# APLIKASI ALGORITMA KNUTH-MORRIS-PRATT PADA MESIN PENCARI KATA UNTUK LINGKUNGAN *WEBSITE* MAHASISWA INFORMATIKA 2005

**Herdyanto Soeryowardhana**

Program Studi Teknik Informatika Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if15095@students.if.itb.ac.id

## ABSTRAK

**Jumlah website yang semakin bertambah setiap tahunnya serta kebutuhan pengguna akan aplikasi yang dapat mencari website-website yang diinginkan membuat mesin pencari (search engine) pada saat ini merupakan aplikasi yang sangat penting. Bahkan hampir di setiap *browser*, atau website-website besar juga sudah menyediakan mesin pencari yang terintegrasi. Salah satu bagian penting dari mesin pencari adalah algoritma pencocokan string. Algoritma pencocokan string saat ini sudah banyak ditemukah. Setiap algoritma memiliki kelebihan dan kekurangan dalam hal kecepatan, kemudahan pengaplikasian dan sebagainya. Salah satu contoh algoritma pencocokan string adalah algoritma Knutt-Morris-Pratt. Berdasarkan hal-hal diatas, penulis kemudian merasa tertarik untuk mengaplikasikan salah satu algoritma pencocokan string yaitu Knutt-Morris-Pratt ke dalam mesin pencari untuk lingkungan *website* mahasiswa Informatika 2005. Mesin pencari ini berbasis *web*. Oleh karena itu, penulis mengaplikasikannya dalam bahasa pemrograman PHP (*PHP Hypertext Preprocessor*). Kemampuan algoritma KMP cenderung lebih baik ketimbang algoritma *BruteForce*. Setelah penulis menguji program tersebut, program tersebut dapat menghasilkan keluaran yang sesuai dengan keinginan pengguna pada umumnya dan memiliki kecepatan pencarian yang cukup baik.**

**Kata kunci:** Knutt-Morris-Pratt, PHP, Mesin Pencari.

## 1. PENDAHULUAN

1 spasi, 10 pt

Mesin pencari atau *search engine* pada saat ini sudah merupakan kebutuhan yang sangat penting. Apalagi setiap tahunnya jumlah *website* di dunia semakin bertambah banyak. Mesin pencari seperti Google atau Yahoo bahkan sudah melengkapinya dengan kata-kata kunci atau sintaks yang memudahkan penggunanya untuk mencari informasi-informasi yang relevan dengan keinginan pengguna.

Algoritma pencocokan string pada mesin pencari, barang tentu sudah menjadi bagian yang sangat penting meskipun faktor-faktor lain juga sangat menentukan dalam pencarian yang lebih akurat lagi. Algoritma pencocokan string pada saat ini sudah sangat banyak ditemukan diantaranya : *Bruteforce*, Knutt-Morris-Pratt, Boyer-Moore, dan sebagainya.

Hubungan antara mesin pencari dan algoritma pencarian string membuat penulis merasa tertarik untuk mengaplikasikan algoritma Knutt-Morris-Pratt ke dalam mesin pencari dalam lingkungan *website* mahasiswa Informatika 2005. Program yang penulis buat menggunakan bahasa pemrograman PHP agar lebih mudah dalam mengaplikasikannya dilingkungan berbasis *web*.

Makalah ini berisi dari bagian-bagian. Pada bagian metode penulis membagi lagi kedalam subbab-subbab yaitu bagian penjelasan mengenai algoritma *Bruteforce* dan Knutt-Morris-Pratt, dan bagian mengenai Aplikasi algoritma KMP di dalam mesin pencari yang penulis buat. Pada bagian selanjutnya, penulis akan menjelaskan tentang hasil keluaran dari program yang penulis buat. Pada bab terakhir, penulis akan memberikan kesimpulan-kesimpulan dari keseluruhan makalah.

