Penerapan Algoritma Pencocokan String Boyer Moore Dalam Melacak Perubahan Informasi Pada Text File

Lionnarta Savirandy 13518128
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518128@std.stei.itb.ac.id

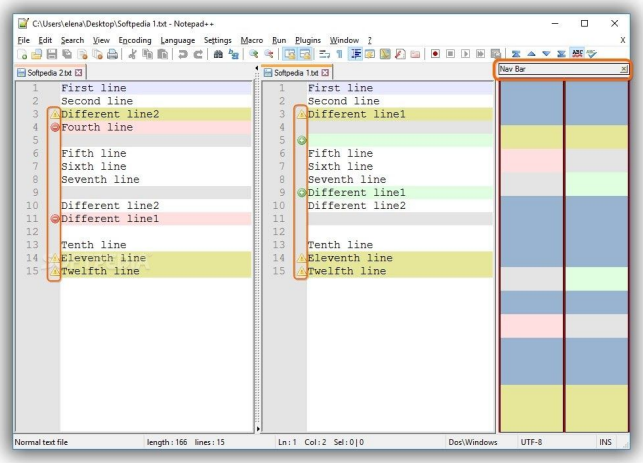
***Abstract*—Algoritma pencocokan string menjadi salah satu algoritma penting dalam mengolah informasi. Salah satu penggunaan algoritma ini adalah mencari perubahan kata yang terjadi dalam informasi dengan cara membandingkan informasi lama dengan yang baru. Boyer Moore merupakan salah satu algoritma pencocokan string yang efektif dalam mengolah string dengan karakter yang beragam. Makalah ini akan menggunakan algoritma Boyer Moore dalam proses membandingkan perubahan terhadap suatu informasi.**

Keywords—Pencocokan String; Boyer Moore; Informasi; Perubahan;

I. PENDAHULUAN

Dalam era teknologi, laju pertumbuhan informasi menjadi semakin cepat. Pertumbuhan yang dimaksud tidak sebatas pada bertambahnya jumlah informasi yang ada, namun juga mencakup perubahan-perubahan yang ada dalam informasi tersebut.

Pencocokan string merupakan algoritma untuk mencari suatu string pada teks yang sesuai dengan *pattern* yang diberikan. Algoritma ini mampu mengolah informasi menjadi informasi lainnya seperti kata penting dalam teks, jumlah kemunculan suatu kata, dan mencari perbedaan antara dua teks yang berbeda.



Gambar 1. Contoh hasil perbandingan perubahan pada *text*.
sumber:https://www.softpedia.com/blog/use-diff-tools-to-compare-text-files-and-spot-the-differences-504794.shtml#sgal_2

Melacak perubahan terhadap *text file* pada dasarnya adalah proses mencari perubahan yang dilakukan pada teks lama dan memberitahu pengguna perubahan apa saja yang dilakukan. Pada makalah ini, teks akan dibagi menjadi beberapa baris, dan perubahan yang diperiksa adalah perubahan yang terjadi pada setiap barisnya.

Pada proses pencocokan string ini, algoritma yang digunakan adalah algoritma Boyer Moore. Algoritma Boyer Moore memiliki keuntungan dalam proses pencocokan string untuk string yang beragam seperti teks berbahasa inggris.

II. TEORI SINGKAT

A. *String*

String merupakan himpunan/kumpulan dari karakter seperti alphabet, angka, space, ataupun karakter spesial lainnya yang merepresentasikan suatu teks. Contoh string: "Saya mengerjakan tugas", "Saya mengerjakan 3 tugas", dan "12.342".

B. *Pencocokan String (String Matching)*

Pencocokan String merupakan suatu algoritma untuk mencari semua kemunculan *pattern* yang diberikan pada suatu teks. *Pattern* yang diberikan harus memiliki panjang karakter lebih pendek atau sama dengan teks yang diperiksa. Contoh kasus pencocokan string:

Teks : "Strategi Algoritma"
Pattern: "trat"

S	t	r	a	t	e	g	i		A	l	g	o	r	i	t	m	a
	t	r	a	t													

Pada hasil dapat disimpulkan *pattern* "trat" ditemukan dalam teks "Strategi Algoritma".

