**PERBANDINGAN PERFORMANSI METODE STRING MATCHING**

**UNTUK PENCARIAN DATA KONSUMEN**

**LAPORAN SKRIPSI**

Logo

Description automatically generated

**Oleh :**

**ISEP LUTPI NUR**

**2113191079**

**FAKULTAS TEKNIK**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**UNIVERSITAS SANGGA BUANA YPKP**

**2023**

# **DAFTAR ISI**

# **DAFTAR GAMBAR**

# **DAFTAR TABEL**

# **BAB I PENDAHULUAN**

* 1. **Latar Belakang Masalah**

Teknologi informasi dan komunikasi telah mengubah cara orang mencari, memproses, dan mengakses informasi. Dalam era digital ini data dapat ditemukan dengan lebih mudah dan cepat daripada sebelumnya. Hal ini memberikan dampak signifikan pada banyak aspek kehidupan manusia, termasuk dalam bidang pendidikan, bisnis, hiburan, dan lain-lain.

Dalam era digital seperti sekarang ini, pencarian data atau informasi menjadi sangat penting dan sering dilakukan oleh banyak orang. Seiring dengan semakin banyaknya data yang tersedia, pencarian data menjadi semakin kompleks dan memerlukan algoritma yang efektif dan efisien untuk dapat mencari data dengan cepat.

Data konsumen menjadi salah satu faktor penting dalam bisnis. Data konsumen dapat membantu perusahaan dalam mengambil keputusan yang lebih tepat dan memaksimalkan potensi bisnis. Salah satu cara untuk mengumpulkan data konsumen adalah dengan melakukan pencarian data pada database perusahaan.

Salah satu jenis pencarian data yang sering digunakan adalah pencarian string atau pola dalam sebuah teks atau data. Pencarian string dapat digunakan dalam berbagai aplikasi, seperti deteksi plagiarism, analisis teks, pencarian teks di dalam database, dan lain sebagainya.

Terdapat penelitian *string matcing* untuk Pencarian Informasi Data Zakat dan Aktivitas Sosial dengan menggunakan algoritma *Knuth Morris Pratt* (KMP). Dalam penelitian tersebut menunjukan bahwa performa algoritma KMP sangat baik dengan rata-rata waktu eksekusi dalam lima kali pengujian yaitu 0.03 ms, 0.03 ms, 0.02 ms, 0.02 ms dan 0.03 ms [1].

Terdapat beberapa metode algoritma yang dapat digunakan dalam pencarian string, antara lain *Naïve String Matching, Knuth-Morris-Pratt (KMP)*, *Boyer-Moore*, dan *Rabin-Karp Algorithm*. Masing-masing algoritma memiliki kelebihan dan kelemahan tertentu, serta berbeda dalam hal waktu eksekusi dan *space complexity*.

Penelitian ini bertujuan untuk membandingkan performa algoritma string matching yang populer, yaitu *Naïve String Matching*, KMP, *Boyer-Moore*, dan *Rabin-Karp* dalam mencari pola pada sebuah string. Performa algoritma akan diukur menggunakan metrik seperti waktu eksekusi dan space complexity. Penelitian ini juga akan membahas tentang performa algoritma *SQL LIKE* dalam mencari pola pada data yang tersimpan di dalam database dan membandingkannya dengan algoritma string matching yang telah disebutkan sebelumnya.

Dalam penelitian ini, akan dikaji kelebihan dan kelemahan masing-masing algoritma string matching dan *SQL LIKE* dalam konteks pencarian data konsumen. Selain itu, penelitian ini juga akan memberikan rekomendasi mengenai algoritma yang tepat digunakan dalam pencarian data berdasarkan karakteristik data dan kebutuhan pengguna.

Diharapkan hasil penelitian ini dapat memberikan kontribusi dalam pengembangan teknologi pencarian data di masa depan, terutama dalam hal pengembangan algoritma string matching dan SQL LIKE yang lebih efektif dan efisien. Selain itu, diharapkan penelitian ini dapat memberikan kontribusi dalam bidang ilmu komputer khususnya dalam pengolahan string dan algoritma. Oleh karena itu, pada penelitian ini dibuat dengan judul “PERBANDINGAN PERFORMANSI METODE STRING MATCHING UNTUK PENCARIAN DATA KONSUMEN”

* 1. **Rumusan Masalah**

Berdasarkan uraian di atas, terdapat beberapa hal yang dapat disimpulkan untuk dijadikan sebagai rumusan masalah yang selanjutnya akan dibuatkan laporan penelitian ini di antaranya:

