Pemanfaatan Algoritma Knuth-Morris-Pratt dan Boyer-Moore Pada Pencarian Kata Dalam Artikel

Giovani Anggasta 13519155
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519155@std.stei.itb.ac.id

***Abstract*—Pencarian kata pada suatu artikel sering kali diperlukan. Pencarian kata pada artikel mempermudah pembaca artikel untuk menemukan materi para artikel sesuai dengan kata kunci yang diinginkan. Membaca suatu artikel dapat lebih efisien dengan adanya pencarian kata karena dapat menemukan materi yang sesuai pada artikel dengan waktu yang lebih singkat tanpa harus membaca keseluruhan artikel. Pencarian kata ini dapat memanfaatkan salah beberapa algoritma string matching yaitu Knuth-Morris-Pratt (KMP) Algorithm dan juga Boyer-Moore Algorithm (BM) sebagai dasarnya.**

Keywords—Pencarian kata, Knuth-Morris-Pratt (KMP), Boyer-Moore(BM), Artikel

I. Pendahuluan

Menurut Kamus Besar Bahasa Indonesia (KBBI), artikel adalah sebuah karya tulis lengkap, misalnya seperti laporan berita atau esai dalam majalah, surat kabar, dan lain sebagainya. Artikel sering kali digunakan sebagai dasar dari suatu penelitian ataupun bahan bacaan sehari-hari. Selain itu, artikel juga dapat dijadikan sebagai sumber informasi. Artikel tentu memiliki banyak manfaat. Beberapa manfaat artikel yaitu sebagai sarana untuk menyampaikan gagasan, menyebarkan informasi, dan juga ilmu pengetahuan kepada masyarakat.

Media penyebaran dan penulisan artikel sangat beragam. Beberapa media yang sering kali digunakan sebagai wadah penyebaran dan penulisan artikel yaitu media cetak, media massa, dan juga media elektronik. Pada zaman kemajuan teknologi seperti saat ini, pembaca artikel sudah dimudahkan dengan banyaknya artikel yang sudah berbentuk digital atau berupa media elektronik. Beberapa contoh kumpulan artikel yang sudah berbentuk digital terdapat detik.com, kompas.com, dan masih banyak lagi.

Pada suatu artikel tentu memuat banyak tulisan. Dengan banyaknya tulisan pada artikel, terkadang pembaca kesulitan untuk mencari kata kunci yang diinginkan pada suatu artikel. Pembaca artikel perlu mencari satu-persatu kata kunci yang diinginkan pada suatu artikel dimana artikel tersebut terdapat banyak kata. Hal tersebut dapat dipermudah dengan menggunakan fitur pencarian kata pada suatu artikel. Pembaca artikel tidak perlu mencari satu-persatu kata kunci yang diinginkan pada suatu artikel.

Fitur pencarian kata pada artikel dapat memanfaatkan beberapa algoritma string matching yaitu Knuth-Morris-Pratt Algorithm dan juga Boyer-Moore Algorithm. Dengan fitur pencarian kata, pembaca artikel dapat mencari kata kunci yang sesuai pada artikel dengan lebih efisien tanpa harus mencari kata satu-persatu. Selain itu dengan adanya fitur pencarian kata pada artikel dapat mempersingkat waktu pengguna dalam pencarian kata kunci yang sesuai.

II. Landasan Teori

A. String Matching (Pencocokan String)

String merupakan sebuah tipe data yang digunakan dalam pemrograman, seperti integer dan juga floating point, namun string digunakan untuk merepresentasikan teks dibandingkan dengan angka dimana panjang dari string itu sendiri sangat abstrak. String terdiri atas sekumpulan karakter yang dapat juga dapat berisi spasi dan angka. Sebagai contoh, kata “kucing” dan kalimat “Terdapat 5 anak kucing” merupakan sebuah string, bahkan “23456” juga merupakan sebuah string apabila dispesifikasikan dengan benar. Biasanya, seorang programmer harus menyertakan tanda kutip untuk menyatakan bahwa variabel adalah sebuah string sehingga variabel tersebut tidak dikenali sebagai integer atau tipe data lainnya.

String matching atau pencocokan string merupakan subjek yang sangat penting dalam pemrosesan teks. Algoritma pencocokan string merupakan komponen dasar yang digunakan dalam implementasi perangkat lunak. Pencocokan string merupakan algoritma yang digunakan untuk menemukan kemunculan pola string dalam suatu string ada teks lainnya.

Terdapat berbagai macam algoritma yang dapat digunakan atau diimplementasikan dalam pengaplikasian pencocokan string. Algoritma-algoritma tersebut adalah:

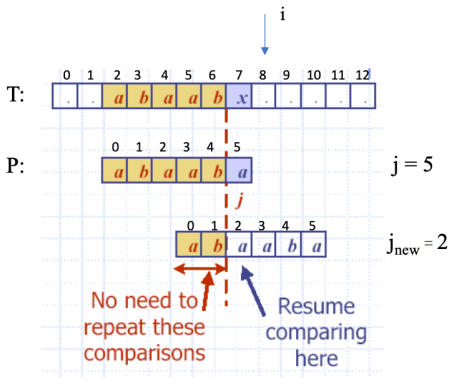
1. Algoritma Brute Fore
2. Algoritma Knuth-Morris-Pratt
3. Algoritma Boyer-Moore
4. Algoritma Zhu-Takaoka
5. Algoritma Karp-Rabin
6. Algoritma Shift-Or
7. Algoritma Aho-Corasick

dan masih banyak algoritma lainnya.