C. Last Occurence

Last occurrence merupakan fungsi untuk mencari letak kemunculan terakhir setiap karakter pada *pattern* yang diberikan. Pada kasus ini, huruf besar dan kecil disamakan lalu letak kemunculan terakhir karakter disimpan dalam suatu penyimpanan seperti larik.

Pada program, larik yang dibuat merepresentasikan seluruh karakter dan karakter yang tidak muncul pada *pattern* pada umumnya diberikan nilai -1. Contoh:

Pattern: "Strategi"

S	T	R	A	T	E	G	I
0	1	2	3	4	5	6	7

A	E	G	I	R	S	T
3	5	6	7	2	0	4

D. Algoritma Boyer Moore

Boyer Moore merupakan salah satu algoritma untuk pencocokan string yang teknik pencocokannya dibagi dua, yaitu:

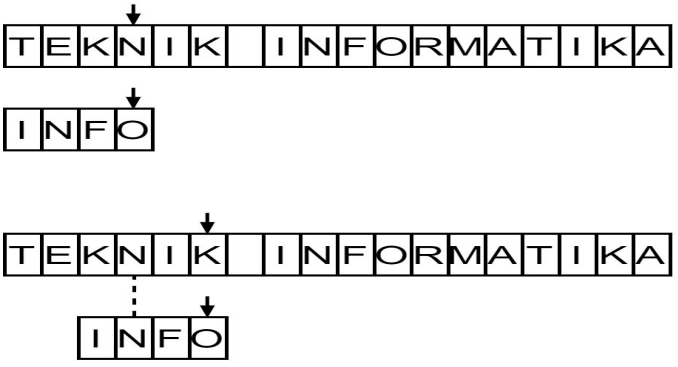
1) Looking-glass

Looking glass merupakan metode untuk melakukan proses pencocokan string dimulai dari akhir *pattern* (bergerak dari belakang ke depan). Jika pada teks T[i] tidak sesuai dengan *pattern* P[j] yang diperiksa, maka *pattern* dapat digeser ke belakang T[i] karena sudah pasti *pattern* tidak cocok dengan bagian teks yang sedang diperiksa. Proses pergeseran pada *pattern* menggunakan teknik character jump.

2) Character-jump

Character jump merupakan teknik untuk menghitung banyaknya pergeseran *pattern* yang dilakukan pada *text* yang sedang diperiksa. Pergeseran yang dilakukan dapat dibagi menjadi tiga kasus:

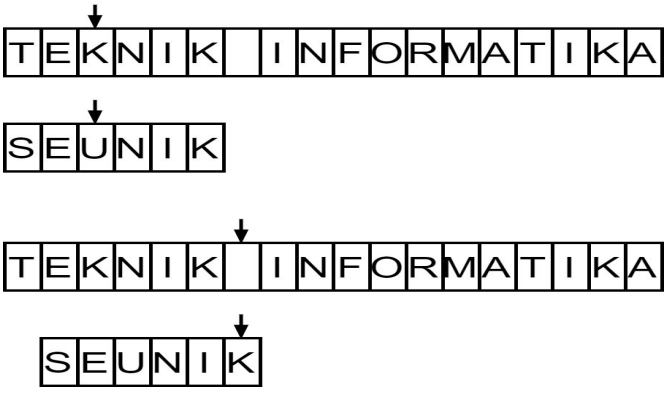
a) Mismatch terjadi pada teks T[i] berisi x dan *pattern* P memiliki karakter x dimana *last occurrence*-nya berada pada sebelah kiri *pattern* yang sedang diperiksa.



Gambar 2. Ilustrasi pergeseran *character jump* 1
sumber: dokumen penulis

Pada kasus ini, *pattern* P digeser ke kanan sehingga lokasi karakter mismatch pada teks sejajar dengan lokasi *last occurrence* karakter tersebut pada *Pattern*.