1. Bagaimana performa dari algoritma string matching SQL LIKE*, Naïve String Matching, KMP, Boyer-Moore, dan Rabin-Karp* dalam mencari pola pada sebuah string untuk mencari data konsumen dalam jumlah besar?
2. Bagaimana mengukur Performa algoritma tersebut dengan menggunakan metrik seperti waktu eksekusi dan space complexity?
3. Bagaimana menentukan algoritma yang paling tepat dalam pencarian data berdasarkan karakteristik data dan kebutuhan pengguna?
   1. **Batasan Masalah**

Ada beberapa batasan masalah dalam menyelesaikan penelitian ini yaitu sebagai berikut :

1. Penelitian ini hanya akan membahas performa dari algoritma string matching SQL LIKE*, Naive String Matching, KMP, Boyer-Moore, dan Rabin-Karp* dalam mencari pola pada sebuah string.
2. Penelitian ini dilakukan dengan membandingkan waktu eksekusi dan space complexity yang dibutuhkan algoritma tersebut.
3. Penelitian ini hanya akan membahas algoritma string matching SQL LIKE*, Naive String Matching, KMP, Boyer-Moore, dan Rabin-Karp* yang paling tepat dalam pencarian data berdasarkan karakteristik data dan kebutuhan pengguna.
   1. **Maksud dan Tujuan**

Maksud penelitian ini adalah untuk melakukan perbandingan dari beberapa algoritma string matcing.

Kemudian tujuan dari penelitian ini adalah:

1. Membandingkan performa dari lima algoritma string matching yang populer, yaitu SQL LIKE*, Naive String Matching, KMP, Boyer-Moore, dan Rabin-Karp* dalam mencari pola pada sebuah string.
2. Memberikan rekomendasi mengenai algoritma yang tepat digunakan dalam pencarian data berdasarkan karakteristik data dan kebutuhan pengguna khususnya yang memiliki data berukuran besar dan diakses oleh banyak penguna sekaligus misalnya toko online.
   1. **Metode Penelitian**

Metode penelitian yang diterapkan dibagi menjadi tiga bagian, yaitu metode pengumpulan data, metode pengembangan sistem dan metode pengembangan perangkat lunak.

* + 1. **Metode Pegumpulan Data**

Metode pengumpulan data yang diambil dalam penelitian ini adalah sebagai berikut:

* + - 1. Studi pustaka

Pada tahap ini penulis mencari referensi teori yang relevan dengan studi kasus atau permasalahan yang ditemukan. Referensi tersebut didapatkan dari jurnal, buku, situs-situs internet dan artikel laporan penelitian.

* + - 1. Eksperimen

Pengumpulan data penelitian dilakukan dengan melakukan percobaan terhadap suatu hal, yaitu pengambilan sampel objek (dataset) yang dilanjutkan dengan observasi, pelatihan, dan pencatatan hasil yang berkaitan dengan penelitian.

* + 1. **Metode Pengembangan Sistem**

Metode pengembangan sistem dalam penelitian ini menggunakan lima metode string matching yaitu Algoritma Boyer-Moore, *Naïve String Matching*, *Knuth-Morris-Pratt, Rabin-Karp* dan *SQL LIKE.*

* + 1. **Metode Pengembangan Perangkat Lunak**

Metode pengembangan perangkat lunak dalam penelitian ini menggunakan metode Agile. Metode Agile adalah metode yang mengutamakan keunggulan teknis saat mengembangkan perangkat lunak. Kesederhanaan dianggap sangat penting bagi Agile dalam mengoptimalkan sumber daya yang dimiliki. Masing-masing tim pengembang Agile melakukan refleksi agar dapat bekerja secara efektif dan memiliki pola kerja yang baik [2].

Diagram

Description automatically generated

Gambar 1. Metode Pengembangan Perangkat Lunak

Adapun tahapan – tahapan dari metode *Agile* adalah sebagai berikut :

1. ***Requirements*:** Langkah pertama untuk mengidentifikasi kebutuhan dalam pengembangan.
2. ***Design*:** pada langkah ini dilakukan dalam desain visual dan arsitektur aplikasi.
3. ***Development*:** Tahapan ini untuk penulisan kode dan tulang punggung dari keseluruhan proses.
4. ***Testing*:** Langkah ini untuk pengujian dan menentukan kualitas dari perangkat lunak yang dibuat.
5. ***Development***: Langkah ini untuk peluncuran perangkat lunak ke pengguna.
6. ***Review*:** Merupakan langkah untuk menilai atau mengulas aplikasi dan meninjau perangkat lunak yang dibangun.
   1. **Sistematika Penulisan**

Dalam tugas akhir ini, sistematika penulisannya dibagi menjadi sejumlah bab, di antaranya:

**BAB I: PENDAHULUAN**

Pendahuluan berisi tentang latar belakang dari permasalahan yang dihadapi, rumusan masalah, batasan masalah, maksud dan tujuan, metode penelitian serta sistematika penulisan.

