BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Sebelumnya

Penelitian yang sudah dilakukan sebelumnya tentang sistem pencarian buku atau bahan pustaka pada perpustakaan sudah banyak dilakukan dan banyak menggunakan berbagai macam algoritma. Salah satu algoritma yang digunakan adalah algoritma *String Matching,* berikut merupakan beberapa penelitian terkait mengenai penerapan algoritma *String Matching*:

Tabel 2. 1. Referensi Penelitian Menggunakan Algoritma *String Matching*

No.	Judul Penelitian	Tujuan Penelitian	Metode	Hasil	Keterangan
1.	Implementasi Algoritma *Horspool* Pada Aplikasi Istilah *Fashion* (2019) [6]	Mempermudah masyarakat awam yang kesulitan untuk mencari informasi terkait istilah-istilah *fashion* dengan lebih mudah.	Algoritma *String Matching* yang digunakan adalah Algoritma *Horspool.*	Algoritma *Horspool* dapat menemukan kecocokan *string* pada tahap ketiga, sesuai nilai pergeseran pada tabel *bad match.*	Penelitian menggunakan data istilah *fashion* dengan menerapkan Algoritma *Horspool* didalamnya.
2.	Penelusuran Katalog Perpustakaan Pada SMA IT Yabis Bontang Dengan Algoritma *Boyer-Moore* (2018) [7]	Membuat sistem yang dapat memudahkan pengunjung perpustakaan.	Algoritma *Boyer-Moore* dan metode pengembangan *Waterfall*.	Sistem penelusuran katalog perpustakaan sekolah dapat membantu meringankan kerja petugas atau *admin* dalam memberikan laporan jumlah buku yang tersedia di perpustakaan.	Penelitian menggunakan data buku pada perpustakaan SMA IT Yabis Bontang dengan menerapkan Algoritma *Boyer-Moore* pada sistem.

Tabel 2. 1. Referensi Penelitian Menggunakan Algoritma *String Matching* (lanjutan)

No.	Judul Penelitian	Tujuan Penelitian	Metode	Hasil	Keterangan
3.	Implementasi Algoritma *Knuth-Morris-Pratt* Pada Fungsi Pencarian Judul Tugas Akhir *Repository* (2017) [8]	Implementasi Algoritma *KMP* pada fungsi pencarian dalam sistem *repository* tugas akhir.	Algoritma *KMP* dan metode pengembangan *Waterfall*.	Hasil pengujian performa menunjukan bahwa rata-rata performa Algoritma *KMP* dalam menemukan kata di form pencarian adalah 0.0138 detik.	Penelitian menggunakan data *repository* tugas akhir dengan menerapkan Algoritma *KMP* didalamnya.
4.	Aplikasi Saran Buku Bacaan Bagi Pengunjung Perpustakaan AMIK STIEKOM Sumatra Utara Berdasarkan Algoritma *Brute Force* (2018) [9]	Pemicu minat para pengunjung untuk tetap membaca buku-buku yang tersedia.	Algoritma *String Matching* yang digunakan adalah Algoritma *Brute Force*.	Pemanfaatan Algoritma *Brute Force* dalam proses pencarian buku bacaan yang disarankan kepada para pengunjung sangat efektif.	Penelitian menggunakan data buku pada perpustakaan AMIK STIEKOM Sumatra Utara dengan menerapkan Algoritma *Brute Force* pada sistem.
5.	Implementasi Algoritma *Zhu-Takaoka* Pada Aplikasi Kamus Istilah Musik Berbasis Android (2017) [10]	Implementasi Algoritma *Zhu-Takaoka* ke dalam kamus digital berbasis android.	Algoritma *Zhu-Takaoka* dan metode pengembangan *Waterfall*.	Algoritma *Zhu-Takaoka* dapat diterapkan didalam aplikasi dan berjalan tanpa *error*.	Penelitian menggunakan data istilah musik dengan menerapkan Algoritma *Zhu-Takaoka* pada sistem.

Berdasarkan Tabel 2.1. diatas, dapat disimpulkan bahwa dengan objek penelitian dan data yang berbeda, Algoritma *String Matcing* dapat diimplementasikan kedalam sebuah sistem aplikasi baik berbasis android maupuan desktop untuk proses pencarian kata atau *string*.