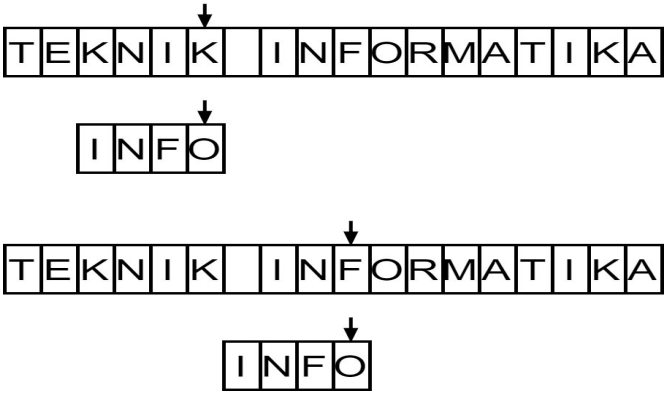
b) Mismatch terjadi pada teks T[i] berisi x dan *pattern* P memiliki karakter x dimana *last occurrence*-nya berada pada sebelah kanan *pattern* yang sedang diperiksa.



Gambar 3. Ilustrasi pergeseran *character jump* 2
sumber: dokumen penulis

Pada kasus ini, *pattern* P digeser ke kanan sebanyak satu karakter terhadap teks.

c) Mismatch terjadi pada teks T[i] berisi x dan *pattern* P tidak memiliki karakter x pada *pattern* tersebut.



Gambar 4. Ilustrasi pergeseran *character jump* 3
sumber: dokumen penulis

Pada kasus ini, *pattern* P digeser terhadap teks T sedemikian sehingga lokasi P[0] berada di sebelah kanan teks yang diperiksa.

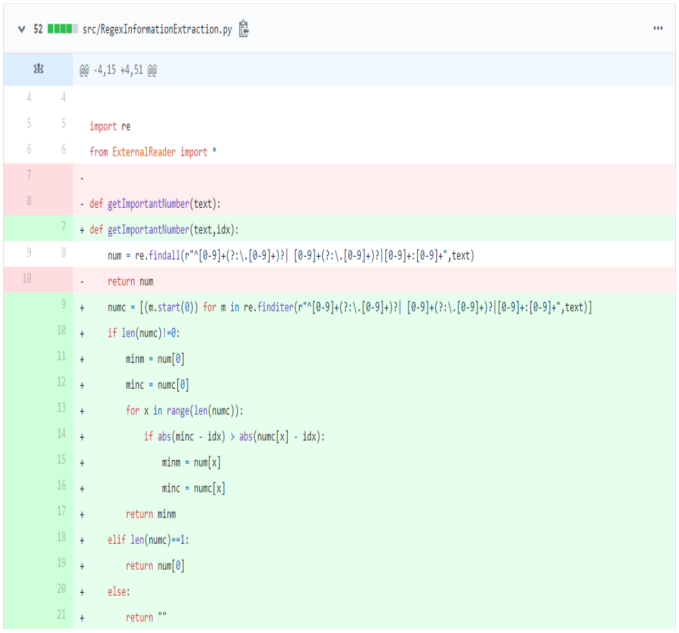
Algoritma Boyer Moore memiliki kompleksitas waktu O(nm+A) dengan A merupakan variasi karakter. Algoritma ini semakin baik jika variasi karakter semakin banyak, namun kurang baik untuk variasi karakter yang sedikit seperti *binary*.

E. Melacak Perubahan Informasi (Track Changes)

Perubahan informasi terhadap suatu *file* terjadi ketika pengguna mengubah isi dari *file* tersebut. Perubahan yang dilakukan dapat berupa penghapusan dan penambahan suatu

string, gambar, atau objek lainnya yang didukung oleh aplikasi.

Pada umumnya setiap aplikasi yang digunakan untuk mengubah *file* memiliki *track changes* dari *file* tersebut. Kegunaan dari *track changes* ini adalah menampilkan perubahan-perubahan pada *file* dalam suatu waktu. Dengan menggunakan *track changes*, pengguna dapat mengetahui perkembangan dari suatu *file* dan memutuskan untuk menggunakan perubahan tersebut atau tidak. Selain itu pada bidang informatika, pengguna seringkali melakukan kesalahan pada saat membuat program. Dengan menggunakan track changes, pengguna dapat kembali ke waktu program belum terdapat kesalahan.



