Penerapan Algoritma String Matching dalam Web Peraturan ITB Finder

Muhammad Fauzan Al-Ghifari 13518112
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518112@std.stei.itb.ac.id

***Abstract*—Banyak mahasiswa yang belum memahami peraturan akademik yang berlaku di ITB, padahal aturan tersebut merupakan sesuatu yang penting untuk ditaati agar menciptakan suasana kampus yang kondusif dan mendukung lingkungan pembelajaran secara maksimal. Masih sering ditemukan mahasiswa yang kebingungan tentang kebijakan kampus padahal aturan tentang permasalahn tersebut sudah jelas tertuang pada aturan ITB. Hal ini terjadi karena mahasiswa malas untuk membaca aturan akademik yang cukup panjang dan berbelit-belit. Oleh karena itu penulis ingin membantu mahasiswa memahami peraturan akademik ITB dengan membuat "Web Peraturan ITB Finder" yang menerapkan algoritma *string matching*.**

***Keywords*—algoritma, *string matching*, Peraturan akademik.**

I. PENDAHULUAN

Peraturan adalah patokan yang dibuat untuk membatasi tingkah laku seseorang dalam suatu lingkup tertentu yang jika dilanggar pelakunya akan dikenakan hukuman. Peraturan sudah disepakati Bersama sebelumnya dan bersifat mengikat sekelompok orang dalam rangka mencapai tujuan hidup bersama. Sangat diharapkan semua elemen di dalam suatu organisasi atau komunitas memahami peraturan yang mengikat.

Sebagai salah satu institusi pendidikan di Indonesia Institut Teknologi Bandung (ITB) juga memiliki aturannya sendiri yang mengikat seluruh civitas akademika di dalamnya. Namun sayang sekali belum semua orang memahami aturan tersebut. Masih banyak mahasiswa yang sering bertanya tentang syarat-syarat kelulusan *cum laude*, batasan untuk mengulang mata kuliah atau jumlah sks minimal yang harus diambil untuk bisa lulus dari ITB.

Setiap civitas akademika di ITB memiliki kepentingannya masing-masing saat membuka dokumen peraturan ITB. Pencarian suatu topik dalam peraturan akademik ITB dapat dipermudah dengan menggunakan algoritma *string matching*. Peraturan akademik yang terbilang cukup panjang dapat dipotong menjadi bagian-bagian yang lebih pendek sesuai dengan topik yang dibutuhkan oleh setiap individu.

Secara garis besar *string matching* adalah algoritma yang digunakan untuk mencari suatu kata atau kalimat yang muncul pada teks. Dengan adanya Web Peraturan ITB Finder diharapkan dapat meningkatkan *awareness* warga ITB tentang pentingnya memahami aturan-aturan yang berlaku dan menciptakan suasana yang kondusif dan lingkungan yang mendukung untuk kegiatan belajar mengajar.

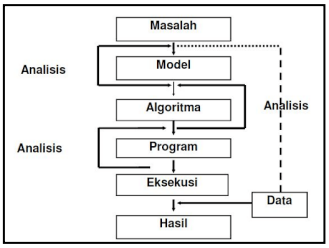
II. LANDASAN TEORI

A. Algoritma

Algoritma adalah deretan instruksi yang jelas dalam memecahkan masalah, yaitu untuk memperoleh keluaran yang diinginkan dari suatu masukan dalam jumlah waktu yang terbatas. Kata Algoritma berasal dari algorism, nama penulis buku arab yang terkenal, yaitu Abu Ja'far Muhammad Ibnu Musa Al Khawarizmi. Orang barat membaca Al Khawarizmi sebagai algorism.

Tidak semua urutan Langkah penyelesaian masalah yang logis dapat disebut sebagai algoritma. Menurut Donald E.Knuth di dalam bukunya yang berjudul *The Art of Computer Programming*, algoritma mempunyai lima ciri penting.

Algoritma harus bersifat *finiteness* (keterbatasan). Algoritma harus berakhir setelah mengerjakan sejumlah langkah berhingga. Barisan instruksi yang dibuat dalam suatu urutan tertentu, dimaksudkan agar masalah dapat diselesaikan. Banyaknya langkah harus terbatas.