**BAB II: LANDASAN TEORI**

Landasan Teori tentang pembahasan tentang teori-teori yang ada kaitannya dengan penelitian yang sedang dilakukan, seperti konsep dasar model pengembangan sistem, konsep dasar pemrograman, dan peralatan pendukung sistem yang dijadikan sebagai rujukan dalam melakukan penelitian.

**BAB III: ANALISIS DAN PERANCANGAN**

Analisis serta perancangan isinya mengenai penjelasan tentang analisa perancangan sistem yang akan dibangun, yang meliputi analisis sistem berjalan, sistem yang diusulkan, perancangan UML, perancangan *database*, serta perancangan antar muka.

**BAB IV: IMPLEMENTASI DAN PENGUJIAN SISTEM**

Berisikan tentang implementasi, pengujian dan analisa hasil dari penelitian terhadap properti dari aplikasi android. Sehingga bisa diketahui apakah sistem tersebut sanggup menuntaskan masalah yang ada serta cocok dengan tujuan dari penelitian ini.

**BAB V: PENUTUP**

Berisikan tentang kesimpulan dari pelaksanaan tugas akhir ini. Penutup juga berisikan tentang saran, di mana saran tersebut bisa bermanfaat untuk pembaharuan sistem pada waktu ke depannya.

# **BAB 2 LANDASAN TEORI**

1. **Data Konsumen**

Data konsumen adalah informasi yang di terima saat konsumen berinteraksi dengan media bisnis. Media ini dapat berupa situs web, aplikasi seluler, halaman media sosial, halaman survei, kampanye, dan versi online atau offline lainnya dari upaya pemasaran bisnis.

Data konsumen penting karena memungkinkan bisnis untuk mengoptimalkan platform melalui pemahaman berbagai aspek interaksi konsumen. Dengan menggunakan data konsumen, bisnis tidak hanya dapat merampingkan situs web yang mereka miliki dan tunjukkan kepada konsumen, tetapi mereka juga dapat mengoptimalkan iklan dan saran produk untuk konsumen tertentu menggunakan data mereka.

Data konsumen yang berjumlah besar biasanya terdapat di toko online. Data konsumen di toko online dapat mencakup berbagai jenis informasi yang dikumpulkan dari interaksi konsumen dengan platform toko online seperti Data Informasi pribadi, Data transaksi, Data Produk dan Data Prilaku.

1. **Algoritma**

Algoritma adalah urutan instruksi atau aturan yang sistematis, terstruktur, dan terdefinisi dengan baik untuk menyelesaikan sebuah masalah atau tugas secara efektif dan efisien. Algoritma digunakan dalam berbagai bidang, termasuk matematika, ilmu komputer, teknologi informasi, fisika, dan banyak lagi.

Algoritma sering digunakan dalam pemrograman komputer sebagai panduan untuk menyelesaikan suatu tugas atau masalah tertentu. Algoritma dapat ditulis dalam bahasa yang mudah dimengerti oleh manusia, yang kemudian diterjemahkan ke dalam bahasa yang dimengerti oleh komputer, seperti bahasa pemrograman.

Algoritma biasanya memiliki tujuan tertentu dan dapat digunakan untuk menyelesaikan berbagai masalah, mulai dari yang sederhana hingga yang kompleks. Beberapa contoh masalah yang dapat diselesaikan dengan algoritma adalah sorting (pengurutan) data, pencarian data, enkripsi data, dan optimisasi pemrosesan data.

1. **Algoritma *String Matching***

Algoritma *string matching* adalah algoritma untuk mencari keberadaan sebuah pola atau substring dalam sebuah string. Pola atau substring tersebut dapat berupa satu karakter atau beberapa karakter yang harus ditemukan dalam sebuah string.

Algoritma string matching sangat umum digunakan dalam pengembangan perangkat lunak, seperti dalam aplikasi pencarian atau manipulasi string. Contoh penggunaan algoritma string matching adalah ketika kita ingin mencari sebuah kata tertentu dalam sebuah teks.

1. **Algoritma *Naïve String Matching***

Naive string matching adalah salah satu algoritma string matching yang paling sederhana dan mudah dipahami. Algoritma naive string matching membandingkan setiap karakter dari sebuah pola dengan setiap karakter di dalam sebuah string. Algoritma ini memulai dengan menempatkan pola di awal string dan kemudian membandingkan karakter demi karakter. Jika karakter-karakter tidak cocok, maka pola dipindahkan satu karakter ke kanan dan proses pencocokan kembali dilakukan.

Algoritma naive string matching memiliki kompleksitas waktu sebesar O(mn), di mana m adalah panjang pola dan n adalah panjang string. Algoritma ini bekerja dengan baik pada string yang relatif pendek dan pola yang singkat.

