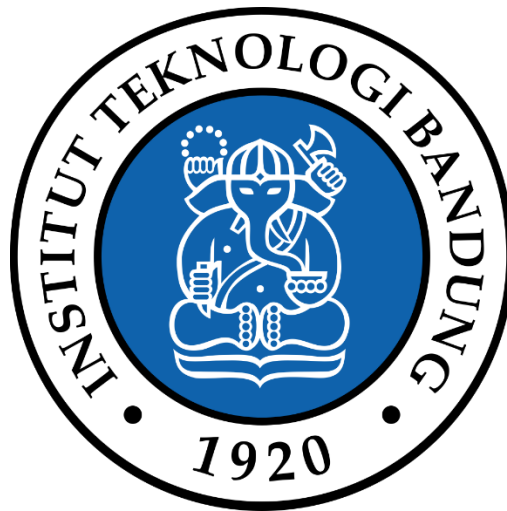


LAPORAN TUGAS KECIL 4
IF2211 – STRATEGI ALGORITMA

**Ekstraksi Informasi dari Artikel Berita
dengan Algoritma Pencocokan String**



Disusun oleh
Tony Eko Yuwono
13518030

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

1. Deskripsi

1.1. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang mencocokkan strik dari kiri ke kanan (seperti Brute Force), namun penggeserannya dilakukan lebih cerdas daripada brute force karena menggunakan larik LPS (*border function*) pada tahap *preprocessing*. Pada tahap preprocessing, dicari sebuah larik LPS (proper prefix terpanjang yang juga suffix) pada setiap substring dari teks *pattern* mulai dari substring dengan panjang 1 hingga N.

Setelah membuat larik LPS, algoritma KMP dimulai dari indeks ke-0 teks dan indeks ke-0 *pattern*. Pencarian dilakukan dengan melakukan penggeseran hingga ditemukan ketidakcocokan hingga pembacaan teks berakhir atau ditemukan substring dimana semua huruf pada substring cocok dengan *pattern*. Jika ditemukan ketidakcocokan pada sebuah indeks (misalkan j), maka pattern akan digeser sehingga dimulai dari indeks LPS(k) dimana k adalah j-1. Jika ditemukan kecocokan terhadap semua huruf pada pattern, indeks pertama kecocokan teks akan dicatat.

1.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore terdiri dari dua teknik, yaitu teknik *looking-glass*, dan teknik *character-jump*. Teknik *looking-glass* adalah teknik untuk menemukan *pattern* dalam teks dengan menelusuri pattern secara mundur (dari kanan ke kiri). Sedangkan teknik *character-jump* adalah teknik yang mengurus pergeseran karakter pada pattern, dimana jika *mismatch* terjadi pada $T[i] \neq P[j]$, terdapat tiga kemungkinan, yaitu:

1. Jika pada pattern terdapat huruf pada $P[j]$, geser pattern ke kanan hingga $T[i] == P[j]$.
2. Jika pada pattern terdapat huruf pada $P[j]$ tetapi pergeseran ke kanan tidak dimungkinkan, maka geser pattern 1 langkah sehingga berada pada $T[i+1]$.
3. Jika kasus pertama dan kedua tidak dapat diaplikasikan, geser $P[0]$ hingga bertemu $T[i+1]$.

Pergeseran tersebut akan dilakukan hingga semua huruf pada pattern cocok dengan substring dari teks, atau pembacaan teks sudah berakhir.

1.3. Regular Expression

Regular Expression (Regex) adalah deretan karakter spesial yang mendefinisikan sebuah *pattern* dalam sebuah string. Regex dapat melakukan pencocokan string dengan efisien. Saat ini Regex telah digunakan pada berbagai bahasa pemrograman, termasuk Python dimana terdapat modul bawaan yang dapat diimport dengan nama `re`. Pattern pada Regex memiliki struktur tersendiri yang harus dibaca misalnya metakarakter titik (`.`) yang menandakan karakter tersebut bisa diisi apa saja.