B. Knuth-Morris-Pratt Algorithm

Algoritma Knuth-Morris-Pratt melakukan pencocokan string dengan mencari pola dalam teks dengan urutan dari kiri ke kanan seperti algoritma brute force. Namun, algoritma Knuth-Morris-Pratt mengubah pola dengan lebih cerdas dibandingkan dengan algoritma brute force.

Jika terdapat ketidaksesuaian antara teks T dan pola P pada P[j], misalkan T[i] ≠ P[j] maka dapat digeser prefix terbesar dari P[0 .. j-1] yang merupakan suffix dari P[1 .. j-1] untuk menghindari pola perbandingan yang boros. Berikut ini merupakan contoh pergeseran prefixnya.



Gambar 2.1 Pergeseran prefix terbesar dari P[0..j-1] yang merupakan suffix dari P[1..j-1].

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Prefix yang mungkin dari P[0..j-1] adalah "a", "ab", "aba", "abaa", dan "abaab". Suffix yang mungkin dari P[1..j-1] adalah "b", "ab", "aab", dan "baab". Kemudian dilihat pattern terbesar yang beririsan antara prefix P[0..j-1] dengan suffix P[1..j-1] dimana pattern tersebut adalah "ab" dengan panjang string 2. Nilai 2 yang diperoleh dari irisan prefix dan suffix digunakan sebagai index j yang baru yang akan digunakan untuk pemeriksaan berikutnya sehingga j = 2. Sedangkan indeks baru untuk memulai pemeriksaan pada teks (T) yaitu indeks i dimana i adalah indeks saat terjadi mismatch sebelumnya.

Algoritma Knuth-Morris-Pratt akan melakukan preproses terhadap pattern P yang akan dicari kecocokannya pada teks T dengan mencari nilai prefix terbesar pada P[0..j-1] yang sama dengan suffix pada P[1..j-1] ketika terjadi mismatch pada P[j]. Dimisalkan j adalah posisi P dimana terjadi mismatch dan k adalah posisi j-1. Border function dengan parameter k akan mengembalikan ukuran prefix terbesar pada P[0..k] yang juga menjadi suffix dari P[1..k]. Border function sering juga disebut dengan failure function.

Berikut ini akan diberikan contoh Knuth-Morris-Pratt border function. Dimisalkan pattern P yaitu "abababababca" dengan panjang string 10 dan indeks dari P dimulai dari 0 sampai 9 dan k merupakan nilai j-1 dimana karakter pada pattern masih sama dengan karakter pada teks.

j	0	1	2	3	4	5	6	7	8	9
P [j]	a	b	a	b	a	b	a	b	c	a

Gambar 2.2 Pattern P

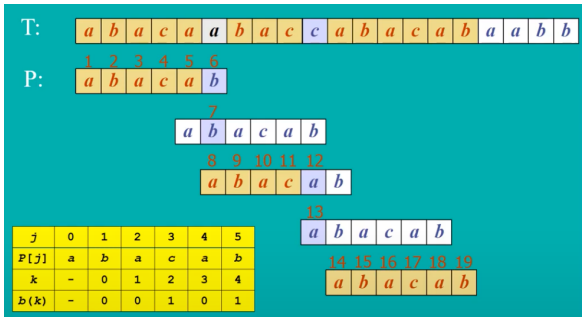
Sumber : https://www.youtube.com/watch?v=HWlSF8f-uWU&t=37s

Jika terjadi mismatch pada indeks j ke 0 maka tidak perlu menghitung border function. Jika j = 1 maka nilai k = , sehingga nilai dari border function adalah 0 karena tidak mungkin terdapat suffix terbesar pada P[1..0]. Begitu pula apabila nilai j = 2 sehingga k = 1, nilai dari border function adalah 0 karena tidak ada karakter yang sama karena suffix P[1..1].

Dimisalkan terjadi mismatch pada j = 4 sehingga didapatkan k = 3, maka prefix P[0..3] yang mungkin adalah "a", "ab", "aba", dan "abab" dan suffix P[1..3] yang mungkin adalah "b", "ab", dan "bab". Maka didapatkan karakter yang sama antara prefix P[0..3] dan suffix P[1..3] adalah "ab" sehingga nilai dari border function adalah 2.

Proses algoritma Knuth-Morris-Pratt secara umum adalah pertama menerima input teks dan juga pattern, kemudian menghitung border function untuk setiap nilai j pattern, dan selanjutnya dilakukan pencocokan string.

Berikut ini merupakan contoh pencocokan string dengan algoritma Knuth-Morris-Pratt.



