LAPORAN TUGAS KECIL 4
IF2211/STRATEGI ALGORITMA
Semester II tahun 2019/2020

Ekstraksi Informasi dari Artikel Berita dengan Algoritma Pencocokan String

Oleh:

13518004 Qurrata A'yuni

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 4

BAB I
Algoritma Pencocokan *String* Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Regular Expression (Regex)

1. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Pencocokan *String* Knuth-Morris-Pratt (KMP) adalah algoritma yang melakukan pencarian sebuah *pattern* di dalam teks dengan urutan kiri-ke-kanan (seperti algoritma *Brute Force*). Tetapi algoritma ini melakukan pergeseran lebih cerdas dibandingkan algoritma *Brute Force* yang melakukan pencocokan pada setiap karakter.

Langkah-langkah yang dilakukan algoritma KMP pada saat pencocokan *string*:

- Algoritma KMP mulai melakukan pencocokan *pattern* pada awal teks.
- Dari kiri ke kanan, algoritma ini akan melakukan pencocokan karakter per karakter pada *pattern* dengan karakter teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
    - Karakter pada *pattern* dan pada teks yang dibandingkan tidak cocok.
    - Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahukan penemuan posisinya.
- Algoritma kemudian menggeser pattern berdasarkan *table next*, lalu mengulangi langkah kedua sampai *pattern* berada di ujung teks.

Preprocess algoritma KMP adalah menemukan prefiks yang cocok dengan *pattern* dengan *pattern* itu sendiri. Dengan j adalah posisi *mismatch* di P dan k adalah posisi sebelum *mismatch* terjadi (k=i-1). Fungsi pinggiran b(k) didefinisikan sebagai ukuran prefiks terbesar dari P[0..k] yang juga merupakan sufiks dari P[1..k].

Kompleksitas waktu untuk menghitung fungsi pinggiran adalah $O(m)$, sedangkan untuk pencarian *string* $O(n)$, sehingga kompleksitas waktu algoritma KMP adalah $O(m+n)$ sangat cepat dibandingkan *Brute Force* ketika *worst case* terjadi kompleksitasnya $O(mn)$.

Kekurangan algoritma ini adalah tidak bekerja dengan baik bila alfabet bertambah. Akan terjadi banyak *mismatch*, dan cenderung terjadi pada awal *pattern,* tetapi algoritma ini lebih cepat ketika mismatch terjadi setelahnya.

2. Algoritma Boyer-Moore (BM)

   Algoritma Pencocokan *String* Boyer-Moore didasari dengan dua teknik.
   - *The looking-glass technique*
     Mencari P pada T dengan berpindah secara *backward* pada P, dimulai dari akhir.
   - *The character-jump technique*
     Ketika *mismatch* terjadi di T[i] == x.
     Karakter pada *pattern* P[j] tidak sama dengan T[i]

   Langkah yang dilakukan algoritma BM pada saat pencocokan *string* adalah:
   - Mencocokkan *pattern* pada awal teks.
   - Dari kanan ke kiri, algoritma akan mencocokkan karakter per karakter dengan karakter teks yang bersesuaian. Sampai salah satu kondisi berikut dipenuhi: karakter pada *pattern* dan pada teks yang dibandingkan tidak cocok dan semua karakter pada *pattern* cocok. Algoritma ini akan memberitahu penemuan posisinya.
   - Algoritma kemudian menggeser *pattern* dengan memaksimalkan nilai pergeseran *good-suffix* dan pergeseran *bad-character*, lalu mengulangi langkah kedua sampai *pattern* berada di ujung teks.

   Running time algoritma ini adalah $O(nm + A)$ ketika kasus terburuk terjadi. Tetapi algoritma ini cepat ketika alfabet (A) besar, dan lama ketika alfabetnya kecil. Misalnya cocok untuk teks berbahasa Inggris, namun tidak bagus untuk *binary.*

3. Regular Expression (Regex)

   Regular Expression adalah notasi standar yang mendeskripsikan suatu pola (*pattern)* berupa urutan karakter atau *string. Regex* digunakan untuk pencocokan *string* dengan efisien. *Regex* sudah menjadi standar yang tersebar di semua *tools* dan bahasa pemrograman.

   Dalam *Regex* terdapat sejumlah karakter khusus yang mempengaruhi proses pencocokan *string.* Contohnya, karakter titik "." akan cocok dengan karakter apapun. Contoh lainnya dapat dipelajari melalui website berikut https://www.regextester.com/ bila menggunakan bahasa Python dokumentasi pustaka *Regex* dapat dilihat di https://docs.python.org/3/library/re.html/

BAB II
Kode Program

1. File kmp.py

```
def compute_fail(pattern):
 fail = [-9999 for i in range (len(pattern))]
 fail[0] = 0

 m = len(pattern)
 j = 0
 i = 1
  while (i < m):
   if (pattern[i] == pattern[j]):
     fail[i] = 1 + j
     i += 1
     j += 1
   elif (j > 0):
     j = fail[j-1]
   else:
     fail[i] = 0
     i += 1

 return fail

def kmp_match(text, pattern):
 n_text = text.lower()
 n_pattern = pattern.lower()
 n = len(n_text)
 m = len(n_pattern)

 fail = compute_fail(n_pattern)
  i = 0
 j = 0
  while (i < n):
   if (n_pattern[j] == n_text[i]):
     if (j == m-1):
       return i-m+1
     i += 1
     j += 1
   elif (j > 0):
     j = fail[j-1]
   else:
     i += 1

 return -999
```

2. File bm.py

```
def build_last(pattern):
   last = [-1]*256

   for i in range(len(pattern)):
       last[ord(pattern[i])] = i;

   return last
```

```
def bm_match(text, pattern):
  n_text = text.lower()
  n_pattern = pattern.lower()

  m = len(n_pattern)
  n = len(n_text)

  last = build_last(n_pattern)

  s = 0

  while(s <= n-m):
      j = m-1

      while j>=0 and n_pattern[j] == n_text[s+j]:
          j -= 1
       if j<0:
          # pattern match
          return s
          s += (m-last[ord(n_text[s+m])] if s+m<n else 1)
      else:
          s += max(1, j-last[ord(n_text[s+j])])

  return -999
```

3. File regex.py

```
import re

def regex_match(text, pattern):
   found = re.findall(pattern.lower(), text.lower())
   for match in re.finditer(pattern, text):
       start, end = match.span()
       return start
   return -999
```

4. File kelas Information.py

```
class Information:

   def __init__ (self):
       self.cases = 0
       self.date = "XX/XX/XXXX"
       self.sentence = "EMPTY"
       self.name_of_file = "NONE"

   def __init__ (self, cases, date, sentence, name_of_file):
       self.cases = cases
       self.date = date
       self.sentence = sentence
       self.name_of_file = name_of_file

   def set_cases (self, cases):
       self.cases = cases

   def set_date (self, date):
       self.date = date
```

```
    def set_sentence (self, sentence):
        self.sentence = sentence

    def set_name_of_file (self, name_of_file):
        self.name_of_file = name_of_file

    def print_info(self):
        print("Waktu        :", self.date)
        print("Kasus        :", self.cases)
        print("Kalimat      :", self.sentence)
        print("Nama file    :", self.name_of_file)
        print()
```

5. File load_txt.py

```
import os
def break_into_sentence(name_file):
    list_of_sentence = []
    sentence = ""
    file = open(name_file, 'r')
    char = ""
    while (True):
        prev_char = char
        char = file.read(1)
        sentence += char
        if (char == ' ' and prev_char == ".") or char == '\n':
            sentence = sentence.replace('\n','')
            sentence += '\n'
            list_of_sentence.append(sentence)
            sentence = ""
        if not char:
            break
    file.close
    return list_of_sentence

def get_file_name(folder):
    list_file_name = []
    for dirname, _, filenames in os.walk(folder):
        for filename in filenames:
            list_file_name.append(os.path.join(dirname,filename))
    return list_file_name
```

