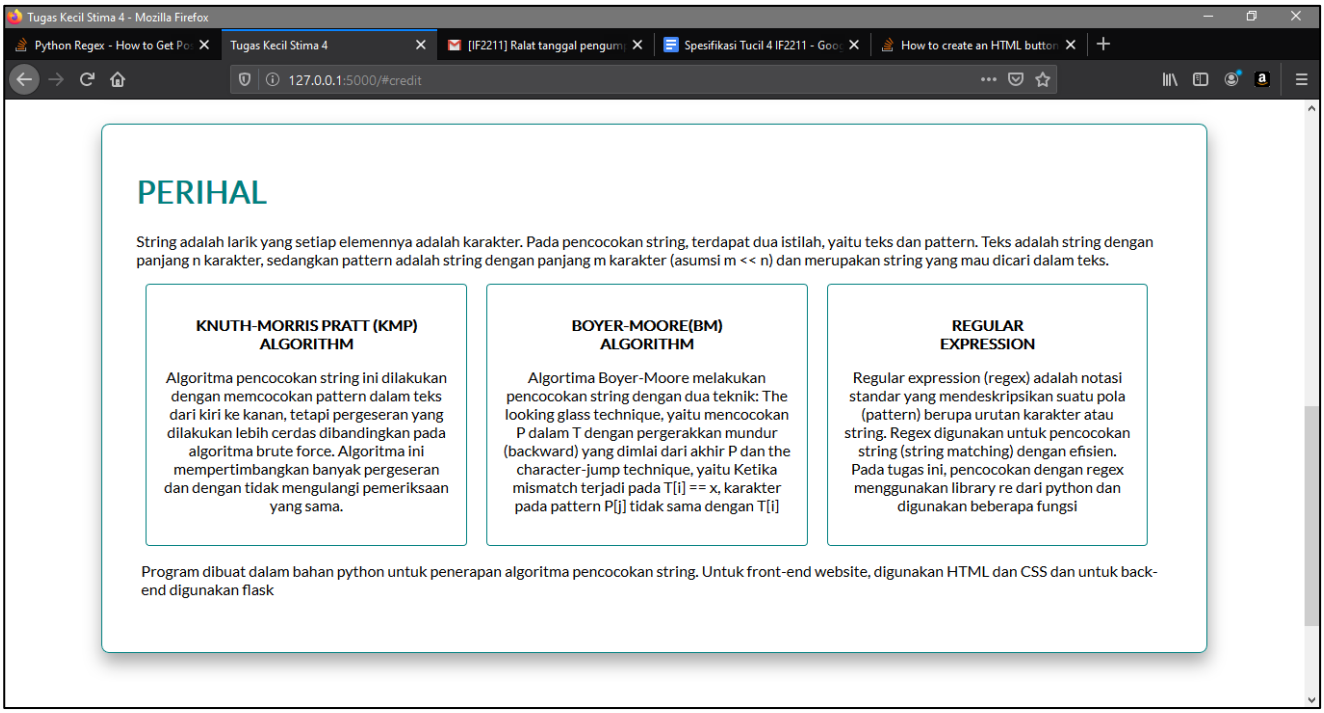
```
    source.clear()
    datesResult.clear()
    numsResult.clear()
    sntcResult.clear()

    for berita in filesOfBerita:
        kalimatBerita = extractBerita("../test/"+berita)

        if (method == "kmp"):
            for kalimat in kalimatBerita :
                foundIndex = KMP.kmpAlgorithm(kalimat, keyword)
                if (foundIndex != -1):
                    prosesKalimat(kalimat, berita, foundIndex)

        elif(method == "bm"):
            for kalimat in kalimatBerita :
                foundIndex = BM.bmAlgorithm(kalimat, keyword)
                if (foundIndex != -1):
                    prosesKalimat(kalimat, berita, foundIndex)
        else : #method == "regex"
            for kalimat in kalimatBerita :
                foundIndex = KMP.kmpAlgorithm(kalimat, keyword)
                if (foundIndex != -1):
                    prosesKalimat(kalimat, berita, foundIndex)
```

