

**Laporan Tugas Kecil 4 - Strategi Algoritma (IF2211)**  
**Ekstraksi Informasi dari Artikel Berita dengan Algoritma**  
**Pencocokan String**



**Oleh:**

**Muhammad Hasan – 13518012 – K03**

**TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**SEMESTER 2 TAHUN 2019/2020**

## Daftar Isi

Daftar Isi.....	2
BAB 1 – Pencocokan <i>String</i> Boyer-Moore, KMP, dan Regex .....	4
1.1 Algoritma Boyer-Moore untuk Pencocokan <i>String</i> .....	4
1.2 Algoritma Knuth-Morris-Pratt (KMP) untuk Pencocokan <i>String</i> .....	4
1.3 Regex untuk Pencocokan <i>String</i> .....	5
BAB 2 – <i>Source Code</i> Program .....	7
2.1 <i>Checklist</i> Tabel Program .....	7
2.2 <i>Source Code</i> Program .....	7
2.2.1 <i>Source code</i> BoyerMooreAlgorithm.py:.....	7
2.2.2 <i>Source code</i> KMPAlgorithm.py: .....	9
2.2.3 <i>Source code</i> Regex.py .....	10
2.2.4 <i>Source code</i> kalimat.py .....	10
2.2.5 <i>Source code</i> jumlah.py .....	11
2.2.6 <i>Source code</i> waktu.py .....	11
2.2.7 <i>Source code</i> result.py .....	13
BAB 3 – Screenshot Input-Output Program .....	15
3.1 Spesifikasi Personal Computer.....	15
3.2 Screenshot Input-Output Program .....	15
3.2.1 Test 1 (Algoritma Boyer-Moore).....	15
3.2.2 Test 2 (Algoritma KMP) .....	16
3.2.3 Test 3 (Regex Exact Match) .....	17
3.2.4 Test 4 (Multiple files).....	18

3.2.5 Test 5 (Multiple files).....	20
3.2.6 Test 6 (Multiple files).....	22

## BAB 1 – Pencocokan *String* Boyer-Moore, KMP, dan Regex

### 1.1 Algoritma Boyer-Moore untuk Pencocokan *String*

Algoritma Boyer-Moore adalah salah satu algoritma untuk *pattern searching*, dibuat oleh Robert S. Boyer dan J Strother Moore pada tahun 1977. Algoritma ini melakukan *preprocess pattern* terlebih dahulu pada *pattern* yang diberikan berdasarkan kombinasi dari dua pendekatan, yaitu *bad character heuristic* dan *good suffix heuristic*. *Preprocess* tersebut dilakukan saat melakukan *searching*, sehingga proses akan dilakukan lebih cepat.

Ide dari *bad character heuristic* ini cukup sederhana, setiap karakter pada *pattern* yang tidak sesuai dengan *text* disebut **bad character**. Saat *bad character* ditemukan, algoritma ini akan melakukan penggeseran sampai:

1. Karakter yang tidak sesuai menjadi sesuai
2. *Pattern* bergerak melewati karakter yang tidak sesuai tersebut

Kompleksitas dari algoritma ini dibandingkan *bruteforce* dapat berbeda jauh, untuk kemungkinan terbaik, jika  $N$  = panjang *text*, dan  $M$  = panjang *pattern*, maka kompleksitas waktu terbaik dari algoritma ini adalah  $O(N/M)$ . Namun, pada kasus terburuk, algoritma ini dapat memiliki kompleksitas waktu  $O(NM)$ .

### 1.2 Algoritma Knuth-Morris-Pratt (KMP) untuk Pencocokan *String*

Algoritma Knuth-Morris-Pratt (KMP) merupakan *string-searching-algorithm* yang sering digunakan. Algoritma ini melakukan *preprocess array of longest prefix suffix* pada *pattern* terlebih dahulu, tentunya *preprocess* tersebut akan berguna saat melakukan *searching* pada suatu *text*. *Array of longest prefix suffix* tersebut kadang disebut juga sebagai *prefix function*.