2.2. Dasar Teori

Sebagai bahan acuan dalam penelitian ini, ada beberapa dasar teori yang digunakan yaitu sebagai berikut:

2.2.1. Perpustakaan

Perpustakaan adalah sekumpulan koleksi yang terdiri dari bahan-bahan yang tertulis, tercetak ataupun grafis lainnya seperti film, *slide*, yang tersimpan di dalam ruangan atau gedung yang diatur dan diorganisasikan dengan sistem tertentu agar dapat digunakan untuk keperluan pembelajaran, penelitian, dan lain-lain [11].

Menurut UU Nomor 43 Tahun 2007 Pasal 1 menyebutkan bahwa yang dimaksud dengan perpustakaan ialah sebuah institusi pengelola koleksi karya tulis, karya cetak, dan/atau karya rekam secara profesional dengan sistem yang baku, guna memenuhi kebutuhan pendidikan, penelitian, pelestarian informasi, dan rekreasi para pemustaka [3].

Tugas utama perpustakaan secara khusus adalah sebagai penyedia layanan informasi yang mencakup layanan referensi, layanan penelusuran, layanan temu kembali informasi, layanan sirkulasi, konsultasi bibliografi, penyediaan fasilitas membaca, media pembelajaran yang nyaman. Perpustakaan juga menyediakan layanan pendidikan untuk pemakainya sebagai upaya meningkatkan wawasan dan keterampilan para pemustaka dalam menggunakan layanan, koleksi, dan fasilitas pendukung perpustakaan, khususnya pemanfaatan teknologi informasi dan komunikasi [9].

2.2.2. Katalog

Katalogisasi (*cataloging*) adalah suatu kegiatan atau proses pembuatan ringkasan dari bahan pustaka atau dokumen (buku, majalah, *file*, film, dll). Istilah ini terkadang juga meliputi klasifikasi bahan pustaka dan secara umum menyiapkan atau menyediakan bahan pustaka untuk digunakan pemakai, terkadang disebut juga dengan istilah pengindeksan (*indexing*). Katalog (*catalog*) adalah representasi ciri-ciri dari sebuah bahan pustaka atau dokumen (misalnya: judul, pengarang, deskripsi fisik, subyek, dll) dari koleksi perpustakaan yang merupakan ringkas bahan pustaka tersebut yang disusun secara sistematis [7].

Katalog sendiri berfungsi untuk menunjukan ketersediaan koleksi yang

dimiliki oleh suatu perpustakaan dengan proses penelusuran. Proses tersebut sangat penting untuk menghasilkan sebuah temuan atau informasi mengenai suatu bahan pustaka yang relevan dan akurat. Pada penelusuran informasi, jumlah dokumen relevan yang ditelusuri akan dipengaruhi oleh jumlah kata kunci yang digunakan [12].

2.2.3. Android

Teknologi perangkat mobile sangat berkembang pesat, didukung dengan berbagai sistem operasi yang digunakan di dalamnya, sistem operasi android merupakan salah satu yang paling populer saat ini [13].

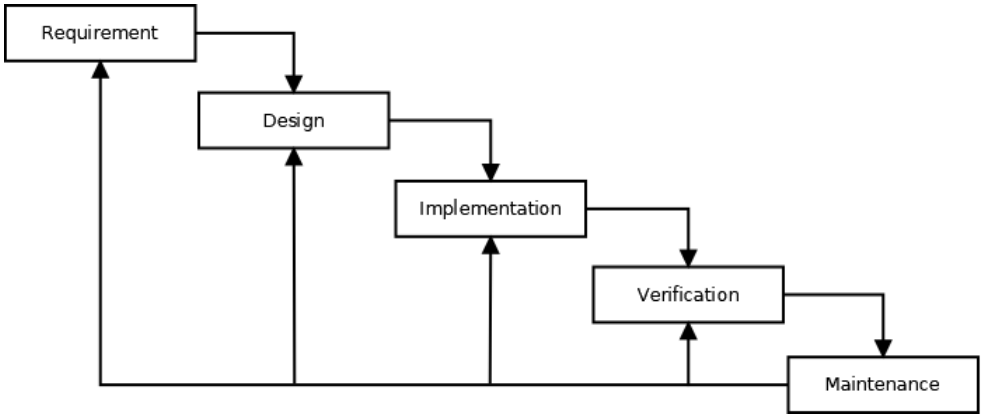
Android adalah sebuah sistem operasi yang digunakan pada perangkat *mobile* yang terdiri dari sistem operasi, *middleware*, dan aplikasi kunci yang dirilis oleh Google sehingga android mencakup keseluruhan aplikasi, mulai dari sistem operasi sampai pada pengembangan aplikasi itu sendiri. Sistem operasi android banyak digunakan pada perangkat *smartphone* dikarenakan sistem operasi android bersifat *open source* yang memungkinkan pengguna untuk mengkustomisasi sendiri [14][15].