Gambar 2.3 Contoh pencocokan string dengan algoritma Knuth-Morris-Pratt

Sumber : https://www.youtube.com/watch?v=HWlSF8f-uWU&t=37s

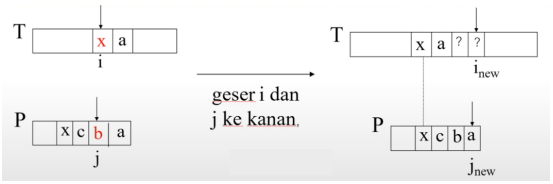
Tabel kuning di pojok kiri bawah pada Gambar 2.3 merupakan table dari nilai border function dan angka-angka yang berada diatas P merupakan urutan pemeriksaan. Pada pemeriksaan ke 6 terjadi mismatch antara T dengan P dimana indeks P saat terjadi mismatch (j) adalah 5 sehingga nilai border functionnya adalah 1. Pemeriksaan selanjutnya dimulai dari j = 1 dan i (indeks T) dimulai dari terjadinya mismatch antara P dan T, namun dipemeriksaan ke 7 terjadi mismatch lagi dengan sehingga akan dilihat nilai border function j = 1 adalah 0. Maka pemeriksaan selanjutnya dimulai dari j = 0. Pada pemeriksaan ke-12 terjadi mismatch kembali dengan indeks j = 4, sehingga nilai border function untuk j = 4 adalah 0. Maka pemeriksaan selanjutnya dimulai dari indeks j = 0. Pada pemeriksaan ke-13 terjadi mismatch lagi pada j = 0 sehingga tidak perlu dicari kembali nilai dari border function. Untuk pemeriksaan selanjutnya sudah ditemukan teks yang sesuai dengan pattern.

C. *Boyer-Moore Algorithm*

Algoritma Boyer-Moore memiliki mekanisme umum yaitu *the looking-glass technique* dan *the character-jump technique*. Untuk *the looking-glass technique* memeriksa kecocokan pattern P dengan teks T dimulai dari indeks terakhir pada P. Pemeriksaan terhadap T tetap dimulai dari awal, dalam hal ini indeks i akan dimulai pada nilai m-1 jika panjang P adalah m.

Untuk *the character-jump* dilakukan ketika terjadi mismatch (P[j] ≠ T[i] dengan T[i] = x). Terdapat 3 kasus yang mungkin terjadi. Kasus pertama yaitu ketika terjadi mismatch pada T[i] dan P[j], dan karakter pada T[i] adalah x. Jika terdapat x di P dengan indeks yang lebih kecil daripada j, maka seolah-olah melakukan pergeseran P ke kanan agar posisi x di T[i] sejajar dengan posisi kemunculan terakhir (lo) x di P. Pemeriksaan berikutnya, indeks j selalu dimulai pada indeks terakhir (m-1). Yang digeser adalah posisi indeks i dengan nilai $i_{new} = i + (m-1) - lo = i + m - (lo + 1)$. Sesuai contoh dibawah, perhitungan i_{new} adalah sebagai berikut.

$$i_{new} = i + (4-1) - 0$$
$$= i + 3$$

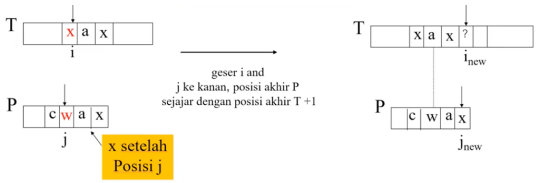


Gambar 2.4 Pergeseran indeks pada kasus 1

Sumber : https://www.youtube.com/watch?v=W-HwIH73Bsk

Kasus yang kedua yaitu ketika terjadi mismatch pada T[i] dan P[j], dan karakter pada T[i] adalah x. Jika terdapat x di P, tapi pada posisi dengan indeks yang lebih besar daripada j, maka seolah-olah melakukan pergeseran P satu karakter ke kanan, agar posisi indeks terakhir P sejajar dengan (posisi akhir T sebelumnya) + 1. Pemeriksaan berikutnya tetap dimulai dari indeks terakhir P yaitu j = m – 1. Yang digeser adalah nilai i menjadi $i_{new} = i + m - j$. Pada contoh dibawah, perhitungan i_{new} adalah sebagai berikut.

$$i_{new} = 1 + 4 - 1$$
$$= i + 3$$

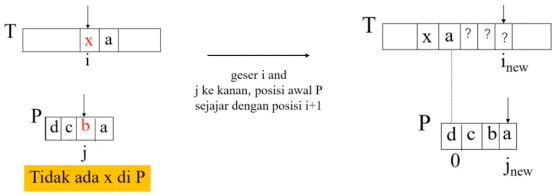


Gambar 2.5 Pergeseran indeks pada kasus 2

Sumber : https://www.youtube.com/watch?v=W-HwIH73Bsk

Kasus yang ketiga yaitu ketika terjadi mismatch pada T[i] dan P[j], dan karakter pada T[i] adalah x. Jika kasus 1 dan 2 tidak ditemukan saat terjadi mismatch (karakter x tidak ditemukan pada P), maka seolah-olah menggeser P agar posisi indeks pertama P (P[0]) sejajar dengan indeks i + 1. Pemeriksaan berikutnya tetap dimulai dari indeks terakhir P yaitu j = m – 1. Yang digeser adalah nilai i menjadi $i_{new} = i + m$. Pada contoh dibawah perhitungan i_{new} adalah sebagai berikut.