Berikut adalah pseudocode algoritma naive string matching:

NAIVE-STRING-MATCHING(T, P)

n = length(T)

m = length(P)

for i = 1 to n-m+1 do

if P[1..m] == T[i..i+m-1] then

print "Pattern occurs with shift " i

Pseudocode di atas menjelaskan bagaimana algoritma naive string matching mencari keberadaan sebuah pola P dalam sebuah string T. Algoritma ini membandingkan setiap m karakter dalam P dengan setiap m karakter di T. Jika ada kecocokan, algoritma akan mencetak pesan bahwa pola ditemukan dengan pergeseran (shift) tertentu.

1. **Algoritma *Knuth Morris Pratt***

Algoritma *Knuth-Morris-Pratt* adalah salah satu algoritma pencarian string, 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.

Jika kita melihat algoritma KMP lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian.

Perhitungan penggeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokkan pada saat pattern sejajar dengan *teks[i..i+n-1]*, dapat menganggap ketidakcocokan pertama terjadi di antara *teks[i+j]* dan *pattern[j]*, dengan *0<j<n*. Berarti, *teks[i..i+j1]=pattern[0..j-1]* dan *a=teks[i+j]* tidak sama dengan *b=pattern[j].* Ketika kita menggeser, sangat beralasan bila ada sebuah awalan *v* dari pattern akan sama dengan sebagian akhiran *u* dari sebagian teks. Sehingga kita bisa menggeser pattern agar awalan *v* tersebut sejajar dengan akhiran dari *u.*

Dengan kata lain, pencocokkan string akan berjalan secara efisien bila kita mempunyai tabel yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokkan di karakter ke- *j* dari pattern. Tabel itu harus memuat *next[j]* yang merupakan posisi karakter *pattern[j]* setelah digeser, sehingga kita bisa menggeser pattern sebesar *j-next[j]* relatif terhadap teks.

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 dipattern dan di teks yang dibandingkan tidak cocok (mismatch).
   2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.

Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

1. **Algoritma *Boyer Moore***

Algoritma Boyer-Moore diperkenalkan oleh Bob Boyer dan J.S. Moore pada tahun 1977. Pada algoritma ini pencocokan kata dimulai dari karakter terakhir kata kunci menuju karakter awalnya. Jika terjadi perbedaan antara karakter terakhir kata kunci dengan kata yang dicocokkan maka karakter-karakter dalam potongan kata yang dicocokkan tadi akan diperiksa satu per satu. Hal ini dimaksudkan untuk mendeteksi apakah ada karakter dalam potongan kata tersebut yang sama dengan karakter yang ada pada kata kunci. Apabila terdapat kesamaan, maka kata kunci akan digeser sedemikian rupa sehingga posisi karakter yang sama terletak sejajar, dan kemudian dilakukan kembali pencocokan karakter terakhir dari kata kunci. Sebaliknya jika tidak terdapat kesamaan karakter, maka seluruh karakter kata kunci akan bergeser ke kanan sebanyak m karakter, di mana m adalah panjang karakter dari kata kunci.

Booyer-Moore merupakan salah satu Algortima Pattern Matching yang cukup terkenal. Algoritma ini menggunakan beberapa kasus pengecekan teks (input karakter yang akan dibaca) dengan Pattern (pola yang akan disaring). Algoritma Boyer-Moore adalah algoritma pencarian string yang mencari dengan cara membandingkan sebuah huruf dengan huruf yang ada di pattern yang dicari, dan menggeser pattern tersebut hingga posisinya sama dengan teks yang dicari dan membandingkan kata tersebut. Cara ini disebut character jump. Algoritma Pattern Matching Booyer-Moore ini berbasis pada 2 metode yaitu :

* + - 1. The Looking-Glass Technique

The Looking-Glass Technique melakukan perbandingkan suatu karakter akhir pada kata w dengan suatu karakter pada teks s. Jika karakter tersebut sama maka jendela karakter akan berjalan mundur pada kedua string dan mengecek kembali kedua karakter. Mencari Suatu kecocokan String pada Teks dengan pola yang akan dicari dengan cara memindahkan atau menggesernya sampai Teks string selesai.

* + - 1. The Character-Jump Technique

Character-jump Technique melakukan suatu aksi ketika perbandingan antara dua karakter yang berbeda. Ada dua aksi yang tergantung pada teks s dan kata w yang dimiliki; jika p yaitu karakter pada s yang sedang diproses yang tidak cocok maka ada dua kemungkinan aksi. Mencari karakter yang sesuai dan cara penggeseran sebuah karakter perbandingan terakhir.