2.2.4. Model Waterfall

Metode yang digunakan adalah metode air terjun atau yang sering disebut metode *waterfall* sering disebut juga siklus hidup klasik *(classic life cycle)*, dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna melalui tahapan- tahapan analisis *(requirement)*, desain *(design)*, implementasi *(implementation)*, serta pengujian sistem *(verification)*, yang diakhiri dengan pemeliharaan sistem *(Maintenance)* [16][17].

Berdasarkan banyak penelitian dijelaskan bahwa metode ini adalah metode yang digunakan untuk menjelaskan fakta dan fenomena objek yang diteliti berdasarkan pandangan peneliti. Dapat juga dikatakan metode penelitian deskriptif adalah penjelasan tentang suatu keadaan fakta yang dikaji berdasarkan perspektif peneliti [18].

Alur dari metode *waterfall* dapat dilihat seperti penjelasan dan Gambar 2.1 dibawah ini:



Gambar 2. 1. Fase Model Waterfall [16]

a. Analisa Persyaratan

Analisa awal yang meliputi seluruh kebutuhan sistem yang diperlukan, meliputi kegunaan dari sistem yang akan dibuat dan batasan-batasannya. Informasi analisis sistem dapat diperoleh melalui jurnal, wawancara atau diskusi dengan para ahli tentang kebutuhan *software* yang sekiranya diperlukan nantinya.

b. Desain Sistem

Desain sistem bertujuan untuk memberikan gambaran dari sistem yang akan dibuat dimulai dari bagaimana tampilannya, sehingga dapat membantu dalam merumuskan spesifikasi kebutuhan *hardware* dan arsitektur sistem secara keseluruhan.

c. Implementasi

Implementasi merupakan tahap memulai proses pemrograman *software* dengan salah satu strateginya adalah memecah menjadi modul-modul kecil sehingga dalam proses pengerjaannya lebih mudah dalam melakukan pemeriksaan terhadap setiap modul yang telah dibuat apakah sudah sesuai yang diinginkan atau belum.

d. Pengujian Sistem

Tahap pengujian sistem berfokus pada malakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian terhadap sistem apakah sudah berjalan sesuai desain dengan baik atau tidak.

e. Pemeliharaan Sistem

Pemeliharaan sistem yang dilakukan adalah melihat dan mengamati *software* yang sudah berjalan apakah ada kesalahan didalamnya yang tidak ditemukan ditahap sebelumnya, dan melakukan pembaharuan konten yang mungkin dibutuhkan oleh *user* dari *software* tersebut.

2.2.5. Algoritma String Matching

String matching adalah suatu proses pencarian semua kemunculan *query* atau *pattern* ke dalam *string* yang lebih panjang. Proses pencocokan *string* merupakan bagian utama dalam proses pencarian *string,* proses ini memegang peranan penting untuk mendapatkan dokumen yang sesuai dengan kebutuhan. Algoritma *String Matching* menjadi salah satu algoritma yang sering digunakan proses pencarian kata yang diinginkan [19][20][21]. Berikut beberapa macam algoritma *string matching,* antara lain:

a) Algoritma *Boyer-Moore*

Algoritma *Boyer-Moore* dikembangkan oleh Bob Boyer dan J Strother Moore pada tahun 1977. Pada proses pencarian *string*, Algoritma *Boyer-Moore* membaca karakter-karakter dari kanan ke kiri. Pada saat dimana jumlah karakter pada *pattern* lebih sedikit dari pada jumlah karakter pada *text*, maka Algoritma *Boyer-Moore* menggunakan dua buah fungsi *precomputed* yang disebut *good-suffix-shift* yang bertujuan ketika terdapat kejadian dimana terdapat pengulangan karakter-karakter pada *pattern* [7].