3. *Screenshot Input-Output* Program

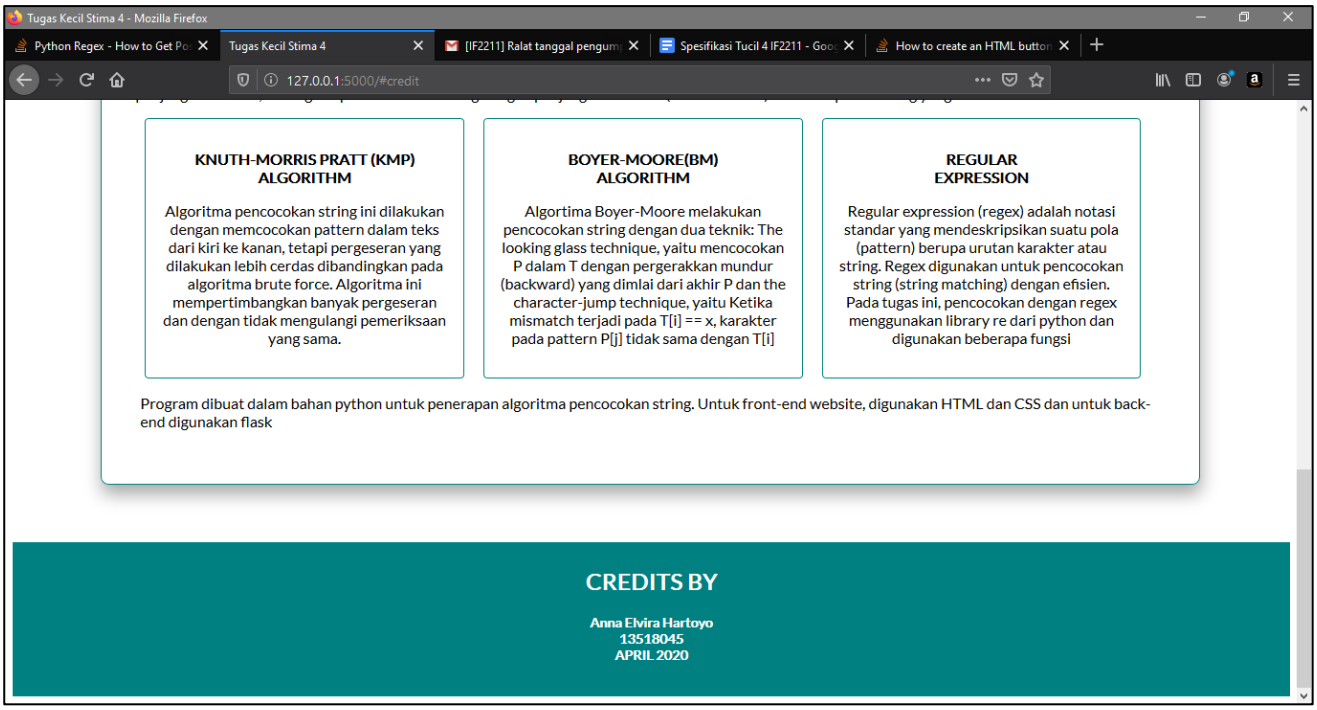
Berikut screen shot tampilan website secara keseluruhan:



Gambar 5. Tampilan awal homepage website



Gambar 6. Tampilan bagian perihal



Gambar 7. Tampilan bagian credit

Seluruh teks berita terdapat pada folder test. Text berita harus memiliki format khusus pada tiga baris pertama, yaitu baris pertama adalah judul berita, baris kedua adalah sumber dan penulis berita, dan baris ketiga adalah tanggal berita. Ketiga baris dipisahkan oleh enter.

Berikut contoh dari file teks berita1.txt

Jumlah Positif Corona Jatim Stagnan, Pasien Sembuh Bertambah
CNN Indonesia
Sabtu, 04/04/2020 20:20 WIB

Sebelumnya, jumlah pasien sembuh Corona COVID-19 semakin hari semakin bertambah di Jawa Timur (Jatim). Terbaru, ada ada penambahan empat orang sehingga total saat ini menjadi 96 orang dari sebelumnya 92 orang.

Rinciannya, dua pasien sembuh dari Kota Surabaya, satu pasien masing-masing dari Kabupaten Bondowoso dan Bangkalan.

"Kami bersyukur hari ini ada empat orang sembuh. Sehingga total ada 96 orang pasien yang sebelumnya positif, sekarang negatif di Jatim, atau setara 18,4 persen," kata Gubernur Jatim, Khofifah Indar Parawansa di Gedung Negara Grahadi Surabaya, Jumat, 17 April 2020.

