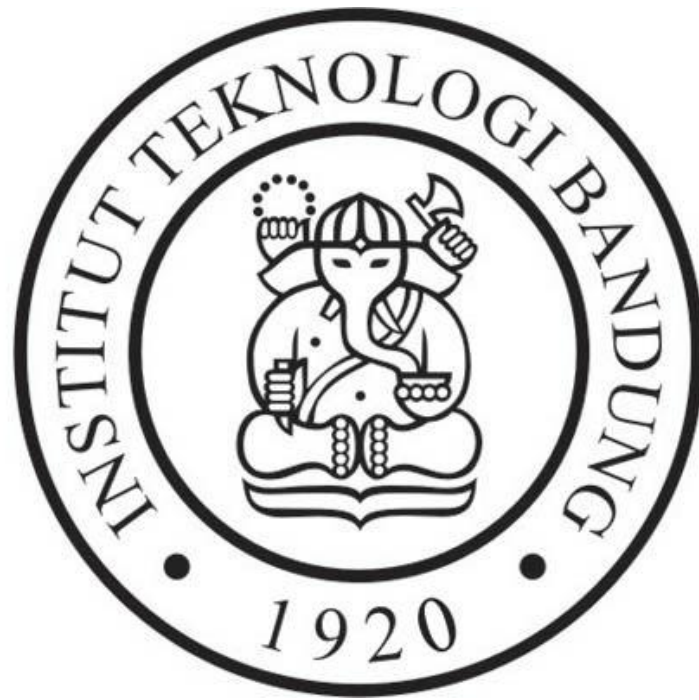


TUGAS KECIL 4 IF2211 STRAREGI ALGORITMA
SEMESTER II TAHUN 2019/2020
EKSTRAKSI INFORMASI DARI ARTIKEL BERITA
DENGAN ALGORITMA PENCOCOKAN STRING



Disusun oleh:
Syarifuddin Fakhri A (13518095)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020

BAB 1

DESKRIPSI DAN SPESIFIKASI TUGAS

Deskripsi :

Pada Tugas Kecil IV kali ini Anda diminta membuat aplikasi sederhana ekstraksi informasi dengan kedua algoritma tersebut, plus menggunakan *regular expression (regex)*. Teks yang akan Anda proses adalah teks berita berbahasa Indonesia seperti contoh berikut ini (jabar11042020.txt).

421 Orang di Jabar Terkonfirmasi Positif COVID-19

Yudha Maulana – detikNews

Sabtu, 11 Apr 2020 20:07 WIB

Bandung - Angka positif virus Corona atau COVID-19 di Jawa Barat menembus angka 400 kasus. Laman Pusat Informasi dan Koordinasi COVID-19 Jabar (Pikobar) pada Sabtu (11/4/2020) pukul 18.43 WIB, mencatat terdapat 421 orang yang terkonfirmasi positif COVID-19.

Dibandingkan sehari sebelumnya, jumlah tercatat yaitu 388 orang. Terjadi penambahan 8,5 persen atau 33 kasus per harinya. Sementara itu, secara nasional terdapat 3.842 kasus positif COVID-19.

Dari 421 kasus tersebut, 40 orang meninggal dunia dengan keterangan terpapar COVID-19. Sedangkan, angka kesembuhan di Jabar masih tetap berada di angka 19 orang.

Per hari jumlah Orang Dalam Pemantauan (ODP) di Jabar mencapai 28.775 orang. Sebanyak 15.363 di antaranya masih menjalani proses pemantauan dan 13.412 orang lainnya telah selesai menjalani proses pemantauan.

Sementara itu jumlah Pasien Dalam Pengawasan (PDP) mencapai 2.278 orang. Tercatat 1.344 orang masih menjalani proses pengawasan dan 934 orang lainnya telah selesai menjalani proses pengawasan.

Pada kumpulan teks berita korban covid-19 ini, informasi penting dari pengguna adalah jumlah korban dan waktunya. Oleh karena itu, informasi yang akan diekstraksi adalah angka (diberi warna biru) dan waktu (diberi warna merah).