Gambar 1. Flow chart algoritma
(Sumber: https://idcloudhost.com)

Setiap langkah pada algoritma harus didefinisikan dengan tepat dan tidak ambigu. Algoritma memiliki nol atau lebih masukan (input). Masukan adalah besaran yang diberikan kepada algoritma mulai bekerja. Algoritma mempunyai nol atau lebih keluaran (output). Keluaran adalah besaran yang memiliki hubungan dengan masukan. Keluaran tersebut tentunya harus merupakan solusi dari masalah yang sedang ingin diselesaikan.

Algoritma harus efektif dan efisien. Setiap langkah harus sederhana sehingga dapat dikerjakan dalam jumlah waktu yang masuk akal. Suatu Algoritma dikatakan efektif jika algoritma tersebut dapat menghasilkan suatu solusi yang sesuai dengan masalah yang diselesaikan, sedangkan algoritma dikatakan efisien jika waktu proses dari algoritma relative lebih singkat dan penggunaan memorinya lebih sedikit.

B. *String Matching Algorithm*

Pencarian string di dalam teks disebut juga pencocokan string (*string matching* atau *pattern matching*). Persoalan pencarian string adalah diberikan teks, yaitu long string yang panjangnya n karakter dan pattern, yaitu string dengan panjang m karakter (m < n) yang akan dicari di dalam teks.

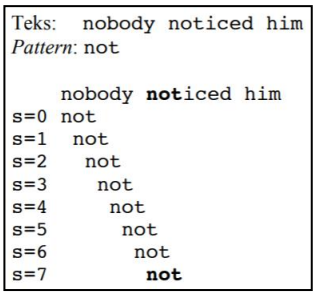
Carilah lokasi di dalam teks yang bersesuaian dengan pattern, aplikasi dari masalah pencocokan string antara lain pencarian suatu kata di dalam dokumen, misalnya fitur find yang disediakan di dalam aplikasi notepad, *web search engine*, misalnya Google, Analisis citra dan *Bioinformatics*, misalnya pencocokan rantai asam amino pada rantai DNA.

String Concepts misalkan panjang dari suatu String S adalah m. S = $x_0x_1…x_{m-1}$. Sebuah prefix dari S adalah substring S[0..k], sebuah suffix dari S adalah substring S[k..m-1] dengan k adalah indeks apa pun antara 0 dan m-1.

Contohnya adalah S = Fauzan. Maka semua prefix yang mungkin dari S adalah “F”, “Fa”, “Fau”, “Fauz”, “Fauza”, “Fauzan” dan semua suffix yang mungkin adalah “n”, “an”, “zan”, “uzan”, “auzan”, “Fauzan”

C. *The Brute Force Algorithm*

Dengan asumsi bahwa teks berada di dalam *array* T[1..n] dan *pattern* berada di dalam *array* P[1..m], maka algoritma *brute force* pencocokan string adalah sebagai berikut.



Gambar 2. String Matching Brute Force
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

1. Mula-mula *pattern* P dicocokkan pada awal teks T.
2. Dengan bergerak dari kiri ke kanan, bandingkan setiap karakter di dalam *pattern* P dengan karakter yang bersesuaian di dalam teks T sampai:
   a. Semua karakter yang dibandingkan cocok atau sama (pencarian berhasil)
   b. Dijumpai sebuah *mismatch* karakter (pencarian belum berhasil)
3. Bula *pattern* P belum ditemukan kecocokannya dan teks T belum habis, geser *pattern* P satu karakter ke kanan dan ulangi langkah 2.

Kompleksitas kasus terbaik adalah O(n). Kasus terbaik terjadi jika pattern ditemukan di awal teks. Kasus terburuk adalah saat pattern ditemukan kesalahannya di akhir pattern dan di teks pattern berada di belakang. Contoh kasus terburuk adalah seperti pattern: aab, teks:aaaaaaaaaaaaaaab. Tetapi kasus yang biasa terjadi adalah kasus rata-rata dengan kompleksitas sebesar O(m+n), yang termasuk sudah cukup cepat untuk menemukan pattern.