Khofifah menyampaikan terimakasih kepada seluruh tim medis telah memberikan pelayanan terbaik, dalam menyembuhkan pasien Corona COVID-19. Khofifah berharap jumlah pasien sembuh korona di Jatim terus bertambah.

"Saya sampaikan terimakasih kepada para tim medis, telah memberikan dedikasi dan pelayanan terbaiknya. Mudah-mudahan jumlah pasien sembuh terus bertambah, seperti harapan kita bersama," ujar dia.

Sementara itu, ada penambahan dua orang meninggal dunia akibat COVID-19, sehingga totalnya menjadi 48 orang atau setara dengan 9,2 persen.

"Kami juga menyambung kabar duka bahwa ada dua orang yang meninggal, satu dari Lamongan dan satu dari Surabaya," ucap Khofifah.

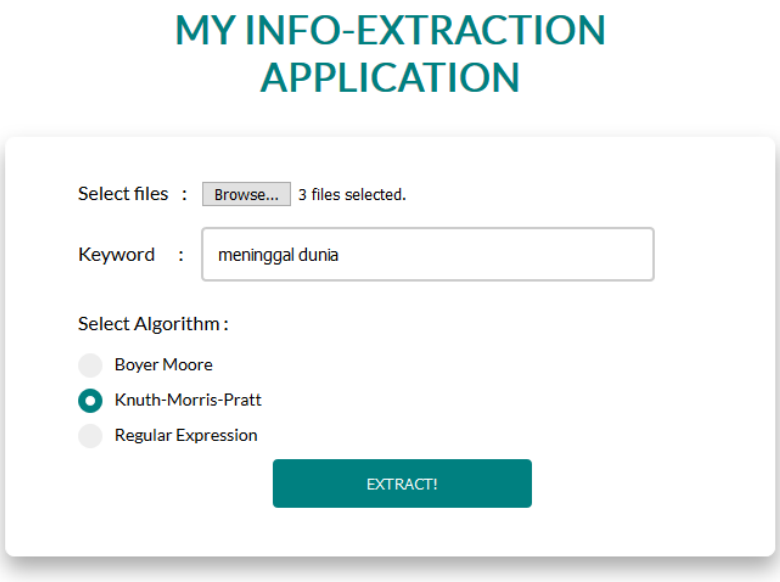
Sementara untuk jumlah Pasien Dalam Pengawasan (PDP) di Jatim ada 1.826 orang, dan 1.014 orang di antaranya masih dalam pemantauan. Sedangkan Orang Dalam Pemantauan (ODP) sebanyak 15.942 orang, dan yang masih dipantau ada 7.278 orang.

Teks lengkap berita lain (berita2.txt sampai berita10.txt) dapat dpilihat pada folder test