6. File kasus_waktu.py

```
import re

def find_jam (sentence):
    results = []
    for match in re.finditer(r'((1[0-2]|0?[1-9])(?:.|:)([0-5][0-9])
?([AaPpWw][MmIi][BbTt](?:|[Aa])))',sentence.upper()):
        results.append(match.group(0))
    for match in
re.finditer(r'^([0-9]|0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]',sentence.lower()):
        results.append(match.group(0))
    print(results)
    if len(results) != 0:
        for res in results:
            return res + " "
        return ""
```

```
   else:
       return ""
def find_hari (sentence):
   results =
re.findall(r'(?:Senin|Monday|Selasa|Tuesday|Rabu|Wednesday|Kamis|Thursday|Jumat|Friday
|Sabtu|Saturday|Minggu|Sunday)',sentence)
   results = sorted(results,key=None,reverse=True)
   if len(results) != 0:
       for res in results:
           return res + " "
       return ""
   else:
       return ""

def find_bulan (sentence):
   res =
re.findall(r'\d{0,2}\s{0,1}(?:Januari|January|Jan|February|Februari|Feb|Maret|March|Ma
r|April|Apr|May|Mei|June|Jun|Juni|July|Jul|Agustus|August|Agu|Aug|September|Sep|Octobe
r|Oktober|Oct|Okt|November|Nov|Desember|December|Des|Dec)\s{0,1}\d{0,2},{0,1}\s{0,1}\d
{0,4}', sentence)
   results = sorted(res,key=None,reverse=True)
   if len(results) != 0:
       for res in results:
           return res + " "
       return ''
   else:
       return ''
 def find_tgl (sentence):
   results =
re.findall(r'\d{1,2}/\d{1,2}/\d{1,4}|\d{1,4}/\d{1,2}/\d{1,2}|\d{1,2}-\d{1,2}-\d{1,4}|\
d{1,4}-\d{1,2}-\d{1,2}',sentence.lower())
   results = sorted(results,key=None,reverse=True)
   if len(results) != 0:
       for res in results:
           return res + " "
       return ""
   else:
       return ""

def find_tanggal (sentence, news_date_update):
   news_date = news_date_update
   jam = find_jam(sentence)
   hari = find_hari(sentence)
   bulan = find_bulan(sentence)
   tanggal = find_tgl(sentence)
   date = hari + bulan + tanggal + jam
   if date != "" and len(date.replace(" ","")) > 3:
       return date
   else:
       return news_date

def find_cases (sentence, idx_keyword):
   cases = re.findall(r'\b\d+\b', sentence.lower()) # menyimpan semua angka
   # print(cases)
   cases_idx = [] # menyimpan idx kemunculan
   idx = 0
   if len(cases) != 0:
       if len(cases) == 1:
           return cases[0]
       else :
```

```
            for match in re.finditer(r'\b\d+\b', sentence.lower()):
                start, end = match.span()
                cases_idx.append(start)
            # print(cases_idx)
            deltas = []
            for case_idx in cases_idx: # memasukkan selisih angka terdekat dengan
keyword
                deltas.append(abs(case_idx-idx_keyword))
            print(deltas)
            # mencari delta[i] terkecil
            min = deltas[0]
            for i in range (len(deltas)):
                if deltas[i] < min:
                    min = deltas[i]
                    idx = i
            # print(cases[idx])
            return cases[idx]
    return "0"
```

7. file extract_information.py

```
from bm import *
from kmp import *
from regex import *
from load_txt import *
from kasus_waktu import *
from Information import *

# menyimpan ke dalam list of information
def extract_information(folder, keyword, algorithm):
    files = get_file_name(folder) # menyimpan list of file
    texts = [] # menyimpan kalimat
    for file in files:
        texts.append(break_into_sentence(file))
    found = -999
    i = 0
    infos = [] # menyimpan list of information
    # mencari kemunculan keyword dalam kalimat
    for text in texts:
        for sentence in text:
            if algorithm.lower() == "kmp":
                found = kmp_match(sentence, keyword)
                if found != -999:
                    cases = find_cases(sentence, kmp_match(sentence, keyword))
                    date = find_tanggal(sentence, find_tanggal(texts[i][2],""))
                    file_name = files[i]
                    info = Information(cases, date, sentence, file_name)
                    infos.append(info)
            elif algorithm.lower() == "bm":
                found = bm_match(sentence, keyword)
                if found != -999:
                    cases = find_cases(sentence, bm_match(sentence, keyword))
                    date = find_tanggal(sentence, find_tanggal(texts[i][2],""))
                    file_name = files[i]
                    info = Information(cases, date, sentence, file_name)
                    infos.append(info)
            elif algorithm.lower() == "regex":
                found = regex_match(sentence, keyword)
                if found != -999:
```

```
                    cases = find_cases(sentence, regex_match(sentence, keyword))
                    date = find_tanggal(sentence, find_tanggal(texts[i][2],""))
                    file_name = files[i]
                    info = Information(cases, date, sentence, file_name)
                    infos.append(info)
        i += 1
    return infos
```

8. File app.py

```
from flask import Flask, render_template, flash, request, url_for
from extract_information import *

app = Flask(__name__)

@app.route('/')
def main():
    return render_template("app.html")

@app.route('/pranala')
def profile():
    return render_template("pranala.html")

@app.route('/send', methods=['POST'])
def send(sum=sum):

    if request.method == 'POST':
        folder = request.form['folderPath']
        keyword = request.form['keyword']
        algoritma = request.form['algoritma']
        infos = []
        if algoritma == 'kmp':
            try:
                infos = extract_information(folder, keyword, "kmp")
            except:
                infos = []
            sum = str(len(infos)) + " kalimat ditemukan"
            return render_template('app.html', sum=sum, items=infos)

        elif algoritma == 'bm':
            try:
                infos = extract_information(folder, keyword, "bm")
            except:
                infos = []
            sum = str(len(infos)) + " kalimat ditemukan"
            return render_template('app.html', sum=sum, items=infos)

        elif algoritma == 'regex':
            try:
                infos = extract_information(folder, keyword, "regex")
            except:
                infos = []
            sum = str(len(infos)) + " kalimat ditemukan"
            return render_template('app.html', sum=sum, items=infos)

if __name__ == "__main__":
    app.run(debug=true)
```

9. File pranala.html

```
<!DOCTYPE html>
<html>
<head>
<title>Halaman Profil</title>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.c
ss">
<style>
.card{
   box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
   max-width:300px;
   margin:auto;
   text-align:center;
   font-family:Arial, Helvetica, sans-serif;
   color:rgb(221, 208, 21);
   background-color: white;
}

.title{
   color:rgb(0, 0, 0);
   font-size:18px;
}
button{
   border:none;
   outline:0;
   display:inline-block;
   padding:8px;
   color:white;
   background-color:#000;
   text-align:center;
   cursor:pointer;
   width:100%;
   font-size:18px;
}

a {
   text-decoration:none;
   color:rgb(221, 208, 21);
}
   button:hover a:hover
{
   opacity:0.7;
}
</style>
</head>
 <body background="../static/background.jpg">
   <div class="card">
   <h2 style="text-align:center">Profil</h2>
       <img src="../static/foto.JPG" alt="yuni" style="width:100%";>
       <h1 style="color:rgb(221, 208, 21);">Qurrata A'yuni<h1>
         <h4 style="color:rgb(0, 0, 0);"> Teknik Informatika </h4>
         <h4 style="color:rgb(0, 0, 0);"> Institut Teknologi Bandung</h4>
       <div style="margin:24px 0;">
         <a href="https://www.twitter.com/"><i class="fa fa-twitter"></i></a>
         <a href="https://www.github.com/qurrata111"><i class="fa fa-github"></i></a>
       </div>
   <form action="/">
       <p><button> Go to My App </button></p>
```

```
    </form>
    </div>
  </body>
</html>
```