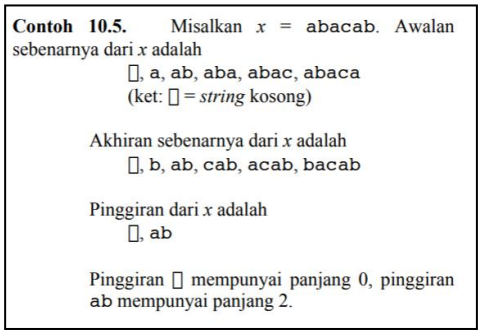
D. *Knuth-Morris-Pratt (KMP) Algorithm*

Pada algoritma *brute force*, setiap kali ditemukan ketidakcocokan pada pattern dengan teks, maka pattern digeser satu karakter ke kanan. Sedangkan pada algoritma KMP, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter seperti pada algoritma brute force.

Dengan algoritma KMP ini, waktu pencarian dapat dikurangi secara signifikan. Algoritma KMP dikembangkan oleh D. E. knuth, Bersama dengan J. H. Morris dan V.R Pratt.

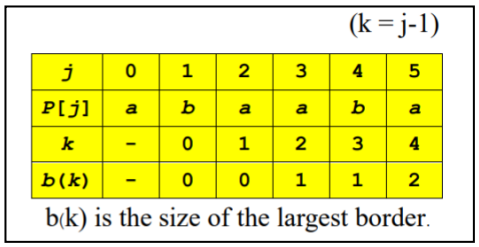
Misalkan Misalkan A adalah alfabet dan x = $x_1x_2…x_k$, k ∈ N, adalah string yang panjangnya k yang dibentuk dari karakter-karakter di dalam alfabet A. Awalan (prefix) dari x adalah upa-string (substring) u dengan u = $x_1x_2…x_{k-1}$, k ∈ {1,2,…k-1} dengan kata lain, x diawali dengan u. Akhiran (suffix) dari x adalah upa string (substring u dengan) u = $x_{k-b}x_{k-b+1}…x_k$, k ∈ {1,2,…k-1} dengan kata lain, x diakhiri dengan v.

Pinggiran (border) dari x adalah upa-string r sedemikian sehingga r = $x_1x_2…x_{k-1}$ dan u = $x_{k-b}x_{k-b+1}…x_k$, k ∈ {1,2,…,k-1}. Pinggiran x adalah upa string yang kedua awalannya dan juga akhirannya sebenarnya dari x.



Gambar 3. Mengitung Pinggiran
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

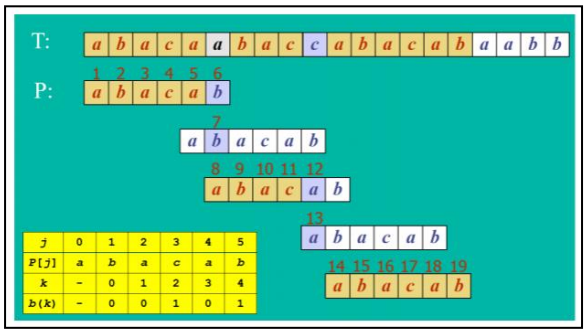
Untuk menerapkan algoritma KMP ini kita perlu mencari fungsi pinggiran dari pattern yaitu kumpulan pinggiran dari tiap panjang substring yang ada, untuk mengetahui jumlah pergeseran yang efisien. Kumpulan pinggiran tersebut dibentuk dalam suatu tabel yang memuat, yang jika terjadi mismatch pada saat pencocokan string kita merujuk ke tabel tersebut untuk menentukan awal pergeseran pattern di pemeriksaan selanjutnya.



Gambar 4. Border Function untuk Pattern abaaba
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Prosedur String Matching dengan KMP:

1. Cocokkan pattern dari awal teks dari kiri ke kanan
2. Karakter pattern di cocokkan karakter per karakter dengan teks, hingga mencapai salah satu kondisi
   a. Sampai akhir pattern semua karakter cocok dengan teks, algoritma akan memberi tahu bahwa ada string yang match pada teks
   b. Terjadi mismatch (karakter di pattern tidak cocok dengan teks)
3. Jika terjadi mismatch maka lakukan pergeseran sesuai dengan tabel dari border function, mulai pencocokan dari pattern dari indeks ke b(k) dimana k adalah indeks j-1 pattern mengalami mismatch



Gambar 5. Contoh penggunaan algoritma KMP
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Kompleksitas waktu saat menghitung fungsi pinggiran sebesar O(m) dan saat pencarian string O(n), sehingga kompleksitas waktu algoritma KMP adalah O(m+n). Sangat cepat jika dibandingkan dengan brute force.