Pengguna aplikasi ini akan memberikan masukan berupa folder yang berisi kumpulan teks berita, *keywords*, dan hasil ekstraksi jumlah dan waktunya. Karena sebagian besar kalimat mengandung angka, aplikasi akan memfilter angka berdasarkan *keywords* dari pengguna, seperti 'terkonfirmasi positif', 'meninggal dunia', 'Orang Dalam Pemantauan', 'ODP', 'Pasien Dalam Pengawasan', 'PDP' atau *keyword* lainnya. Hasilnya berupa pasangan angka dan waktu, serta kalimat yang mengandung informasi tersebut. Waktu yang diambil harus berada dalam satu kalimat dengan angka tersebut. Jika tidak ada, gunakan tanggal artikel yang tercantum. Jika terdapat lebih dari satu angka, pilih angka yang paling dekat dengan *keyword*. Berikut contohnya.

Keyword: terkonfirmasi positif

Hasil ekstraksi informasi:

Jumlah: 421; Waktu: Sabtu, 11 Apr 2020 20:07 WIB

421 Orang di Jabar Terkonfirmasi Positif COVID-19. (jabar11042020.txt)

Jumlah: 421; Waktu: Sabtu (11/4/2020) pukul 18.43 WIB

Laman Pusat Informasi dan Koordinasi COVID-19 Jabar (Pikobar) pada Sabtu (11/4/2020) pukul 18.43 WIB, mencatat terdapat 421 orang yang terkonfirmasi positif COVID-19. (jabar11042020.txt)

Keyword: meninggal dunia

Hasil ekstraksi informasi:

Jumlah: 40; Waktu: Sabtu, 11 Apr 2020 20:07 WIB

Dari 421 kasus tersebut, 40 orang meninggal dunia dengan keterangan terpapar COVID-19. (jabar11042020.txt)

Terdapat dua jenis pencocokan *string* yang Anda lakukan. Pertama, *exact match* dengan *keyword* yang diberikan pengguna untuk memfilter kalimat yang akan diproses informasinya. Semua teknik (KMP, BM, dan *regex*) bisa digunakan untuk fitur ini. Kedua, ekstraksi jumlah dan waktu dari kalimat hasil *exact match* dengan menggunakan *regex*.

Pencarian tidak bersifat *case sensitive*, jadi huruf besar dan huruf kecil dianggap sama (hal ini dapat dilakukan dengan *mengganggap* seluruh karakter di dalam *pattern* dan teks sebagai huruf kecil semua atau huruf kapital semua).

Spesifikasi program :

1. Aplikasi yang Anda buat merupakan aplikasi web yang menerima *keyword* pencarian, misalnya “terkonfirmasi positif”. Tampilan antarmuka pengguna seperti berikut.



My InfoExtraction App

Folder : <browse>

Keyword : <keyword>

Algoritma :

- ☐ Boyer-Moore
- ☐ KMP
- ☐ Regex

1. Jumlah: ... ; Waktu:
<kalimat> (<namafile>)

2. Jumlah: ... ; Waktu:
<kalimat> (<namafile>)

...

[Perihal](#)

Perihal : *link* ke halaman tentang program dan pembuatnya

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya

2. Aplikasi menggunakan hasil implementasi algoritma KMP, *Boyer-Moore*, dan *Regex* dengan menggunakan bahasa python. Pencocokan *string* dilakukan pada konten berita (teks).

Data Uji :

Data uji yang digunakan dapat Anda tentukan sendiri, minimal terdapat folder yang berisi 10 teks berita. *Posting* dapat berbahasa Indonesia atau Inggris.

