TUGAS KECIL IV
IF2211 STRATEGI ALGORITMA
EKSTRAKSI INFORMASI DARI ARTIKEL BERITA DENGAN ALGORITMA PENCOCOKAN STRING

LAPORAN TUGAS KECIL

Diajukan untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma

Oleh

MUHAMMAD FAUZAN AL-GHIFARI

(13518112 - K01)

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2020

A. Deskripsi Singkat Pncocokan String KMP

Algoritma Knuth-Morris-Pratt merupakan salah satu algoritma pencarian string (string matcing) yang dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977 (Wikipedia : Knuth-Morris-Pratt). Langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan string yaitu (modifikasi):

String pattern (kata yang dicari) akan dipecah menjadi array karakter String text (teks, artikel, dsb) akan dipecah menjadi array karakter Menentukan lompatan yang akan dilakukan ketika pencarian (funsi preKMP()) Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks. ()

Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi: Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.

Algoritma kemudian menggeser pattern berdasarkan tabel next (lompat), lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Cuplikan program bagian algoritma KMP buatan saya

```
from copy import deepcopy

# Knuth Morris Pratt Algorithm
def KMP (T, P):
    bf = borderFunction(P)
    result = []
    n = len(T)
    m = len(P)

    i = 0 # for teks
    j = 0 # for pattern
    temp = j
    while (i < n):
        while (j < m and i < n):
            if (j == -99):
                j = temp+1
                i += 1

            elif (j == m-1 and P[j] == T[i]):
                result.append(i-m+1)
```

```
                i += 1
                j = 0

            elif (T[i] == P[j]):
                i += 1;
                j += 1;

            else: # miss match
                temp = j
                j = bf[j]
        i+=1

    return result

# Make Border Function
def borderFunction(pattern):
    j = len(pattern);
    bf = []
    for i in range (j):
        bf.append(calculateBF(pattern, i))
        bf[0] = -99
    return bf

# Calculate BF for specific j
def calculateBF(pattern, j):
    ret = 0
    stop = False
    pref = []
    suf = []

    # calculate pref
    input = []
    for i in range (0, j, 1):
        input.append(pattern[i])
        pref.append(deepcopy(input))

    # calculate suf
    input.clear()
    for i in range (j-1, 0, -1):
        input.insert(0,pattern[i])
        suf.append(deepcopy(input))

    # looking for b[k]
    i = len(suf)-1
    while i >= 0 and not(stop):
        if (pref[i] == suf[i]):
            ret = len(pref[i])
            stop = True
```

```
        i -= 1
    return ret

from nltk.tokenize import sent_tokenize

def main():
    f = open ("text.txt", "r")
    T = f.read()
    # P = "COVID-19"
    # T = "saya adalah fauzan keren, memang fauzan keren, sudah tentu fauzan
keren"
    # P = "fauzan"
    # print(KMP(T,P))


if __name__ == "__main__":
    main()
```