$$i_{new} = i + m$$
$$= i + 4$$



Gambar 2.6 Pergeseran indeks pada kasus 3

Sumber : https://www.youtube.com/watch?v=W-HwIH73Bsk

Untuk kasus 1 dan kasus 2, perlu informasi mengenai di mana kemunculan terakhir suatu karakter pada P untuk menentukan berapa banyak pergeseran yang perlu dilakukan untuk indeks i yang baru pada teks. Informasi kemunculan terakhir bisa diporses saat P sudah diketahui. Dilakukan preproses fungsi *Last Occurrence* dengan menentukan posisi kemunculan terakhir semua karakter pada teks T di dalam pattern P. Jika suatu karakter pada T tidak muncul di P (kasus 3), maka nilai fungsi *Last Occurrence* untuk karakter tersebut adalah -1.

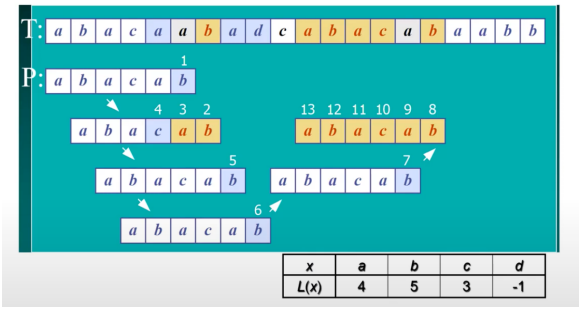
Dimisalkan terdapat T dengan karkater pada T adalah A = {a, b, c, d} dan dimisalkan P adalah "abacab". Maka fungsi *Last Occurrence* L(x) untuk masing-masing karkater pada T adalah sebagai berikut.

- L(a) = 4 → kemunculan terakhir karakter a pada P ada di indeks 4.
- L(b) = 5 → kemunculan terakhir karakter b pada P ada di indeks 5.
- L(c) = 3 → kemunculan terakhir karakter c pada P ada di indeks 3.
- L(d) = -1 → karakter d tidak muncul pada P

Semua nilai tersebut disimpan dalam table atau array. Parameter fungsi L(x) adalah semua karakter pada T.

Algoritma *Boyer-Moore* terdapat tiga tahapan yaitu input teks dan pattern. Kemudian dilakukan perhitungan nilai dari fungsi *Last Occurrence* untuk setiap karakter T. Selanjutnya dilakukan pencocokan string dengan menggunakan dua teknik yaitu *the looking-glass technique* dan *the character-jump technique*.

Berikut ini merupakan ilustrasi pencocokan string menggunakan algoritma Boyer-Moore.



Gambar 2.7 Contoh pencocokan string dengan algoritma Boyer-Moore

Sumber : https://www.youtube.com/watch?v=W-HwIH73Bsk

Diketahui bahwa variasi karakter yang terdapat pada teks hanya ada 4 yaitu a, b, c, dan d. Kemudian dilakukan proses penghitungan last occurrence untuk setiap karakter tersebut. Hasil perhitungan tersebut dapat dilihat pada table di kanan bawah gambar 2.7. Berikutnya dilakukan proses pencocokan string dengan teknik *looking glass technique* yaitu dimulai dari indeks terakhir pattern P sehingga j = 5 dan i = 5. Pada pemeriksaan pertama ditemukan mismatch. Kemudian ditentukan apakah mismatch tersebut merupakan kasus pertama, kedua, atau ketiga. Pada pemeriksaan pertama termasuk pada kasus pertama karena karakter pada a muncul disebelah kiri j dimana karakter a memiliki nilai *occurrence* 4. Pemeriksaan selanjutnya dimulai dengan j ada pada karakter terakhir jadi P dan i = 6. Pada pemeriksaan ke 2 dan ke 3 tidak terjadi mismatch sehingga pemeriksaan dilanjutkan. Pada pemeriksaan ke 4 terjadi mismatch dimana karakter pada teks adalah a dan karakter pada pattern adalah c, sehingga di cek kembali apakah kemunculan terakhir karkater a terdapat disebelah kanan atau sebelah kiri j dan diapatkan kemunculan terakhir karakter a terdapat disebelah kanan j sehingga merupakan kasus ke dua. Sehingga akan ditentukan indeks i yang baru untuk pemeriksaan selanjutnya sehingga i = 7. Sedangkan untuk indeks j selalu mulai dari karakter terakhir P. Pada pemeriksaan ke 5 terdapat mismatch dimana karakter pada T adalah a dan karkater pada P adalah b. Karakter a terakhir pada P muncul disebelah kiri j sehingga merupakan kasus pertama. Maka didapatkan i = 8 untuk pemeriksaan selanjutnya. Pada pemeriksaan ke 6 kembali terjadi mismatch dimana karakter T adalah d dan karakter P adalah b. Nilai occurrencenya adalah -1 karena tidak terdapat karakter d pada P, sehingga didapatkan nilai i yang baru yaitu i = 14. Pada pemeriksaan ke 7 terdapat mismatch kembali dimana karakter T adalah a dan karakter P adalah b. Karakter a terakhir pada P muncul disebelah kiri j sehingga merupakan kasus pertama sehingga nilai i yang baru adalah i = 15. Pada pemeriksaan ke 8-13 tidak terjadi mismatch sehingga ditemukan pattern yang sesuai pada teks.