Beberapa kasus yang ada pada algoritma ini antara lain:

1. Jika suatu karakter Pola (P) mengandung karakter x dimana x adalah anggota dari Teks yang telah dibandingkan maka perbandingan karakter selanjutnya dimulai karakter P yang sama dengan Misal : T = ..xa..??
   1. = xcba

Diagram

Description automatically generated

Gambar 2. 1 Kinerja Algoritma BM dalam Pencarian String

Keterangan Gambar :

Pada Gambar 1 dapat dijelaskan bahwa T yang akan dicocokkan dengan pattern P adalah kata “xa”. Dengan menggunakan algoritma Boyer Moore, pencocokan akan dimulai dari kanan ke kiri sesuai dengan T yang diminta yaitu “xa” pada pattern P “xcba”. Jika T mengalami ketidakcocokan maka T akan melompat sejauh n karakter T pada Pattern P. jika terjadi kecocokan (macth) maka proses akan berhenti.

1. Jika Perbandingan karakter terakhir pada suatu pola sama dengan teks adalah sama. Maka pergeseran karakter selanjutnya bergeser satu kali.

Misal :

T = ..wax..??

* 1. = cwax

Tabel 1. Kinerja Algoritma BM dalam Pencarian String

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P | C | w | A | x |
| T | W | a | X |  |
|  |  | w | A | x |

Keterangan Tabel :

Pada tabel di atas dapat dijelaskan bahwa T yang akan dicocokkan dengan pattern P adalah kata “cwax”. Dengan menggunakan algoritma Boyer Moore, pencocokan akan dimulai dari kanan ke kiri sesuai dengan T yang diminta yaitu “wax” pada pattern P “cwax”. Jika T mengalami ketidakcocokan maka T akan melompat sejauh n karakter T pada Pattern P untuk selanjutnya memulai pencocokan. jika terjadi kecocokan (macth) maka proses akan berhenti.

Algoritma Boyer-Moore ini adalah pembandingan karakter dalam sebuah string yang dilakukan dari belakang ke depan atau dari kanan ke kiri karakter. Jika algoritma Boyer Moore membandingkan teks “MAKALAH” misalnya, algoritma ini melakukan pengecekan apakah karakter ke tujuh dari teks yang dibandingkan adalah karakter ‘H’. Jika karakter ketujuh adalah ‘H’ maka ia akan melakukan pengecekan apakah karakter sebelumnya (ke-6) adalah ‘A’. Demikian seterusnya hingga menemukan bahwa karakter pertama adalah ‘M’.

Alasan kenapa Boyer-Moore melakukan pengecekan dari belakang akan lebih jelas jika kita mengamati apa yang terjadi jika pengecekan menghasilkan nilai yang tidak sama. Misalnya, algoritma mendapati karakter ke-8 dengan karakter ‘X’ bukannya ‘H’. ‘H’ tidak muncul sama sekali pada “MAKALAH”, dan ini berarti tidak ada kesamaan sama sekali karakter ‘X’ dengan semua karakter dalam “MAKALAH”. Sehingga, dengan hanya melakukan sekali pengecekan pada karakter ke-8 kita dapat mengabaikan pengecekan karakter ke-1 sampai ke-7 dan langsung melanjutkan pengecekan karakter dimulai dari karakter ke-9, tepat setelah ‘X’. Algoritma ini pada awalnya melakukan perhitungan sebuah tabel untuk menentukan banyaknya ‘loncatan’ karakter yang akan dilakukan setelah mendapati sebuah perbandingan yang tidak cocok.

Karakteristik utama dari algoritma Boyer-Moore adalah algoritma ini melakukan pencocokan string mulai dari kanan (belakang). dengan karakteristik tersebut, ketidakcocokan saat terjadi perbandingan string akan membuat pergerakan pattern melompat lebih jauh untukmenghindari perbandingan karakter pada string yang diperkirakan gagal. Sehingga proses pencarian string akan lebih optimal.

Contoh cara kerja algoritma Boyer Moore ini adalah sebagai berikut :

Teks : mendaftar di unijoyo

Pattern : daftar

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | m | e | n | d | A | f | t | a | r |  | d | i |  | U | S | B | Y | P | K | P |
| 1 | d | a | f | t | A | r |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  | d | a | f | T | a | r |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  | d | a | F | t | a | r |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  | d | A | f | t | a | r |  |  |  |  |  |  |  |  |  |  |  |

Gambar 2. 2 Contoh Algoritma Boyer Moore

Dalam Gambar 2 dengan menggunakan Algoritma Boyer-Moore dalam pencarian string yang mencari dengan cara membandingkan sebuah huruf dengan huruf yang ada di pattern (m karakter = daftar) yang dicari, dan menggeser pattern sejauh m karakter tersebut hingga posisinya sama dengan teks = n karakter yang dicari dan membandingkan kata tersebut. Dalam pergeserannya, m karakter akan mencocokkan huruf dimulai dari kanan ke kiri. Jika huruf di kanan tidak ada kecocokan dengan n karakter maka m karakter atau pettern akan menggeser sejauh m karakter untuk mencocokkan kembali huruf yang dimaksud, sehingga menjadi cocok.