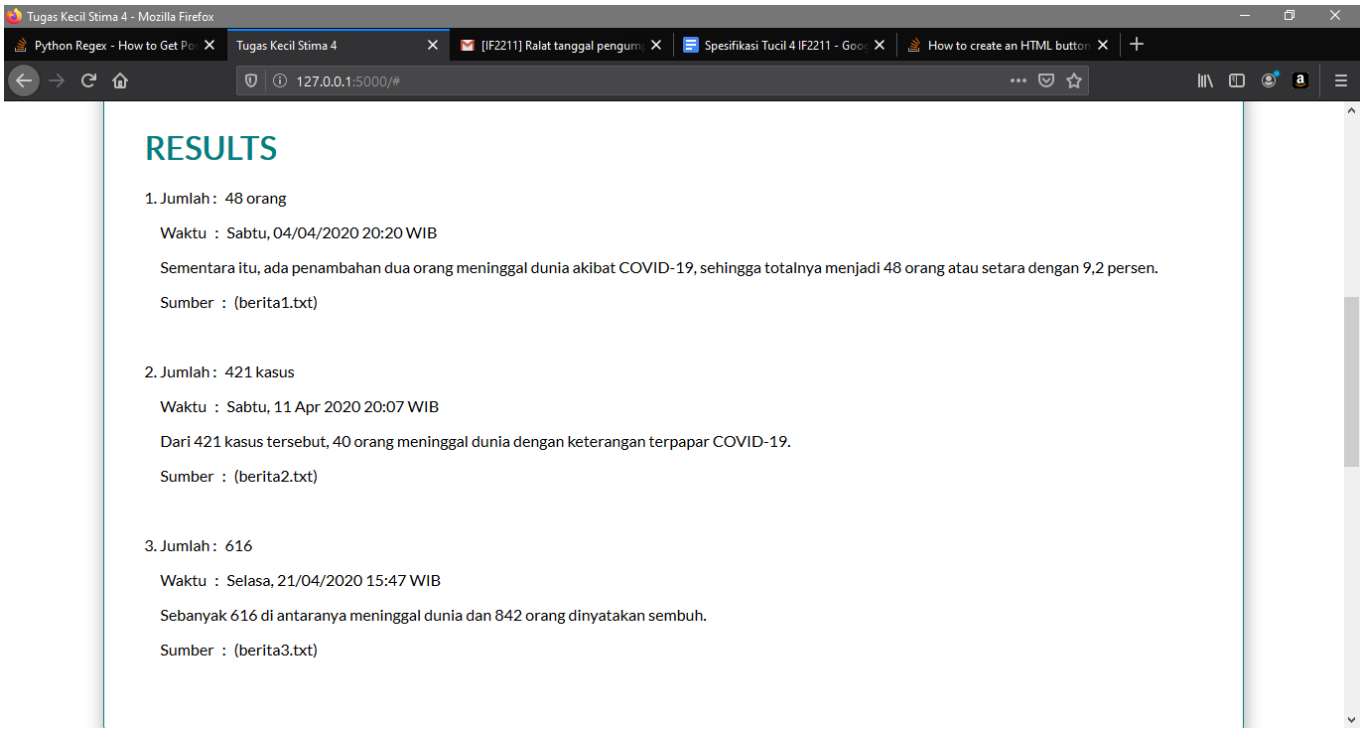
Berikut input-output program untuk beberapa keyword yang dicari pada teks berita dengan berbagai metode algoritma:

a) Pengujian 1

Keyword “meninggal dunia” dari teks berita1.txt, berita2.txt, dan berita3.txt dengan metode KMP



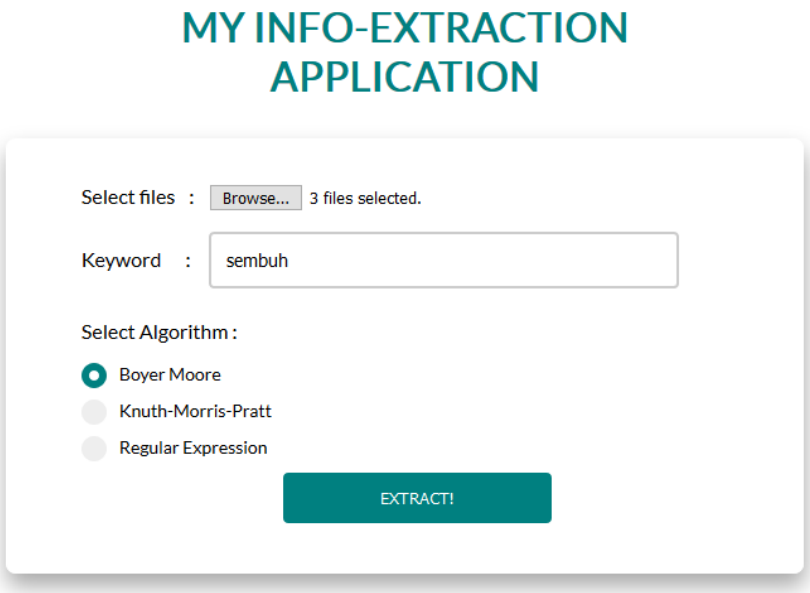
Gambar 8. Input program untuk pengujian 1