2. Kode Program

Program dibuat dengan bahasa Python dan framework Flask. Berikut adalah kode program algoritma KMP, BM, atau Regex pada aplikasi ini:

2.1. kmp.py

```
def computeLPS(pattern):
    '''Compute LPS (longest proper prefix which is also suffix) for preprocessing in KMP Algorithm'''
    lps = [0 for i in range(len(pattern))]
    lastLps = 0 #variabel untuk mencatat lps
    indexNow = 1 #pencarian lps dari index-1 (lps[0] pasti 0)
    while indexNow < len(pattern):
        if pattern[indexNow].lower() == pattern[lastLps].lower():
            lastLps+=1 #geser lastLps
            lps[indexNow] = lastLps
            indexNow+=1 #geser index saat ini
        elif lastLps > 0: #karakter pada pattern ke-i dan ke-lastLps tidak sama
            lastLps = lps[lastLps-1]
        else:
            lps[indexNow] = 0
            indexNow+=1

    return lps

def search(text, pattern):
    '''String matching KMP Algorithm modified, can search multiple pattern occurrence in a text'''
    lps = computeLPS(pattern)
    idxText = idxPattern = 0
    idxFound = []
    while idxText < len(text):
        if (pattern[idxPattern].lower() == text[idxText].lower()):
            idxText+=1
            idxPattern+=1
            if idxPattern == len(pattern):
```

```

        idxFound.append(idxText-idxPattern)
        idxPattern = 0
    elif idxPattern > 0:
        idxPattern = lps[idxPattern-1]
    else:
        idxText+=1
return idxFound

```

2.2. bm.py

```

def buildLast(pattern):
    '''The preprocessing function for Boyer-Moore Algorithm'''
    last = [-1 for i in range(128)]
    for i in range(len(pattern)):
        last[ord(pattern[i].lower())] = i
    return last

def search(text, pattern):
    '''String matching Boyer Moore Algorithm modified, can search multiple pattern occurrence in a text'''
    last = buildLast(pattern)
    idxText = idxPattern = len(pattern)-1
    idxFound = []
    while idxText < len(text):
        if pattern[idxPattern].lower() == text[idxText].lower():
            if idxPattern == 0:
                idxFound.append(idxText)
                idxText = idxText+2*len(pattern)-1
            #geser text ke posisi string setelah matched
            idxPattern = len(pattern)-1 #reset idxPattern
        else:
            idxText-=1
            idxPattern-=1
    else:
        lastOcc = last[ord(text[idxText].lower())]
        idxText += len(pattern)-min(idxPattern, 1+lastOcc)
        idxPattern = len(pattern)-1
    return idxFound