Keuntungan dari menggunakan algoritma KMP adalah algoritma tidak perlu bergerak mundur dalam input teks, T. Hal ini membuat algoritma baik dalam pemrosesan file besar yang dibaca dari luar perangkat melalui network stream.

Kekurangannya adalah KMP tidak bekerja dengan baik saat ukuran dari alfabet meningkat karena lebih banyak kemungkinan mismatch dan mismatch cenderung terjadi di awal pola, tetapi KMP bekerja lebih cepat saat mismatch terjadi di tengah-tengah pola.

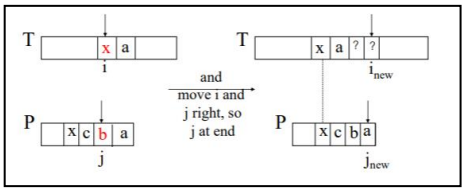
E. Boyer-Moore Algorithm

Algoritma Boyer-Moore adalah salah satu algoritma untuk mencari suatu string di dalam teks, dibuat oleh R.M Boyer dan J.S Moore. Algoritma Boyer-Moore melakukan perbandingan dimulai dari kanan ke kiri, tetapi pergeseran window tetap dari kiri ke kanan. Jika terjadi kecocokan maka dilakukan perbandingan karakter teks dan karakter pola yang sebelumnya, yaitu dengan sama-sama mengurangi indeks teks dan pola masing-masing sebanyak satu. Dengan menggunakan algoritma ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingkan dengan algoritma lainnya. Algoritma Boyer-Moore didasari dua buah teknik

1. The looking-glass technique, temukan P di T dengan cara bergerak dari belakang melalui P, dimulai dari akhirannya
2. The charactec-jump technique, karakter di pattern P[j] tidak sama dengan T[i], mismatch occurs at T[i] == x

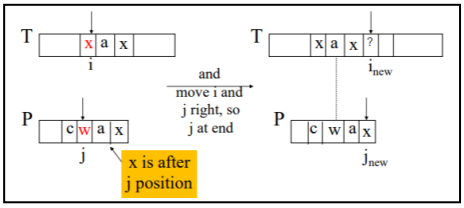
Ada 3 kemungkinan di character-jump technique

a. Case 1, Jika P mengandung x di suatu tempat, maka coba untuk menggeser P ke kanan, sejajarkan kejadian terakhir x dalam P dengan T[i]



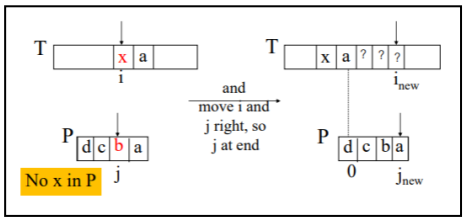
Gambar 6. Case 1 Character-jump Technique
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

b. Case 2, Jika P mengandung x di suatu tempat, tetapi penggeseran ke kejadian terakhir tidak dimungkinkan, maka geser P ke kanan dengan 1 karakter ke T[i+1]



Gambar 7. Case 2 Character-jump Technique
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

c. Case 3, Jika Case 1 dan Case 2 tidak berlaku, maka geser P untuk menyelaraskan P[0] dengan T[i+1]



Gambar 8. Case 3 Character-jump Technique
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Untuk memudahkan dalam penggunaan algoritma Boyer-Moore kita sebaiknya mencari *Last-Occurrence Function* terlebih dahulu. Boyer-Moore Algotithm memproses pattern P dan alphabet A untuk membuat last occurrence function L() yang merupakan maps dari semua letter di A menjadi sebuah integer. L(X) didefinisikan sebagai indeks I terbesar yang mengakibatkan P[i] – x, atau -1 jika character tidak ditemukan

Contoh dari pembentukan fungsi last occurrence dari A = {a,b,c,d} dan P = “abcab” adalah sebagai berikut

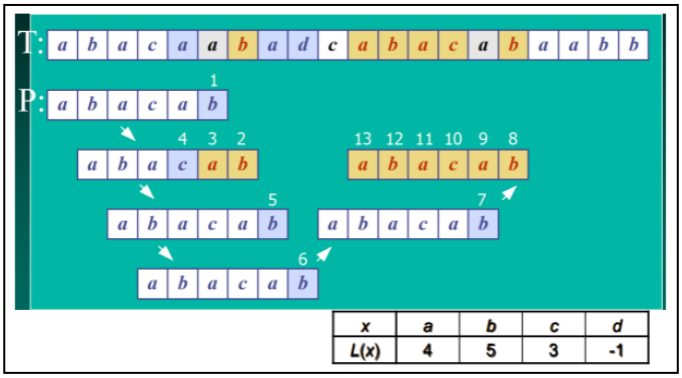
a. Berikan indeks untuk P

a	b	a	c	a	b
0	1	2	3	4	5

b. Cari indeks terbesar dari masing-masing A di P

x	a	b	c	d
L(x)	4	5	3	-1

L(d) bernilai -1 karena d tidak terdapat di pattern, A adalah kumpulan character yang ada di dalam Teks.