B. Deskripsi Pencocokan String dengan Boyeer Moore

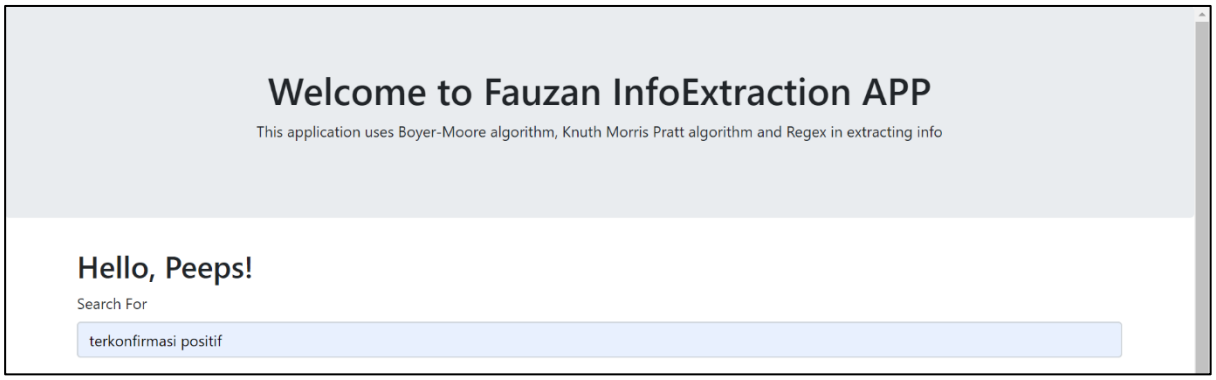
Algoritma Boyer-Moore diperkenalkan oleh Bob Boyer dan J.S. Moore pada tahun 1977. Pada algoritma ini pencocokan kata dimulai dari karakter terakhir kata kunci menuju karakter awalnya. Jika terjadi perbedaan antara karakter terakhir kata kunci dengan kata yang dicocokkan maka karakter-karakter dalam potongan kata yang dicocokkan tadi akan diperiksa satu per satu. Hal ini dimaksudkan untuk mendeteksi apakah ada karakter dalam potongan kata tersebut yang sama dengan karakter yang ada pada kata kunci. Apabila terdapat kesamaan, maka kata kunci akan digeser sedemikian rupa sehingga posisi karakter yang sama terletak sejajar, dan kemudian dilakukan kembali pencocokan karakter terakhir dari kata kunci. Sebaliknya jika tidak terdapat kesamaan karakter, maka seluruh karakter kata kunci akan bergeser ke kanan sebanyak m karakter, di mana m adalah panjang karakter dari kata kunci. Booyer-Moore merupakan salah satu Algortima Pattern Matching yang cukup terkenal. Algoritma ini menggunakan beberapa kasus pengecekan teks (input karakter yang akan dibaca) dengan Pattern (pola yang akan disaring). Algoritma Boyer-Moore adalah algoritma pencarian string yang mencari dengan cara membandingkan sebuah huruf dengan huruf yang ada di pattern yang dicari, dan menggeser pattern tersebut hingga posisinya sama dengan teks yang dicari dan membandingkan kata tersebut. Cara ini disebut character jump

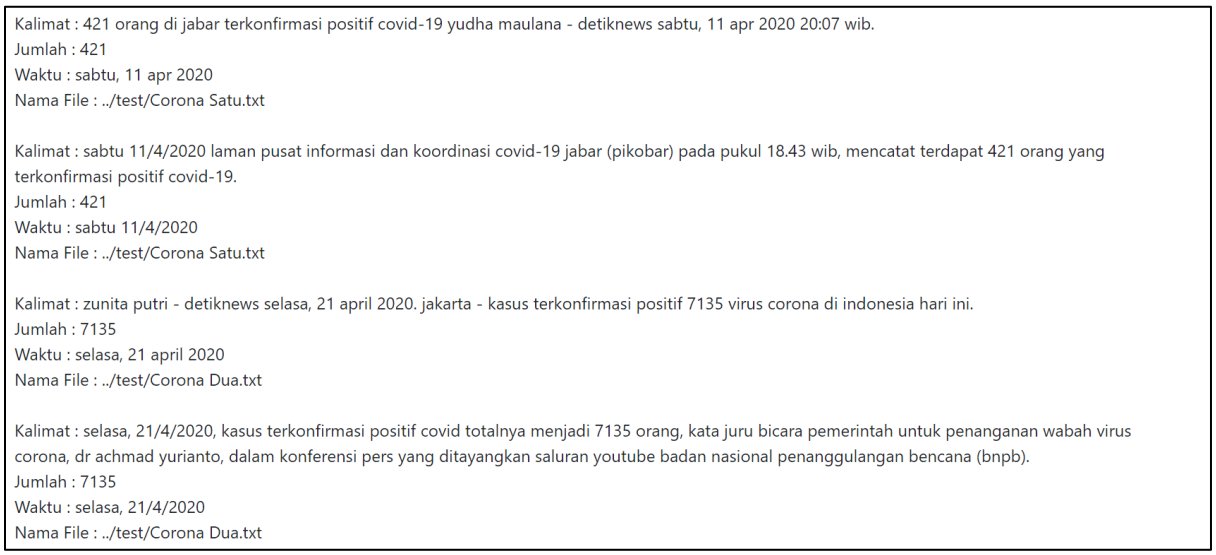
Contoh Algoritma Boyer Moore buatan saya

```
# tuple for LO
class tuple:
    def __init__(self):
        self.var = ""
        self.val = 0

# Make Lo Table
def lastOccurence(teks, pattern):
    variation = varChar(teks)
    lo = []

    for i in range (len(variation)):
        item = tuple()
        item.var = variation[i]
        item.val = last(pattern, variation[i])
        lo.append(item)

    return lo


# looking for last occurence char in list
def last(list, char):
    ret = -1
    found = False
    i = len(list)-1
    while i >= 0 and not(found):
        if (list[i] == char):
            ret = i
            found = True
        i -= 1
    return ret


# return variation char in teks
def varChar(teks):
    var = []
    for i in range (len(teks)):
        ret = findVar(var ,teks[i])
        if (ret == -1):
            var.append(teks[i])
    return var


# looking for variation char index
def findVar(var, findvar):
    ret = -1
    for i in range(len(var)):
```

```
        if (var[i] == findvar):
            ret = i
    return ret

# looking for char value in lo
def valueLo(char, lo):
    for i in range (len(lo)):
        if (lo[i].var == char):
            return lo[i].val

# algoritma boyer moore
def BM(T, P):
    lo = lastOccurence(T, P)
    result = []
    n = len(T)
    m = len(P)
    j = m-1 # for pattern
    i = j # for teks

    while(i < n):
        while(j < m and i < n):
            if (j == 0 and T[i] == P[j]):
                result.append(i)
                i = i + m
                j = m-1
            elif (T[i] == P[j]):
                i-=1
                j-=1

            # missmatch
            elif (T[i] != P[j]):
                # kasus ketiga
                if (valueLo(T[i], lo) == -1):
                    j = m-1
                    i = i + m
                # kasus pertama
                elif (valueLo(T[i], lo) < j):
                    j = m-1
                    i = i + m - (valueLo(T[i], lo) + 1)
                # kasus kedua
                elif (valueLo(T[i], lo) > j):
                    i = i + m - j
                    j = m-1

    return result
```