1. **Algoritma *Rabin Karp***

Algoritma Rabin-Karp diciptakan oleh Michael O. Rabin dan Richard. Karp pada tahun 1987 yang menggunakan fungsi hashing untuk menemukan pattern di dalam string teks.

Karakteristik Algoritma Rabin-Karp :

1. Menggunakan sebuah fungsi hashing
2. Fase prepocessing menggunakan kompleksitas waktu O(m)
3. Untuk fase pencarian kompleksitasnya : O(mn)
4. Waktu yang diperlukan O(n+m)

Algoritma Rabin-Karp adalah algoritma pencocokan string yang menggunakan fungsi hash sebagai pembanding antara string yang dicari (*m*) dengan substring pada teks (*n*). Apabila hash value keduanya sama maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila hasil keduanya tidak sama, maka substring akan bergeser ke kanan. Pergeseran dilakukan sebanyak (*n-m*) kali. Perhitungan nilai hash yang efisien pada saat pergeseran akan mempengaruhi performa dari algoritma ini.

Langkah-langkah dalam algoritma Rabin Karp:

1. Menghilangkan tanda baca dan mengubah ke teks sumber dan kata yang ingin dicari menjadi kata-kata tanpa huruf kapital. Contoh:

Aplikasi Deteksi Source Code C++

Akan dirubah menjadi

aplikasi deteksi source code c

1. Membagi teks ke dalam gram-gram yang ditentukan nilai k-gram nya. Misalkan ukuran k = 7 maka hasilnya seperti dibawah ini:

{aplikas} {plikasi} {likasi } {ikasi d} {kasi de} {asi det} {si dete} {i detek} { deteks} {deteksi} {eteksi } {teksi s} {eksi so} {ksi sou} {si sour} {i sourc} { source} {source } {ource c} {urce co} {rce cod}

{ce code} {e code } { code c}

1. Mencari nilai hash dengan fungsi rolling hash dari tiap gram yang terbentuk.

191614241217501169209949739

204832761207007517191955506

22125508019278132274441702

195269355199245837223500051 197988836209673042221276279 19301450377006679223484029

217299753227221413219436636

* 1. 185691736 74404924

1. Mencari nilai hash yang sama antara 2 teks.

191614241 217501169 209949739

204832761 207007517 191955506 221255080 192781322 74441702

195269355 199245837 223500051

197988836 209673042 221276279

193014503 77006679 223484029

217299753 227221413 219436636

* 1. 185691736 74404924

1. Menentukan persamaan 2 buah teks dengan persamaan Dice's Similarity Coefficient.

Untuk menghitung kemiripan dapat digunakan *Dice Similarity Coeficients.* dengan cara menghitung jumlah K-Gram yang digunakan pada kedua teks nanti yang diuji. Nilai kemiripan tersebut dapat dihitung dengan rumus :

Diagram

Description automatically generated with medium confidence

Dimana:

**S** = nilai kemiripan

**C** = jumlah K-Gram yang sama

**A** serta **B** = jumlah K-Gram dari masing masing string

1. **SQL LIKE**

SQL LIKE adalah sebuah pernyataan (statement) yang digunakan dalam bahasa SQL untuk melakukan pencarian data yang cocok dengan pola tertentu pada sebuah tabel atau database. Dalam SQL LIKE, kita dapat menggunakan wildcard characters atau karakter joker seperti % (untuk merepresentasikan nol atau lebih karakter) dan \_ (untuk merepresentasikan satu karakter) untuk mencari data yang cocok dengan pola yang diinginkan.

Contohnya, kita dapat menggunakan pernyataan SQL LIKE untuk mencari semua data yang memiliki kata "apple" pada nama buah, seperti:

SELECT \* FROM fruits WHERE name LIKE '%apple%';

Pernyataan ini akan mengembalikan semua data pada tabel "fruits" yang memiliki kata "apple" pada kolom "name".

1. **UML (Unified Modelling Language)**

Untuk merancang sistem yang berorientasi objek, dibutuhkan suatu metode pemodelan secara visual, metode ini dinamakan Bahasa Pemodelan Terpadu (UML). UML adalah standar bahasa untuk pendokumentasian, perancangan, dan visualisasi.