*Prefix function* pada suatu *string* yang panjangnya  $n$  didefinisikan sebagai array  $\pi$  dengan panjang  $n$ , dengan  $\pi[i]$  adalah panjang dari *longest proper prefix* dari *substring*  $s[0 \dots i]$  yang juga merupakan *suffix* dari *substring* tersebut. Secara definisi  $\pi[0] = 0$ .

Secara matematis, definisi dari *prefix function* adalah sebagai berikut:

$$\pi[i] = \max_{k=0..i} \{k: s[0 \dots k-1] = s[i-(k-1) \dots i]\}$$

Sebagai contoh, *prefix function* dari *string* “abcbcd” adalah [0,0,0,1,2,3,0]. Dengan *prefix function* tersebut, algoritma KMP ini dapat menentukan posisi selanjutnya secara lebih cepat ketika terdapat *mismatch* saat *searching* berlangsung.

Dengan *preprocess* dan *searching* yang dilakukan, masing-masing memiliki kompleksitas  $O(M)$  ( $M$  = panjang *pattern*) dan  $O(N)$  ( $N$  = panjang *text*), sehingga secara keseluruhan, total kompleksitas waktu algoritma KMP adalah  $O(N + M)$

### 1.3 Regex untuk Pencocokan *String*

*Regular Expression* (Regex) adalah suatu rangkaian karakter yang mendefinisikan suatu *search pattern*. Biasanya *pattern* tersebut digunakan *string searching algorithm* untuk operasi “*find*” atau “*replace*” pada suatu *string*, atau untuk validasi input.

Suatu *regular expression*, yang sering disebut *pattern*, menentukan himpunan dari *string* yang dibutuhkan untuk suatu tujuan tertentu. Dalam formalisme yang lebih umum, terdapat beberapa operasi untuk menyusun suatu regex:

#### 1) OR Operator (|)

Suatu garis vertikal yang memisahkan alternatif. Contohnya `gray|grey` dapat mencocokkan ‘gray’ atau ‘grey’

#### 2) Grouping

Buka kurung dan tutup kurung biasanya digunakan untuk mendefinisikan cakupan dan prioritas dari suatu operator. Contohnya `gray|grey` dan `gr(a|e)y` adalah *pattern* yang *equivalent*

#### 3) Quantification

Suatu *quantifier* setelah *token* tertentu (seperti karakter) atau suatu kelompok mendefinisikan seberapa banyak suatu element diperbolehkan untuk muncul. Contohnya, quantifier yang sering digunakan adalah tanda tanya `?` (mengindikasikan ada nol atau satu buah elemen sebelumnya), *asterisk* `*`

(mengindikasikan ada nol atau lebih buah elemen sebelumnya), dan tanda plus +  
(mengindikasikan ada minimal satu buah elemen sebelumnya)

Regex ini tentunya memiliki banyak operasi dan ketentuan lain, sehingga biasanya jika terdapat suatu regex yang memenuhi string tertentu, maka terdapat tak hingga regex lain yang bisa memenuhi string tersebut juga.

## BAB 2 – Source Code Program

### 2.1 Checklist Tabel Program

Poin	Ya	Tidak
Program berhasil dikompilasi	✓	
Program berhasil <i>running</i>	✓	
Program dapat menerima input dan menuliskan output	✓	
Luaran sudah benar untuk semua data uji	✓	

### 2.2 Source Code Program

Program dibuat dengan menggunakan bahasa pemrograman **python**, berikut adalah beberapa *source code* yang penting untuk disebutkan:

#### 2.2.1 Source code BoyerMooreAlgorithm.py:

Kode ini digunakan untuk *exact match* saat melakukan *filter* kalimat

```
# Declare constants
NUM_OF_CHAR = 256

def searchPatternInTextWithBM(pattern, text):
    # Preprocess bad character heuristic
    # Initialize all occurrences with -1
    badChar = [-1 for _ in range(NUM_OF_CHAR)]

    # Filling the listed value with last occurrences
    patLength = len(pattern)
    for i in range(patLength):
        badChar[ord(pattern[i])] = i
```

```
# Search if there is the pattern in the text
textLength = len(text)
iterShift = 0
while iterShift <= textLength - patLength:
    # set iterPattern to last index on pattern
    iterPattern = patLength - 1

    # keep reducing iterPattern when the characters are matching
    # in this iterShift
    while iterPattern >= 0 and pattern[iterPattern] == text[iterPattern + iterShift]:
        iterPattern -= 1

    # if the pattern occurs then iterPattern < 0
    if iterPattern < 0:
        # pattern found
        return True
    else:
        # shift the pattern
        iterShift += max(1, iterPattern - badChar[ord(text[iterPattern + iterShift])])

# pattern not found
return False
```