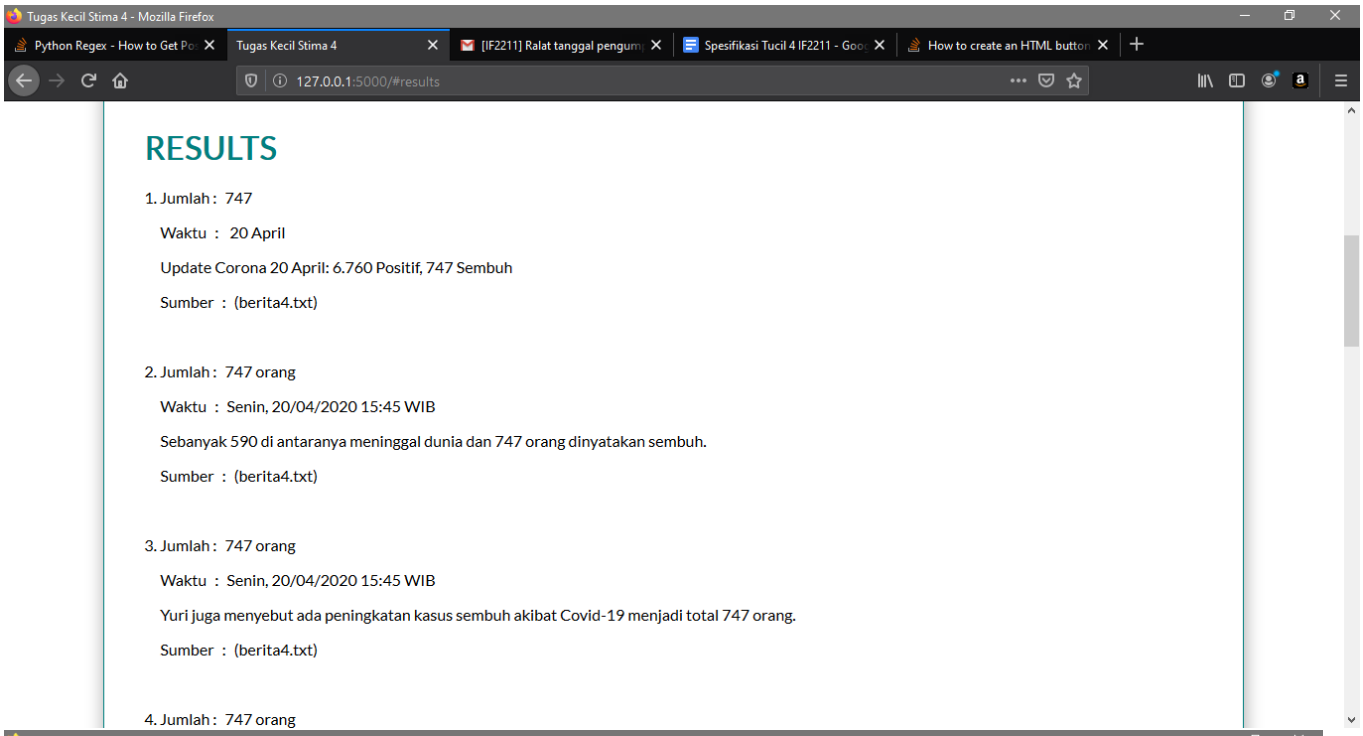
Gambar 9. Output program untuk pengujian 1