```

2.3. rgx.py

```
import re
import datetime

def search(text, pattern):
    '''String matching with Regex'''
    regex = re.compile(pattern, re.IGNORECASE)
    return [m.start() for m in regex.finditer(text)]

def searchDigit(sentence):
    number = searchDigit2(sentence)
    if number[0] == ':': number = searchDigit1(sentence)
    return number

def searchDigit1(text):
    '''Search number with digit format'''
    return [x.group() for x in re.finditer(r'[\d+\.]?(\d+)?(?:[pP]uluh|[rR]atus|[rR]ibu|[jJ]uta)?(?:\d+)?(\d+\.)?(\d+)?(?:[pP]uluh|[rR]atus|[rR]ibu|[jJ]uta)?(?:\d+)?', text)]

def searchDigit2(sentence):
    '''Search number with alphabetic format'''
    all = re.findall(r'((?:^(?:\d+)(?:\d+)?|(?:[lL]ebih|[kK]urang(?:dari)?(?:[sS]atu|[dD]ua|[tT]iga|[eE]mpat|[lL]ima|[eE]nam|[tT]ujuh|[dD]elapan|[sS]embilan))?(?:%|[pP]ersen|[pP]uluh|[rR]atus|[rR]ibu|[jJ]uta)?)', sentence)
    return all

def searchDate(sentence):
    result = searchDate5(sentence)
    if not result: result = searchDate4(sentence)
    if not result: result = searchDate3(sentence, '/')
    if not result: result = searchDate3(sentence, '-')
    if not result: result = searchDate2(sentence, '/')
    if not result: result = searchDate2(sentence, '-')
    return result

def searchDate2(sentence, separator):
    '''Search date with pattern: DD/MM/YY or DD-MM-YY'''
    result = []
    pattern = '\d{1,2}' + separator + '\d{1,2}' + separator + '\d{2}'
    dateformat = "%d" + separator + "%m" + separator + "%y"
    regex = re.compile(pattern)
    for match in regex.finditer(sentence):
        try:
            datetime.datetime.strptime(match.group(), dateformat)
            result.append(match.group())
        except ValueError:
            pass
    return result

def searchDate3(sentence, separator):
    '''Search date with pattern: DD/MM/YYYY or DD-MM-YYYY'''
    result = []
    pattern = '\d{1,2}' + separator + '\d{1,2}' + separator + '\d{4}'
    dateformat = "%d" + separator + "%m" + separator + "%Y"
    regex = re.compile(pattern)
    for match in regex.finditer(sentence):
        try:
            datetime.datetime.strptime(match.group(0), dateformat)
            result.append(match.group(0))
        except ValueError:
            pass
```

```

return result

def searchDate4(sentence):
    '''Search date with pattern: Day Date Month Year Time TimeRegion'''
    result = []
    pattern = "(?:Kemarin(?:lusa)?,)?(?:([sS]enin)?(?:[sS]elasa)?(?:[rR]abu)?(?:[kK]amis)?(?:[jJ]umat)?(?:[sS]abtu)?(?:[mM]inggu)?),-)?(?:((0?[1-9])|([12][0-9])|3[01])[ -]/)(?:[jJ]an(?:uari)?|[fF]eb(?:uari)?|[mM]ar(?:et)?|[aA]pr(?:il)?|[mM]ei|[jJ]uni|[jJ]uli|[aA]gustus|[sS]ept(?:ember)?|[oO]kt(?:ober)?|[nN]ov(?:ember)?|[dD]es(?:ember)?)[- -]/)(?:\d{4})?[- -]/)?(?:([pP]ukul)?(?:\d{2}:\d{2})?(?:([wW][iI])([tT][aA]?[bB]))?(?:(:yang)?lalu)?"
    regex = re.compile(pattern)
    if regex.search(sentence): result.append(regex.search(sentence).group())
    return result

def searchDate5(sentence):
    '''Search date with pattern: Day (Date/Month/Year or Date-Month-Year) Time TimeRegion'''
    result = []
    pattern = "(?:Kemarin(?:lusa)?,)?(?:([sS]enin)?(?:[sS]elasa)?(?:[rR]abu)?(?:[kK]amis)?(?:[jJ]umat)?(?:[sS]abtu)?(?:[mM]inggu)?),-)?(?:((0?[1-9])|([12][0-9])|3[01])[ -]/)(0?[1-9]|1[12])[ -]/)(?:\d{4})?[- -]/)?(?:([pP]ukul)?(?:\d{2}[:.]?\d{2})?(?:([wW][iI])([tT][aA]?[bB]))?(?:(:yang)?lalu)?"
    regex = re.compile(pattern)
    if regex.search(sentence): result.append(regex.search(sentence).group())
    return result