### 2.2.2 Source code KMPAlgorithm.py:

Kode ini digunakan untuk *exact match* saat melakukan *filter* kalimat

```
def searchPatternInTextWithKMP(pattern, text):
    # precompute longest prefix suffix of the pattern
    lps = [0]
    patLength = len(pattern)
    for i in range(1, patLength):
        j = lps[i - 1]
        while j > 0 and pattern[j] != pattern[i]:
            j = lps[j - 1]
        if pattern[j] == pattern[i]:
            j += 1
        lps.append(j)

    # search if there is pattern in the text
    textLength = len(text)
    iterText = 0
    iterPattern = 0
    while iterText < textLength:
        # if current character in the pattern matches with text
        # increment both iter
        if text[iterText] == pattern[iterPattern]:
            iterText += 1
            iterPattern += 1

        if iterPattern == patLength:
            # pattern found
            return True

        # mismatch after iterPattern matches
        elif iterText < textLength and text[iterText] != pattern[iterPattern]:
            # use lps so we won't have to check every character
            if iterPattern > 0:
                iterPattern = lps[iterPattern - 1]
```

```
        else:
            iterText += 1

    # pattern not found
    return False
```

### 2.2.3 Source code Regex.py

Kode ini digunakan untuk *exact match* saat melakukan *filter* kalimat

```
import re

def searchPatternInTextWithRegex(pattern, text):
    return re.search(pattern, text) != None
```

### 2.2.4 Source code kalimat.py

Kode ini digunakan untuk memisahkan teks berita menjadi beberapa kalimat

```
import nltk

def getKalimat(path):
    path = "../test/" + path
    try:
        with open(path) as file:
            data = file.read().replace('\n', '')
            return nltk.tokenize.sent_tokenize(data)
    except Exception as e:
        print(e)
        return None
```

### 2.2.5 Source code jumlah.py

Kode ini digunakan untuk menentukan jumlah yang bersangkutan dengan *keyword* pada kalimat yang diberikan

```
import re
# Regex for jumlah
regexJumlah = '(?<!\S)\d{1,3}(?:\.\d{3})*(?!\\S)'

def getJumlah(kalimat):
    list_jumlah = re.findall(regexJumlah, kalimat)
    for jumlah in list_jumlah:
        if len(jumlah) == 0:
            continue
        return jumlah
    return None
```

### 2.2.6 Source code waktu.py

Kode ini digunakan untuk menentukan waktu yang bersangkutan dengan *keyword* pada kalimat yang diberikan

```
import re

# declare constant regex times
hari = '(?:senin|selasa|rabu|kamis|jumat|sabtu|minggu)?'
bulan = '(?:januari|februari|maret|april|mei|juni|juli|agustus|september|oktober|november|desember)'
ketwaktu = '(?:sekarang|kemarin|pagi|sore|malam|siang)?'
lanjutwaktu = '(?:sampai|hingga|pada|saat)'
jamkata = '(?:jam|pukul)'
jamangka = '(?:\d{2}\.\d{2})'
jamket = '(?:wib|wit|wita|pm|am)'
jamwaktu = '(?:' + jamkata + '\s' + jamangka + '\s' + jamket + ')?'
tanggalangka = '(?:\(\d{1,2}/\d{1,2}/\d+\))?'
tanggalhuruf = '(?:\d{1,2}\s(?:' + bulan + ')\s\d+)?'
```

```
rwaktu = '(:' + hari + '\s?' + ketwaktu + '\s?' + tanggalhuruf + '\s?' + tangg  
alangka + ',?\s?' + jamwaktu + ')'  
regexwaktu = rwaktu + '\s?(?:' + lanjutwaktu + '\s?' + rwaktu + ')?'  
  
# method to get waktu  
def getWaktu(kalimat):  
    list_waktu = re.findall(regexwaktu, kalimat)  
    invalid_list = ["sampai", "hingga", "pada", "saat"]  
    for waktu in list_waktu:  
        waktu = waktu.strip()  
        if len(waktu) <= 1:  
            continue  
        if waktu in invalid_list:  
            continue  
        return waktu.replace(',', '').strip()  
  
    return None  
  
# method to get date  
def getTanggal(kalimat):  
    list_date = re.findall(tanggalangka, kalimat)  
    for date in list_date:  
        date = date.strip()  
        if len(date) <= 1:  
            continue  
        return date  
  
    list_date = re.findall(tanggalhuruf, kalimat)  
    for date in list_date:  
        date = date.strip()  
        if len(date) <= 1:  
            continue  
        return date  
  
    return None
```