D. Artikel

Menurut Kamus Besar Bahasa Indonesia (KBBI), artikel adalah sebuah karya tulis lengkap, misalnya seperti laporan berita atau esai dalam majalah, surat kabar, dan lain sebagainya. Artikel dibuat untuk dipublikasikan dengan tujuan menyampaikan gagasan dan fakta yang dapat mendidik atau menghibur.

Ada beberapa unsur-unsur artikel yang harus ada seperti kata, kalimat, gaya bahasa, dan isinya. Berikut merupakan unsur dan ciri-ciri artikel secara umum.

1. Isi artikel harus berdasarkan pada fakta yang benar-benar terjadi dan dapat dipertanggungjawabkan
2. Menggunakan metode penulisan yang tepat dan sistematis agar informasi mudah diterima oleh pembaca
3. Memiliki sifat factual dan informatif berdasarkan riset dan hasil penelitian yang telah dilakukan
4. Dapat mengandung opini dan analisa pemikiran, namun tetap harus dilandasi data dan teori yang valid
5. Menggunakan ragam kalimat yang lugas, logis, denotatif, dan efektif.

Dilihat dari isinya, terdapat 5 macam artikel. Lima macam artikel tersebut berupa narasi, deskripsi, persuasi, argumentasi, dan eksposisi sesuai dengan jenis-jenis paragraph berdasarkan isinya. Berikut ini adalah pengertian dari kelima jenis-jenis artikel tersebut.

1. Artikel Narasi

   Artikel narasi berisi runtutan dan cerita dari suatu kejadian atau peristiwa yang telah terjadi dengan jelas dan informatif, bisa juga berupa karya narasi fiksi.

2. Artikel Deskripsi

   Artikel deskripsi berisi gambaran mengenai objek atau keadaan sehingga pembaca seolah-olah melihat, mendengar, dan merasakannya.

3. Artikel Argumentasi

   Artikel argumentasi bertujuan membuktikan kebenaran suatu pendapat dengan data dan fakta ilmiah sebagai alasan atau buktinya.

4. Artikel Persuasi

   Artikel persuasi bertujuan untuk mempengaruhi pembaca untuk berbuat sesuai baik berupa ajakan atau himbauan terhadap sebuah teori.

5. Artikel Eksposisi

   Artikel eksposisi berisi uraian atau penjelasan tentang suatu topik dengan tujuan memberi informasi atau pengetahuan tambahan bagi pembaca.

III. IMPLEMENTASI

Pengimplementasian aplikasi pencarian kata atau pattern ini diimplementasikan dengan menggunakan bahasa pemrograman Python. Aplikasi ini menerima input satu file artikel dimana artikel tersebut sudah diubah formatnya menjadi file dengan ekstensi txt.

Aplikasi ini memanfaatkan algoritma Knut-Morris-Pratt dan juga algoritma Boyer-Moore untuk melakukan pencarian pattern yang sesuai pada artikel yang diberikan. Pattern merupakan masukkan pengguna sedangkan teks merupakan artikel. Jika

terdapat pattern yang sesuai program akan mengembalikan pada paragaf berapa sajakah pattern tersebut ditemukan. Jika tidak ditemukan pattern maka akan dikembalikan pesan bahwa pattern yang dicari tidak ditemukan. Pengguna aplikasi juga dapat memilih akan menggunakan algoritma apa untuk melakukan pencarian.

A. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt memiliki dua parameter yaitu text atau pada aplikasi ini merupakan artikel dan pattern dimana pada aplikasi ini merupakan pattern yang akan dicari yang merupakan masukkan dari pengguna. Pada awal program pattern dan text diubah menjadi string dengan huruf kecil seluruhnya agar pada saat pencarian tidak terdapat masalah jika dilakukan pencarian dengan huruf kapital atau huruf kecil. Kemudian diinisialisasi panjang jadi text dan juga pattern. Selanjutnya dibentuk array dengan nama border yang nantinya akan menyimpan nilai border function dari setiap karakter pada pattern. Kemudian dideklarasikan variabel j dan i dimana j merupakan indeks dari pattern P dan i merupakan indeks dari teks T. Selanjutnya dipanggil BorderFunction untuk menginisialisasi nilai border function untuk setiap karakter pada pattern pada array border. Dideklarasikan juga variabel found dengan nilai false untuk menentukan apakah pattern ditemukan pada teks atau tidak. Kemudian dilakukan iterasi dengan syarat i < panjang text dan found tidak sama dengan true. Jika pada iterasi P[j] = T[i] maka iterasi dilanjutkan ke indeks selanjutnya. Jika P[i] ≠ T[i] dan i < panjang teks serta j tidak bernilai nol maka nilai j akan diinisialisasi dengan nilai border function karakter yang mengalami mismatch, namun jika j sama dengan 0 maka akan dilakukan pemeriksaan ke indeks teks selanjutnya. Jika nilai j sudah sama dengan panjang pattern maka pattern ditemukan pada teks. Berikut ini merupakan potongan kode algoritma Knuth-Morris-Pratt.

```
def KMPSearch(text, pattern):
    pattern = pattern.lower()
    text = text.lower()
    longText = len(text)
    longPat = len(pattern)
    border = [0]*longPat
    j = 0
    BorderFunction(pattern, longPat, border)
    i = 0
    found = False
    while (i < longText and not found):
        if (pattern[j] == text[i]):
            i += 1
            j += 1

        if (j == longPat):
            j = border[j-1]
            found = True
        elif (i < longText and pattern[j] !=
text[i]):
            if (j != 0):
                j = border[j-1]
            else:
                i += 1
    if (found):
        return i-len(pattern)
    else:
        return -1
```

