IF2211 Strategi Algoritma

Tugas Besar III

Pencarian Berita pada *News Aggregator* dengan Algoritma Pencocokan String

oleh:

Wenny Yustalim	13515002
Reinhard Benjamin L.	13515011
Rachel Sidney Devianti	13515124

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

2017

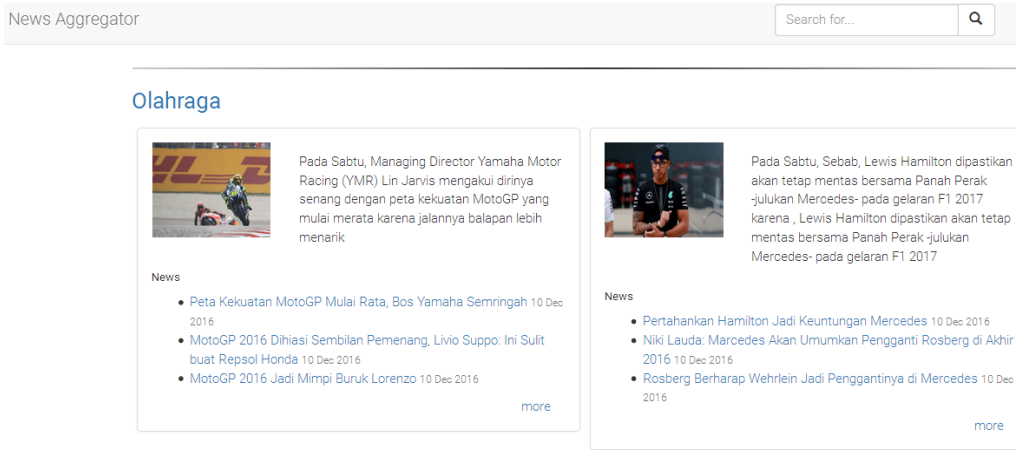
DAFTAR ISI

DAFTAR GAMBAR

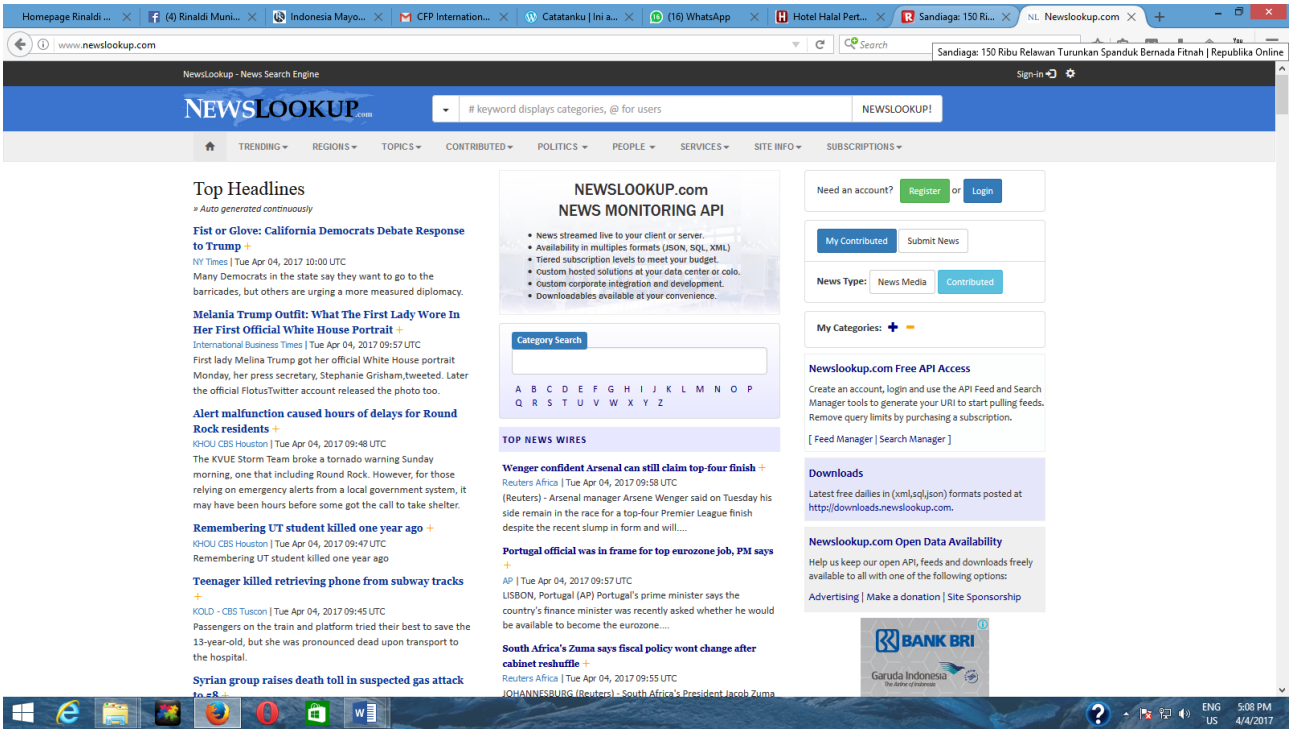
BAB I
DESKRIPSI MASALAH

Sistem agregasi berita (*news aggregator*) dikembangkan untuk membantu pembaca berita dengan mengumpulkan informasi berita dari berbagai sumber dan menyajikannya dalam satu tempat. Dengan sistem ini, pembaca tidak perlu mencari berita sendiri, dan aplikasi dapat mengambil berita sesuai kebutuhan dari pembaca (Lasica, 2003). Pencarian berita merupakan salah satu fitur yang terdapat pada sistem ini.