### 2.2.7 Source code result.py

Kode ini digunakan untuk mendapatkan hasil yang dibutuhkan, yaitu jumlah, waktu, kalimat, dan sumber teks berita

```
from algorithms.BoyerMooreAlgorithm import searchPatternInTextWithBM
from algorithms.KMPAlgorithm import searchPatternInTextWithKMP
from algorithms.Regex import searchPatternInTextWithRegex
from utils.kalimat import getKalimat
from utils.jumlah import getJumlah
from utils.waktu import getWaktu, getTanggal

def getResults(form):
    # get input results
    filenames = form.get('filenames')
    keyword = form.get('keyword')
    algochoice = int(form.get('algochoice'))

    # get choice of algorithm
    algorithmsChoice = [searchPatternInTextWithBM, searchPatternInTextWithKMP,
                        searchPatternInTextWithRegex]
    searchPattern = algorithmsChoice[algochoice - 1]

    # split filenames
    filenames = filenames.strip().split(',')

    # get jumlah, waktu, kalimat, and source
    results = []
    for filename in filenames:
        list_kalimat = getKalimat(filename)
        if list_kalimat == None:
            continue

        # filter list_kalimat
        list_kalimat = [x for x in list_kalimat if searchPattern(keyword.lower(
        ), x.lower())]

        # get date first
        tanggal = None
        for kalimat in list_kalimat:
            tanggal = getTanggal(kalimat)
            if tanggal != None:
                break
```

```
# get result
for kalimat in list_kalimat:
    total_jumlah = getJumlah(kalimat)
    if total_jumlah == None:
        continue
    waktu = getWaktu(kalimat.lower())
    if waktu == None:
        if tanggal == None:
            continue
        waktu = tanggal
    result = {}
    result['jumlah'] = total_jumlah
    result['waktu'] = waktu
    result['kalimat'] = kalimat
    result['source'] = filename
    results.append(result)

if len(results) == 0:
    return None

return results
```

## BAB 3 – Screenshot Input-Output Program

### 3.1 Spesifikasi Personal Computer

Spesifikasi *Personal Computer* (PC) yang digunakan dalam menjalankan program adalah sebagai berikut:

Processor	RAM	HardDrive
Intel® Core™ i7-8550U CPU @ 1.80 GHz – 1.99 GHz	DDR3 – 16GB	SSD SATA 3 – 512 GB

### 3.2 Screenshot Input-Output Program

#### 3.2.1 Test 1 (Algoritma Boyer-Moore)

## Information Extractor App

Filenames

Masukkan nama file beserta tipe ekstensinya dipisahkan dengan koma (contoh: test1.txt,test2.txt,test3.txt)

Keyword

Algorithms

- ☒ Boyer-Moore
- ☐ KMP
- ☐ Regex

Jumlah: 6.575

Waktu: hingga minggu

Kalimat :

Dengan penambahan kasus ini, total kasus Covid-19 di Tanah Air menjadi 6.575 pasien hingga Minggu.

(test1.txt)

Jumlah: 47

Waktu: sabtu hingga minggu siang

Kalimat :

Ada 47 pasien Covid-19 yang meninggal dunia dalam kurun waktu Sabtu hingga Minggu siang ini sehingga totalnya menjadi 582 orang.