Lain – lain :

1. Anda dapat menambahkan fitur fungsional lain yang menunjang program yang Anda buat (unsur kreativitas diperbolehkan/dianjurkan).
2. Program berbasis web dan dapat dikembangkan dengan salah satu kakas: php, flask, javascript.
3. Program implementasi Boyer-Moore dan KMP menggunakan bahasa python.
4. Program harus modular dan mengandung komentar yang jelas.
5. Mahasiswa harus membuat program sendiri kecuali *library* file dan *regex*, tetapi belajar dari contoh-contoh program serupa yang sudah ada tidak dilarang (tidak boleh mengkopi *source code* dari program orang lain). Program harus dibuat sendiri, tidak boleh sama dengan teman. Keterlambatan pengumpulan akan mengurangi nilai.
6. Program disimpan di dalam folder StrAlgo4-xxxxx. Lima digit terakhir adalah NIM Anda. Di dalam folder tersebut terdapat tiga folder bin, src dan doc yang masing-masing berisi :
 - a. Folder src berisi *source code* dari program
 - b. Folder test berisi data uji.
 - c. Folder doc berisi dokumentasi program dan readme
7. Semua pertanyaan menyangkut tugas ini harus dikomunikasikan melalui *milis* agar dapat dicermati oleh semua peserta kuliah IF2211 (*milis* IF2211@students.if.itb.ac.id).

BAB 2

DASAR TEORI

Algoritma Pencocokan *String*

Algoritma pencocokan *string* adalah sebuah algoritma yang digunakan untuk melakukan pencarian semua kemunculan *string* pendek atau yang disebut *pattern* pada sebuah *string* panjang atau yang disebut *text*.

Algoritma *Knuth-Morris-Pratt* (KMP)

Algoritma *Knuth-Morris-Pratt* adalah sebuah algoritma pencocokan *string* yang dikembangkan secara terpisah oleh Donald E-Knuth pada tahun 1967 James H. Morris bersama Vaughan R.Pratt pada tahun 1966, namun keduanya mempublikasikannya pada tahun 1977. Algoritma ini terlihat seperti algoritma pencocokan *string* dengan *brute force* tetapi lebih cepat dan lebih cerdas algoritma KMP dengan memanfaatkan konsep *prefix* (awalan) dan *suffix* (akhiran). Cara kerja algoritma ini adalah mencocokkan *string* dari kiri ke kanan. Cara kerja algoritma ini adalah sebagai berikut:

1. Apabila sama maka geser i dan j ke kanan sebanyak satu, jika j sama dengan panjang *pattern*, maka *pattern* sudah ketemu di *text*.
2. Apabila terjadi *mismatch* antara *pattern* P di $P[j]$ dengan *text* T di $T[i]$, jika $j > 0$ maka j diganti menjadi ukuran *string* terbesar yang menjadi *prefix* $P[0 .. j-1]$ dan *suffix* $P[1 .. j-1]$.
3. Sedangkan lainnya, maka geser i ke kanan sebanyak satu.

Kompleksitas algoritma ini adalah $O(m+n)$ dengan m panjang *pattern* dan n panjang *text*.

Algoritma *Boyer-Moore* (BM)

Algoritma *Boyer-Moore* (BM) adalah sebuah algoritma pencocokan *string* yang dikembangkan oleh Robert S. Boyer and J Strother Moore pada tahun 1977. Algoritma ini memulai pencarian dari ujung kanan *pattern* kemudian berjalan ke kiri. Cara kerja algoritma ini adalah sebagai berikut:

1. Cari semua kemunculan alamat terakhir semua huruf yang ada di *pattern*, simpan pada sebuah *list* sebagai pasangan huruf dan alamat
2. Apabila sama maka geser i dan j ke kiri sebanyak satu, jika j sama dengan -1 maka *pattern* sudah ketemu di *text*
3. Apabila terjadi *misssmatch* antara *pattern* P di $P[j]$ dengan *text* T di $T[i]$, jika $T[i]$ ada di *list* yang telah dibuat pada nomor 1 maka:
 - a. Jika alamat huruf $T[i]$ di *list* berada di kiri j , maka $i_{baru} = i_{lama} + \text{panjang } pattern - 1 - \text{alamat huruf } T[i]$ dan $j_{baru} = \text{panjang } pattern - 1$
 - b. Jika alamat huruf $T[i]$ di *list* berada di kanan j , maka $i_{baru} = i_{lama} + \text{panjang } pattern - j_{lama}$ dan $j_{baru} = \text{panjang } pattern - 1$
4. Sedangkan lainnya, maka $i_{baru} = i_{lama} + \text{panjang } pattern$ dan $j_{baru} = \text{panjang } pattern - 1$