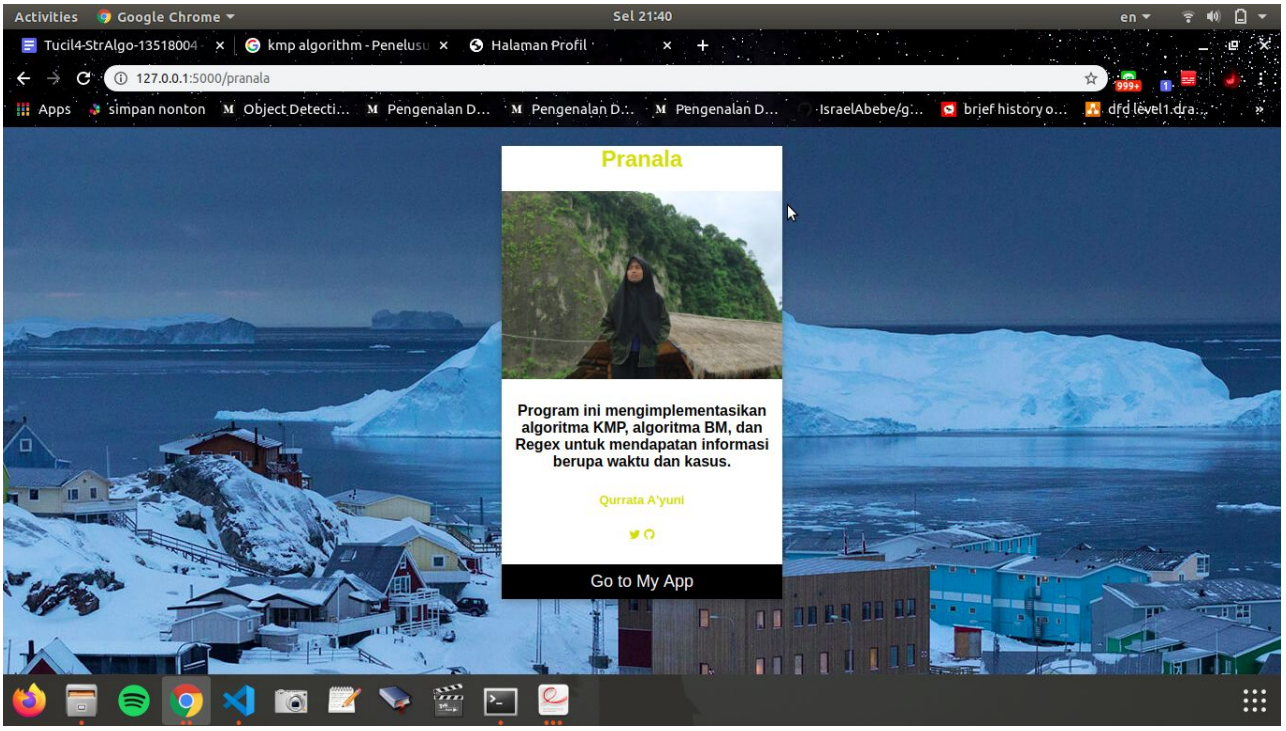
10. File app.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <meta http-equiv="X-UA-Compatible" content="ie=edge" />
   <title style="font-family: Arial, Helvetica, sans-serif;">Extraction Information
Application</title>
   <link
     rel="stylesheet"
     href="https://cdnjs.cloudflare.com/ajax/libs/skeleton/2.0.4/skeleton.min.css"
   />

   <style>
     .alert {
       background: rgb(221, 208, 21);
       padding: 1rem;
       border-radius: 5px;
       color: rgb(7, 7, 7);
       margin: 1rem;
       font-family: Arial, Helvetica, sans-serif;
     }

     .container {
       text-align:left;
       font-family:Arial, Helvetica, sans-serif;
       color:white;
     }

   </style>
 </head>
 <body background="../static/background.jpg">
   <div class="container">
     <h3 style="text-align: center">Extraction Information App</h3>
   </div>

   <!-- Form -->
   <div class="container">
     <form action="/send" method="POST">
       <label for="Alamat Folder">Alamat Folder</label>
       <input
         type="text"
         placeholder="Alamat Folder"
         class="u-full-width"
         name="folderPath"
       />

       <label for="Keyword">Keyword</label>
       <input
         type="text"
         placeholder="Keyword"
```

```
          class="u-full-width"
          name="keyword"
        />

        <label for="Algoritma">Algoritma</label>
        <select class="u-full-width" name="algoritma">
          <option value="kmp">Knuth-Morris-Pratt</option>
          <option value="bm">Boyer-Moore</option>
          <option value="regex">Regular Expression</option>
        </select>

        <input type="submit" value="Search" id="search_btn" style="color: white;"/>
        <br />
        <div class="alert">
          {{ sum }}
          <ul>
            {% for item in items %}
              <br> Kasus      : {{item.cases}} </br>
              <br> Waktu      : {{item.date}} </br>
              <br> Nama file  : {{item.name_of_file}} </br>
              <br> {{item.sentence}} </br>
              <hr> </hr>
            {% endfor %}
          </ul>
        </div>
      </form>