Gambar 1 News aggregator dari Google



Gambar 2 Prototipe news aggregator dari Lab GAIB ITB



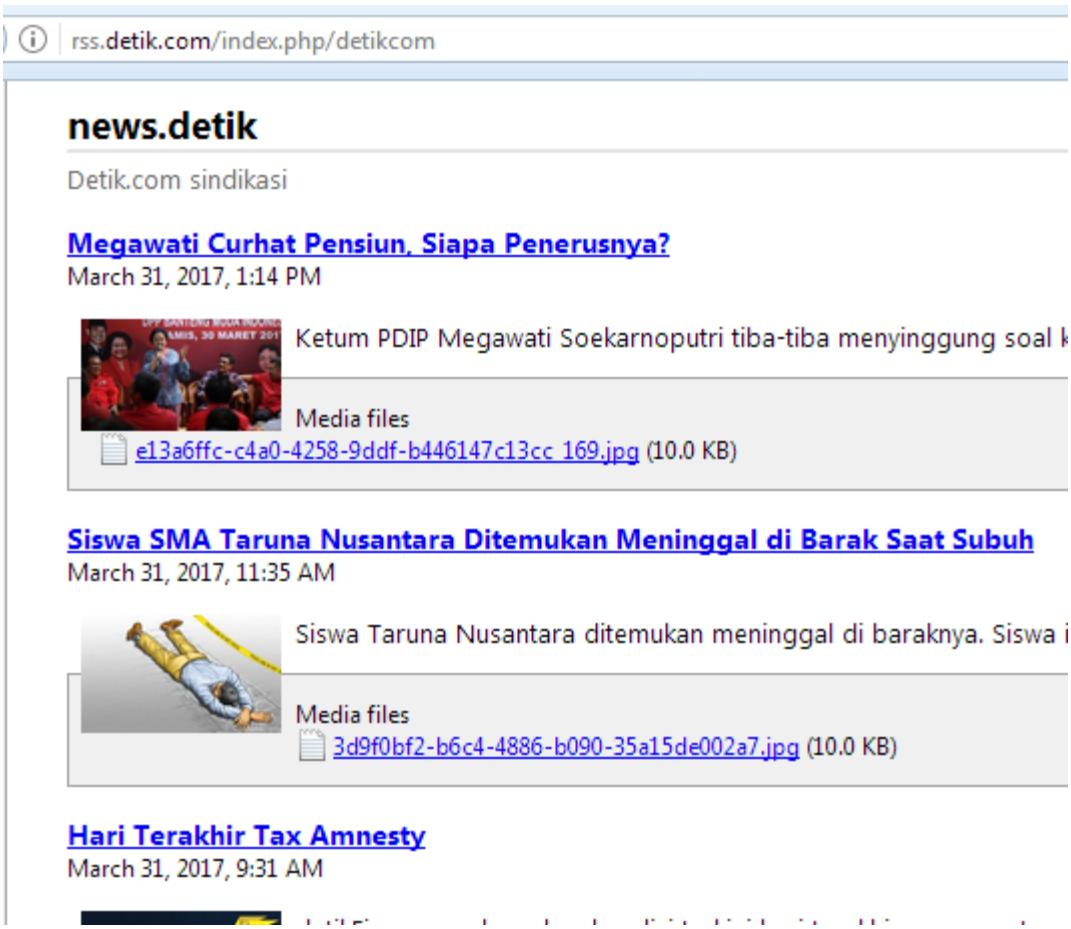
Gambar 3 News aggregator dari situ www.newslookup.com

Algoritma pencocokan *string* (*pattern*) Knuth-Morris-Pratt (KMP) dan Algoritma Boyer-Moore merupakan algoritma yang lebih baik daripada *brute force*. Pada Tugas Besar III kali ini Anda diminta membuat aplikasi sederhana pencarian berita pada *news aggregator* dengan kedua algoritma tersebut, plus menggunakan *regular expression* (*regex*). Teks yang akan Anda proses adalah kumpulan berita berbahasa Indonesia. Pengguna aplikasi ini akan memberikan masukan berupa keyword pencarian, dan menghasilkan daftar berita yang diurut berdasarkan tanggal berita.

Pencocokan *string* yang anda buat adalah *exact matching* (untuk KMP dan BM) jadi artikel berita yang diproses mengandung string yang tepat sama dengan *keyword* yang dimasukkan oleh pengguna. Sedangkan bila menggunakan *regex* maka tidak selalu *exact matching*. Pencarian juga tidak bersifat *case sensitive*, jadi huruf besar dan huruf kecil dianggap sama (hal ini dapat dilakukan dengan mengganggap seluruh karakter di dalam *pattern* dan teks sebagai huruf kecil semua atau huruf kapital semua).

Kumpulan berita diambil secara otomatis menggunakan *crawler* berbasis *RSS* (*rich site summary* atau *really simple syndication*) dari situs berita daring berbahasa Indonesia. Saat membaca *RSS* dengan *XML parser*, informasi yang dibutuhkan berupa judul, tanggal berita, dan *URL* berita. Berikut daftar *RSS* yang dapat digunakan:

http://rss.detik.com/index.php/detikcom
http://tempo.co/rss/terkini
http://rss.vivanews.com/get/all
http://www.antaranews.com/rss/terkini



Gambar 4 Isi RSS detik.com jika dilihat dari browser

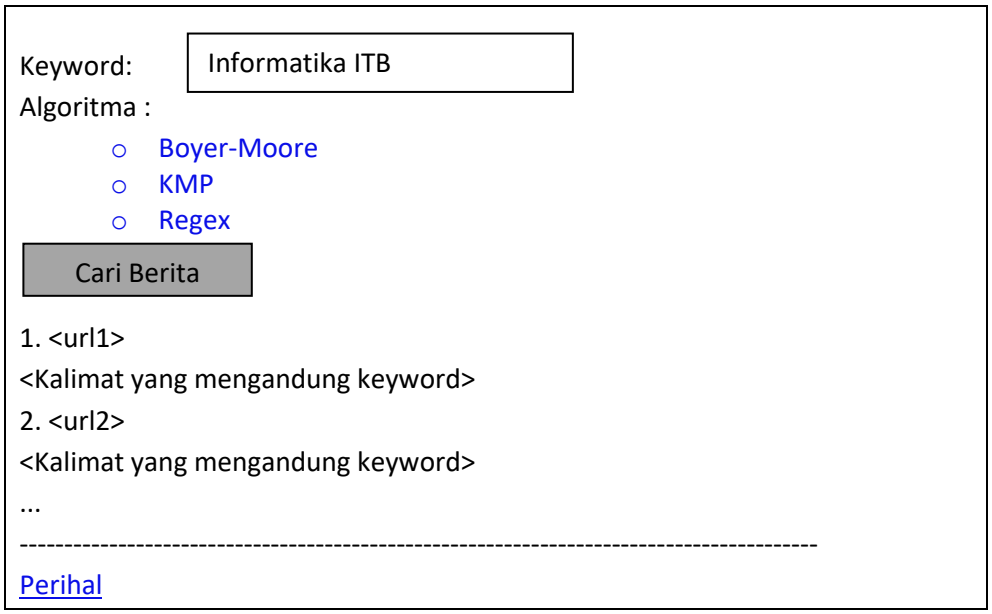
Untuk setiap *URL*, unduh artikelnya dan lakukan *parsing*. Artikel berupa file HTML dan tidak hanya mengandung konten berita, tetapi masih mengandung *header*, *footer*, iklan, dan tambahan informasi pada situs berita tersebut. Untuk itu, dilakukan *parsing* HTML untuk mendapatkan hanya teks konten berita dan foto yang terkait berita tersebut. Salah satu library *html parser* yang dapat digunakan misalnya adalah https://jsoup.org/ untuk Java, *RSS parser* untuk .NET