Algoritma *Boyer-Moore* mempunyai empat konsep dasar di dalam proses pencarian *string*, yaitu:

1. *Preprocessing*
2. *Right-to-left-rule*
3. *Bad-character-rule*
4. *Good-suffix-rule*

Tahap *bad-character preprocessing* dan *good-suffix prepocessing* pada Algortima *Boyer-Moore* disebut *Precomputation.* Prinsip dasar yang pertama dari Algoritma *Boyer-Moore* adalah melakukan perbandingan antara *pattern* yang dicari dengan *text* yang dilakukan dari arah kanan ke kiri. Perbandingan ini diawali dengan mambandingkan antara karakter dari *pattern* paling kanan dengan *text*. Apabila terjadi kecocokkan, maka perbandingan akan dilanjutkan dengan karakter yang disebelah kiri dari yang dibandingkan sampai semua karakter dari *pattern.* Apabila terjadi ketidakcocokkan maka akan dilakukan pergeseran yang ditentukan oleh *Precomputation*, yaitu dua fungsi pergeseran *bad character shift* dan *good character shift.* Aturan *bad character shift* berfungsi untuk menghindari pengulangan perbandingan yang gagal dari suatu karakter dalam *text* dengan *pattern.* Sedangkan aturan *good character shift* berfungsi untuk menangani kasus apabila terdapat pengulangan karakter pada *pattern* [7].

b) Algoritma *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt* dikembangkan oleh D.E. Knuth, J.H. Morris, dan V.R. Pratt. Algoritma ini merupakan jenis *extract string matching algorithm* yang melakukan pencocokan *string* dengan tepat sesuai dengan susunan karakter [8]. Apabila terjadi ketidakcocokan saat *pattern* sejajar dengan *text*[i..i+n-1], maka dapat disimpulkan ketidakcocokan pertama terjadi di antara *text*[i+j] dan *pattern*[j], dengan i<j<n. Berarti, *text*[i..i+j] = *pattern*[0..j+1] dan a = *text*[i+j] tidak sama dengan b = *pattern*[j] saat terjadi pergeseran [22].

Pencocokan *string* akan berjalan secara efisien apabila terdapat tabel yang menentukan berapa panjang jarak pergeseran seandainya terdeteksi ketidakcocokan di karakter ke-j dari *pattern.* Tabel itu harus memiliki nilai *next*[j] yang merupakan posisi karakter *pattern*[j] setelah digeser, sehingga pergeseran *pattern* secara relatif sebesar j-*next*[j] terhadap *text* [22].

Secara sistematis, proses langkah-langkah Algoritma *Knuth-Morris-Pratt* pada saat pencocokan *string* adalah sebagai berikut [22]:

1. Algoritma *Knuth-Morris-Pratt* mulai mencocokan *pattern* pada awal *text*.
2. Algoritma ini akan mencocokan setiap karakter *pattern* dengan karakter pada *text* yang bersesuaian dari kiri ke kanan sampai salah satu kondisi berikut terpenuhi:
   (a) Karakter dari *pattern* dan *text* yang dibandingkan tidak cocok *(missmatch).*
   (b) Semua karakter dari *pattern* cocok. Kemudian algoritma akan memberitahukan posisi penemuan tersebut.
3. *Pattern* digeser berdasarkan tabel *next,* dan menghitung langkah 2 sampai *pattern* berada di ujung *text*.

c) Algoritma *Brute-Force*

Algoritma *Brute-Force* merupakan algoritma yang digunakan untuk mencocokan *pattern* dengan semua *string* untuk menemukan keberadaan *string* target, pemecahan masalah yang sangat sederhana, langsung, dan jelas dapat dilakukan berdasarkan algoritma ini. Secara rinci langkah-langkah Algoritma *Brute- Force* untuk mencocokan *string* adalah [9]:

1. Pencocokan *pattern* dimulai dari awal *text.*
2. Algoritma *Brute-Force* akan mencocokan setiap karakter *pattern* dengan karakter pada *text* yang bersesuaian yang dimulai dari kiri ke kanan, hingga salah satu kondisi berikut terpenuhi:

(a) Karakter di *pattern* yang dibandingkan cocok, maka pencarian selesai.
(b) Berdasarkan hasil pencocokan, apabila ditemukan ketidakcocokan antara *pattern* dengan *text*, maka pencarian belum berhasil.