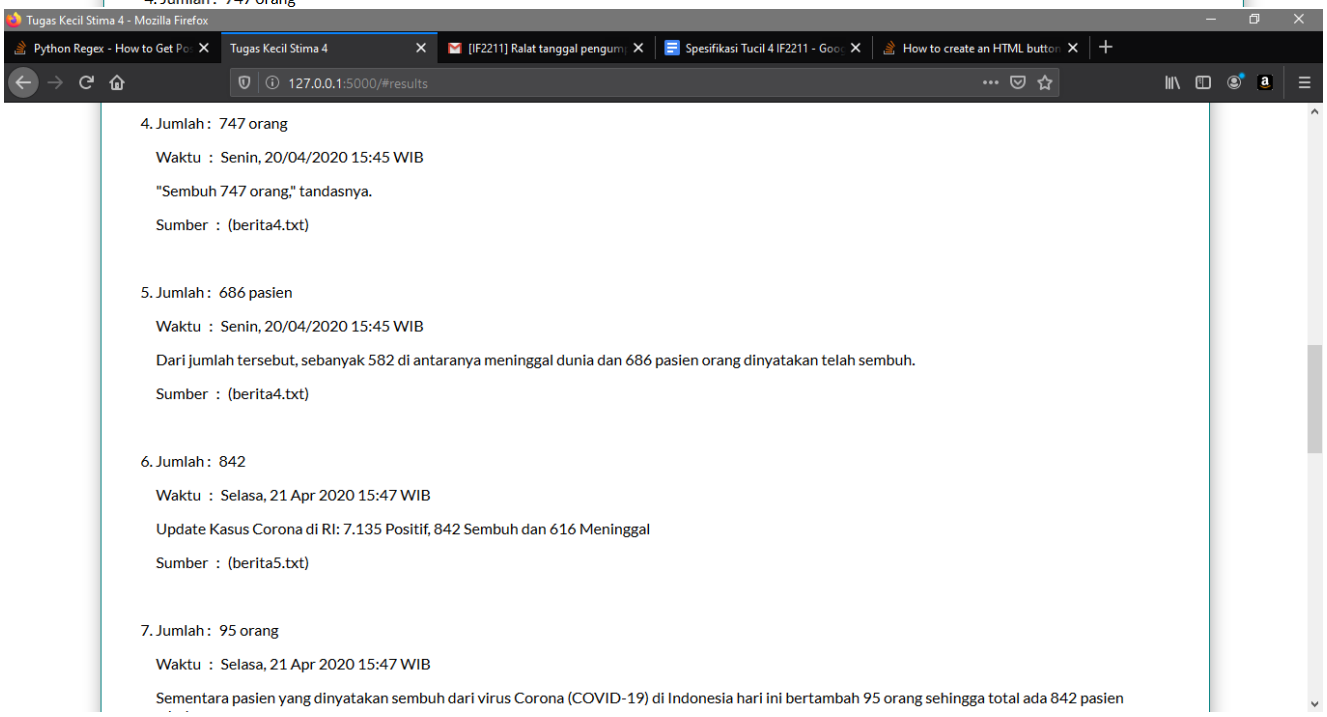
b) Pengujian 2

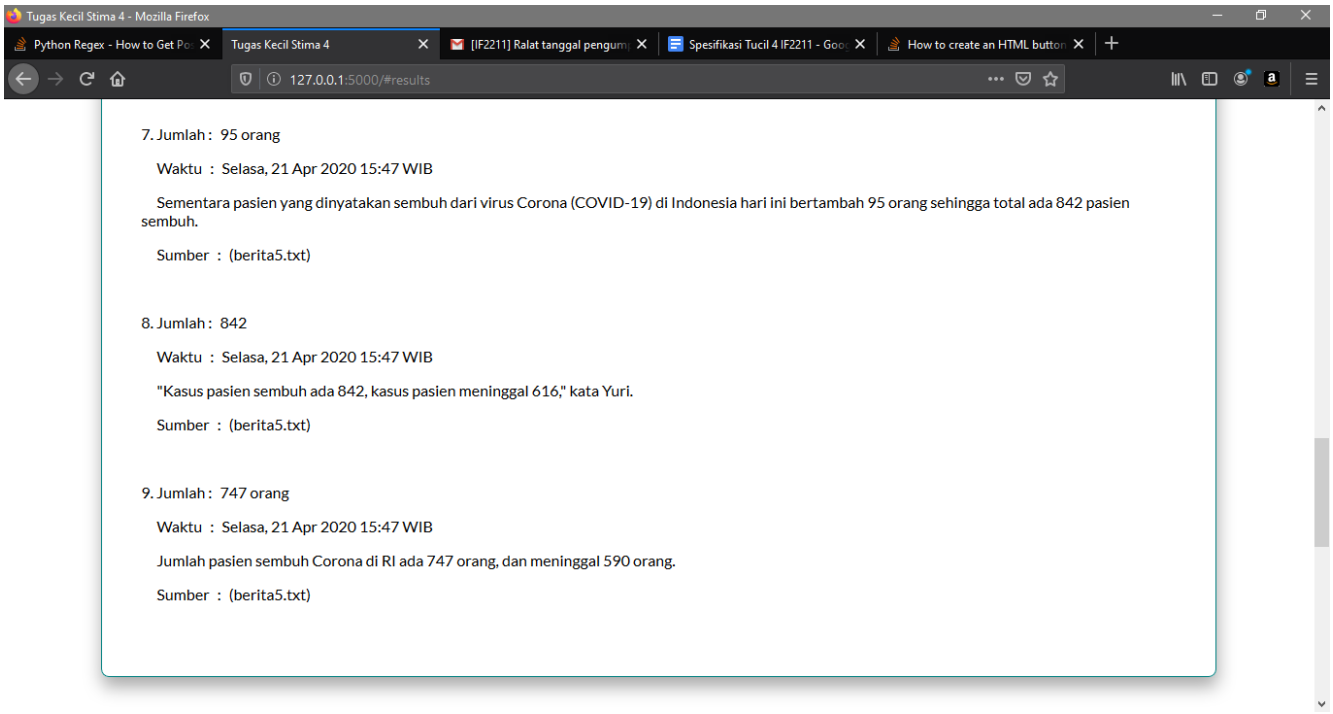
Keyword “sembuh” dari teks berita4.txt, berita5.txt, dan berita6.txt dengan metode BM