(http://stackoverflow.com/questions/684507/rss-parser-in-net) dan *XML parser* dalam Bahasa C# (http://stackoverflow.com/questions/642293/how-do-i-read-and-parse-an-xml-file-in-c),

Spesifikasi program:

1. Aplikasi pencarian berita pada *news aggregator* yang anda buat merupakan aplikasi berbasis web yang menerima *keyword* pencarian, misalnya "Informatika ITB". Tampilan antarmuka pengguna-komputer kira-kira seperti **Error! Reference source not found.** di bawah ini:

My Search App



Gambar 5 Isi RSS detik.com jika dilihat dari browser

Perihal : link ke halaman tentang program dan pembuatnya

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya

2. Pencarian berita menggunakan hasil implementasi algoritma KMP, Boyer-Moore, dan *Regex* dengan menggunakan Bahasa C#. Pencocokan string dilakukan tidak hanya pada judul berita, tetapi juga isi berita tersebut.
3. Kumpulan berita yang akan diproses diambil secara otomatis berbasis RSS.

Data Uji

Data uji yang digunakan dapat anda tentukan sendiri, minimal terdapat 20 berita yang diproses. Berita dapat berbahasa Indonesia atau Inggris.

Lain – lain:

1. Anda dapat menambahkan fitur fungsional lain yang menunjang program yang anda buat (unsur kreatifitas diperbolehkan/dianjurkan).
2. Program berbasis *web* dan dapat dikembangkan dengan salah satu kakas: PHP, JSP (*Java Server Pages*), atau ASP.
3. Program implementasi Boyer-Moore dan KMP menggunakan Bahasa C# dan dapat dikembangkan dengan kakas C# dalam *platform* .NET.
4. Akses konten berita dapat menggunakan library *XML* parser dan *html parser* yang ada.
5. Tugas dikerjakan per kelompok dengan jumlah anggota maksimal 3 orang dan tidak boleh sama dengan anggota kelompok sebelumnya.
6. Program harus modular dan mengandung komentar yang jelas.
7. Mahasiswa harus membuat program sendiri kecuali library *XML* dan *HTML parser*, tetapi belajar dari contoh-contoh program serupa yang sudah ada tidak dilarang (tidak boleh mengkopi *source code* dari program orang lain).
8. Keterlambatan pengumpulan akan mengurangi nilai.

9. Program disimpan di dalam *folder* StrAlgo3-xxxxx. Lima digit terakhir adalah NIM anggota terkecil. Didalam *folder* tersebut terdapat tiga folder bin, src dan doc yang masing-masing berisi :
    a. Folder *bin* berisi *executable file* (exe)
    b. Folder *src* berisi *source code* dari program
    c. Folder *test* berisi data uji.
    d. Folder *doc* berisi dokumentasi program dan *readme*

    Folder ini disimpan dalam bentuk CD untuk dikumpulkan bersama berkas laporan dimasukan kedalam amplop coklat.

10. Semua pertanyaan menyangkut tugas ini harus dikomunikasikan melalui milis agar dapat dicermati oleh semua peserta kuliah IF2211 (milis IF2211@students.if.itb.ac.id).
11. Demo program akan dilaksanakan pada tanggal yang dimumkan oleh asisten. Peserta mengisi jadwal demo yang disediakan pada saat pengumpulan tugas.
12. Tiap anggota harus memahami proses pembuatan program, karena akan ada pertanyaan-pertanyaan yang harus dijawab per individu.
13. Pada saat demo, asisten akan memanggil per kelompok sesuai jadwal yang telah diisi sebelumnya. Kelompok yang tidak berkepentingan dilarang masuk. Demo dilakukan di Lab IRK.
14. **Bonus 1** (nilai maksimal 10): jika judul berita mengandung kata tempat yang menunjukkan lokasi, maka gunakan *Google Map API* untuk menuju ke lokasi yang dimaksudkan.
15. **Bonus 2** (nilai maksimal 10): Setiap kelompok membuat video aplikasi yang mereka buat kemudian mengunggahnya ke *Youtube*. Pada waktu demo aplikasi di depan asisten, mahasiswa mengakses video *Youtube* tersebut dan memutarnya di depan asisten sebelum memulai demo.

BAB II
DASAR TEORI

Algoritma KMP *(Knuth-Morris-Pratt)*

1. Definisi
   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):[1]

   1. String pattern (kata yang dicari) akan dipecah menjadi array karakter
   2. String text (teks, artikel, dsb) akan dipecah menjadi array karakter
   3. Menentukan lompatan yang akan dilakukan ketika pencarian (funsi preKMP())
   4. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks. ()
   5. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   6. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
   7. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
   8. Algoritma kemudian menggeser pattern berdasarkan tabel next (lompat), lalu mengulangi langkah 2 sampai pattern berada di ujung teks.
2. Cara kerja
   Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan string:

   1. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.
   2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
      1. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
      2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
   3. Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

3. Pseudocode
   a. Fase pra-pencarian

```
procedure preKMP(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output kmpNext : array[0..n] of integer
)
Deklarasi:
i,j: integer

Algoritma
  i := 0;
```

```
  j := kmpNext[0] := -1;
  while (i < n) {
     while (j > -1 and not(P[i] = P[j]))
        j := kmpNext[j];
     i:= i+1;
     j:= j+1;
     if (P[i] = P[j])
        kmpNext[i] := kmpNext[j];
     else
        kmpNext[i] := j;
     endif
  endwhile
```

b. Fase pencarian

```
procedure KMPSearch(
    input m, n : integer,
    input P : array[0..n-1] of char,
    input T : array[0..m-1] of char,
    output ketemu : array[0..m-1] of boolean
)

Deklarasi:
i, j,next: integer
kmpNext : array[0..n] of interger

Algoritma:
preKMP(n, P, kmpNext)
i:=0
while (i<= m-n) do
    j:=0
    while (j < n and T[i+j] = P[j]) do
       j:=j+1
    endwhile
    if(j >= n) then
       ketemu[i]:=true;
    endif
    next:= j - kmpNext[j]
    i:= i+next
endwhile
```

4. Kompleksitas
   Algoritma ini menemukan semua kemunculan dari pattern dengan panjang n di dalam teks dengan panjang m dengan kompleksitas waktu O(m+n). Algoritma ini hanya membutuhkan O(n) ruang dari memory internal jika teks dibaca dari file eksternal. Semua besaran O tersebut tidak tergantung pada besarnya ruang alpabet. [2]

Algoritma BM *(Boyer-Moore)*

1. Definisi

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977.

Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritma ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat. [3]

2. Cara kerja
   Secara sistematis, langkah-langkah yang dilakukan algoritma Boyer-Moore pada saat mencocokkan string adalah:

   1. Algoritma Boyer-Moore mulai mencocokkan pattern pada awal teks.
   2. Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
      1. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
      2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
   3. Algoritma kemudian menggeser pattern dengan memaksimalkan nilai penggeseran good-suffix dan penggeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

3. Pseudocode
   a. Fase pra-pencarian

```
procedure preBmBc(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output bmBc : array[0..n-1] of integer
)
Deklarasi:
  i: integer

Algoritma:
  for (i := 0 to ASIZE-1)
     bmBc[i] := m;
  endfor
  for (i := 0 to m - 2)
     bmBc[P[i]] := m - i - 1;
  endfor
procedure preSuffixes(
    input P : array[0..n-1] of char,
    input n : integer,
```

```
    input/output suff : array[0..n-1] of integer
)

Deklarasi:
  f, g, i: integer

Algoritma:
  suff[n - 1] := n;
  g := n - 1;
  for (i := n - 2 downto 0) {
     if (i > g and (suff[i + n - 1 - f] < i - g))
        suff[i] := suff[i + n - 1 - f];
     else
        if (i < g)
           g := i;
        endif
        f := i;
        while (g >= 0 and P[g] = P[g + n - 1 - f])
           --g;
        endwhile
        suff[i] = f - g;
     endif
  endfor
procedure preBmGs(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output bmBc : array[0..n-1] of integer
)
Deklarasi:
  i, j: integer
  suff: array [0..RuangAlpabet] of integer

  preSuffixes(x, n, suff);

  for (i := 0 to m-1)
     bmGs[i] := n
  endfor
  j := 0
  for (i := n - 1 downto 0)
     if (suff[i] = i + 1)
        for (j:=j to n - 2 - i)
           if (bmGs[j] = n)
              bmGs[j] := n - 1 - i
           endif
        endfor
     endif
  endfor
  for (i = 0 to n - 2)
     bmGs[n - 1 - suff[i]] := n - 1 - i;
  endfor
```

b. Fase pencarian

```
procedure BoyerMooreSearch(
    input m, n : integer,
    input P : array[0..n-1] of char,
    input T : array[0..m-1] of char,
```

```
    output ketemu : array[0..m-1] of boolean
)

Deklarasi:
i, j, shift, bmBcShift, bmGsShift: integer
BmBc : array[0..255] of interger
BmGs : array[0..n-1] of interger

Algoritma:
preBmBc(n, P, BmBc)
preBmGs(n, P, BmGs)
i:=0
while (i<= m-n) do
    j:=n-1
    while (j >=0 n and T[i+j] = P[j]) do
       j:=j-1
    endwhile
    if(j < 0) then
       ketemu[i]:=true;
    endif
    bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
    bmGsShift:= BmGs[j]
    shift:= max(bmBcShift, bmGsShift)
    i:= i+shift
```

4. Kompleksitas
   Tabel untuk penggeseran bad-character dan good-suffix dapat dihitung dengan kompleksitas waktu dan ruang sebesar O(n + σ) dengan σ adalah besar ruang alfabet. Sedangkan pada fase pencarian, algor itma ini membutuhkan waktu sebesar O(mn), pada kasus terburuk, algoritma ini akan melakukan 3n pencocokkan karakter, namun pada performa terbaiknya algoritma ini hanya akan melakukan O(m/n) pencocokkan. [3]

Regex (Regular Expression)

Regular expression (regex) merupakan sekumpulan notasi dan karakter yang digunakan untuk mendeskripsikan suatu pola pada pencarian berbasis huruf. Dengan regex dimungkinkan untuk mengenali suatu string yang mempunya karakteristik dan pola tertentu, seperti email, tanggal, nomor kartu kredit, dll. Misal dengan notasi regex p.+ing maka akan menyaring semua kata-kata kecuali kata yang diawali huruf "p" dan mempunyai akhiran ing, seperti playing, praying, pulling, dll.
Notasi regex dapat dikombinasikan sedemikian kompleksnya sehingga dapat menemukan kata yang mempunya pola-pola cukup rumit.Adapun contohnya:

```
^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$
```

akan menampilkan error atau warning ketika string yang dimasukkan tidak memiliki format seperti alamat email, misal john@yahoo.com[5]

`e([:digit:])? –` mencari 'e' diikuti dengan nol atau 1 digit. Semua kelas karakter yang telah dinamai seperti `[:digit:]` harus dibungkus oleh *parentheses*. `^([:digit:])$ --` menemukan baris atau sel dengan digit tepat 1. Istilah pencarian dapat dikombinasikan untuk membentuk pencarian yang kompleks. Untuk menemukan nomor 3 digit sendiri pada suatu paragraf, `^[:digit:]{3}$` `^` berarti kecocokan harus ditemukan di awal paragraf, `[:digit:]` cocok dengan digit desimal apapun, `{3}` berarti harus ada tepat 3 salinan dari “digit”, `$` berarti kecocokan harus ditemukan di akhir paragraf.

BAB III
ANALISIS PEMECAHAN MASALAH

I. Langkah-langkah pemecahan masalah

Pemecahan masalah persiapan tools

Tools yang kami gunakan adalah Visual Studio 2017 agar kami dapat dengan mudah menulis program berbasis web dalam bahasa C# dan html. Tools lain yang kami gunakan adalah Microsoft Word 2013 untuk memudahkan kami membuat dokumentasi berupa laporan secara mudah.

Masalah yang timbul dalam persiapan tools adalah kami sempat bingung dalam menentukan package yang harus diinstall pada Visual Studio kami. Terdapat sedikit masalah di tengah proses pengimplementasian kode kami karena ketika diuji, program tidak dapat berjalan dengan baik. Hal ini dikarenakan terdapat package yang tidak dicentang saat proses instalasi.

Pemecahan masalah algoritma

Algoritma yang kami gunakan telah kami pelajari pada kuliah IF2211 Strategi Algoritma, yaitu algoritma KMP (Knuth-Morris-Pratt), algoritma BM (Boyer Moore), dan Regex (regular expression). Ketiga algoritma ini kami tulis dalam bahasa C# dengan melihat referensi pada slide yang telah diajarkan di kuliah Strategi Algoritma. Langkah pemecahan string matching telah terdapat pada bagian dasar teori.

Pemecahan masalah frontend

User interface kami terdiri dari 2 buah halaman. Halaman pertama yaitu homepage yang memunculkan logo, nama *news aggregator* yang kami buat, tombol “search” untuk mencari berita, tanggal dan waktu saat ini (real time). Ketika user mengetik tombol “search”, akna muncul tampilan berupa logo beserta field untuk user agar dapat mengetik *keyword*/kata kunci yang ingin dicari, serta sebuah drop-down menu yang dapat diklik untuk menampilkan 3 buah pilihan, yaitu KMP, Boyer-Moore, dan Regex. Terdapat juga tombol update agar user dapat mengupdate berita sesuai dengan waktu saat itu.