Gambar 9. Contoh penggunaan algoritma BM
(Sumber: Diktat Ir. Rinaldi Munir, M.T.)

Kompleksitas waktu menjalankan kasus terburuk oleh algoritma Boyer moore adalah O(nm+A). Algoritma cepat Ketika alphabet (A) besar, lambat saat alphabet kecil, misalnya baik untuk teks Bahasa inggris, buruk untuk biner. Boyer-Moore secara signifikan lebih cepat daripada algoritma brute force dalam pencocokan string.

III. Implementasi dan Pengujian

Pada penyelesaian masalah kali ini penulis akan mengimplementasikan algoritma Boyer-Moore dalam Web Aplikasi Peraturan ITB Finder menggunakan Bahasa python.

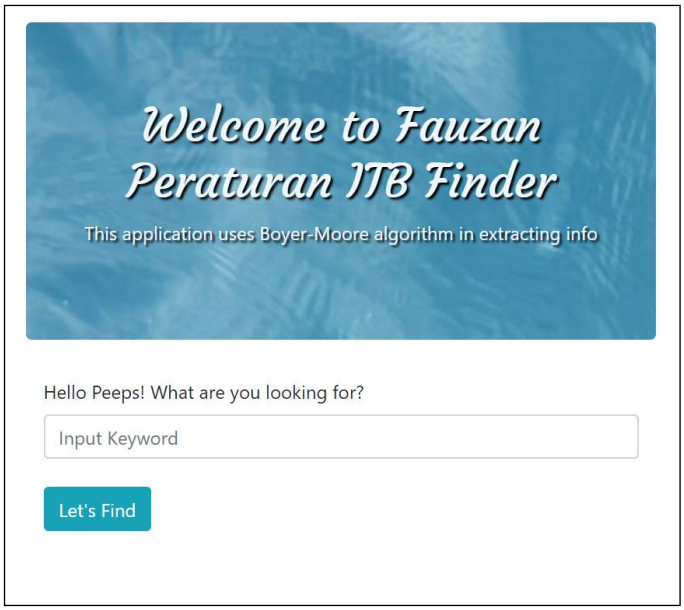
```python
# algoritma boyer moore
def BM(T, P):
    lo = lastOccurence(T, P)
    result = []
    n = len(T)
    m = len(P)
    j = m-1 # for pattern
    i = j # for teks

    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

            # mismatch
            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
```

Gambar 9. Potongan Algoritma Booyer-Moore
(Sumber:Penulis)

Penulis memutuskan untuk menggunakan algoritma Boyer moore pada pengaplikasian string matching kali ini, karena variasi character pada teks (yang pada kasus ini adalah peraturan akademik ITB) cukup banyak dan menggunakan Bahasa Indonesia. Kedua karakteristik ini sangat cocok dalam hal mengambil keuntungan dari Algoritma Booyer-more dan meningkatkan kecepatan pencarian string pada aplikasi.

Data uji peraturan akademik institute teknologi bandung penulis ambil dari ditdik.itb.ac.id sejumlah 12 Pasal tentang Kode Etik Mahasiswa dan 65 Pasal tentang Peraturan Akademik. Salah satu contoh isi dari data uji adalah sebagai berikut.