Kompleksitas algoritma ini adalah $O(nm + A)$

Regular Expression

Regular Expression adalah sebuah set alat atau aturan yang digunakan untuk mencari *pattern* pada sebuah *text* menggunakan ekspresi-ekspresi tertentu.

BAB 3

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Program

a. Modul KMP

```
from backend.ModulLanguage import LowerCase

def KMPSearch(sentence, keyword):
    #Implementasi algoritma KMP dengan parameter sentence untuk text dan keyword untuk pattern

def BorderFuntion(keyword, j):
    #Implementasi border function pada KMPSearch untuk mencari kecocokan prefix P[0..j-1] dengan suffix P[1..j-1]
```

b. Modul BM

```
from backend.ModulLanguage import LowerCase

def BMSearch(sentence, keyword):
    #Implementasi algoritma BM dengan parameter sentence untuk text dan keyword untuk pattern

def GetDictOfChar(string):
    # Mencari semua kemunculan alamat terakhir semua huruf yang ada di pattern, simpan pada sebuah list sebagai pasangan huruf dan alamat
```

c. Modul Regex

```
import re

def RegexSearch(sentence, keyword):
    #Implementasi regex dengan parameter sentence untuk text dan keyword untuk pattern

def GetDate(sentence):
    #Mendapatkan string date pada sentence

def GetDay(sentence):
    #Mendapatkan string day pada sentence

def GetTime(sentence):
    # Mendapatkan string time pada sentence

def GetTotal(sentence, keyword):
    #Mendapatkan int total angka yang berkaitan atau berkaitan dengan keyword pada sentence

def IsCovidStr(sentence, pos):
    #Mencari tahu apakah angka di posisi po adalah angka yang merupakan substring Covid-19
```

d. Modul Language

```
import nltk

def SplitSentence(paragraph):
    #Memecah paragraf menjadi list kalimat-kalimat

def LowerCase(sentence):
    #Mengubah sentence menjadi lowercase (menggunakan huruf kecil semua)
```

e. Modul File

```
import os

def GetListFile(path):
    #Mendapatkan list alamat file dari sebuah alamat folder

def GetFileContents(path):
    #Mendapatkan list paragraf file dari sebuah alamat file

def GetFileName(path):
    #Mendapatkan sebuah nama file dari sebuah alamat file
```

f. Extrac Info

```
from backend.ModulFile import GetFileContents
from backend.ModulLanguage import SplitSentence
from backend.ModulBM import BMSearch
from backend.ModulKMP import KMPSearch
from backend.ModulReg import RegexSearch, GetDate, GetTotal

def Extrac(listFile, algorithm, keyword):
    #Mengekstrak list alamat file untuk mencari sebuah keyboard dengan suatu algoritma
```

g. Interface

```
from flask_wtf import FlaskForm
from wtforms.fields import StringField, RadioField, SubmitField, FileField, SelectField
from wtforms.widgets import ListWidget, RadioInput
from wtforms.validators import DataRequired

class InterfaceForm(FlaskForm):
    #class yang digunakan untuk membuat form pengisian pada interface home web
```

h. Result

```
class ResultPage():
    #class yang digunakan untuk membuat page hasil pada result page web
    def __init__(self, keyword, algorithm, totalFind, listResult):
        #method memanggil class ResultPage dengan parameter keyword, algorithm, totalFind (jumlah listResult yang dihasilkan), dan listResult
```

i. About

```
class AboutPage():  
    # class yang digunakan untuk membuat page about pada About page web
```