Gambar 5. Track changes pada github
sumber: dokumen penulis

III. Pencocokan String Dalam Perbandingan Perubahan Informasi

Pada *track changes*, dilakukan proses pencocokan string data lama terhadap data yang baru. Dalam kasus ini disediakan dua *file* berbeda yaitu *file* baru dan *file* lama. Pertama, seluruh kata pada *file* baru akan didata untuk dijadikan sebagai daftar *pattern* yang akan digunakan.

```
new_kosakata = []
for i in newfile:
    for j in i.split():
        new_kosakata.append(j)
new_kosakata = list(set(new_kosakata))
```

Gambar 6. Proses pendaftaran *pattern* pada Python
sumber: dokumen penulis

Setelah diperoleh data *pattern*, proses pencocokan string dilakukan pada setiap *pattern* yang terdapat pada data tersebut. Proses ini dilakukan dengan menggunakan algoritma pencocokan string Boyer Moore. Hal ini dikarenakan algoritma ini efektif untuk kata yang bervariasi. Berikut merupakan implementasi algoritma Boyer Moore pada bahasa Python

```
def boyer_moore(text,pattern):
    lo = last_occurrence(pattern)
    m = len(pattern)
    n = len(text)
    i = m - 1
    res = []

    if (i > n-1):
        return res
    j = m - 1
    while (i <= n-1):
        if (text[i].lower() == pattern[j].lower()):
            if (j == 0):
                res.append(i)
                i = i + m
            else:
                i -= 1
                j -= 1
        else:
            clo = lo[ord(text[i].lower())]
            i = i + m - min(j,clo+1)
            j = m - 1
    return res
```

Gambar 7. Implementasi algoritma Boyer Moore
sumber: dokumen penulis

Seperti yang telah dijelaskan pada bab II, bahwa algoritma Boyer Moore menggunakan fungsi *last occurrence* untuk menentukan kasus yang sesuai pada teknik *character jump*. Berikut implementasi fungsi *last occurrence* pada bahasa Python.

```
def last_occurrence(pattern):
    lo = [-1 for i in range(128)]
    for i in range(len(pattern)):
        lo[ord(pattern[i].lower())] = i
    return lo
```

Gambar 8. Implementasi fungsi *last occurrence*
sumber: dokumen penulis

Pada kasus ini, algoritma Boyer Moore akan mencari *pattern* pada teks secara keseluruhan sehingga fungsi akan mengembalikan larik yang berisi sebuah pasangan nilai baris dengan larik posisi *pattern* ditemukan.

Pencarian dengan algoritma Boyer Moore dilakukan pada *file* lama dan *file* baru. Hal ini bertujuan untuk mendapatkan lokasi setiap *pattern* yang ada sehingga dapat dilakukan perbandingan antara kedua *file* yang ada. Setelah itu, proses dilanjutkan dengan mencari perbedaan diantara kedua larik yang diperoleh sehingga diperoleh larik baru yang berisi daftar *pattern* yang berbeda pada *file* baru. Berikut implementasi

untuk mencari perbedaan diantara kedua larik pada bahasa Python.

```
listbeda = []
for i in range(len(resnew)):
    if(resold[i] != resnew[i]):
        listbeda.append([item for item in resnew[i] if item not in resold[i]])
```

Gambar 9. Implementasi proses untuk mencari perbedaan di antara *file* baru dan *file* lama
sumber: dokumen penulis

Hasil yang diperoleh kemudian akan diambil nilai pertama dari pasangan nilai yang diperoleh, yaitu posisi baris perbedaan ditemukan. Setelah mendapatkan posisi baris, kemudian *file* baru akan ditampilkan dengan baris yang mengalami perubahan diberikan sebuah tanda seperti warna yang berbeda.

Untuk lebih memahami hasil kerja dari program, berikut akan diberikan contoh pengaplikasian pada *file*.