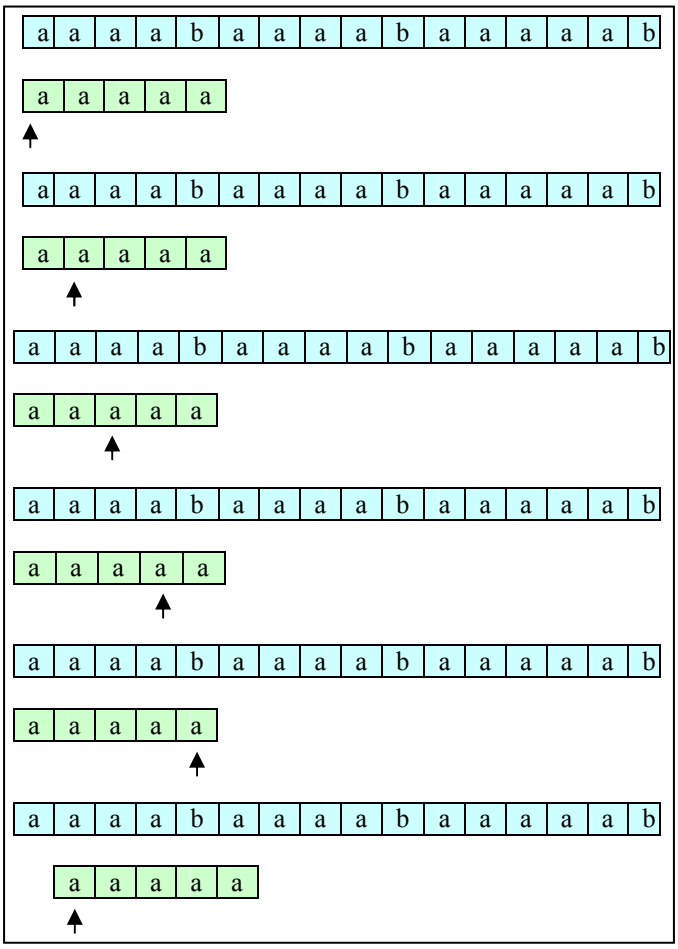
## 2. METODE

### 2.1 Algoritma *BruteForce* dan Knutt-Morris Pratt

Algoritma KMP menggunakan informasi mengenai karakter yang berada di dalam string yang dicari untuk menentukan seberapa jauh untuk menggerakkan string yang dicari setelah ketidakcocokan terjadi. Untuk mengilustrasikan hal ini, perhatikan contoh dibawah ini.

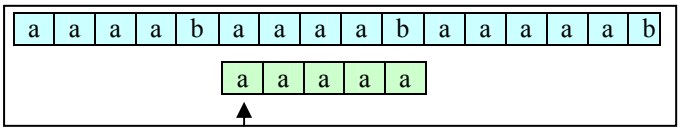
Misalkan terdapat dua string yaitu : s1 = “aaaabaaaabaaaaab” dan s2 =”aaaaa”. Dengan

menggunakan algoritma *BruteForce* maka string akan dicocokkan dengan cara seperti dibawah ini:




**Gambar 1. Pencocokan string menggunakan secara BruteForce.**

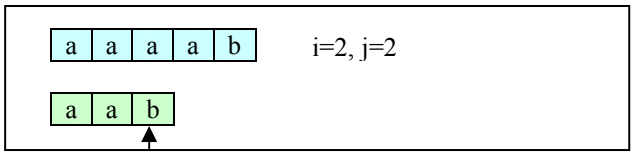
Tetapi faktanya jika kita melihat pada s2 kita dapat mengatakan bahwa disana tidak ada kesempatan yang membuat posisi kedua cocok. 'b' akan berakhir jika dicocokkan dengan karakter keempat pada s2, yaitu 'a'. Berdasarkan pada pengalaman kita pada s2 bahwa kita ingin iterasi terakhir diatas digantikan dengan :




**Gambar 2. Iterasi pengganti.**

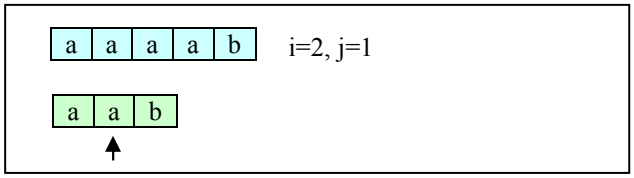
Kita dapat mengimplementasikan ide ini dengan cukup efisien dengan menghubungkan dengan setiap posisi elemen dalam pencarian string dengan jumlah yang bisa kita pindahkan dengan aman jika kita mendapat ketidakcocokan dalam posisi elemen. Contohnya pada pencocokan a yang pertama jika tidak cocok pada elemen pertama maka pindahkan string satu langkah. Contoh lainnya jika tidak cocok pada elemen kedua lalu pindahkan 2 langkah.