j. Main (Pengolahan backend dan frontend)

```
from flask import Flask, render_template, request, flash, redirect, url_for  
from frontend import app  
from frontend.form.Interface import InterfaceForm  
from frontend.form.Result import ResultPage  
from frontend.form.About import AboutPage  
from backend.ModulFile import GetListFile, GetFileName  
from backend.ExtracInfo import Extrac  
  
@app.route('/', methods=['GET', 'POST'])  
def Interface():  
    #Untuk merender Interface.html, menerima input dari user, dan mengolah datanya  
  
@app.route('/Result')  
def Result():  
    #Untuk merender Result.html dan menampilkan hasil dari fungsi Interface  
  
@app.route('/About')  
def About():  
    #Untuk menampilkan About.html
```

k. Layout.html (Layout untuk implementasi frontend web desain)

```
<!DOCTYPE html>  
<html>  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <style>  
        .navbar {  
            overflow: hidden;  
            background-color: #4CAF50;  
        }  
        .navbar a {  
            float: left;  
            color: #f2f2f2;  
            text-align: center;  
            padding: 10px 12px;  
            text-decoration: none;  
            font-size: 13px;  
        }  
        .navbar a:hover {  
            background-color: #333;  
            color: white;  
        }  
    </style>  
    <head>  
        {% if title %}  
            <title>{{ title }}</title>  
        {% else %}  
            <title>SFA-Blog</title>  
        {% endif %}
```



```

        <strong>{{form.algorithm.label}} :</strong> {{ form.algorithm(size=32)
    }}

    {% for error in form.algorithm.errors %}
        <span style="color: red;">Warning!!! <br> [{{ error }}]</span>
    {% endfor %}
</p>
<p> {{form.submit()}} </p>
</form>
{% endblock %}

```

o. `__init__` (app)

```

from flask import Flask
from Config import Configuration

app = Flask(__name__)
app.config.from_object(Configuration)

from frontend import Main

#Untuk menjalankan kakas flask sebagai app

```

p. Config

```

import os

class Configuration(object):
    #Class untuk mengkonfigurasi sebuah flask

```

q. App

```

from frontend import app

if __name__ == '__main__':
    app.run(debug = True)

#Menjalankan program keseluruhan dari aplikasi ini

```

4.2 Pengujian

Spesifikasi komputer dalam pengujian ini sebagai berikut:

- a. Prosesor : AMD A6-3400M APU with Radeon(tm) HD Graphics 1,40 GHz
- b. RAM : 4 GByte (*dual chanel*)
- c. System : 64-bit Operating System
- d. OS : Windows 7 Professional

Dalam program yang telah dibuat, program python akan menjalankan *localhost*. *Localhost* ini digunakan untuk sinkronisasi dan pemrosesan data *input output* dari pengguna. Masukan dari pengguna adalah sebagai berikut

1. Alamat folder yang berisi file txt yang akan diekstrak infonya
2. *Keyword* info yang akan dicari pengguna

3. Memilih 1 algoritma yang digunakan untuk pemrosesan

Keluaran program adalah *list* kecocokan file dengan *keyword* yang digunakan.