(test1.txt)

### 3.2.2 Test 2 (Algoritma KMP)

## Information Extractor App

Filenames

Masukkan nama file beserta tipe ekstensinya dipisahkan dengan koma (contoh: test1.txt,test2.txt,test3.txt)

Keyword

Algorithms

- ☐ Boyer-Moore
- ☒ KMP
- ☐ Regex



- ☒ KMP
- ☐ Regex

Extract

Jumlah: 6.575

Waktu: hingga minggu

Kalimat :

Dengan penambahan kasus ini, total kasus Covid-19 di Tanah Air menjadi 6.575 pasien hingga Minggu.

(test1.txt)

Jumlah: 47

Waktu: sabtu hingga minggu siang

Kalimat :

Ada 47 pasien Covid-19 yang meninggal dunia dalam kurun waktu Sabtu hingga Minggu siang ini sehingga totalnya menjadi 582 orang.

(test1.txt)

### 3.2.3 Test 3 (Regex Exact Match)

## Information Extractor App

Filenames

test1.txt

Masukkan nama file beserta tipe ekstensinya dipisahkan dengan koma (contoh: test1.txt,test2.txt,test3.txt)

Keyword

pasien

Algorithms

- ☐ Boyer-Moore
- ☐ KMP
- ☒ Regex

Extract

- ☐ Boyer-Moore
- ☐ KMP
- ☒ Regex

Jumlah: 6.575

Waktu: hingga minggu

Kalimat :

Dengan penambahan kasus ini, total kasus Covid-19 di Tanah Air menjadi 6.575 pasien hingga Minggu.

(test1.txt)

Jumlah: 47

Waktu: sabtu hingga minggu siang

Kalimat :

Ada 47 pasien Covid-19 yang meninggal dunia dalam kurun waktu Sabtu hingga Minggu siang ini sehingga totalnya menjadi 582 orang.

(test1.txt)

### 3.2.4 Test 4 (Multiple files)

## Information Extractor App

Filenames

Masukkan nama file beserta tipe ekstensinya dipisahkan dengan koma (contoh: test1.txt,test2.txt,test3.txt)

Keyword

Algorithms

- ☒ Boyer-Moore
- ☐ KMP
- ☐ Regex

## Laporan Tugas Kecil 4 - Strategi Algoritma – Muhammad Hasan (13518012) – K03

Extract

Jumlah: 6.575

Waktu: hingga minggu

Kalimat :

Dengan penambahan kasus ini, total kasus Covid-19 di Tanah Air menjadi 6.575 pasien hingga Minggu.

(test1.txt)

Jumlah: 47

Waktu: sabtu hingga minggu siang

Kalimat :

Ada 47 pasien Covid-19 yang meninggal dunia dalam kurun waktu Sabtu hingga Minggu siang ini sehingga totalnya menjadi 582 orang.

(test1.txt)

Jumlah: 3.112

Waktu: senin (20/4/2020)

Kalimat :

JAKARTA, KOMPAS.com - Jumlah pasien positif Covid-19 di Jakarta mencapai 3.112 orang per Senin (20/4/2020) ini.

(test2.txt)

Jumlah: 2

Waktu: pada minggu (19/4/2020)

Kalimat :

Pasien yang terjangkit virus corona tipe 2 (SARS-CoV-2) ini bertambah 79 orang dibandingkan data terakhir pada Minggu (19/4/2020), sebanyak 3.033 orang.

(test2.txt)

Jumlah: 3.112

Waktu: (20/4/2020)

Kalimat :

Kepala Dinas Kesehatan DKI Jakarta Widyastuti mengatakan, dari 3.112 pasien positif Covid-19, 237 pasien dinyatakan sembuh.

(test2.txt)

Jumlah: 30

Waktu: pada minggu

Kalimat :

Pasien sembuh bertambah 30 orang dibandingkan data pada Minggu, yakni 207 orang.

(test2.txt)

Jumlah: 297

Waktu: kemarin

Kalimat :

Sementara itu, jumlah pasien meninggal sebanyak 297 orang, bertambah 5 orang dibandingkan data kemarin atau sebanyak 292 orang.

(test2.txt)

Jumlah: 1.826

Waktu: (20/4/2020)

Kalimat :

Kemudian, 1.826 pasien masih dirawat di rumah sakit.