Tujuan dibuatnya UML adalah agar mempermudah dalam mengembangkan suatu perangkat lunak. Selain itu UML juga diharapkan dapat mempermudah semua kebutuhan pengguna dengan tepat, lengkap dan efektif. merancang dan juga memodelkan sistem secara matang akan menghasilkan sistem yang baik. UML penting sekali bagi para pengembang sistem karena UML akan menjadi jembatan untuk menerjemahkan antara pengembang sistem dengan pengguna [3] Beberapa macam UML yang ada antara lain:

### *Use Case* Diagram

Use case diagram adalah diagram UML (Unified Modeling Language) yang digunakan dalam rekayasa perangkat lunak untuk menggambarkan interaksi antara sistem atau aplikasi dengan pengguna atau aktor-aktor lain yang terkait. Use case diagram dapat digunakan untuk menggambarkan berbagai skenario penggunaan (use case) dari sebuah sistem atau aplikasi, sehingga dapat membantu dalam memahami kebutuhan fungsional dari sistem atau aplikasi tersebut.

Dalam use case diagram, setiap use case direpresentasikan sebagai oval dan setiap aktor direpresentasikan sebagai kotak. Garis yang menghubungkan use case dan aktor menunjukkan interaksi antara mereka. Use case diagram dapat digunakan untuk mengidentifikasi dan memvisualisasikan aktor-aktor yang terlibat dalam sebuah sistem atau aplikasi, serta skenario-skenario penggunaan yang dapat terjadi.

Use case diagram juga dapat digunakan untuk mengidentifikasi dan memperjelas persyaratan sistem atau aplikasi dengan menggambarkan interaksi antara sistem atau aplikasi dan pengguna atau aktor-aktor lain yang terkait. Use case diagram sering digunakan sebagai langkah awal dalam proses pengembangan perangkat lunak, dan dapat digunakan sebagai alat komunikasi yang efektif antara pengembang perangkat lunak, klien, dan pemangku kepentingan lainnya. [4].



Gambar 1. 2 Contoh Use Case Diagram

### *Activity Diagram*

Activity diagram adalah jenis diagram UML (Unified Modeling Language) yang digunakan untuk menggambarkan alur aktivitas atau tindakan dalam sebuah proses bisnis atau sistem. Diagram ini memperlihatkan urutan tindakan yang terjadi dalam sebuah proses atau sistem dan kondisi yang terjadi pada setiap langkah. Activity diagram sangat berguna dalam menggambarkan aliran kerja sistem, urutan aktivitas yang terjadi, serta pengambilan keputusan dalam suatu proses.

Pada activity diagram, aktivitas direpresentasikan oleh persegi panjang dengan nama aktivitas di dalamnya. Tindakan sederhana dapat direpresentasikan oleh lingkaran kecil di dalam aktivitas. Keputusan direpresentasikan oleh berlian dan memiliki beberapa jalur keluar. Sedangkan fork digunakan untuk merepresentasikan percabangan dalam sebuah proses.

Activity diagram sangat berguna dalam menggambarkan aktivitas dan tindakan dalam sebuah proses, menggambarkan kondisi yang mungkin terjadi, dan menggambarkan percabangan dalam suatu proses. Diagram ini dapat membantu pengembang perangkat lunak untuk memahami alur kerja dalam sistem, mengidentifikasi masalah dan perbaikan dalam alur kerja, serta merancang sistem yang lebih efisien dan efektif. Activity diagram sering digunakan dalam analisis dan perancangan sistem, dan dapat digunakan sebagai alat komunikasi yang efektif antara pengembang perangkat lunak, klien, dan pemangku kepentingan lainnya.



Gambar 2. 3 Contoh Activity Diagram

### *Sequence Diagram*

Sequence diagram adalah jenis diagram UML (Unified Modeling Language) yang digunakan untuk menggambarkan interaksi antara objek dalam sebuah sistem. Diagram ini memperlihatkan urutan pemanggilan metode antara objek, serta urutan pesan yang dikirimkan antara objek dalam sebuah skenario. Sequence diagram sangat berguna dalam memodelkan interaksi antara objek dalam sistem dan memperlihatkan urutan pemanggilan metode yang terjadi.

Pada sequence diagram, objek direpresentasikan oleh sebuah kotak vertikal, dengan nama objek di bagian atas. Urutan pemanggilan metode antara objek direpresentasikan oleh panah horizontal, dengan tanda kurung kurawal di atasnya untuk menunjukkan nama metode. Pesan yang dikirim antara objek direpresentasikan oleh panah vertikal dengan tanda kurung kurawal di atasnya untuk menunjukkan nama pesan.

Sequence diagram berguna dalam menggambarkan interaksi antara objek dalam sebuah sistem, serta memperlihatkan urutan pemanggilan metode yang terjadi. Diagram ini dapat membantu pengembang perangkat lunak untuk memahami bagaimana objek dalam sistem saling berinteraksi, mengidentifikasi masalah dan perbaikan dalam interaksi antara objek, serta merancang sistem yang lebih efisien dan efektif. Sequence diagram sering digunakan dalam analisis dan perancangan sistem, dan dapat digunakan sebagai alat komunikasi yang efektif antara pengembang perangkat lunak, klien, dan pemangku kepentingan lainnya. [4].