SFA InfoExtraction App

HOME ABOUT

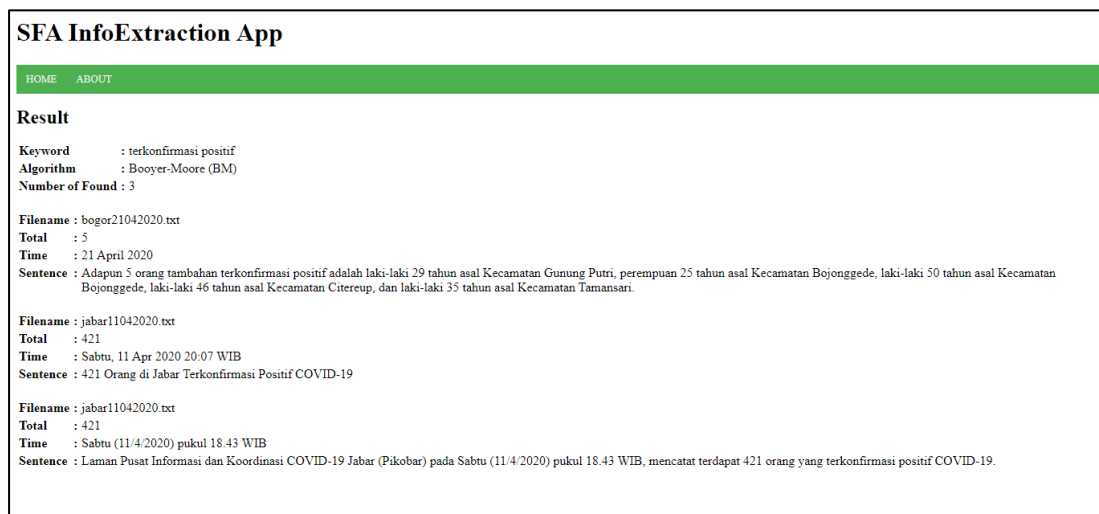
Type Directory of Folder :

Keyword :

Choose Algorithm :

- ☐ Booyer-Moore
- ☐ Knuth-Morris-Pratt
- ☐ Regex

Gambar 4.1 Tampilan Home Aplikasi



SFA InfoExtraction App

HOME ABOUT

Result

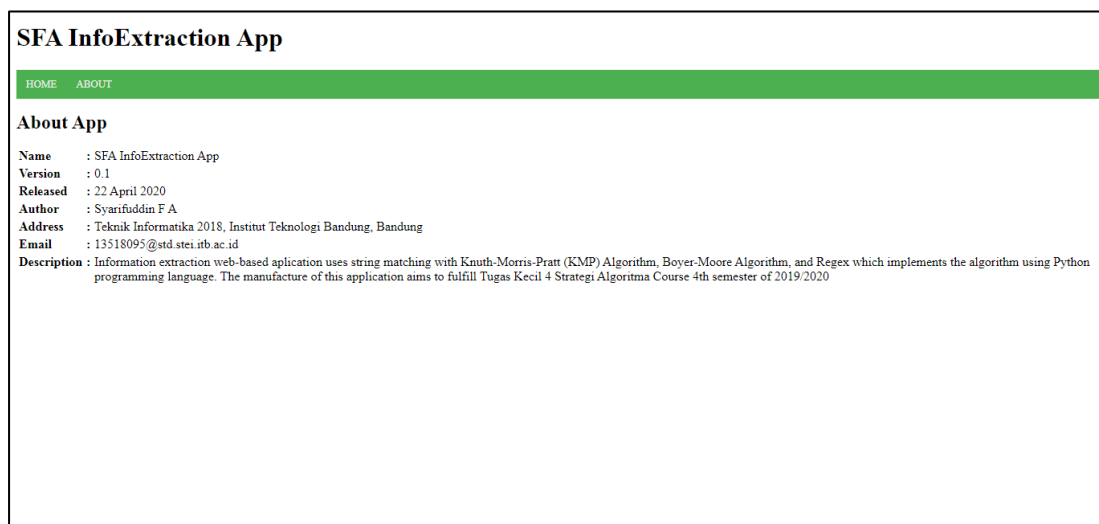
Keyword : terkonfirmasi positif
Algorithm : Booyer-Moore (BM)
Number of Found : 3

Filename : bogor21042020.txt
Total : 5
Time : 21 April 2020
Sentence : Adapun 5 orang tambahan terkonfirmasi positif adalah laki-laki 29 tahun asal Kecamatan Gunung Putri, perempuan 25 tahun asal Kecamatan Bojonggede, laki-laki 50 tahun asal Kecamatan Bojonggede, laki-laki 46 tahun asal Kecamatan Citereup, dan laki-laki 35 tahun asal Kecamatan Tamansari.