Pemecahan masalah exception

Saat kami sedang mencoba untuk mengetes hasil kode yang telah kami buat, banyak waktu yang terbuang karena ketika kami inging mengupdate berita dari webpage yang telah kami buat, terdapat exception yang muncul pada beberapa news article sehingga kami harus mencari cara untuk meng*bypass* exception yang muncul.

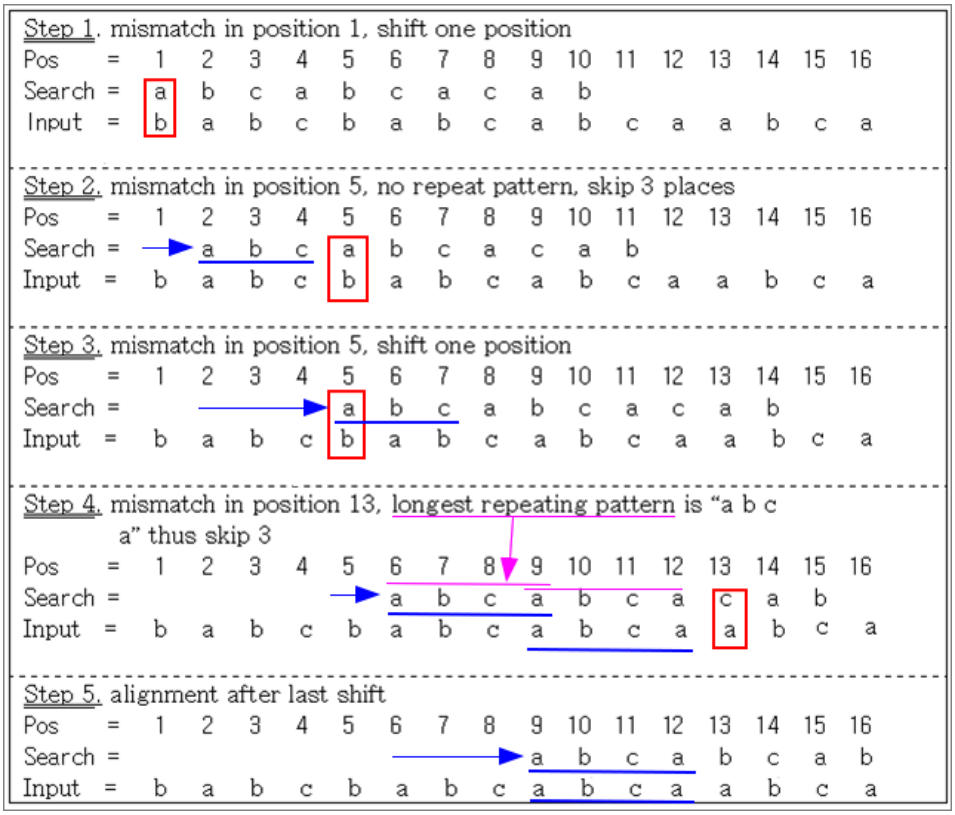
Pemecahan masalah parsing

Parsing RSS untuk mendapatkan judul berita dan url berita cukup mudah. Tantangan yang kami dapat adalah saat parsing HTML. Hal ini dikarenakan setiap news source memiliki gaya penuisan html yang berbeda-beda, sehingga harus di *hardcode.* Kami mencari pattern yang ada dari setiap news source, yaitu dengan cara mebuuka berita-berita yang ada dan secara manual melihat dimana konten berita ditulis pada kode html tersebut.

Sesudah kami menemukan patternnya, kami harus mencari cara untuk menghapus *tags* yang ada pada kode html tersebut agar kami dapat memperoleh konten berita yang bersih tanpa tags.

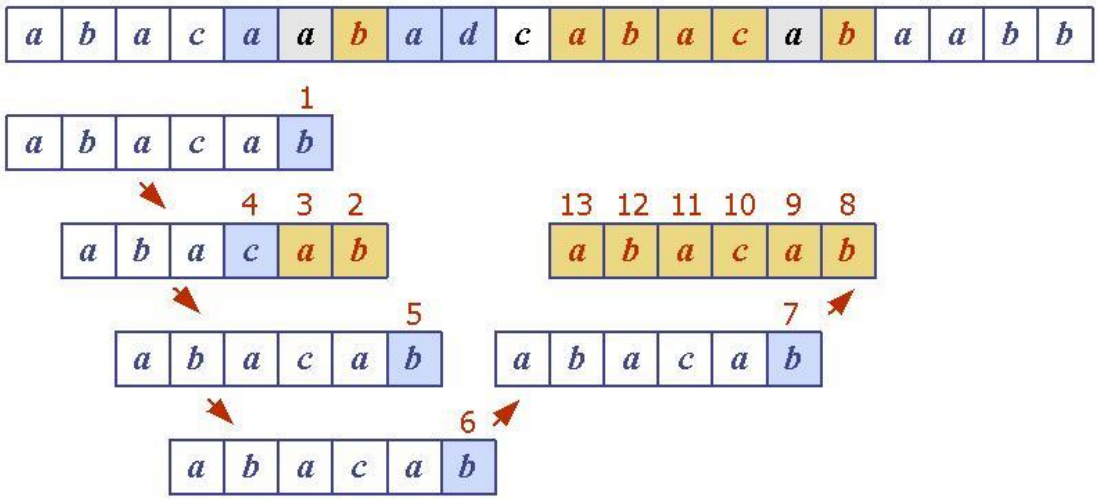
II. Con toh ilustrasi algoritma

Ilustrasi Algoritma KMP



Gambar 5 Ilustrasi algoritma KMP

Sumber: http://cfile22.uf.tistory.com/image/150A2E034BADEEA2C4B600



Gambar 6 Ilustrasi algoritma BM

Source: https://koding4fun.files.wordpress.com/2010/05/complete_example.jpg

BAB IV
IMPLEMENTASI DAN PENGUJIAN

I. Spesifikasi teknis program

1. Struktur data

JSON

JSON merupakan struktur data yang digunakan untuk menyimpan informasi dari berita-berita yang telah diambil menggunakan RSS Parser dan HTML Parser. Informasi ini meliputi: judul, tanggal, *link*, isi, dan lain sebagainya. Data-data dari berita ini disimpan dalam suatu format tertentu sehingga memudahkan ketika proses pengambilan data berita untuk melakukan *string matching* dan menampilkan hasil pencarian ke layar.