File lama:
(Baris 1)
Dari hasil Seleksi Nasional Masuk Perguruan Tinggi Negeri (SNMPTN) tahun 2020 yang diumumkan Rabu (8/4/2020),
(Baris 2)
Lembaga Tes Masuk Perguruan Tinggi (LTMPT) merilis data 20 besar prodi Sains dan Teknologi (Saintek) dengan keketatan tertinggi di SNMPTN 2020.
(Baris 3)
Di ranah Sains dan Teknologi (Saintek), Teknik Informatika dan Farmasi menjadi prodi populer pilihan siswa SNMPTN 2020,
(Baris 4)
berada di peringkat peringkat 20 besar prodi dengan keketatan paling tinggi.

sumber:https://edukasi.kompas.com/read/2020/04/14/140108471/prospek-kerja-teknik-informatika-jurusan-terketat-di-snmptn-2020

File baru:
(Baris 1)
Dari hasil Seleksi Nasional Masuk Perguruan Tinggi Negeri (SNMPTN) tahun 2020 yang diumumkan Rabu (8/4/2020),
(Baris 2)
Lembaga Tes Masuk Perguruan Tinggi (LTMPT) merilis data 20 besar prodi Sains dan Teknologi dengan keketatan paling tinggi di SNMPTN 2020.
(Baris 3)
Di ranah Sains dan Teknologi, Teknik Informatika dan Farmasi menjadi prodi populer pilihan siswa SNMPTN 2020,
(Baris 4)
berada di peringkat peringkat 20 besar prodi dengan keketatan paling tinggi.

Kata yang disoroti warna merah merupakan kata yang berbeda pada *file* lainnya.

Program akan mendata daftar *pattern* yang terdapat pada *file* baru. Berikut hasil data yang diperoleh.

['siswa', 'menjadi', 'peringkat', 'populer', 'yang', 'Negeri', 'Tes', 'diumumkan', 'Rabu', 'data', '(LTMPT)', '2020.', 'tinggi.', 'Teknik', 'prodi', '2020,', 'besar', 'Informatika', 'Lembaga', 'Sains', 'keketatan', 'Teknologi', 'ranah', 'hasil', 'SNMPTN', '2020', 'dengan', 'paling', 'dan', 'Farmasi', 'Teknologi,', 'Masuk', 'pilihan', 'Dari', 'Di', 'Tinggi', 'merilis', 'di', 'tinggi', 'Seleksi', 'tahun', '20', '(SNMPTN)', 'Nasional', '(8/4/2020),', 'Perguruan', 'berada']

Setelah itu algoritma Boyer Moore diterapkan pada *file* lama dan baru sehingga menghasilkan larik sebagai berikut.

File lama:
[[[2, [101]]], [[2, [71]]], [[3, [10, 20]]], [[2, [85]]], [[0, [78]]], [[0, [51]]], [[1, [8]]], [[0, [83]]], [[0, [93]]], [[1, [51]]], [[1, [35]]], [[1, [138]]], [[3, [69]]], [[2, [40]]], [[1, [65]], [2, [79]], [3, [39]]], [[2, [114]]], [[1, [59]], [3, [33]]], [[2, [47]]], [[1, [0]]], [[1, [71]], [2, [9]]], [[1, [108]], [3, [52]]], [[1, [81]], [2, [19]]], [[2, [3]]], [[0, [5]]], [[0, [59]], [1, [131]], [2, [107]]], [[0, [73, 103]], [1, [138]], [2, [114]]], [[1, [101]], [3, [45]]], [[3, [62]]], [[1, [77]], [2, [15, 59]]], [[2, [63]]], [], [[0, [28]], [1, [12]]], [[2, [93]]], [[0, [0]]], [[0, [83]], [1, [68, 128]], [2, [0, 76, 82]], [3, [7, 42]]], [[0, [44]], [1, [28, 121]], [3, [69]]], [[1, [43]]], [[0, [83]], [1, [68, 128]], [2, [0, 76, 82]], [3, [7, 42]]], [[0, [44]], [1, [28, 121]], [3, [69]]], [[0, [11]]], [[0, [67]]], [[0, [73, 75, 103, 105]], [1, [56, 138, 140]], [2, [114, 116]], [3, [30]]], [[0, [58]]], [[0, [19]]], [[0, [98]]], [[0, [34]], [1, [18]]], [[3, [0]]]]