    <div style="text-align: center" style="font-size:20px">
        <b><a href="/pranala" style="color: white; font-size:30px;"> Klik pranala ini
</a></b>
    </div>
  </body>
</html>
```

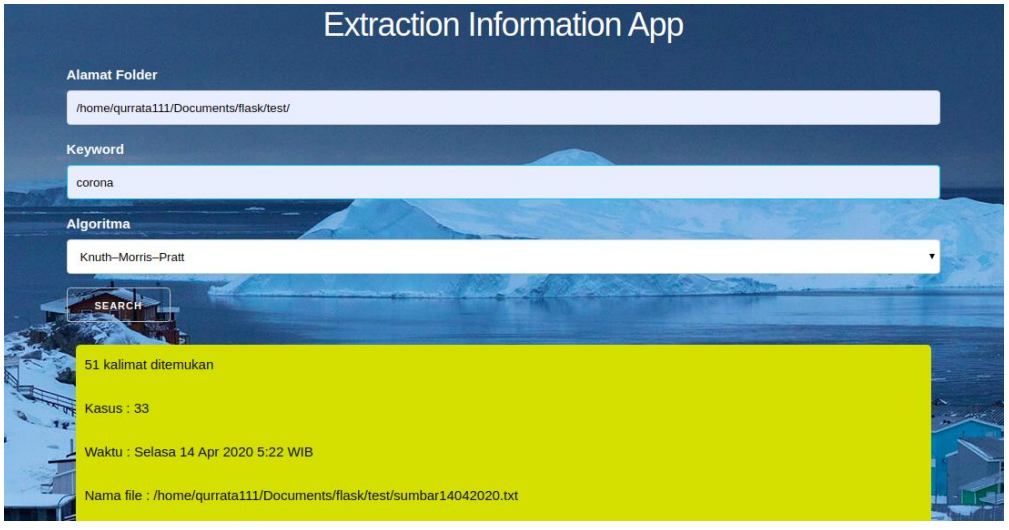
BAB III
Screenshot Input-Output Program

1. Pranala

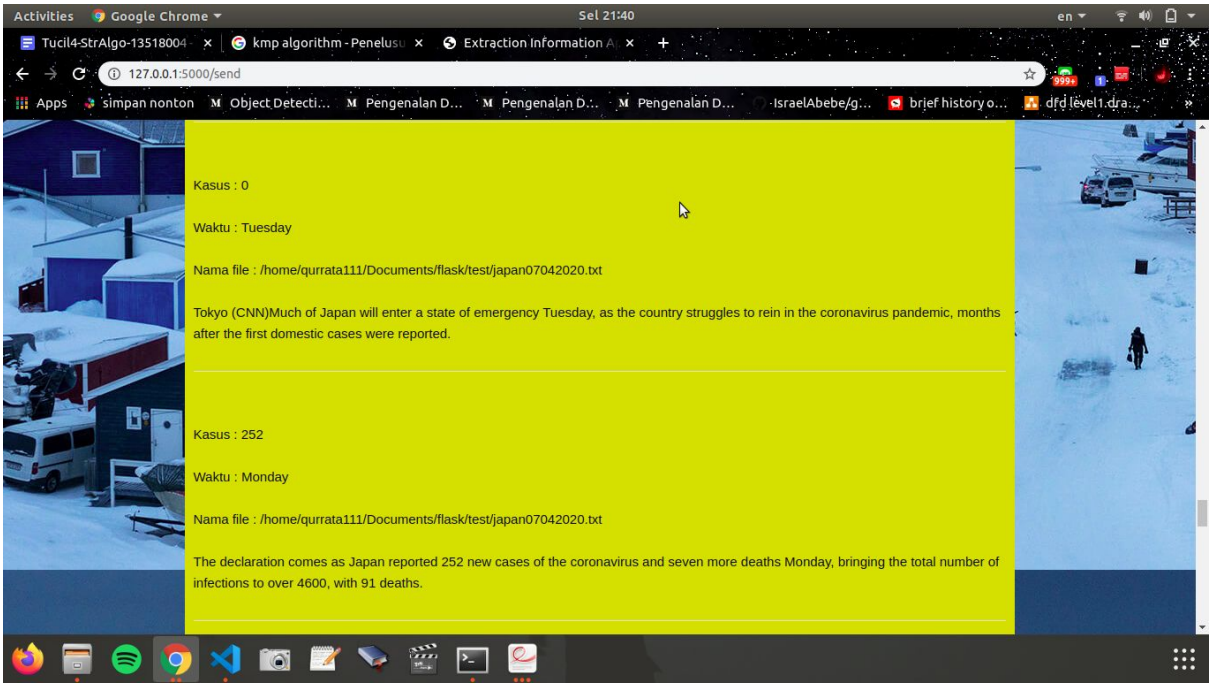


Gambar 1 Pranala

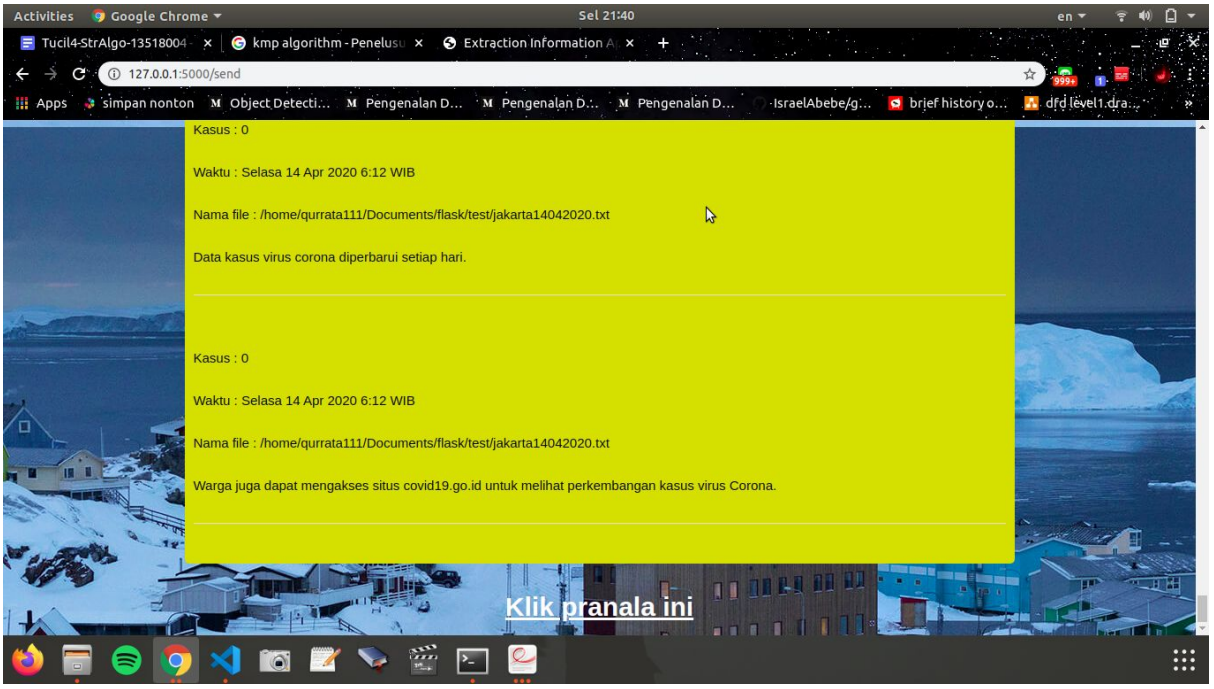
2. Pengujian program dengan algoritma KMP
    - Keyword ditemukan, dengan mencoba keyword “corona”



Gambar 2 Keyword ditemukan dengan algoritma KMP

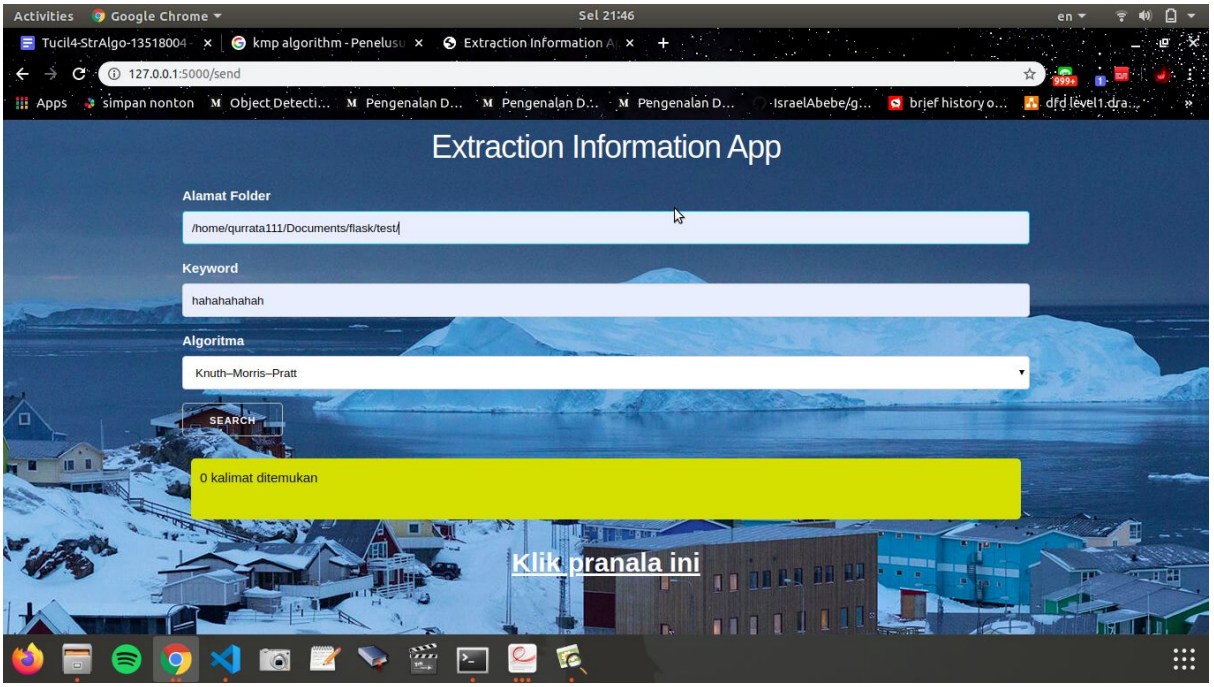