Penulis menggunakan flask untuk mengimplementasikan kode program ke dalam Web Application. Flask adalah kerangka kerja aplikasi web mikro yang ditulis dalam Bahasa pemrograman Python dan berdasarkan Wekzeug toolkit. Berikut adalah tampilan awal dari Web Peraturan ITB Finder. Web ini sudah dapat diakses online sampai tanggal 2 Agustus. Penulis menggunakan web hosting pythonanywhere yang menyediakan fitur untuk mengupload file dengan flask ke dalam sebuah website secara gratis selama 3 bulan. Web dapat diakses dengan mengcopy link bit.ly/FauzanFinder atau http://mfauzanalg.pythonanywhere.com/. Berikut adalah tampilan awal dari website penulis



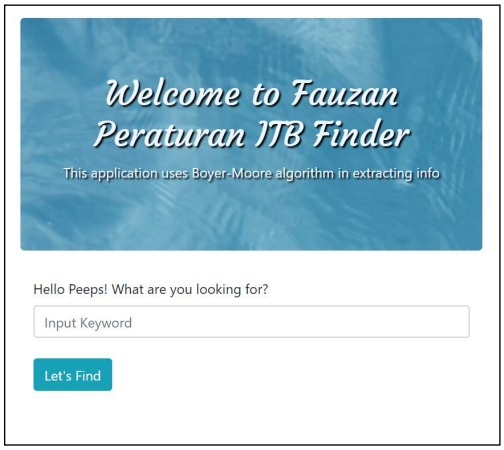
Gambar 10. Homepage Website
(Sumber:Penulis)

Cara penggunaannya cukup mudah, user hanya perlu memasukkan topik apa yang ingin di cari pada peraturan ITB, maka web akan menampilkan peraturan dengan topik yang berkaitan. Perlu diperhatikan bahwa database yang user masukkan hanya berupa peraturan Akademik dan peraturan Kode Etik saja, jadi yang akan ditampilkan hanya bersumber dari dua topik di atas.

Untuk melakukan pencarian dengan topik lain, user dapat me-refresh page atau memasukkan topik yang baru pada input keyword dan menekan lagi tombol Let's Find

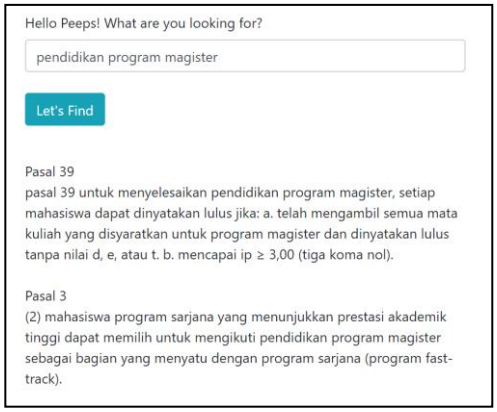
Berikut adalah cara Penggunaan Web Peraturan ITB Finder

1. Pada browser buka halaman bit.ly/FauzanFinder, maka akan terbuka tampilan WebApp seperti gambar yang ada di bawah



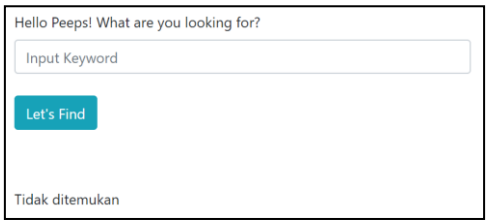
Gambar 11. Homepage Website
(Sumber: Penulis)

2. User Memasukkan input keyword dan menekan tombol Let's Find
   Jika peraturan yang berhubungan dengan input user ditemukan, maka web akan menampilkan daftar peraturan yang ada



Gambar 12. Hasil pencarian ditemukan
(Sumber: Penulis)

   Jika tidak, maka web akan menampilkan Tidak ditemukan



Gambar 13. Hasil pencarian tidak ditemukan
(Sumber:Penulis)

Cara Kerja Program di balik layer

1. Pada form input user memasukkan string yang akan ditangkap program dan di passing sebagai pattern
2. Setelah program mendapat input pattern, program akan melakukan loading database sebagai teks. Berikut adalah salah satu contoh dari teks database yang digunakan.

Kode Etik Mahasiswa Pasal 1
"Pasal 1
Dalam Peraturan Majelis Wali Amanah Institut Teknologi Bandung ini yang dimaksud dengan:

a. Dosen adalah pendidik profesional dan ilmuwan dengan tugas utama mentransformasikan, mengem bangkan, dan menyebarluaskan ilmu pengetahuan dan teknologi melalui pendidikan, penelitian, dan pengabdian kepada masyarakat.
b. Mahasiswa Institut Teknologi Bandung yang selanjutnya disebut sebagai mahasiswa adalah peserta didik pada jenjang Pendidikan di ITB.
c. Kode etik Mahasiswa ITB adalah pedoman yang berisi norma yang mengikat Mahasiswa secara individual dalam melaksanakan kegiatan akade mik dan kemahasiswaan di ITB.
d. Organisasi Kemahasiswaan adalah wadah pembi naan dan pengembangan bakat minat, kepri badian, jati diri, serta kegiatan-kegiatan lain yang dilaksanakan dalam kerangka referensi penca paian Visi dan Misi ITB."