3. Algoritma *Brute-Force* akan terus menggeser *pattern* sebesar satu ke kanan, dan mengulangi langkah ke-2 sampai *pattern* berada di ujung *text.*

d) Algoritma *Horspool*

Algortima *Horspool* merupakan algoritma turunan dari Algoritma *Boyer-Moore* yang melakukan pencocokan *string* dari karakter paling kanan dari *pattern*, algoritma ini dibuat oleh R. Nigel Horspool tepatnya pada tahun 1980. Apabila terjadi ketidakcocokan antara karakter *pattern* dengan karakter *text*, Algoritma *horspool* menggunakan *bad-match* secara berulang hingga seluruh karakter *pattern* memiliki kecocokan terhadap *text*. Melakukan pencocokan *string* menggunakan Algoritma *Horspool* terdapat dua tahap, yaitu tahap *praprocessing* dan tahap *searching* [4][6].

Algoritma *Horspool* mencari *pattern* dari kiri ke kanan dan untuk *shif value* berdasarkan ukuran dari *pattern* yang dicari dalam *bad character shift* tabel. Algoritma *Horspool* tidak efisien untuk *text* yang lebih pendek dibandingkan *pattern*. Apabila *text* yang lebih panjang dibandingkan dengan *pattern* yang dicari, Algoritma *Horspool* menjadi sangat berguna. Algortima *Horspool* hanya menggunakan pergeseran *bad-character* pada bagian kanan karakter [5].

e) Algoritma *Zhu-Takaoka*

Algoritma *Zhu-Takaoka* merupakan algoritma pencocokan *string* modifikasi dari Algoritma *Boyer-Moore* yang dibuat oleh Boyer R.S dan Moore J.S pada tahun 1977. Algoritma *Zhu-Takaoka* dikembangkan oleh Zhu Rui Feng dan Tadao Takaoka yang dipublikasikan pada tahun 1986. Ide dibalik algoritma ini adalah dengan melakukan pencocokan karakter dari kanan ke kiri, sehingga akan lebih banyak informasi yang didapat. Algoritma *Zhu-Takaoka* menggunakan array dua dimensi untuk menghitung nilai pergeseran dan melakukan pencocokan dari kanan ke kiri [10].

Algoritma *Zhu-Takaoka* memiliki ciri-ciri yang sama dengan Algoritma *Boyer- Moore* dalam proses pencarian *string*, yaitu adanya

preprocessing, right-to-left scan, bad character rule, dan *good suffixes rule.* Dua algoritma ini tetap memiliki perbedaan, yaitu terletak pada tahap penentuan *bad character rule.* Pada Algoritma *Boyer- Moore* menggunakan *array* satu dimensi, sedangkan pada Algoritma *Zhu-Takaoka* dimodifikasi menjadi *array* dua dimensi [23].

Tahap *preprocessing* Algoritma *Zhu-Takaoka* membangun tabel *bad character* dua dimensi karena algoritma tersebut melakukan perhitungan 2 karakter. Kompleksitas waktu dari tahap *preprocessing* adalah O(m+o2) dan kompleksitas waktu fase pencarian adalah O(mn). Proses inti Algoritma *Zhu-Takaoka* yaitu melakukan pencarian dengan teknik *right-to-left scan rule.* Teknik ini membanding-kan *pattern* yang dicari dengan sumber *text* dimulai dari kanan ke kiri [14]. Menjalankan prosedur *preZTBc (Preprocessing Zhu-Takaoka Bad Character)* dan *preBmGs (Preprocessing Boyer-Moore Good Suffixes)* untuk mendapatkan inisialisasi [23].

1. Menjalankan prosedur *preZTBc.* Fungsi dari prosedur ini adalah untuk menentukan berapa besar pergeseran yang dibutuhkan untuk mencapai karakter tertentu pada *pattern* dari dua karakter *pattern* terakhir/terkanan. Hasil dari prosedur *preZTBc* disimpan pada tabel *ztBc.*
2. Sebelum menjalankan prosedur *preBmGs,* prosedur *suffix* dijalankan terlebih dulu pada *pattern.* Fungsi dari prosedur *suffix* adalah memeriksa kecocokan sejumlah karakter yang dimulai dari karakter paling kanan dengan sejumlah karakter yang dimulai dari setiap karakter sebelah kiri dari karakter tersebut. Hasil dari prosedur *suffix* disimpan pada tabel *suff.* Jadi *suff*[i] mencatat panjang dari *suffix* yang cocok dengan segmen dari *pattern* yang diakhiri karakter ke-i.
3. Berdasarkan prosedur *preBmGs,* dapat diketahui berapa banyak langkah pada *pattern* dari sebuah segmen ke segmen lain yang letaknya disebelah kiri dengan karakter. Prosedur *preBmGs* menggunakan tabel *suff* untuk mengetahui semua pasangan segmen yang sama. Hasil dari prosedur *preBmGs* disimpan pada tabel *bmGs.* Proses pencarian *string* dilakukan dengan menggunakan hasil dari prosedur *preBmGc* dan *preBmGs,* yaitu tabel *bmBc* dan *BmGs.*