Gambar 3 Keyword ditemukan dengan algoritma KMP



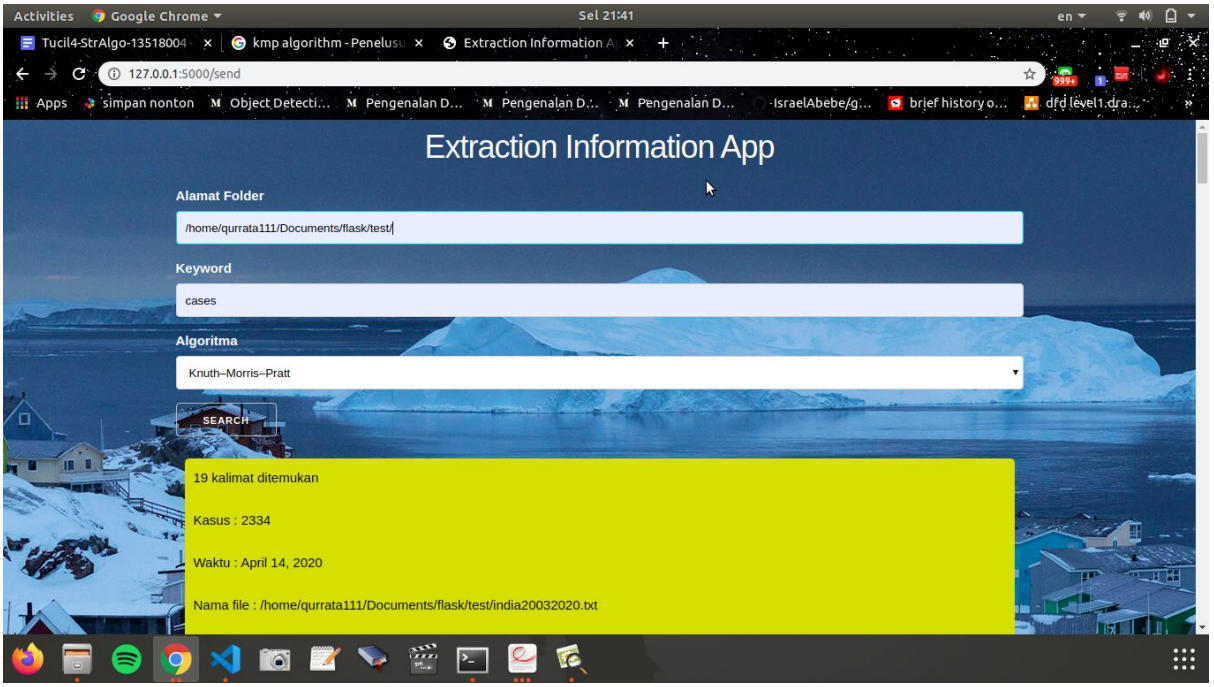
Gambar 4 Keyword ditemukan dengan algoritma KMP

- Keyword tidak ditemukan, dengan mencoba keyword “hahahahahah”

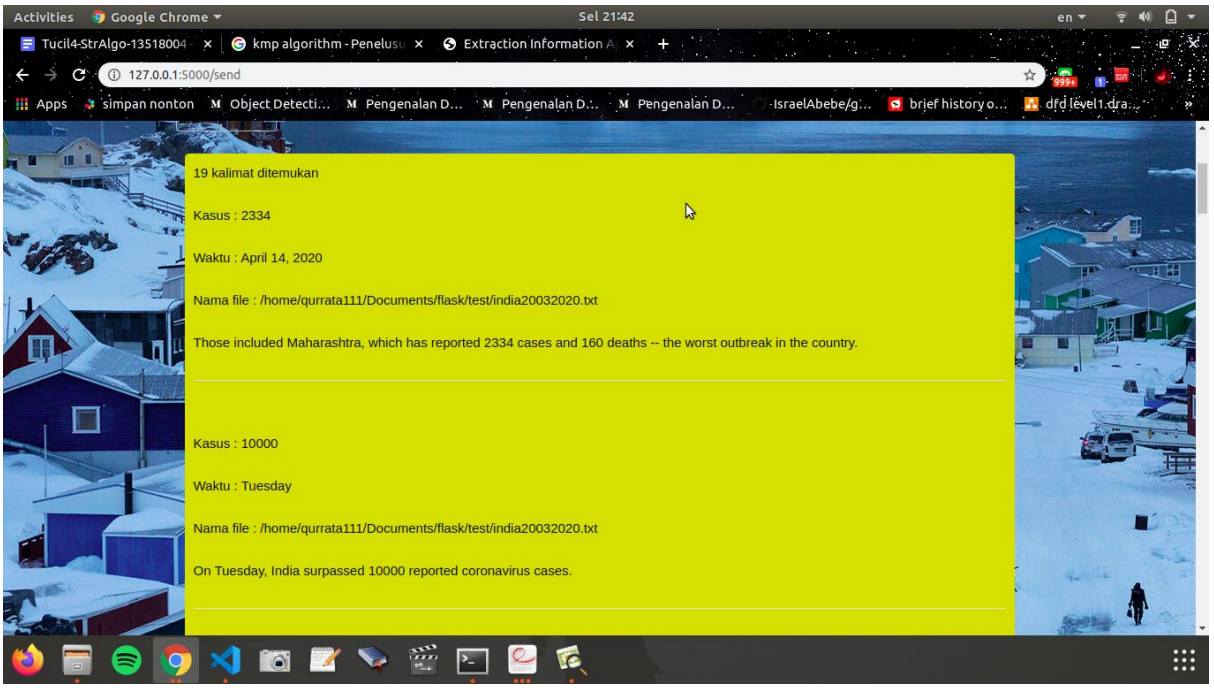


Gambar 5 Keyword tidak ditemukan dengan algoritma KMP

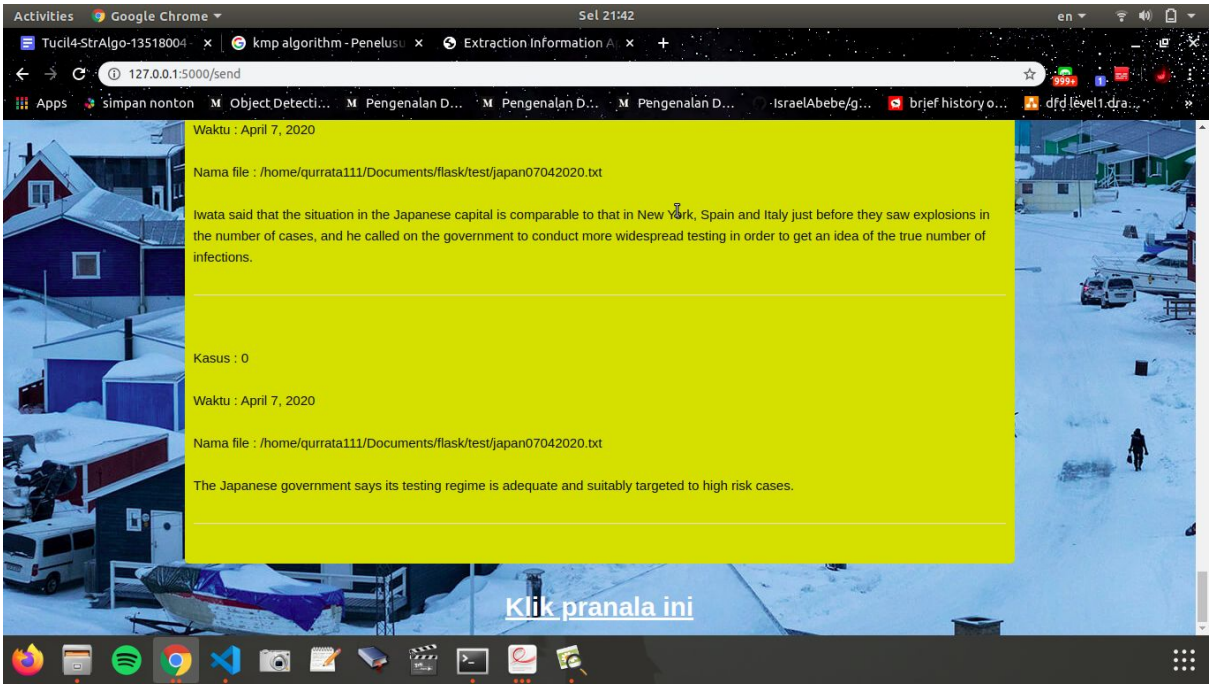
3. Pengujian program dengan algoritma BM
    - Keyword ditemukan, dengan mencoba keyword “cases”



Gambar 6 Keyword ditemukan dengan algoritma BM

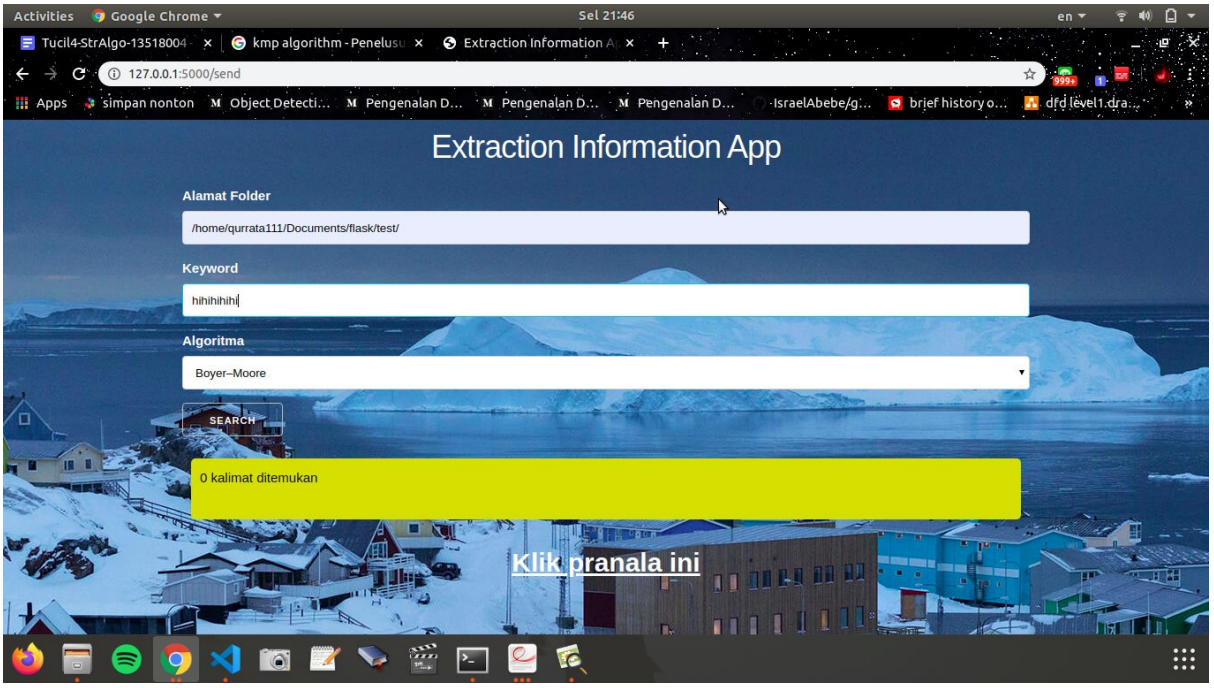


Gambar 7 Keyword ditemukan dengan algoritma BM



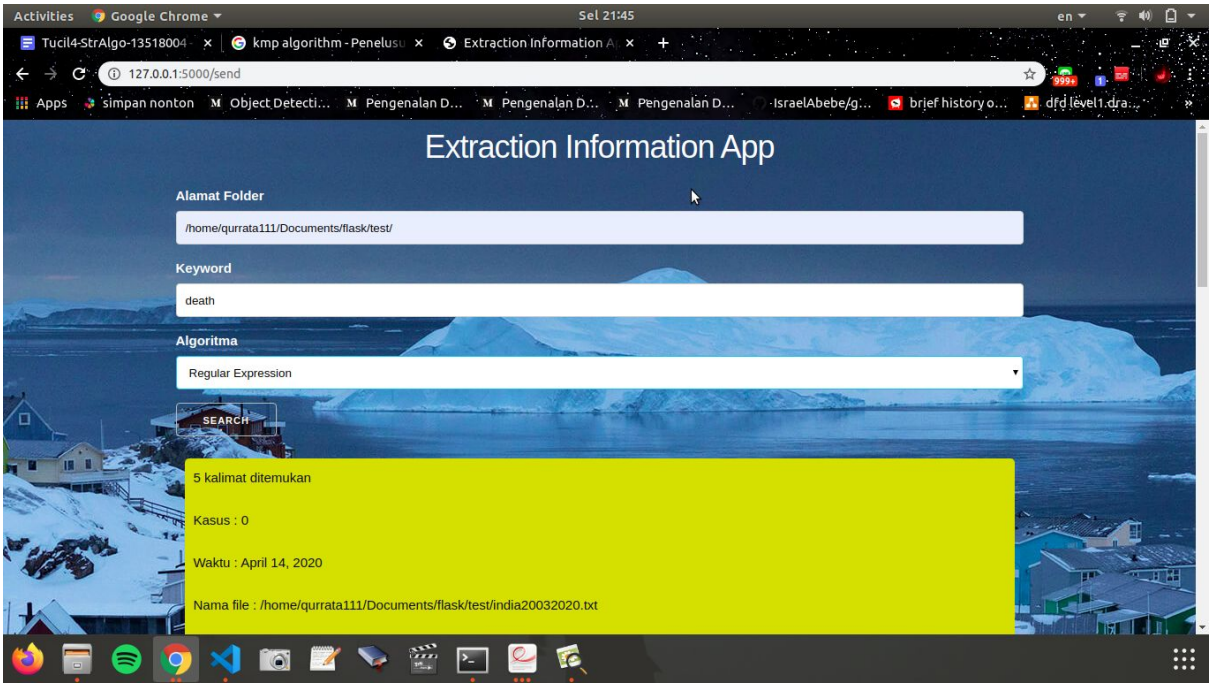