News

Kami membuat struktur data news, dengan isi:

- Title
  Title merupakan variabel bertipe string yang digunakan untuk menyimpan judul dari sebuah berita.
- Summary
  Summary merupakan variabel bertipe string yang digunakan untuk menampung garis besar isi dari sebuah berita.
- Imagelink
  Imagelink merupakan variabel bertipe string yang digunakan untuk menampung alamat *url* gambar berita.
- Link
  Link merupakan variabel bertipe string yang digunakan untuk menampung alamat *url* berita.
- Description
  Description merupakan variabel bertipe string yang digunakan untuk menampung isi dari sebuah berita.

Berikut ini adalah implementasi struktur data yang telah kami buat pada News.cs.

```
class News
{
    public string title;
    public string summary;
    public string imagelink;
    public string link;
    public string description;
}
```

2. Kelas Objek beserta Fungsi dan Prosedur

No.	Nama Kelas	Fungsi/Prosedur	Keterangan

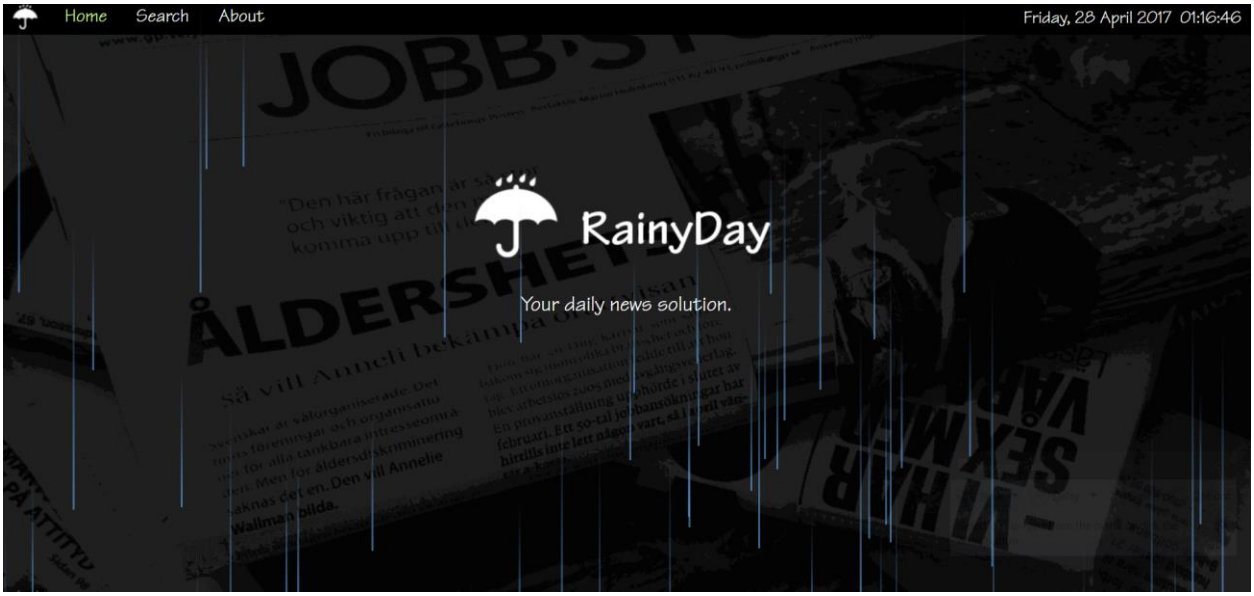
No.	Nama Kelas	Fungsi/Prosedur	Keterangan
1.	Program	`public static void Main(string[] args)`	Main program untuk melakukan pengaturan lingkungan.
2.	Parser	`public void ReadWebsitesFromFile(string filename)`	Prosedur untuk membaca *list website* dari file eksternal dan memasukkan ke dalam *array of string*.
3.	Parser	`public void ParseRSS()`	Prosedur yang melakukan *parsing* terhadap RSS untuk mendapatkan judul, tanggal, dan *link* dari sebuah berita.
4.	Parser	`public void ParseHTML()`	Prosedur untuk melakukan *parsing* terhadap HTML untuk mengambil berita.
5.	Parser	`public void WriteJSONToFile(string filename)`	Prosedur untuk menuliskan *json string* ke *file* News.json.
6.	`HomeController`	`public IactionResult Index()`	Fungsi untuk melakukan *render* terhadap Index.cshtml sehingga dapat dijalankan ke layar.
7.	`HomeController`	`public IactionResult Search()`	Fungsi untuk melakukan *render* terhadap Search.cshtml sehingga dapat dijalankan ke layar.
8.	`HomeController`	`public void StartUpdate()`	Prosedur untuk melakukan *parsing* terhadap RSS dan HTML, menuliskan hasil *parsing* ke News.json.
9.	`News`	`public string title`	Fungsi yang mengembalikan judul dari sebuah beritaa.
10.	`News`	`public string date`	Fungsi yang mengembalikan tanggal dari sebuah berita.

No.	Nama Kelas	Fungsi/Prosedur	Keterangan
11.	News	`public string link`	Fungsi yang mengembalikan *link* dari sebuah berita.
12.	News	`public string content`	Fungsi yang mengembalikan isi dari sebuah berita.
13.	News	`public News(string _title, string _date, string _link, string _content)`	Konstruktor yang menginisialisasi judul, tanggal, *link*, dan isi dari sebuah berita.

3. Antarmuka

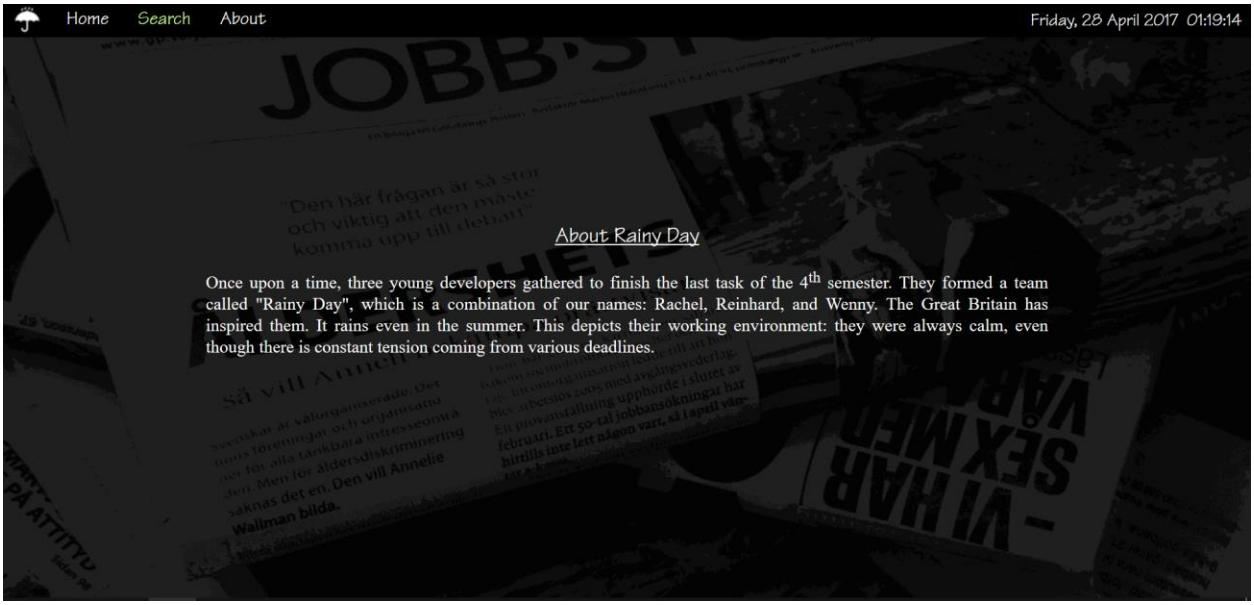
Yang kami gunakan untuk menampilkan halaman web kami adalah cshtml, CSS, JavaScript dan Ajax. Html berguna untuk memberi layout pada seluruh objek yang berada pada layar kami. CSS berguna untuk mengatur font dan style dari seluruh halaman kami. JavaScript merupakan logika dari web kami dan Ajax berguna mengupdate hasil search kami secara realtime.