Faktanya algoritma KMP adalah lebih sedikit cerdik dibanding contoh diatas. Lihatlah contoh dibawah ini:




**Gambar 3.**

Kita hanya dapat memindahkan string kedua, tetapi kita tahu bahwa karakter pertama akan cocok, seperti dua elemen pertama yang identik, jadi kita ingin iterasi berikutnya seperti :



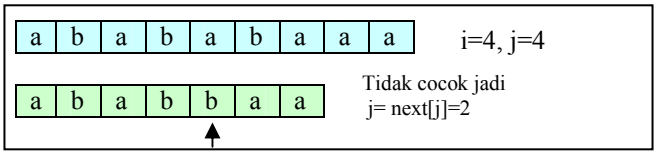

**Gambar 4.**

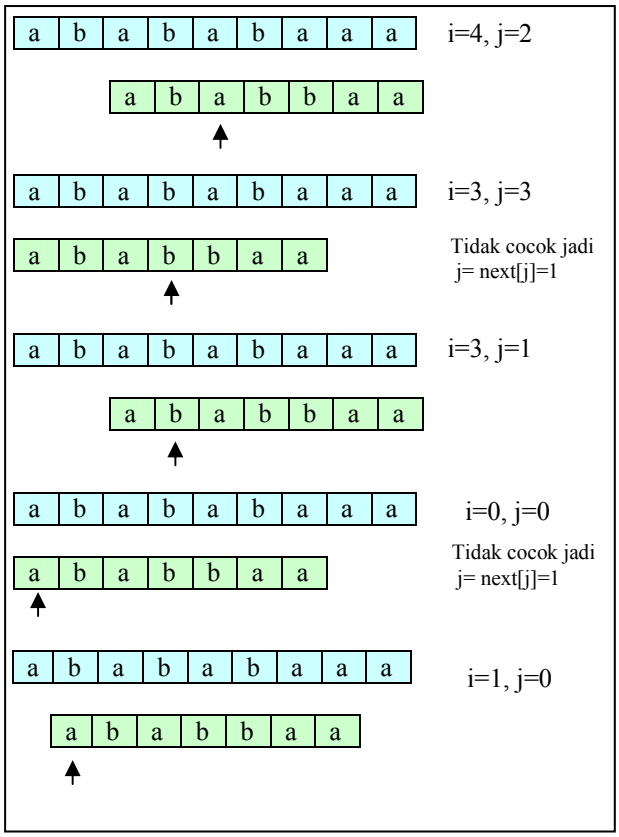
Dengan catatan bahwa i tidak berubah. Hal itu menunjukkan bahwa kita jangan pernah memajukan nilai i tetapi pada suatu ketidakcocokan, hanya perlu menambahkan nilai j sesuai jumlahnya, untuk mendapatkan fakta bahwa kita menggerakkan s2 lebih dari 1 bit sepanjang s1, jadi hubungan antara posisi pada s2 dan posisi i adalah rendah. Kita dapat mempunyai suatu *array* next[j] yaitu untuk setiap posisi dalam s2, posisi dalam s2 yang harus kita simpan dalam s2 yang tidak cocok.

**Tabel 1. Tabel nilai j, s2[j] dan Next[j]**

| j | s2[j] | Next[j] |
|---|---|---|
| 0 | a | -1 |
| 1 | b | 0 |
| 2 | a | 0 |
| 3 | b | 1 |
| 4 | b | 2 |
| 5 | a | 0 |
| 6 | a | 1 |

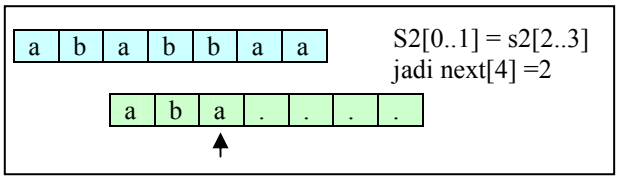
Dalam faktanya next[0] adalah kasus khusus. Jika perbandingan pertama gagal, kita akan menjaga j selalu tetap dan menambahkan nilai i. Jika kita ingin menambahkan nilai i dan j pada setiap kalang, ini dapat didapatkan dengan mudah jika next[0]=-1.






**Gambar 5. Pencocokan String menggunakan KMP.**

Cukup mudah untuk mengimplementasikan algoritma ini jika kita sudah mempunyai *array* next[]. Langkah berikutnya adalah bagaimana kita menghitung next[] pada suatu string. Kita dapat melakukan hal ini dengan mencoba untuk mencocokkan suatu string dengan dirinya. Ketika melihat next[j] kita dapat mencari indeks k pertama sehingga s2[0...k-1] = s2[j-k,..,j-1], contohnya :




**Gambar 6. Contoh pencocokan pattern dengan dirinya.**

Berdasarkan gambar di atas didapatkan s2[0..1] = s2[2..3]. Jadi next[4] = 2. Jika tidak ada daerah yang cocok maka kembalikan 0. Jika j=0, kembalikan -1. Intinya kita cari next[j] dengan menggeser *pattern* ke depan melewati dirinya sendiri, sampai kita menemukan suatu kecocokan antara karakter k pertama dengan karakter k sebelum posisi j.