Gambar 8 Keyword ditemukan dengan algoritma BM

- Keyword tidak ditemukan, dengan mencoba keyword “hihihihihi”

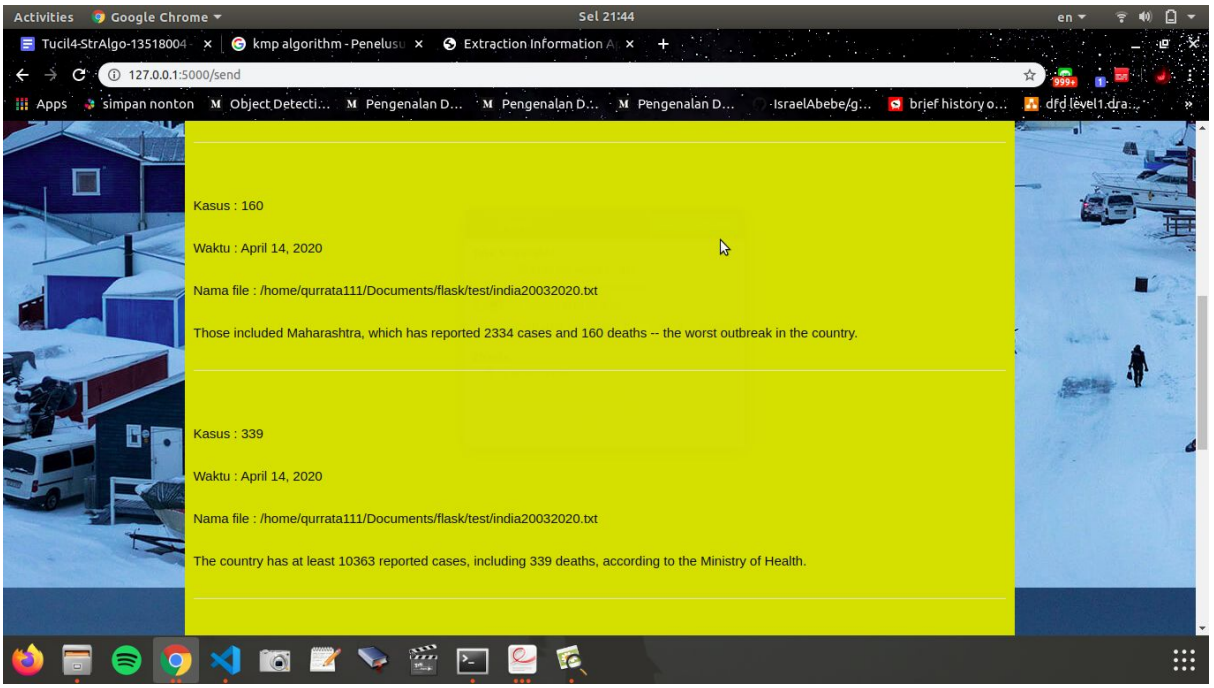


Gambar 9 Keyword tidak ditemukan dengan algoritma BM

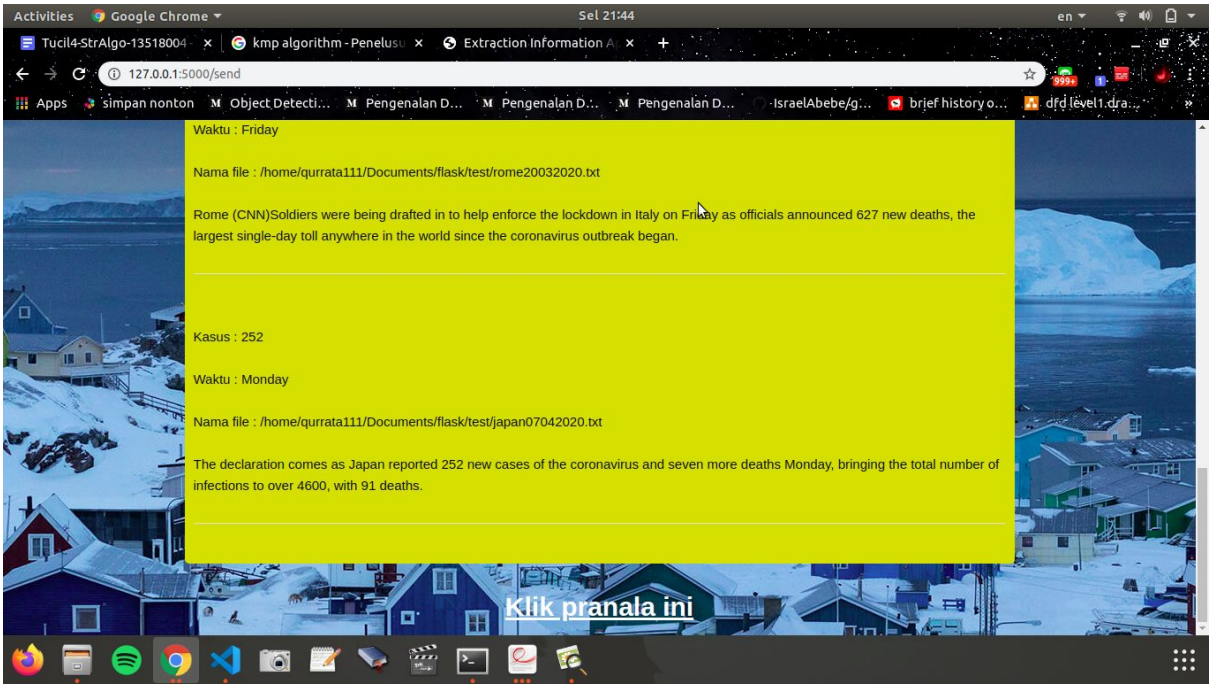
4. Pengujian program dengan regex
    - Keyword ditemukan, dengan mencoba keyword “death”



Gambar 10 Keyword ditemukan dengan Regex

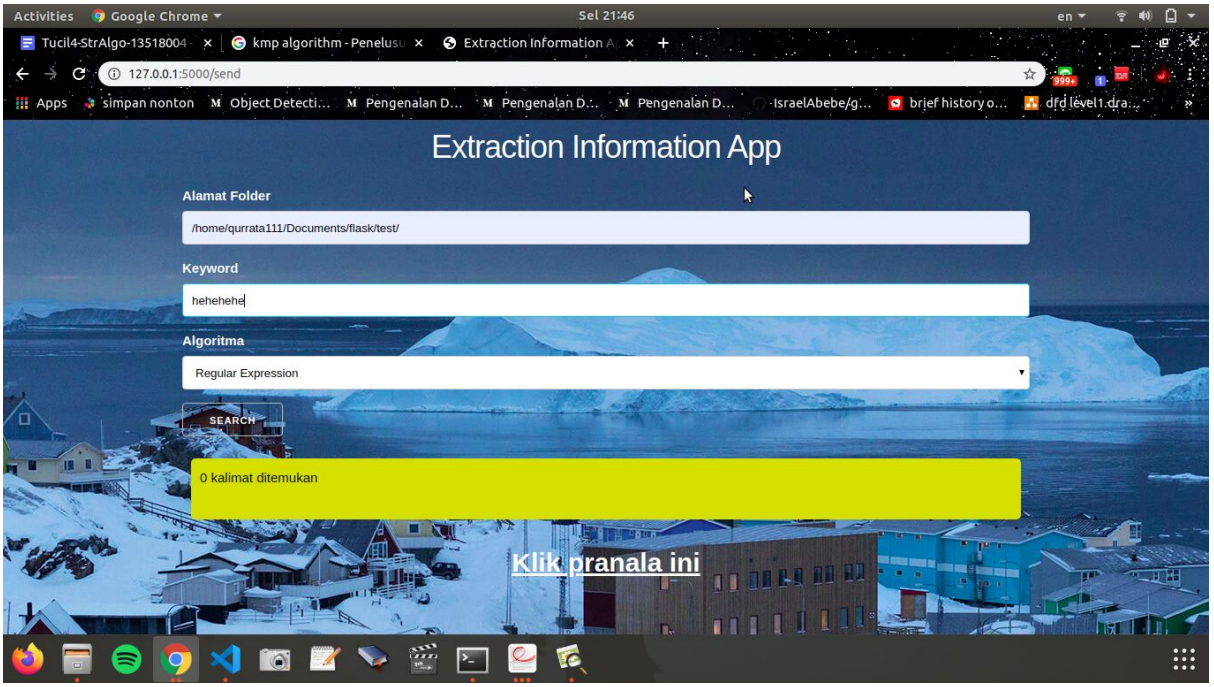


Gambar 11 Keyword ditemukan dengan Regex



Gambar 12 Keyword ditemukan dengan Regex

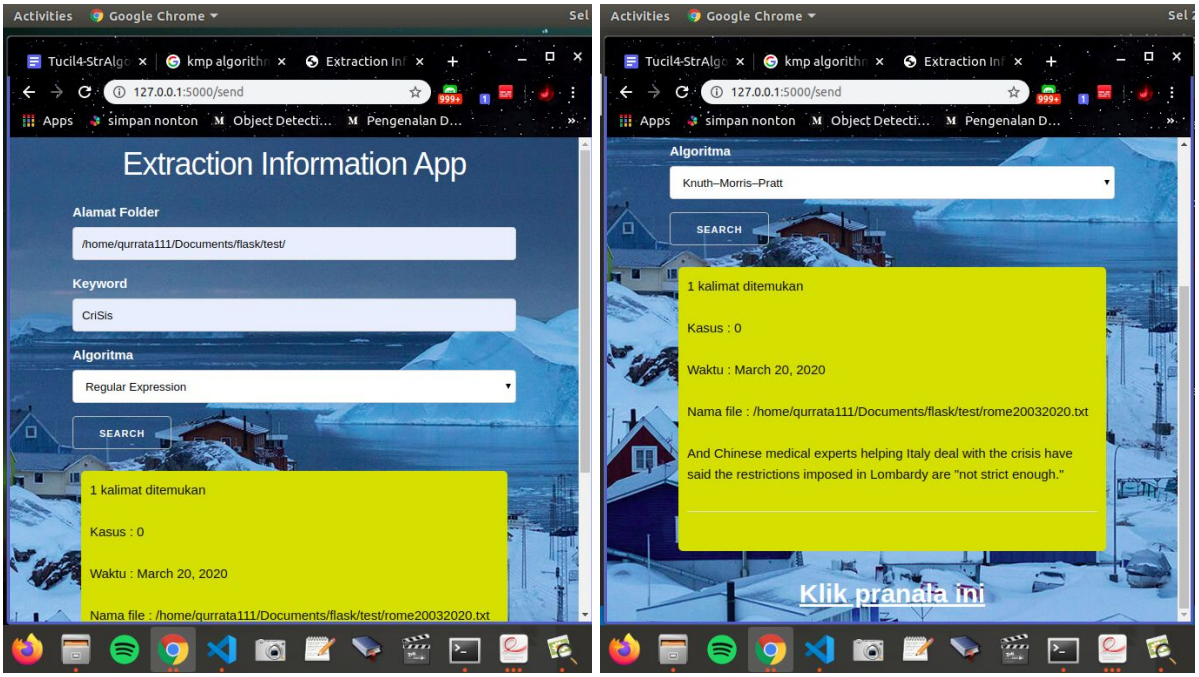