Berikut ini merupakan potongan kode dari BorderFunction yang berfungsi untuk membantu pergeseran indeks pada algoritma Knuth-Morris-Pratt.

```
def BorderFunction(pattern, longPat, border):
    len = 0
    border[0]
    i = 1
    while (i < longPat):
        if (pattern[i] == pattern[len]):
            len += 1
            border[i] = len
            i += 1
        else:
            if (len != 0):
                len = border[len-1]
            else:
                border[i] = 0
                i += 1
```

B. Algoritma Boyer-Moore

Algoritma Boyer-Moore memiliki dua mekanisme umum yaitu teknik *looking-glass* dan teknik *character-jump*. Pada teknik *character-jump* terdapat tiga kasus kemungkinan saat terjadi mismatch. Kasus yang pertama yaitu ketika karakter pada Teks T ketika terjadi mismatch dimisalkan x dimana kemunculan terakhir x pada pattern P berada di kiri j (*current index*). Jika terjadi kasus pertama pergeseran indeks T yaitu i menjadi i_{new} = i + (m-1) – lo = i + m – (lo + 1) dimana m merupakan panjang karakter P dan lo merupakan indeks kemunculan terakhir karakter x pada P. Kasua kedua yaitu ketika karakter pada Teks T ketika terjadi mismatch dimisalkan x, dimana kemunculan terakhir x pada pattern P berada di sebalah kanan j (*current index*). Jika terjadi kasus kedua maka pergeseran indeks T untuk pengecekan selanjutnya yaitu i menjadi i_{new} = i + m – j dimana m merupakan panjang karkater P dan j adalah *current index* dari P. Kasus ketiga adalah ketika karakter pada Teks T ketika terjadi mismatch tidak terdapat pada pattern P sehingga pergeserannya indeks T yaitu i menjadi i_{new} = i + m dimana m merupakan panjang karakter pattern P. Berikut ini merupakan potongan kode algoritma Boyer-Moore.

```
def BMSearch(text, pattern):
    pattern = pattern.lower()
    text = text.lower()
    longPat = len(pattern)
    longText = len(text)
    occ = OccFunction(pattern, longPat)
    i = 0
    found = False
```

```
    while (i <= longText-longPat and not found):
        j = longPat-1
        while (j >= 0 and pattern[j] ==
text[i+j]):
            j -= 1
        if (j < 0):
            i += (longPat-
occ[ord(text[i+longPat])] if i+longPat else 1)
            found = True
        else:
            i += max(1, j-occ[ord(text[i+j])])
    if (found):
        return i
    else:
        return -1
```

Untuk memudahkan lompatan indeks pada teks dibutuhkan fungssi bantuan yaitu *Occurrence Function* dimana *occurrence function* akan mengembalikan array yang berisi nilai kemunculan karakter pada teks T yang ada pada pattern P. Berikut ini merupakan potongan kode dari *Occurrence Function.*

```
def OccFunction(string, size):
    noChar = 256
    occ = [-1]*noChar
    for i in range(size):
        occ[ord(string[i])] = i
    return occ
```