File baru:
[[[2, [91]]], [[2, [61]]], [[3, [10, 20]]], [[2, [75]]], [[0, [78]]], [[0, [51]]], [[1, [8]]], [[0, [83]]], [[0, [93]]], [[1, [51]]], [[1, [35]]], [[1, [132]]], [[3, [69]]], [[2, [30]]], [[1, [65]], [2, [69]], [3, [39]]], [[2, [104]]], [[1, [59]], [3, [33]]], [[2, [37]]], [[1, [0]]], [[1, [71]], [2, [9]]], [[1, [98]], [3, [52]]], [[1, [81]], [2, [19]]], [[2, [3]]], [[0, [5]]], [[0, [59]], [1, [125]], [2, [97]]], [[0, [73, 103]], [1, [132]], [2, [104]]], [[1, [91]], [3, [45]]], [[1, [108]], [3, [62]]], [[1, [77]], [2, [15, 49]]], [[2, [53]]], [[2, [19]]], [[0, [28]], [1, [12]]], [[2, [83]]], [[0, [0]]], [[0, [83]], [1, [68, 122]], [2, [0, 66, 72]], [3, [7, 42]]], [[0, [44]], [1, [28, 115]], [3, [69]]], [[1, [43]]], [[0, [83]], [1, [68, 122]], [2, [0, 66, 72]], [3, [7, 42]]], [[0, [44]], [1, [28, 115]], [3, [69]]], [[0, [11]]], [[0, [67]]], [[0, [73, 75, 103, 105]], [1, [56, 132, 134]], [2, [104, 106]], [3, [30]]], [[0, [58]]], [[0, [19]]], [[0, [98]]], [[0, [34]], [1, [18]]], [[3, [0]]]]

Pada kedua hasil di atas, dilakukan proses untuk mencari perbedaan dari kedua larik. Program yang dibuat hanya mencari perbedaan setiap baris saja sehingga ketika terdapat perbedaan pada *file* baru, maka kata yang berbeda dan kata

setelahnya akan masuk ke dalam larik untuk diproses lebih lanjut dan menghasilkan.

```
[[[2, [91]]], [[2, [61]]], [[2, [75]]], [[1, [132]]], [[2, [30]]],
[[2, [69]]], [[2, [104]]], [[2, [37]]], [[1, [98]]], [[1, [125]],
[2, [97]]], [[1, [132]], [2, [104]]], [[1, [91]]], [[1, [108]]],
[[2, [15, 49]]], [[2, [53]]], [[2, [19]]], [[2, [83]]], [[1, [68,
122]], [2, [0, 66, 72]]], [[1, [28, 115]]], [[1, [68, 122]], [2,
[0, 66, 72]]], [[1, [28, 115]]], [[1, [56, 132, 134]], [2, [104,
106]]]]
```

Untuk mempermudah, hasil disederhanakan sebagai berikut.

Lokasi	File lama	File baru
Baris 1	(Saintek)	
Baris 1	tertinggi	paling tinggi
Barsis 2	(Saintek)	

Catatan: Pada program, komponen baris pada teks dimulai dari nol

Hasil akhir dari program adalah daftar baris sebagai berikut.

```
Hasil baris:
[1, 2]
```

Maka Program akan menampilkan kembali *file* baru beserta dengan hasil analisis program yang dilakukan sebagai berikut.

```
Perubahan pada teks baru :

Dari hasil Seleksi Nasional Masuk Perguruan Tinggi Negeri (SNMPTN
) tahun 2020 yang diumumkan Rabu (8/4/2020),
Lembaga Tes Masuk Perguruan Tinggi (LTMPT) merilis data 20 b
esar prodi Sains dan Teknologi dengan keketatan paling tinggi di
SNMPTN 2020.
Di ranah Sains dan Teknologi, Teknik Informatika dan Far Far
masi menjadi prodi populer pilihan siswa SNMPTN 2020,
berada di peringkat peringkat 20 besar prodi dengan keketatan
 paling tinggi.
```

Gambar 10. Hasil program
sumber: dokumen penulis

Pada hasil program tersebut, baris yang diwarnai merah merupakan baris yang berbeda dengan *file* sebelumnya.