3. Pertama-tama setiap database akan dipotong potong menjadi sebuah kalimat menggunakan library nltk
4. Setelah itu setiap kalimatnya diperiksa apakah kalimat itu mengandung pattern atau tidak menggunakan algoritma Boyer-Moore
5. Jika kalimat mengandung pattern maka kalimat tersebut akan disimpan dalam suatu array of tuple yang menyimpan kalimat dan nomor Pasal.
6. Pemeriksaan dilakukan pada seluruh database.
7. Saat pemeriksaan selesai, maka array of tuple akan ditampilkan pada website.

IV. SIMPULAN

Algoritma String Matching adalah algoritma yang digunakan untuk menemukan pattern string yang terdapat pada teks jika ada. Algoritma ini dapat diselesaikan dengan beberapa strategi misalnya menggunakan algoritma brute force, algoritma knutt morris pratt atau pun algoritma boyer moore. Ada banyak pemanfaatan Algoritma String Matching di dunia saat ini seperti untuk mencari teks pada dokumen, mendeteksi Bahasa atau pun mendeteksi plagiarism. Salah satu pemanfaatan String Matching yang penulis terapkan adalah pada Web Peraturan ITB Finder yang berguna untuk mencari suatu topik peraturan yang berlaku di ITB. Dengan

adanya aplikasi web ini penulis harapkan *awareness* civitas akademika ITB terhadap peraturan akan lebih meningkat dan menciptakan lingkungan yang kondusif untuk menunjang produktivitas ITB.

V. Kumpulan Link

1. Link Website (Sampai 2 Agustus 2020)
   http://mfauzanalg.pythonanywhere.com/
2. Link Source Code
   https://github.com/mfauzanalg/Peraturan-Akademik-ITB-Finder
3. Link Video
   https://youtu.be/D-QtgfYgP_s

VI. Ucapan Terima Kasih

Penulis mengucapkan puji dan syukur kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunianya makalah berjudul "Penerapan Algoritma String Matching dalam Web Peraturan ITB Finder" dapat penulis selesaikan. Penulis mengucapkan terima kasih sebesar-besarnya kepada ibu Dr. Masayu Leylia Khodra, S.T,. M. T. selaku dosen mata kuliah IF2211 Strategi Algoritma untuk kelas K-01, yang telah membimbing saya dalam memahami mata kuliah ini. Tidak lupa pula saya mengucapkan terima kasih kepada bapak Dr. Ir. Rinaldi Munir, M.T. dan ibu Dr. Nur Ulfa Maulidevi S.T., M.Sc. yang juga telah membimbing kelas K-01 selama perkuliahan online berlangsung. Penulis juga mengucapkan terima kasih keluarga dan teman-teman Teknik informatika ITB 2018, terutama teman-teman di kelas K-01 yang sudah membuat perkuliahan strategi algoritma menjadi lebih mudah dan menyenangkan. Semoga pandemic ini segera berakhir dan kita dapat berkumpul lagi di kelas 7602 tercinta.

Referensi

[1] Munir, Rinaldi, Pencocokan-String-2018 http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf Diakses pada 1 Mei 2020, pukul 21.00 GMT+7

[2] Tim penulis idcloudhost https://idcloudhost.com/mengenal-apa-itu-algoritma-definisi-ciri-ciri-dan-contohnya/ Diakses pada 1 Mei 2020, pukul 21.30 GMT+7

[3] Tim penulis researchgate https://www.researchgate.net/publication/301548098_Implementasi_Algoritma_BoyerMoore_pada_Aplikasi_Kamus_Kedokteran_Berbasis_Android Dialses pada 2 Mei 2020, pukul 07.00 GMT+7

[4] Munir, Rinaldi, Pencocokan-String-2020

[5] Peraturan Akademik ITB 2017

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.

Jakarta, 2 Mei 2020

Muhammad Fauzan Al-Ghifari
13518112