Filename : jabar11042020.txt
Total : 421
Time : Sabtu, 11 Apr 2020 20:07 WIB
Sentence : 421 Orang di Jabar Terkonfirmasi Positif COVID-19

Filename : jabar11042020.txt
Total : 421
Time : Sabtu (11/4/2020) pukul 18.43 WIB
Sentence : Laman Pusat Informasi dan Koordinasi COVID-19 Jabar (Pikobar) pada Sabtu (11/4/2020) pukul 18.43 WIB, mencatat terdapat 421 orang yang terkonfirmasi positif COVID-19.

Gambar 4.2 Tampilan Result Aplikasi



SFA InfoExtraction App

HOME ABOUT

About App

Name : SFA InfoExtraction App
Version : 0.1
Released : 22 April 2020
Author : Syarifuddin F A
Address : Teknik Informatika 2018, Institut Teknologi Bandung, Bandung
Email : 13518095@std.stei.itb.ac.id
Description : Information extraction web-based application uses string matching with Knuth-Morris-Pratt (KMP) Algorithm, Boyer-Moore Algorithm, and Regex which implements the algorithm using Python programming language. The manufacture of this application aims to fulfill Tugas Kecil 4 Strategi Algoritma Course 4th semester of 2019/2020

Gambar 4.3 Tampilan About Aplikasi

Pengetesan program menggunakan folder yang berisi 10 teks berita dalam bahasa Indonesia. Pengetesan dilakukan sebanyak 3 kali untuk mencoba 3 *keyword* dan 3 algoritma yang ada.

1. Tes 1

Keyword : terkonfirmasi positif

Algoritma : *Booyer-Moore*

Hasil :

Filename : bogor21042020.txt

Total : 5

Time : 21 April 2020

Sentence : Adapun 5 orang tambahan terkonfirmasi positif adalah laki-laki 29 tahun asal Kecamatan Gunung Putri, perempuan 25 tahun asal Kecamatan Bojonggede, laki-laki 50 tahun asal Kecamatan Bojonggede, laki-laki 46 tahun asal Kecamatan Citareup, dan laki-laki 35 tahun asal Kecamatan Tamansari.

Filename : jabar11042020.txt

Total : 421

Time : Sabtu, 11 Apr 2020 20:07 WIB

Sentence : 421 Orang di Jabar Terkonfirmasi Positif COVID-19

Filename : jabar11042020.txt

Total : 421

Time : Sabtu (11/4/2020) pukul 18.43 WIB

Sentence : Laman Pusat Informasi dan Koordinasi COVID-19 Jabar (Pikobar) pada Sabtu (11/4/2020) pukul 18.43 WIB, mencatat terdapat 421 orang yang terkonfirmasi positif COVID-19.

2. Tes 2

Keyword : terkonfirmasi positif

Algoritma : *Knuth-Morris-Pratt*

Hasil :

Filename : jabar11042020.txt

Total : 28

Time : Sabtu, 11 Apr 2020 20:07 WIB

Sentence : Per hari jumlah Orang Dalam Pemantauan (ODP) di Jabar mencapai 28.775 orang.

Filename : jogja21042020.txt

Total : 3

Time : 21 April 2020 17:02 WIB

Sentence : Total data orang dalam pemantauan (ODP) 3.733 orang

Filename : tangsel12042020.txt
Total : 480
Time : 11 April 2020
Sentence : Orang Dalam Pemantauan (ODP) juga bertambah dari 480 orang menjadi 492 orang.

Filename : tangsel12042020.txt
Total : 480
Time : 11 April 2020
Sentence : Orang Dalam Pemantauan (ODP) juga bertambah dari 480 orang menjadi 492 orang.

3. Tes 3

Keyword : terkonfirmasi positif

Algoritma : *Regex*

Hasil :

Filename : amerika12042020.txt
Total : 19.700
Time : 12 Apr 2020
Sentence : Data dari Johns Hopkins University menemukan, pada Sabtu waktu AS, lebih dari 19.700 meninggal dunia akibat komplikasi dari virus corona.