C. *Program Utama*

Pada program utama, artikel akan dimunculkan terlebih dahulu dengan memanggil prosedur printArtikel. Kemudian artikel akan dibagi perparagraf dimana pembagian tersebut akan berupa array yang berisi teks tiap paragraf dari artikel tersebut. Selanjutnya pengguna aplikasi dapat memilih untuk menggunakan algoritma Knuth-Morris-Pratt atau algoritma Boyer-Moore. Pengguna selanjutnya dapat memasukkan pattern yang akan dicari. Kemudian akan dilakukan iterasi sepanjang array paragraf dan kemudian akan dilakukan pencarian menggunakan algoritma Knuth-Morris-Pratt atau algoritma Boyer-Moore. Jika fungsi KMPSearch atau BMSearch tidak mengembalikan -1 maka indeks+1 dari paragraf akan dimasukkan kedalam array where. Selanjutnya akan dikeluarkan pada paragraf apa aja pattern yang dicari. Jika panjang dari array where = 0 maka akan dikeluarkan pesan bahwa pattern yang dicari tidak ditemukan. Berikut ini merupakan potongan program utama.

```
printArtikel("artikel1.txt")
paragraf = inputArtikel("artikel1.txt")
print("Algoritma apa yang akan digunakan?")
print("  1. Knuth-Morris-Pratt Algorithm")
print("  2. Boyer-Moore Algorithm")
algo = input("Algoritma yang akan digunakan: ")
if (algo == "1"):
    pattern = input("Masukkan pattern yang
ingin dicari: ")
    where = []
    for i in range(len(paragraf)):
        if (KMPSearch(paragraf[i], pattern) !=
-1):
            where.append(i+1)
    if(len(where) != 0):
        print("Pattern '" + pattern +"'
terdapat pada paragraf ", end='')
        for i in range(len(where)):
            print(str(where[i]) + " ", end='')
        print('')
    else:
        print("Tidak ditemukan dalam artikel")
else:
    pattern = input("Masukkan pattern yang ingin
dicari: ")
    where = []
    for i in range(len(paragraf)):
        if (BMSearch(paragraf[i], pattern) != -1):
            where.append(i+1)
    if(len(where) != 0):
        print("Pattern '" + pattern +"' terdapat
pada paragraf ", end='')
        for i in range(len(where)):
            print(str(where[i]) + " ", end='')
        print('')
    else:
        print("Tidak ditemukan dalam artikel")
```

D. *Pengujian*

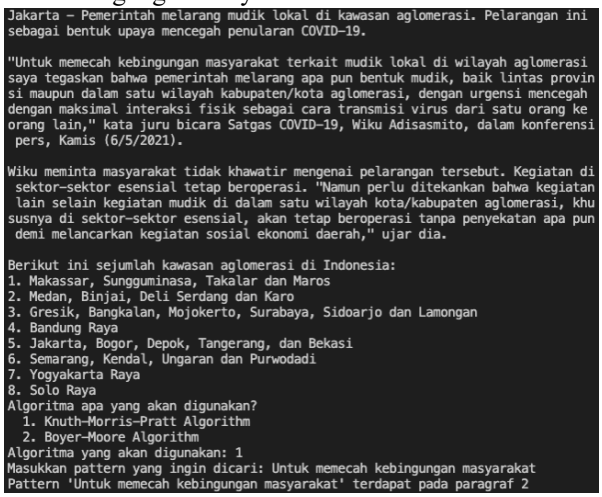
Pengujian akan dibagi menjadi dua yaitu pengujian dengan algoritma Knuth-Morris-Pratt dan pengujian dengan algoritma Boyer-Moore. Pada masing-masing pengujian akan terdapat 3 kasus yaitu pattern berupa kata, pattern berupa kalimat, dan pattern yang tidak terdapat pada artikel.

Berikut ini merupakan pengujian dengan algoritma Knuth-Morris-Pratt untuk kasus pattern berupa kata yaitu “Jakarta”.

Gambar 3.1 Test Case 1

Hasil pengujian menunjukan bahwa pattern “Jakarta” berada pada paragraf pertama dan keempat.

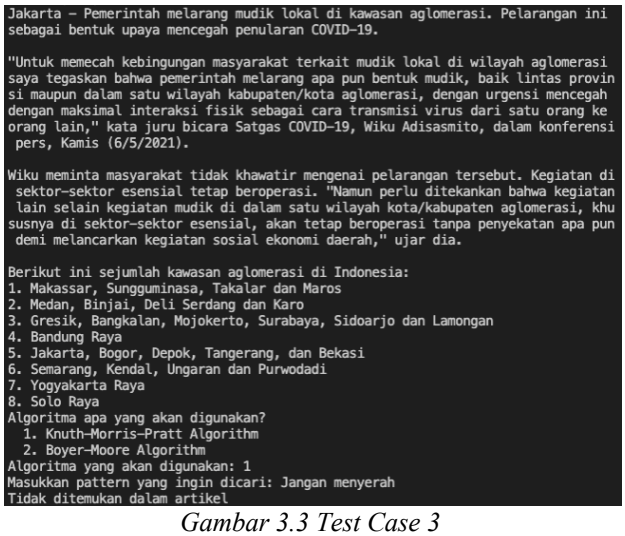
Berikut ini merupakan pengujian dengan algoritma Knuth-Morris-Pratt untuk kasus pattern berupa kalimat yaitu “Untuk memecah kebingungan masyarakat”.



Gambar 3.2 Test Case 2

Hasil pengujian menunjukan bahwa pattern “Untuk memecah kebingungan masyarakat” berada pada paragraf kedua.