Gambar 10. Input program untuk pengujian 2


Tugas Kecil Stima 4 - Mozilla Firefox
Python Regex - How to Get Po
Tugas Kecil Stima 4
[IF2211] Ralat tanggal pengum
Spesifikasi Tucil 4 IF2211 - Goo
How to create an HTML button
127.0.0.1:5000/#results
RESULTS
1. Jumlah : 747
Waktu : 20 April
Update Corona 20 April: 6.760 Positif, 747 Sembuh
Sumber : (berita4.txt)
2. Jumlah : 747 orang
Waktu : Senin, 20/04/2020 15:45 WIB
Sebanyak 590 di antaranya meninggal dunia dan 747 orang dinyatakan sembuh.
Sumber : (berita4.txt)
3. Jumlah : 747 orang
Waktu : Senin, 20/04/2020 15:45 WIB
Yuri juga menyebut ada peningkatan kasus sembuh akibat Covid-19 menjadi total 747 orang.
Sumber : (berita4.txt)
4. Jumlah : 747 orang

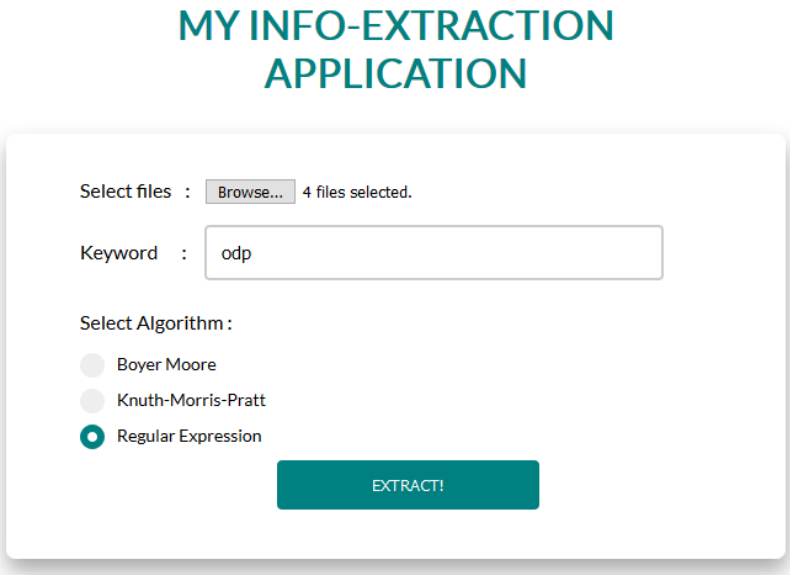

Tugas Kecil Stima 4 - Mozilla Firefox
Python Regex - How to Get Po
Tugas Kecil Stima 4
[IF2211] Ralat tanggal pengum
Spesifikasi Tucil 4 IF2211 - Goo
How to create an HTML button
127.0.0.1:5000/#results
4. Jumlah : 747 orang
Waktu : Senin, 20/04/2020 15:45 WIB
"Sembuh 747 orang," tandasnya.
Sumber : (berita4.txt)
5. Jumlah : 686 pasien
Waktu : Senin, 20/04/2020 15:45 WIB
Dari jumlah tersebut, sebanyak 582 di antaranya meninggal dunia dan 686 pasien orang dinyatakan telah sembuh.
Sumber : (berita4.txt)
6. Jumlah : 842
Waktu : Selasa, 21 Apr 2020 15:47 WIB
Update Kasus Corona di RI: 7.135 Positif, 842 Sembuh dan 616 Meninggal
Sumber : (berita5.txt)
7. Jumlah : 95 orang
Waktu : Selasa, 21 Apr 2020 15:47 WIB
Sementara pasien yang dinyatakan sembuh dari virus Corona (COVID-19) di Indonesia hari ini bertambah 95 orang sehingga total ada 842 pasien