Waktu yang diperlukan untuk melakukan proses pencocokan string dengan algoritma Boyer Moore.

```
Waktu: 16 ms
```

Gambar 11. Waktu program
sumber: dokumen penulis

Program juga mencoba menghitung waktu algoritma Boyer Moore untuk *file* dengan isi 300 kata. waktu yang dibutuhkan adalah sebagai berikut.

```
Waktu: 411 ms
```

Gambar 12. Waktu program untuk *file* besar
sumber: dokumen penulis

IV. Kesimpulan

Pada era teknologi di mana informasi terus berkembang dengan sangat cepat, manusia perlu untuk menyaring informasi yang didapatkan sehingga dapat dipastikan nilai kebenarannya. Untuk memastikan informasi yang diperoleh, tentunya perlu dilakukan penelusuran informasi-informasi terkait. Penelusuran ini dapat dengan mudah dilakukan dengan menggunakan metode pencocokan string.

Algoritma Boyer Moore merupakan salah satu algoritma yang digunakan dalam proses pencocokan string. Algoritma ini proses pencocokan nya sama seperti brute force yaitu dari kiri ke kanan, namun *pattern* yang diperiksa mulai dari akhir *pattern* ke awal *pattern*. Hal ini memungkinkan untuk melakukan pergeseran yang efektif ketika *pattern* dan teks tidak sesuai dengan menggunakan fungsi *last occurrence*.

Pada makalah ini *pattern* yang digunakan adalah seluruh daftar kata pada *file* baru masukkan. Kemudian proses pencocokan string dilakukan pada *file* lama dan baru untuk setiap *pattern* yang ada. Hasil keluarannya adalah daftar baris beserta dengan posisi karakter *pattern* ditemukan. Dengan membandingkan hasil dari *file* lama dan *file* baru, perbedaan dari kedua *file* dapat ditemukan sehingga keluaran akhir dari *file* adalah teks pada *file* baru dengan perubahan yang dilakukan.

Keuntungan dari menggunakan algoritma Boyer Moore adalah proses pencocokan yang cepat untuk teks dengan karakter yang beragam. Sebaliknya algoritma ini menjadi kurang efektif ketika teks masukkan memiliki karakter yang tidak bervariasi.

Link Video Pada Youtube

Penjelasan lebih lanjut tentang jalan kerja program dapat dilihat pada *link* berikut: https://youtu.be/6Rji3zcOdRw

Ucapan Terima Kasih

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas segala berkat dan pertolongannya sehingga penulis dapat menyelesaikan makalah ini. Selain itu penulis juga mengucapkan terima kasih kepada :

1) Ibu Dr. Nur Ulfa Maulidevi, ST., M.Sc., Bapak Dr. Ir. Rinaldi Munir, MT. dan Ibu Dr. Masayu Leylia Khodra, ST., MT., atas bimbingannya terkhusus Ibu Ulfa selaku dosen pengajar strategi algoritma di kelas K-02 yang telah memberikan ilmunya kepada kami sehingga makalah ini dapat diselesaikan.

2) Keluarga dan teman-teman yang selalu mendukung dan menolong penulis.

REFERENSI

[1] https://techterms.com/definition/string diakses pada 1 Mei 2020.

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf diakses pada 2 Mei 2020

[3] https://edukasi.kompas.com/read/2020/04/14/140108471/prospek-kerja-teknik-informatika-jurusan-terketat-di-snmptn-2020 diakses pada 3 Mei 2020

[4] http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/Text/Matching-Boyer-Moore1.html diakses pada 2 Mei 2020

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, 3 Mei 2020

Lionnarta Savirandy
13518128