Kami memiliki 3 page pada website kami. Page pertama berisi logo dari RainyDay beserta tagline kami yaitu “Your daily news solution.”, dengan animasi menyerupai hujan turun sesuai dengan nama kelompok kami. Berikut ini adalah screenshot dari page pertama pada website RainyDay.

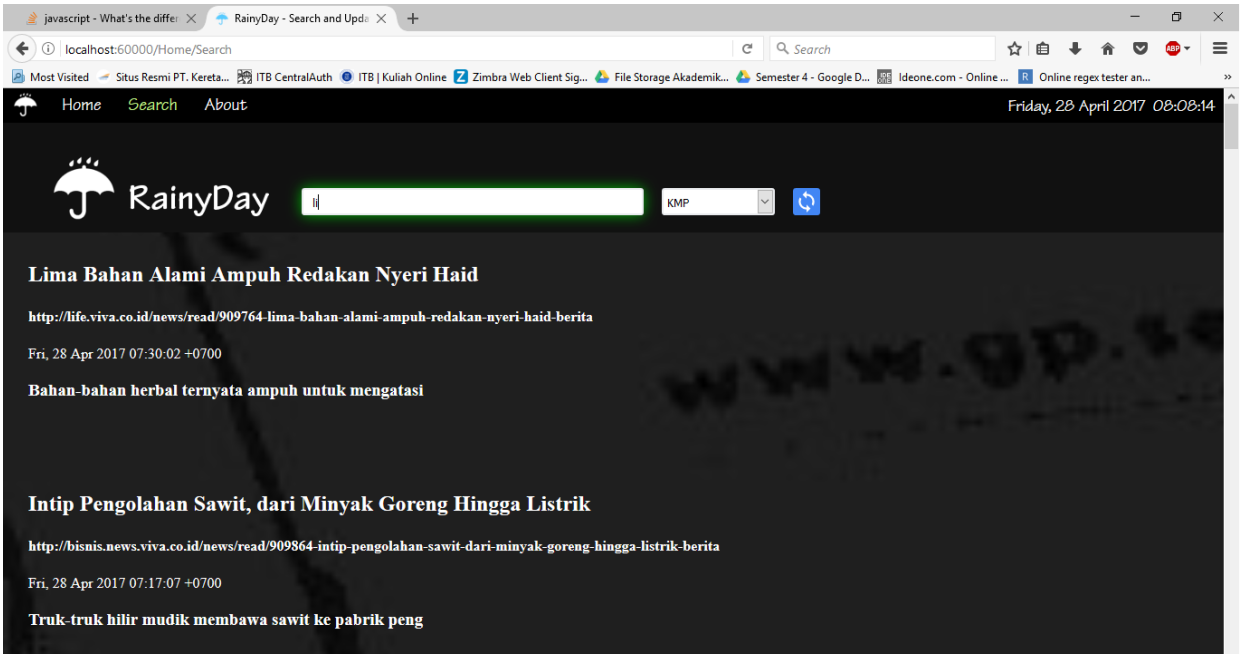


Gambar 7 Screenshot homepage rainy day dengan animasi hujan

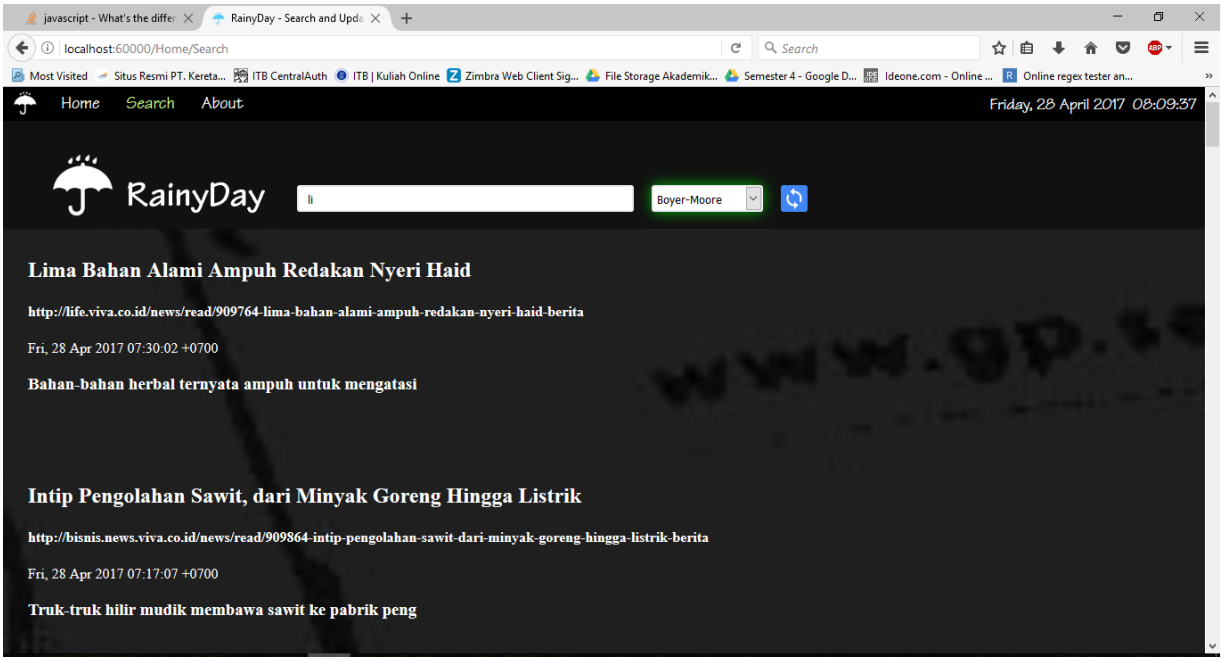
Halaman About menampilkan latar belakang dari pengerjaan proyek ini, beserta alasan pemilihan nama kelompok kami.