Dibawah ini ditampilkan Algoritma KMP dalam bahasa C.

```
void preKmp(char *x, int m, int kmpNext[]) {
  int i, j;
  i = 0;
  j = kmpNext[0] = -1;
  while (i < m) {
    while (j > -1 && x[i] != x[j])
      j = kmpNext[j];
    i++;
    j++;
    if (x[i] == x[j])
      kmpNext[i] = kmpNext[j];
    else
      kmpNext[i] = j;
  }
}


void KMP(char *x, int m, char *y, int n,int *jml) {
  int i, j, kmpNext[XSIZE];

  preKmp(x, m, kmpNext);
  i = j = 0;
  jml=0;
  while (j < n) {
    while (i > -1 && x[i] != y[j])
      i = kmpNext[i];
    i++;
    j++;
    if (i >= m) {
      jml++;
      i = kmpNext[i];
    }
  }
}
```

## 2.2 Aplikasi Algoritma Knutt-Morris-Pratt Pada Bahasa PHP

Untuk memudahkan dalam mengaplikasikan algoritma tersebut pada pencarian kata di lingkungan *website* mahasiswa IF, saya memilih untuk menggunakan bahasa PHP (PHP Hypertext Preprocessor).

Berdasarkan algoritma yang saya tampilkan dalam bahasa C dalam subbab sebelumnya, kemudian saya mengaplikasikan algoritma tersebut ke dalam bahasa PHP. Di bawah ini saya tampilkan kode sumber yang saya buat.

```
<html>
    <body>
    <form method="post" action="index.php">
            <input type="text" name="kata">
            <input type="submit" value="cari">
    </form>

<?php
    function preKmp($x,$m,$kmpNext) {
            $i = 0;
```

```
            $j = -1;
            $kmpNext[0] = -1;
            while ($i < $m) {
        while (($j > -1) && ($x[$i] != $x[$j])){
          $j = $kmpNext[$j];
            }
        $i++;
        $j++;
        if ($x[$i] == $x[$j]){
          $kmpNext[$i] = $kmpNext[$j];
                    }
        else{
          $kmpNext[$i] = $j;
                    }
            }
    }
    function KMP($x, $m, $y, $n) {
            $jml=0;
            preKmp($x, $m, $kmpNext);
            $i = 0;
            $j = 0;
            while ($j < $n) {
        while ($i > -1 && ($x[$i] != $y[$j])){
          $i = $kmpNext[$i];
            }
        $i++;
        $j++;
        if ($i >= $m) {
          $jml++;
                    $i = $kmpNext[$i];
        }
            }
            return $jml;
    }

    $katakunci=$_POST['kata'];
    for ($i=1;$i<=126;$i++){
            $url = "http://students.if.itb.ac.id/~if15";
            if ($i<10){
                    $url = $url."00";
            }
            if ($i>9 && $i<100){
                    $url = $url."0";
            }
            $url = $url . $i;
            $url = $url ."/";
            $url1 = $url."index.php";
            $url2 = $url."index.html";
            $adafile =0;
            if (fopen("$url1","r")!=false){
                    $pegangan =
fopen("$url1","r");
                    $adafile = 1;
            }else if (fopen("$url2","r")!=false){
                    $pegangan =
fopen("$url2","r");
                    $adafile = 2;
            }
            if ($adafile != 0){
                    $Text ="";
                    while
(feof($pegangan)==false){

    $Text=$Text.fgetc($pegangan);
                    }
                    $jml=0;
                    $mm=strlen($katakunci);
                    $nn=strlen($Text);
                    $jml=KMP($katakunci, $mm,
$Text, $nn);
                    if ($jml!=0){
                            if ($adafile==1){

    printf("<br>%s\n",$url1);
                            }
                            else {

    printf("<br>%s\n",$url2);
                            }
                            echo $jml;
                    }
                    fclose($pegangan);
            }
    }
?>
  </body>
</html>
```