```

2.4. algo.py

```

from stringmatcher import decomposer, kmp, bm, rgx

def search(listFile, keyword, method, currentFolder):
    fileResult = []
    for file in listFile:
        searchResult = []
        text = decomposer.decomposition(currentFolder+"/"+file)
        if method == "kmp":
            for sentence in text:
                idxFound = kmp.search(sentence, keyword);
                if idxFound: searchResult.append((rgx.searchDate(sentence), rgx.searchDigit(sentence), sentence))
        elif method == "bm":
            for sentence in text:
                idxFound = bm.search(sentence, keyword);
                if idxFound: searchResult.append((rgx.searchDate(sentence), rgx.searchDigit(sentence), sentence))
        elif method == "regex":
            for sentence in text:
                idxFound = rgx.search(sentence, keyword);
                if idxFound: searchResult.append((rgx.searchDate(sentence), rgx.searchDigit(sentence), sentence))
        fileResult.append((file, searchResult))
    return fileResult

```

2.5. decomposer.py

```
from nltk import tokenize

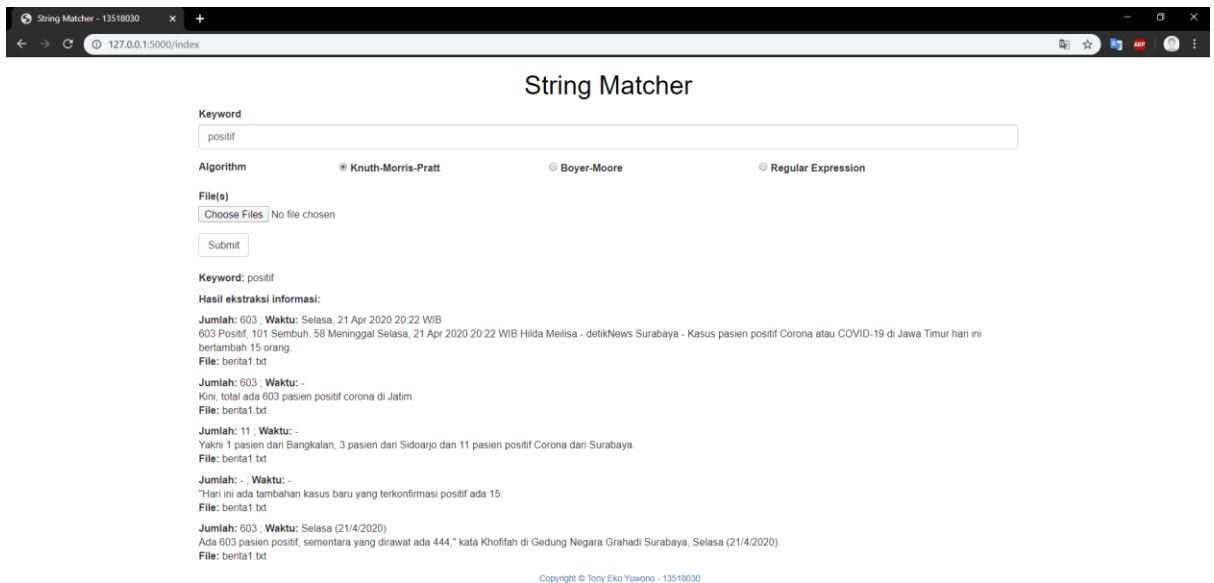
def decomposition(path):
    '''Fungsi untuk mendekomposisi teks paragraf menjadi array of kalimat'''
    file = open(path)
    text = tokenize.sent_tokenize(file.read())
    result = []
    for sentence in text:
        result.append(sentence.replace("\n", " "))
    return result
```

3. Screenshot Program

Program dijalankan pada komputer dengan spesifikasi:

Processor : Intel Core i7-8585U
Memori : 16 GB DDR4
Hard Disk : SSD M.2 PCIe

1. Berita1.txt



2. Berita9.txt

The screenshot shows the String Matcher web application interface. The browser address bar displays '127.0.0.1:5000/index'. The page title is 'String Matcher'. The 'Keyword' input field contains 'trump'. The 'Algorithm' section has three radio buttons: 'Knuth-Morris-Pratt' (selected), 'Boyer-Moore', and 'Regular Expression'. The 'File(s)' section has a 'Choose Files' button and the text 'No file chosen'. Below the 'Submit' button, the 'Keyword: trump' is displayed. The 'Hasil ekstraksi informasi:' section contains the following text:

Jumlah: 15 ; **Waktu:** Rabu, 15 Apr 2020 06:48 WIB
 Trump Umumkan Penghentian Dana untuk WHO Rabu, 15 Apr 2020 06:48 WIB Rita Uli Hutapea - detikNews Washington - Presiden Amerika Serikat Donald Trump mengumumkan penghentian pendanaan untuk Organisasi Kesehatan Dunia (WHO).
File: berita9.txt

Jumlah: - ; **Waktu:** -
 Trump beritaskan badan kesehatan PBB tersebut telah menutup-nutupi keseriusan wabah virus Corona di China sebelum menyebar ke seluruh dunia.
File: berita9.txt

Jumlah: - ; **Waktu:** -
 Dalam konferensi pers, Trump mengatakan dirinya telah memerintahkan pemerintahannya untuk menghentikan pendanaan selagi "peninjauan dilakukan untuk menilai peran WHO dalam salah kelola yang parah dan menutupi penyebaran virus Corona."
File: berita9.txt

Jumlah: 400 juta ; **Waktu:** -
 Menurut Trump, WHO tidak transparan mengenai wabah tersebut dan AS -- pendana terbesar WHO yang menyediakan US\$ 400 juta tahun lalu -- kini akan membahas "apa yang akan kami lakukan dengan semua uang yang ditujukan ke WHO itu."
File: berita9.txt

Jumlah: - ; **Waktu:** -
 Trump menyebut WHO bias terhadap China dan berkolusi untuk mencegah saingan utama ekonomi AS itu terbuka tentang bencana kesehatan yang sedang berlangsung.
File: berita9.txt

Jumlah: - ; **Waktu:** -
 Menurut Trump, hal ini telah menyebabkan negara-negara lain kehilangan waktu krusial untuk bersiap dan menunda keputusan untuk menghentikan perjalanan internasional.
File: berita9.txt

Jumlah: - ; **Waktu:** -
 "Sementara WHO melakukan tugasnya untuk membawa para ahli medis ke China untuk menilai secara obyektif situasi di lapangan dan memberikan kurangnya transparansi China, wabah itu bisa diatasi pada sumbernya dengan kematian yang sangat sedikit," cetus Trump.

3. Berita10.txt

The screenshot shows the String Matcher web application interface. The browser address bar displays '127.0.0.1:5000/index'. The page title is 'String Matcher'. The 'Keyword' input field contains 'luhut'. The 'Algorithm' section has three radio buttons: 'Knuth-Morris-Pratt' (selected), 'Boyer-Moore', and 'Regular Expression'. The 'File(s)' section has a 'Choose Files' button and the text 'No file chosen'. Below the 'Submit' button, the 'Keyword: luhut' is displayed. The 'Hasil ekstraksi informasi:' section contains the following text:

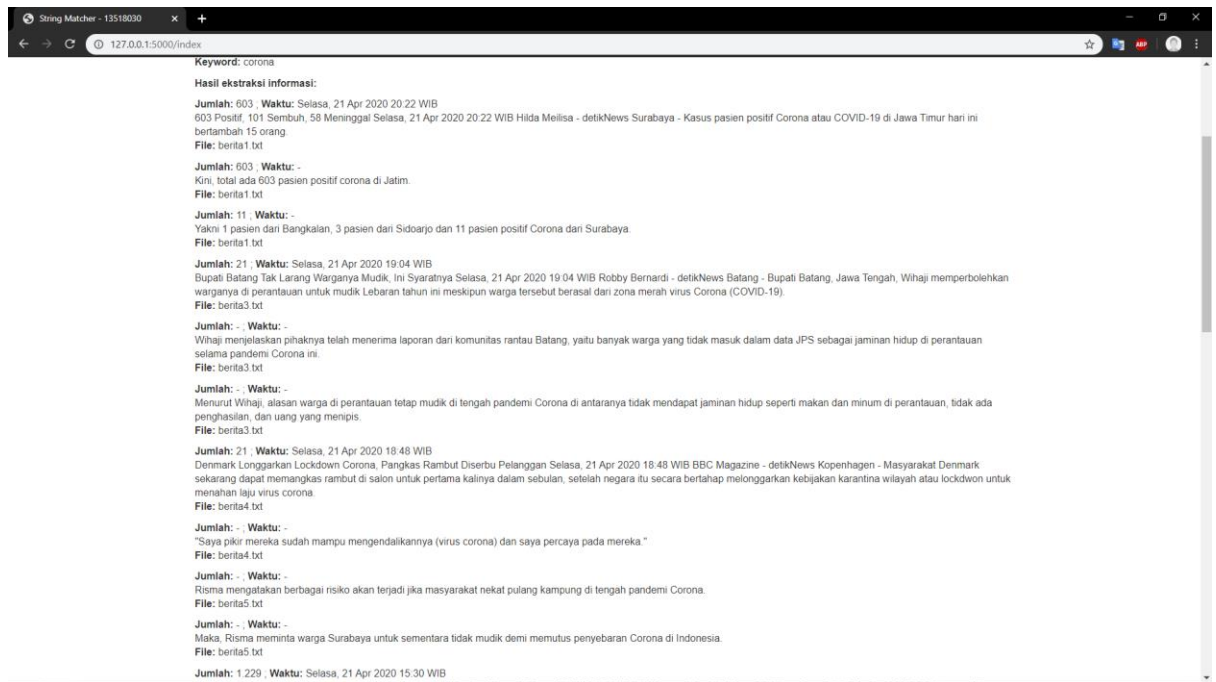
Jumlah: 500 Ribu ; **Waktu:** Selasa, 21 Apr 2020 20:45 WIB
 Luhut Resah Ada 500 Ribu TKI di Malaysia yang Mau Pulang Selasa, 21 Apr 2020 20:45 WIB Anisa Indrainsi - detikFinance Jakarta - Menteri Koordinator Bidang Kemaritiman dan Investasi Luhut Binsar Pandjaitan mewaspadai kepulangan ratusan ribu tenaga kerja Indonesia (TKI) dari luar negeri.
File: berita10.txt

Jumlah: - ; **Waktu:** Selasa (21/4/2020)
 Kemudian dari Taiwan," kata Luhut yang juga sebagai Menteri Perhubungan Ad Interim saat rapat virtual dengan Komisi V DPR RI, Selasa (21/4/2020).
File: berita10.txt

Jumlah: - ; **Waktu:** -
 Luhut khawatir kepulangan mereka tidak melalui cara-cara yang sesuai prosedur.
File: berita10.txt

Copyright © Tony Eko Yawono - 13518030

4. 11 File Text Berita



5. About Me (Perihal)



String Matcher

Hai! Nama saya adalah Tony Eko Yuwono. saya lahir di Kota Madiun, Jawa Timur, kota yang terkenal dengan kota Pecel. Saya bersekolah di kota tersebut dari TK hingga SMP, kemudian saat SMA merantau ke Kota Malang. Setelah menyelesaikan SMA, saya berkuliah di Institut Teknologi Bandung.

Sedikit pengantar, website ini adalah website yang dibuat untuk memenuhi tugas Strategi Algoritma dengan topik **"String Matching"** dengan menggunakan algoritma *Knuth-Morris-Pratt*, *Boyer-Moore*, atau *Regular Expression*.

Saya dapat dihubungi di:
 Email: tony.yuwono@gmail.com
 Github: [tonyekko](#)
 LinkedIn: [Tony Eko Yuwono](#)

4. Tabel Penilaian

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 data uji	✓	