(test2.txt)

Jumlah: 6.575

Waktu: (20/4/2020)

Kalimat :

Sementara di level nasional, total ada 6.575 pasien positif Covid-19 yang tersebar di semua provinsi di Indonesia.

(test2.txt)

Jumlah: 6.575

Waktu: (20/4/2020)

Kalimat :

Sementara di level nasional, total ada 6.575 pasien positif Covid-19 yang tersebar di semua provinsi di Indonesia.

(test2.txt)

Jumlah: 686

Waktu: (20/4/2020)

Kalimat :

Dari total pasien, 686 pasien dinyatakan sembuh, sedangkan 582 pasien meninggal dunia.

(test2.txt)

### 3.2.5 Test 5 (Multiple files)

# Information Extractor App

Filenames

test4.txt,test5.txt,test6.txt

Masukkan nama file beserta tipe ekstensinya dipisahkan dengan koma (contoh: test1.txt,test2.txt,test3.txt)

Keyword

positif

Algorithms

- ☒ Boyer-Moore
- ☐ KMP
- ☐ Regex

Extract

Jumlah: 8

Waktu: hingga senin (20/4/2020) pukul 12.00 wib

Kalimat :

Berdasarkan data pemerintah hingga Senin (20/4/2020) pukul 12.00 WIB, ada penambahan 8 pasien yang tutup usia setelah dinyatakan positif virus corona dalam 24 jam terakhir.

(test5.txt)

Jumlah: 185

Waktu: (20/4/2020)

Kalimat :

"Kasus positif yang kita dapatkan pada hari ini 185 orang, sehingga totalnya 6.760 orang," ujar Achmad Yurianto.

(test5.txt)

Jumlah: 15

Waktu: selasa (21/4/2020)

Kalimat :

Kasus Positif Corona COVID-19 Bertambah 15 OrangKonferensi pers perkembangan kasus virus corona baru yang memicu COVID-19 di Gedung Grahadi, Selasa (21/4/2020) (Foto: Liputan6.com/Dian Kurniawan)Sebelumnya, pasien terkonfirmasi positif Corona COVID-19 kembali mengalami lonjakan di Jawa Timur (Jatim).

(test6.txt)

Jumlah: 603

Waktu: Selasa malam (21/4/2020)

Kalimat :

"Ada 603 pasien terkonfirmasi positif, sementara yang dirawat 444 pasien," ujar Gubernur Jatim, Khofifah Indar Parawansa di Gedung Negara Grahadi Surabaya, Selasa malam (21/4/2020).

(test6.txt)

Jumlah: 2.255

Waktu: (21/4/2020)

Kalimat :

"Tambahan kasus baru yang terkonfirmasi positif dari Bangkalan, Sidoarjo, Surabaya," ucap mantan Menteri Sosial kabinet Indonesia Kerja ini. Sementara itu, Pasien Dalam Pengawasan (PDP) menjadi 2.255 pasien.

(test6.txt)

### 3.2.6 Test 6 (Multiple files)

#### Information Extractor App

Filenames

test7.txt,test8.txt,test9.txt,test10.txt

Masukkan nama file beserta tipe ekstensinya dipisahkan dengan koma (contoh: test1.txt,test2.txt,test3.txt)

Keyword

daerah

Algorithms

- ☒ Boyer-Moore
- ☐ KMP
- ☐ Regex

Extract

Jumlah: 29

Waktu: pada 19 April 2020

Kalimat :

Jumlah kabupaten/kota yang memiliki kasus positif itu bertambah 29 daerah pada hari ini. Detail update data kasus positif corona di Indonesia pada 19 April 2020: Jumlah kasus positif baru: 327 pasien Total jumlah kasus positif: 6.575 pasien Total jumlah pasien positif sembuh: 686 orang Total jumlah pasien positif meninggal: 582 jiwa Total jumlah PDP: 15.646 orang Total jumlah ODP: 178.883 orang Total spesimen dites uji PCR: 47.478 spesimen Total jumlah kasus diperiksa spesimen: 42.219 orang Daftar 10 Provinsi dengan kasus positif corona terbanyak per 19 April 2020: 1.

(test9.txt)