Filename : bogor21042020.txt
Total : 16
Time : 21 April 2020
Sentence : Lalu PDP yang terdata meninggal hingga kini sebanyak 16 orang.

Filename : bogor21042020.txt
Total : 5
Time : 21 April 2020
Sentence : Kemudian, total kasus positif 63 pasien, bertambah sebanyak 5 pasien, meninggal 5 orang, positif aktif yang dirawat di rumah sakit 51 orang.

Filename : indonesia21042020.txt
Total : 616
Time : 21 April 2020
Sentence : Penambahan itu membuat total pasien Covid-19 meninggal dunia yaitu 616 orang.

Filename : jabar11042020.txt

Total : 40
Time : Sabtu, 11 Apr 2020 20:07 WIB
Sentence : Dari 421 kasus tersebut, 40 orang meninggal dunia dengan keterangan terpapar COVID-19.

Filename : jatim10042020.txt
Total : 5
Time : 10 April 2020
Sentence : Adapun pasien meninggal dunia per Jumat petang juga bertambah 5 orang sehingga total pasien Covid-19 yang meninggal dunia sudah mencapai 22 orang.

Filename : jatim10042020.txt
Total : 2
Time : 10 April 2020
Sentence : "5 pasien yang meninggal dunia, 2 pasien dari Surabaya, 1 pasien dari Sidoarjo, 1 pasien dari Lumajang, dan 1 pasien dari Bojonegoro," terang Khofifah.

Filename : jogja21042020.txt
Total : 46
Time : 21 April 2020 17:02 WIB
Sentence : c. 46 orang meninggal.

Filename : jogja21042020.txt
Total : 23
Time : 21 April 2020 17:02 WIB
Sentence : 400 orang negatif (Meninggal Dunia: 23)

Filename : jogja21042020.txt
Total : 7
Time : 21 April 2020 17:02 WIB
Sentence : 72 orang positif (Dirawat: 35, Sembuh: 30, Meninggal Dunia: 7), dan

Filename : jogja21042020.txt
Total : 1
Time : 21 April 2020 17:02 WIB
Sentence : Terdapat penambahan 1 pasien PDP yang meninggal dunia dan hasil uji lab belum diketahui, yaitu Perempuan, 71 tahun, warga Sleman.

Filename : jogja21042020.txt

Total : 2

Time : 21 April 2020 17:02 WIB

Sentence : Terdapat penambahan 2 hasil uji lab yang diketahui negatif bagi PDP yang meninggal saat uji lab masih berlangsung.

Filename : solo08042020.txt

Total : 2

Time : Rabu, 8 April 2020

Sentence : 1 di antaranya sembuh, 2 meninggal dunia, dan 2 lainnya masih menjalani perawatan di rumah sakit.

Filename : solo08042020.txt

Total : 7

Time : Rabu, 8 April 2020

Sentence : Sampai saat ini ada 7 di antara mereka yang meninggal dunia.

Filename : tangsel12042020.txt

Total : 41

Time : 11 April 2020

Sentence : Hingga 11 April 2020, 41 Orang di Tangsel Meninggal Akibat Covid-19

Filename : tangsel12042020.txt

Total : 41

Time : 11 April 2020

Sentence : Liputan6.com, Jakarta - Gugus Tugas Covid-19 Kota Tangerang Selatan menyebut, hingga 11 April 2020 ada 41 orang meninggal akibat corona.

Filename : tangsel12042020.txt

Total : 41

Time : 11 April 2020

Sentence : Liputan6.com, Jakarta - Gugus Tugas Covid-19 Kota Tangerang Selatan menyebut, hingga 11 April 2020 ada 41 orang meninggal akibat corona.

BAB 4
LIST KEBERHASILAN DALAM Pengerjaan Tugas Kecil 3

No.	Poin	Ya	Tidak
1.	Program berhasil dikompilasi	√	
2.	Program berhasil <i>running</i>	√	
3.	Program dapat menerima input dan menuliskan output	√	
4.	Luaran sudah benar untuk semua data uji	√	