C. Screenshot input-output program





Kalimat : 421 orang di jabar terkonfirmasi positif covid-19 yudha maulana - detiknews sabtu, 11 apr 2020 20:07 wib.
Jumlah : 421
Waktu : sabtu, 11 apr 2020
Nama File : ../test/Corona Satu.txt

Kalimat : sabtu 11/4/2020 laman pusat informasi dan koordinasi covid-19 jabar (pikobar) pada pukul 18.43 wib, mencatat terdapat 421 orang yang terkonfirmasi positif covid-19.
Jumlah : 421
Waktu : sabtu 11/4/2020
Nama File : ../test/Corona Satu.txt

Kalimat : zunita putri - detiknews selasa, 21 april 2020. jakarta - kasus terkonfirmasi positif 7135 virus corona di indonesia hari ini.
Jumlah : 7135
Waktu : selasa, 21 april 2020
Nama File : ../test/Corona Dua.txt

Kalimat : selasa, 21/4/2020, kasus terkonfirmasi positif covid totalnya menjadi 7135 orang, kata juru bicara pemerintah untuk penanganan wabah virus corona, dr achmad yurianto, dalam konferensi pers yang ditayangkan saluran youtube badan nasional penanggulangan bencana (bnpb).
Jumlah : 7135
Waktu : selasa, 21/4/2020
Nama File : ../test/Corona Dua.txt

Kalimat : per hari jumlah orang dalam pemantauan odp di jabar mencapai 2.8775 orang.
Jumlah : 2.8775
Waktu : sabtu, 11 apr 2020
Nama File : ../test/Corona Satu.txt

Kalimat : jumlah orang dalam pemantauan odp 499 orang.
Jumlah : 499
Waktu : selasa, 31 maret 2020
Nama File : ../test/Corona Tiga.txt

Kalimat : jumlah orang dalam pemantauan odp 498 orang.
Jumlah : 498
Waktu : rabu, 1 april 2020
Nama File : ../test/Corona Empat.txt

Kalimat : jumlah orang dalam pemantauan odp 504 orang.
Jumlah : 504
Waktu : kamis, 2 april 2020
Nama File : ../test/Corona Lima.txt

Spesifikasi komputer yang digunakan

Nama Laptop : Lenovo IdeaPad S340

OS : Windows 10

RAM : 8192 MB / 8 GB

Processor : AMD Ryzen 3 3200U with Radeon Vega Mobile Gfx (4CPUs), ~2.6GHz

No	Point	Ya	Tidak
1	Program berhasil dikompilasi	✓	
2	Program berhasil *running*	✓	
3	Program dapat menerima input dan menuliskan output	✓	
4	Luaran sudah benar untuk semua data uji	✓	