- Keyword tidak ditemukan, dengan mencoba keyword “hehehehehe”



Gambar 13 Keyword tidak ditemukan dengan Regex

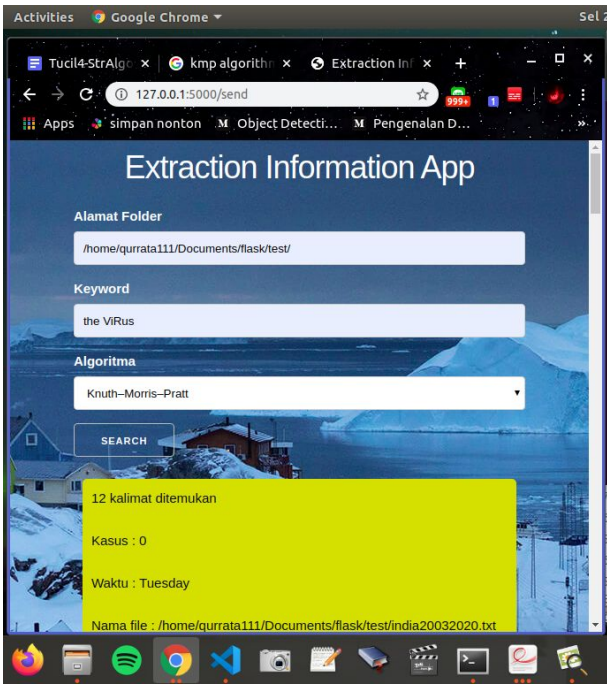
5. Pengujian dengan keyword lain

- Keyword “CriSis”

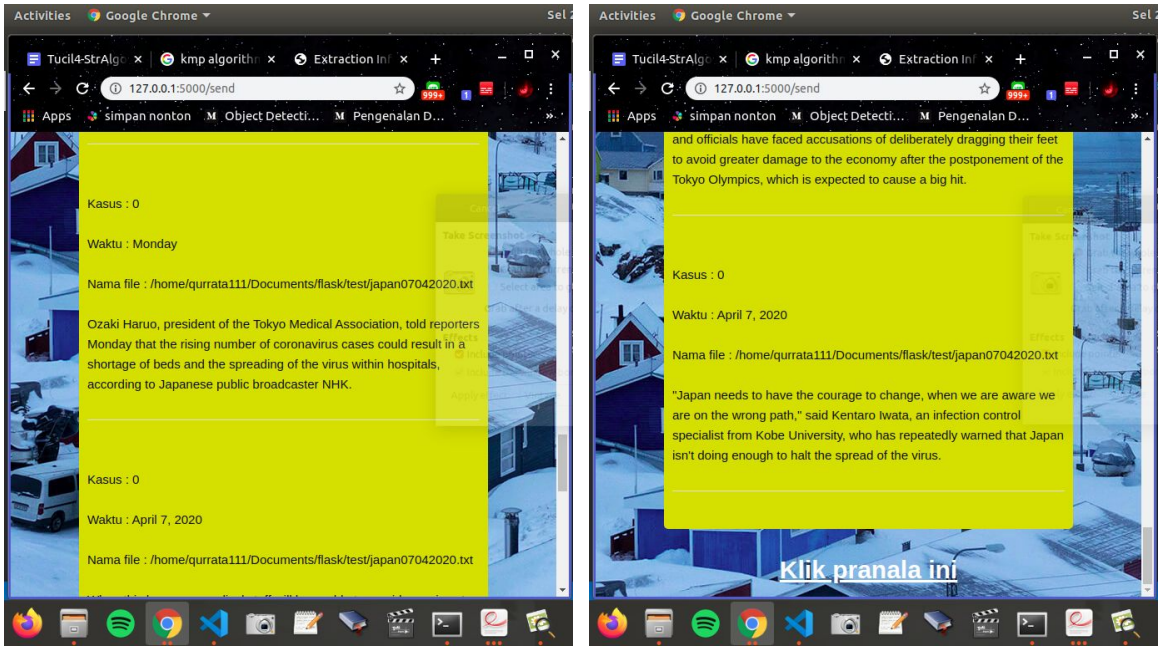


Gambar 14 Keyword ditemukan dengan Regex

- Keyword “the ViRus”

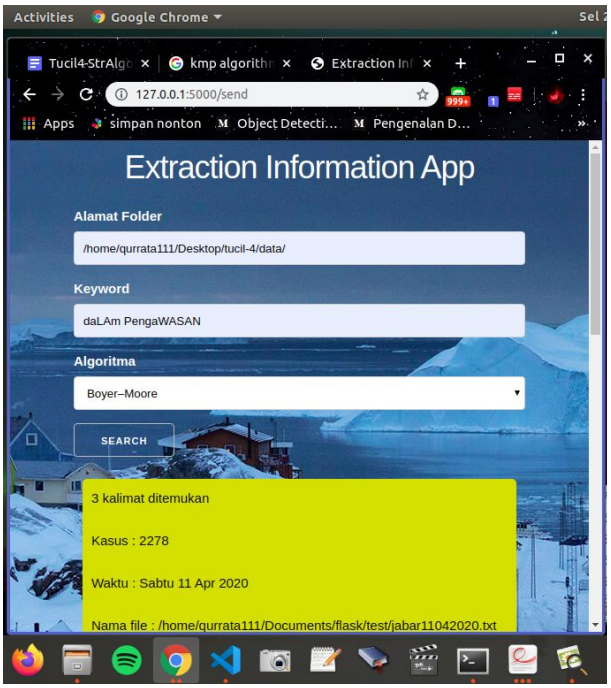


Gambar 15 Keyword ditemukan dengan Algoritma KMP

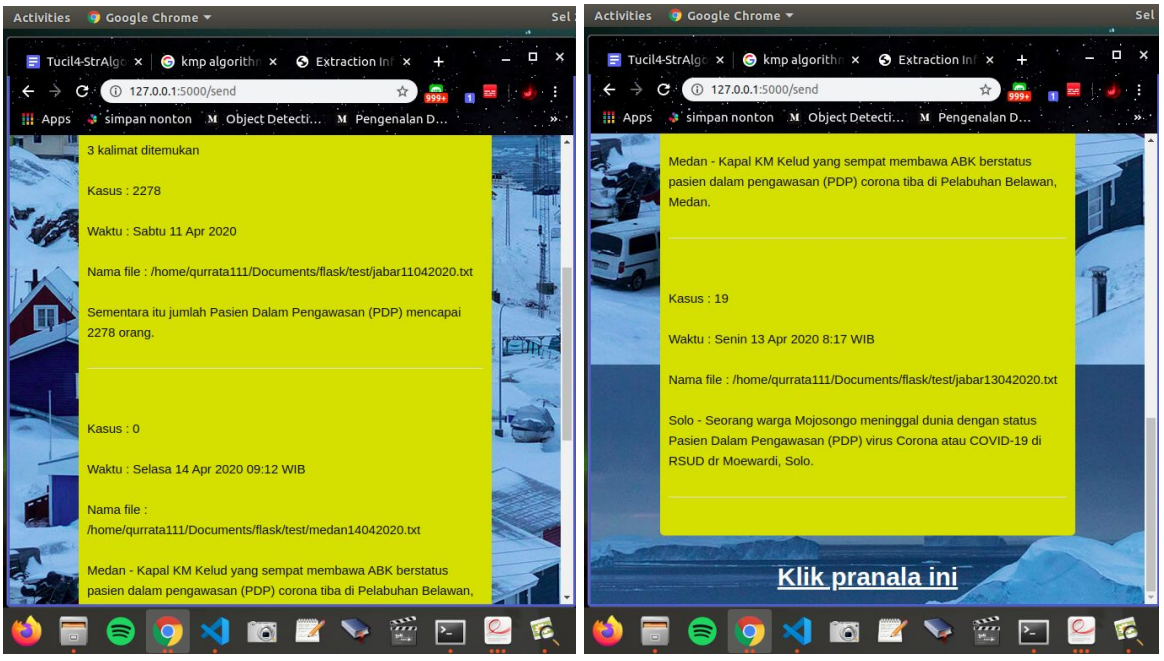


Gambar 16 Keyword ditemukan dengan Algoritma KMP

- Keyword “daLAm PengaWASAN”



Gambar 17 Keyword ditemukan dengan Algoritma BM



Gambar 18 Keyword ditemukan dengan Algoritma BM

Spesifikasi Personal Computer yang digunakan

CPU : Intel core i5-8250U
RAM : 4 Gb
Core : i5
OS : Linux (Ubuntu 18.04.3 LTS)

Tabel 1 Ceklist Uji Coba Program

Poin	Ya	Tidak
Program berhasil dikompilasi	✓	
Program berhasil running	✓	
Program dapat menerima input dan menuliskan output	✓	
Luaran sudah benar untuk semua n	✓	

REFERENSI

https://flask-doc.readthedocs.io/en/latest/
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf
https://id.wikipedia.org/wiki/Algoritme_Knuth-Morris-Pratt
https://id.wikipedia.org/wiki/Algoritme_Boyer-Moore
https://id.wikipedia.org/wiki/Regular_Expression