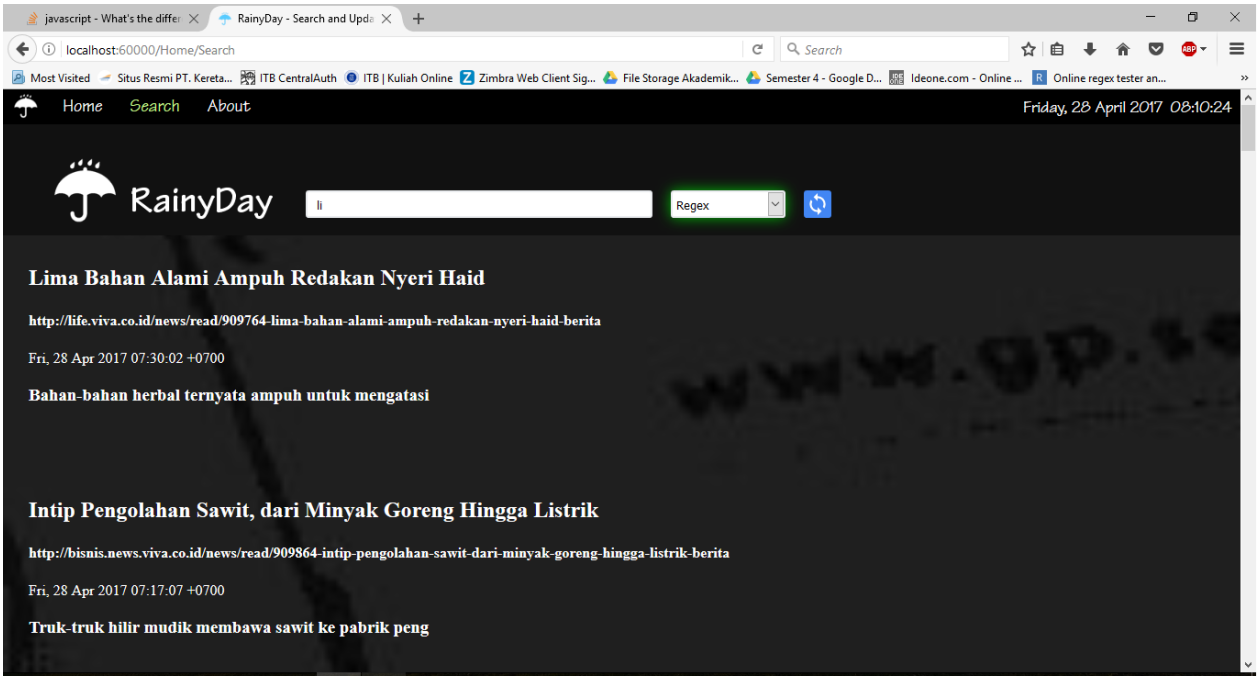
Gambar 8 Halaman About



Gambar 9 Halaman Search untuk KMP



Gambar 10 Halaman Search untuk Boyer-Moore



Gambar 11 Halaman Search untuk Regex

II. Eksperimen/pengujian dengan contoh query

No.	Testcase	Hasil
1.	ahok kalah pemilu	sesuai

No.	Testcase	Hasil
2.	pengumuman SNMPTN	Sesuai
3.	bunga untuk ahok	Sesuai
4.	tips cegah diabetes	Sesuai
5.	kasus korupsi e-ktp	Sesuai
6.	investasi Arab Saudi ke Indonesia	Sesuai
7.	liga spanyol barcelona vs real madrid	Sesuai
8.	kurs dollar kembali normal	Sesuai
9.	universitas terbaik bangsa	Sesuai
10.	warga negara singapura dideportasi	Sesuai
11.	smartphone dual camera	Sesuai
12.	anak ahok girang ahok kalah	Sesuai
13.	google hapus aplikasi gratis	sesuai
14.	harga bbm melonjak di korea utara	Sesuai
15.	vaksin malaria pertama	Sesuai
16.	bArAck kennedY oBaMA	sesuai
17.	teknik informatika itb	Sesuai
18.	rinaldi munir	Sesuai
19.	kapal raksasa TURki	Sesuai
20.	djarot tak maju pilkada jatim	sesuai

III. Analisis hasil pengujian

Dari seluruh pengujian yang kami lakukan dengan testcase yang kami buat, berita yang muncul pada webpage kami sesuai harapan. Pada pencarian dengan KMP dan Boyer-Moore, berita yang ditemukan dari keyword yang diketik oleh user mengandung substring yang sama persis dengan keyword yang diinginkan. Pada pencarian dengan Regex, substring yang ditemukan tidak hanya substring yang sama persis dengan keyword tetapi juga substring

yang mengandung kata-kata lain yang berada di antara keyword yang dimasukkan oleh user, jika keyword terdiri dari lebih dari 1 (satu) kata atau memiliki spasi sebelum atau setelah atau di antara kata-kata pada keyword. Screenshot dari hasil pencarian untuk KMP, Boyer Moore, dan Regex telah kami tampilkan pada bagian antarmuka.

BAB V
KESIMPULAN

No.	Pertimbangan	Keterangan
1.	Program berjalan dengan akurat.	Terpenuhi
2.	Laporan sesuai spesifikasi.	Terpenuhi
3.	Penjelasan pada kode program.	Terpenuhi
4.	User interface yang menarik.	Terpenuhi
5.	Usaha memudahkan asisten agar tidak menambah beban asisten terutama pada semester 6 dengan membuat tabel ini untuk mempermudah penilaian. (Semangat!)	Terpenuhi

REFERENSI

[1] http://www.elangsakti.com/2013/03/algoritma-pencocokan-pencarian-string.html diakses pada 24 April 2017.
[2] https://id.wikipedia.org/wiki/Algoritma_Knuth-Morris-Pratt diakses pada 24 April 2017.
[3] Boyer, Robert Moore, J. 1977. A Fast String Searching Algorithm.
[4] https://id.wikipedia.org/wiki/Algoritma_Boyer-Moore#cite_note-2 diakses pada 25 April 2017.
[5] Ophie, Edmund. 2012. Aplikasi Algoritma String Matching dan Regex untuk Validasi Formulir. Bandung: Institut Teknologi Bandung.

.