Gambar 2. 4 Contoh Sequence Diagram

### *Class Diagram*

Class diagram adalah jenis diagram UML (Unified Modeling Language) yang digunakan untuk memodelkan struktur kelas atau objek dalam sebuah sistem. Diagram ini memperlihatkan kelas-kelas dalam sistem, serta hubungan antara kelas-kelas tersebut. Class diagram sangat berguna dalam memodelkan struktur kelas dalam sistem dan memperlihatkan hubungan antara kelas-kelas tersebut.

Pada class diagram, kelas direpresentasikan oleh sebuah persegi panjang, dengan nama kelas di dalamnya. Atribut kelas direpresentasikan oleh nama atribut di dalam kelas, sedangkan metode kelas direpresentasikan oleh nama metode di dalam kelas. Hubungan antara kelas direpresentasikan oleh panah yang mengarah dari kelas yang satu ke kelas yang lain [4].



Gambar 2. 5 Contoh Class Diagram

1. **PHP (PHP: Hypertext Preprocessor)**

PHP adalah sebuah bahasa pemrograman ***server side scripting*** yang bersifat ***open source***. Sebagai sebuah scripting language, PHP menjalankan instruksi pemrograman saat proses runtime. Hasil dari instruksi tentu akan berbeda tergantung data yang diproses.PHP merupakan bahasa pemrograman server-side, maka script dari PHP nantinya akan diproses di server. Jenis server yang sering digunakan bersama dengan PHP antara lain ***Apache, Nginx***, dan [*LiteSpeed*](https://www.niagahoster.co.id/blog/apa-itu-litespeed/)**.**Selain itu, PHP juga merupakan bahasa pemrograman yang bersifat open source. Pengguna bebas memodifikasi dan mengembangkan sesuai dengan kebutuhan mereka.

1. **Basis Data (MYSQL)**

Database atau basis data adalah kumpulan informadi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Kegunaan utama sistem basis data adalah agar pemakai mampu menyusun suatu pandangan (view) abstraksi data.

Database juga bisa diartikan sebagai sebuah sistem yang berfungsi mengumpulkan data, arsip, atau tabel yang disimpan dan terhubung ke media elektronik, seperti aplikasi atau situs web. Database membuat penyimpanan dan pengelolaan data lebih efisien.

MySQL dibaca MY-ES-KYOO-EL [maɪˌɛsˌkjuːˈɛl]. Beberapa orang bahkan membaca MySQL sebagai “my sequel”. Kegunaan atau fungsi MySQL adalah untuk data warehousing (gudang data), yaitu pengumpulan data dari berbagai sumber, untuk e-commerce, maupun aplikasi logging.

Pengembang pertama MySQL adalah MySQL AB, sebuah perusahaan asal Swedia, yang memulai perjalanannya di tahun 1994. Hak kepemilikan MySQL kemudian diambil secara menyeluruh oleh perusahaan teknologi Amerika Serikat, Sun Microsystems, ketika mereka membeli MySQL AB pada tahun 2008.

Di tahun 2010, Oracle yang adalah salah satu perusahaan teknologi terbesar di Amerika Serikat mengakuisisi Sun Microsystems. Sejak saat itu, MySQL sepenuhnya dimiliki oleh Oracle.

# **BAB III ANALISIS DAN PERANCANGAN**

1. **Analisis Sistem Berjalan**
2. **Sistem Yang Diusulkan**
   * 1. **Data Konsumen**
     2. **Algoritma yang digunakan**
3. **Perbandingan Hasil**
4. **Proses perhitungan**

# **BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM**

# **BAB V PENUTUP**

# **DAFTAR PUSTAKA**

[1] F. Riawan, “Knuth Morris Pratt String Matching Algorithm in Searching for Zakat Information and Social Activities,” *Journal of Applied Data Sciences*, vol. 3, no. 1, 2022, doi: 10.47738/jads.v3i1.49.

[2] K. Prihandani, “Tinjauan Kualitas Pengembangan Sistem Informasi Dengan Metode Agile .,” *Tinjauan Kualitas Pengembangan Sistem Informasi Dengan Metode Agile .*, no. October, 2016.

[3] N. Koch and A. Kraus, “The expressive power of uml-based web engineering,” *Second International Workshop on Weboriented Software Technology IWWOST02*, vol. 16, pp. 105–119, 2002.

[4] Rendi Juliarto, “Apa itu UML? Beserta Pengertian dan Contohnya,” *https://www.dicoding.com/*, 2022. https://www.dicoding.com/blog/apa-itu-uml/ (accessed Jan. 08, 2023).