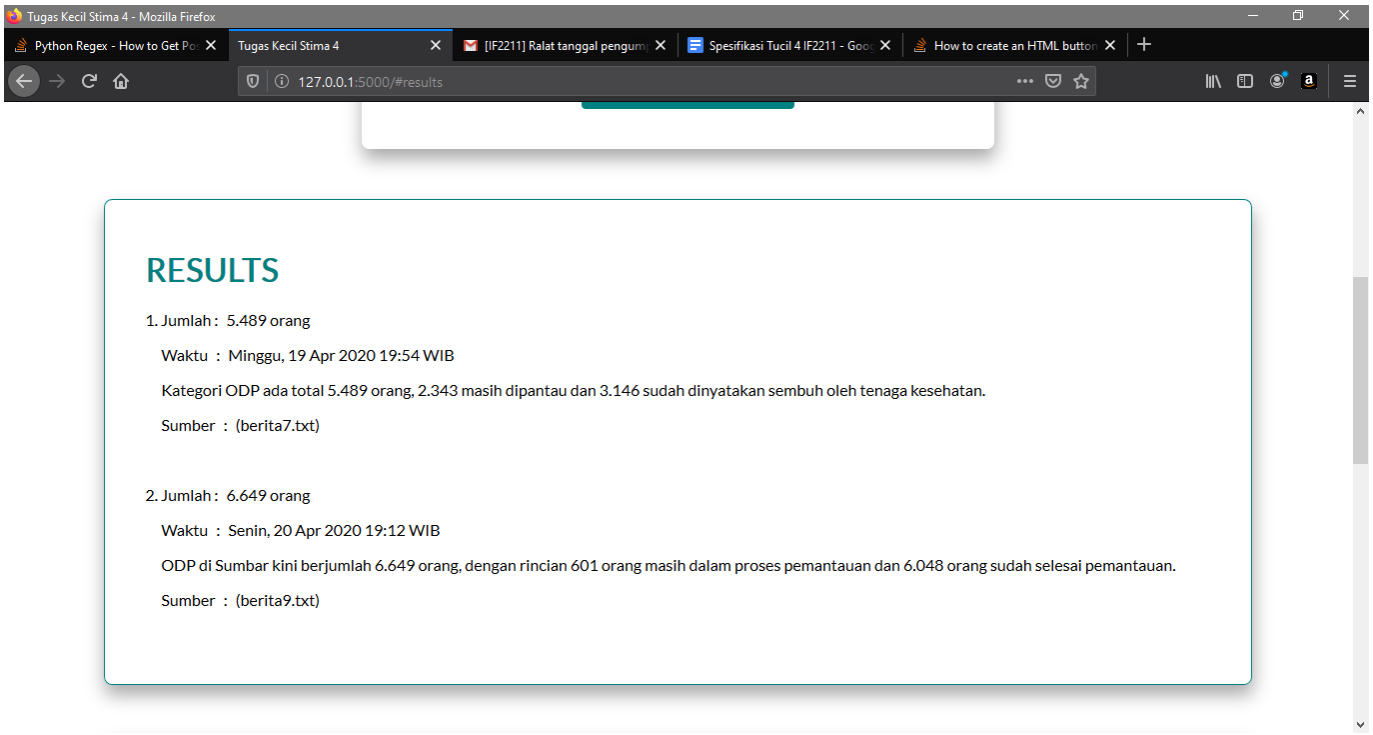
Gambar 11. Output program untuk pengujian 2

c) Pengujian 3

Keyword “odp” dari teks berita7.txt, berita8.txt, berita9.txt, dan berita10.txt dengan mentode regex



Gambar 12. Input program untuk pengujian 3



Gambar 13. Output program untuk pengujian 3

4. Lampiran

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil *running*	√	
3. Program dapat menerima input dan menuliskan output	√	
4. Luaran sudah benar untuk data uji	√	

5. Referensi

Munir, Rinaldi. 2018 *Pattern Matching*. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf diakses tanggal 21 April 2020

Munir, Rinaldi. 2019. *String Matching dengan Regular Expression*. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf diakses tanggal 21 April 2020