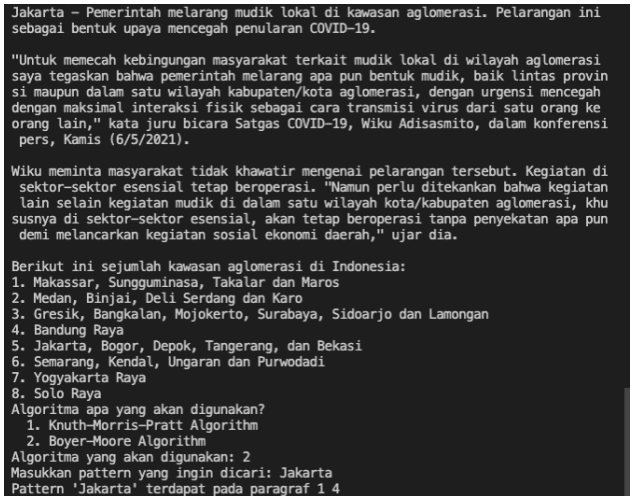
Berikut ini merupakan pengujian dengan algoritma Knuth-Morris-Pratt untuk kasus pattern yang tidak terdapat pada artikel.



Gambar 3.3 Test Case 3

Hasil pengujian menunjukan bahwa tidak ditemukan pattern yang dicari dengan mengembalikan pesan yaitu “Tidak ditemukan dalam artikel”.

Berikut ini merupakan pengujian dengan algoritma Boyer-Moore untuk kasus pattern berupa kata yaitu “Jakarta”.



Gambar 3.4 Test Case 4

Hasil pengujian test case 4 menunjukan bahwa terdapat pattern “Jakarta” pada paragraf pertama dan keempat di artikel.

Berikut ini merupakan pengujian dengan algoritma Boyer-Moore untuk kasus pattern berupa kalimat yaitu “Untuk memecah kebingungan masyarakat”.

Gambar 3.5 Test Case 5

Hasil pengujian test case 5 menunjukan bahwa terdapat kalimat “Untuk memecah kebingngan masyarakat” pada paragraf kedua.

Berikut ini merupakan pengujian menggunakan algoritma Boyer-Moore untuk kasus pattern yang tidak terdapat pada artikel.

Gambar 3.6 Test Case 6

Hasil pengujian test case 6 menunjukan bahwa tidak terdapat pattern yang sesuai pada artikel dengan mengembalikan pesan yaitu "Tidak ditemukan dalam artikel".

IV. Kesimpulan

Berdasarkan hasil pengujian yang dilakukan, dapat disimpulkan bahwa algoritma Knuth-Morrs-Pratt dan algoritma Boyer-Moore dapat digunakan untuk melakukan pencarian pattern masukan pengguna yang sesuai pada artikel dan menunjukan diparagraf manakah pattern tersebut ada. Pemanfaat algoritma Knuth-Morris-Pratt dan algoritma Boyer-Moore membantu aplikasi dalam melakukan pencocokan string untuk mencari pattern yang sesuai untuk setiap paragrafnya. Namun, kedua algoritma ini tentu bukanlah algoritma yang sempurna sehingga dapat digunakan untuk kebutuhan yang besar. Masih banyak hal yang perlu dikembangkan dari aplikasi ini agar dapat digunakan untuk kepentingan serta kebutuhan yang lebih besar.

Video Link at Youtube

Penjelasan mengenai makalah ini dapa dilihat pada link video youtube https://youtu.be/cN6uMhNmI4s.

Acknowledgment

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas berkat dan bantuannya selama penulis menyusun makalah dengan judul "Pemanfaatan Algoritma Knuth-Morris-Pratt dan Boyer-Moore Pada Pencarian Kata Dalam Artikel". Penulis juga mengucapkan terima kasih kepada Bapak dan Ibu Dosen pengampu mata kuliah IF2211 Strategi Algoritma Institut Teknologi Bandung atas bimbingan dan juga pengajarannya sehingga penulis mendapatkan banyak ilmu mengenai Strategi Algoritma di perkuliahan. Selain itu penulis juga berterima kasih kepada teman-teman Teknik Informatika Angkatan 2019 atas dukungannya sehingga penulis mampu menyelesaikan makalah ini dengan baik. Tidak lupa penulis berterima kasih kepada kedua orang tua penulis atas dukungan dan doa yang diberikan kepada penulis.

References

[1] https://www.geeksforgeeks.org/python-program-for-kmp-algorithm-for-pattern-searching-2/. Diakses 6 Mei 2021 pukul 23.55 WIB

[2] https://techterms.com/definition/string. Diakses 7 Mei 2021 pukul 15.15 WIB

[3] https://www-igm.univ-mlv.fr/~lecroq/string/node2.html#SECTION0020. Diakses 7 Mei 2021 pukul 15.30 WIB

[4] https://xlinux.nist.gov/dads/HTML/stringMatching.html. Diakses 7 Mei 2021 pukul 15.32 WIB

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses 10 Mei 2021 pukul 18.00 WIB

[6] https://www.youtube.com/watch?v=HWlSF8f-uWU&t=37s. Diakses 10 Mei 2021 pukul 21.00 WIB

[7] https://www.zonareferensi.com/pengertian-artikel/. Diakses 10 Mei 2021 pukul 21.24 WIB

[8] https://www.youtube.com/watch?v=W-HwIH73Bsk. Diakses 10 Mei 2021 pukul 23.48 WIB

[9] https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/. Diakses 11 Mei 2021 pukul 01.20 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Mei 2021

Giovani Anggasta
13519155