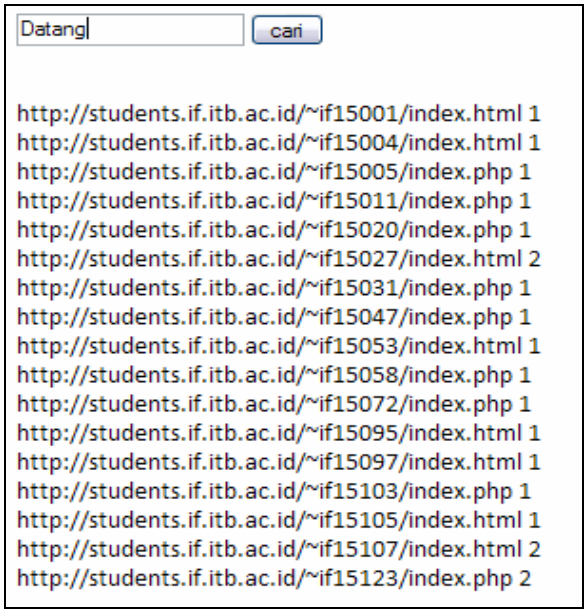
Maksud dari program di atas adalah program tersebut akan menampilkan form pengisian teks untuk teks yang dicari dan juga menampilkan tombol ‘cari’ untuk mengeksekusi pencarian teks tersebut.

Setelah pengguna menekan tombol ‘cari’ maka program akan mencari kata tersebut di file index.php atau index.html yang berada di *website* mahasiswa informatika 2005 dan hanya melakukan pencarian dari alamat http://students.if.itb.ac.id/~if15001/ sampai dengan http://students.if.itb.ac.id/~if15126/. Pencarian kata tidak termasuk file-file di dalam website selain file index.html dan file index.php.

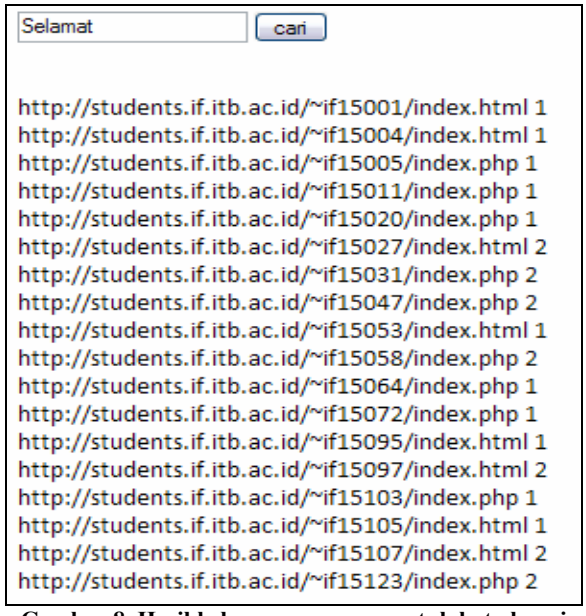
Jika program menemukan kata kunci yang diinginkan oleh pengguna maka akan ditampilkan alamat website dan juga ditampilkan pula jumlah kata yang ditemukan di dalamnya.

## 3. HASIL

Program yang penulis buat akan menampilkan URL letak kata ditemukan dan jumlah kemunculan kata kunci tersebut. Berikut hasil contoh keluaran dari program.




**Gambar 7. Hasil keluaran program untuk kata kunci "Datang"**




**Gambar 8. Hasil keluaran program untuk kata kunci "Selamat"**

Dari gambar diatas dapat dilihat kotak pengisian kata kunci, tombol cari, hasil URL yang ditemukan dan hasil jumlah kata yang ditemukan. Contoh pada gambar 8 : http://students.if.itb.ac.id/~if15123/index.php 2 artinya adalah di dalam URL tersebut ditemukan kata "Selamat" sebanyak 2 buah.

## 4. KESIMPULAN

Algoritma Knutt-Morris-Pratt cenderung lebih efisien dibanding dengan algoritma *Bruteforce* pada umumnya. Namun, KMP cenderung lebih sulit untuk dihapalkan dibanding *Bruteforce.* Pembuatan aplikasi mesin pencari sederhana tidak terlalu sulit. Oleh karena itu, diharapkan setelah pembaca membaca makalah ini, pembaca dapat mengaplikasikan mesin pencari pada khususnya lingkungan *web* dengan mudah.

## REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika, 2006.

[2] Kadir, Abdul, "Dasar Pemrograman Web Dinamis Menggunakan PHP", Penerbit Andi, 2002.

[3] Alison Cawsey (1998) *http://www.macs.hw.ac.uk/~alison/ds98/node77.html*. Tanggal akses: 16 Mei 2007 pukul 22:11

[4] Christian.Charras *(1997) http://www-igm.univ-mlv.fr/~lecroq/string/node8.html#SECTION0080*. Tanggal akses: 16 Mei 2007 pukul 22:11

[5] Wikipedia The Free Encyclopedia (2007) *http://en.wikipedia.org/wiki/Knuth-Morris-Pratt_algorithm*. Tanggal akses: 16 Mei 2007 pukul 22:11