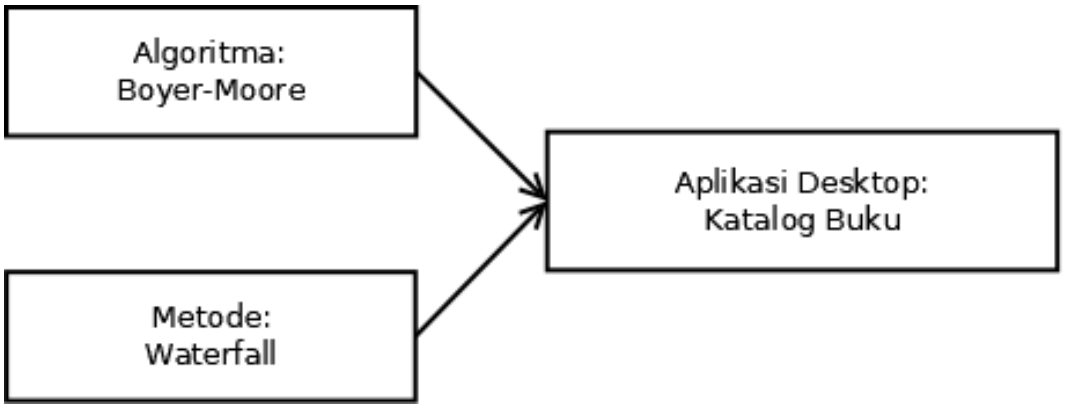
2.3. Kerangka Pemikiran

Proses pencocokan *string* diperlukan suatu algoritma *String Matching* untuk mempercepat proses pencarian kata, supaya hasil dari proses pencarian sebuah

string sesuai dengan kebutuhan informasi. Karena proses pencocokan *string* merupakan kunci dari proses pencarian *string* untuk memperoleh kata yang diinginkan menjadi lebih efektif dan efisien.

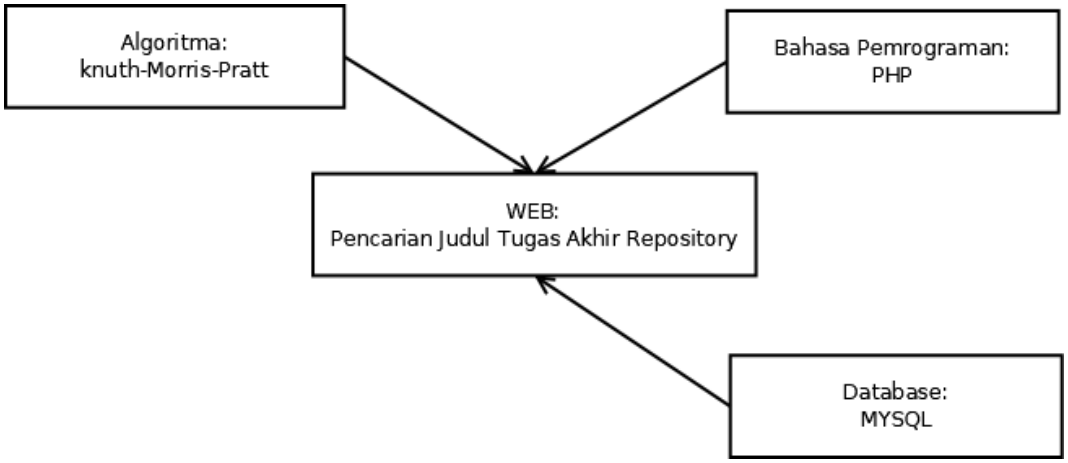
Berdasarkan referensi penelitian sebelumnya dan dasar teori yang relevan dengan penelitian ini, maka diperoleh beberapa kesimpulan dan kerangka pemikiran sebagai berikut:

1. Penelitian [7], Algoritma *Boyer-Moore* dianggap sebagai algoritma pencocokan *string* yang paling efisien pada penggunaan biasa, karena Algoritma *Boyer-Moore* telah menjadi standar untuk pencarian *string.* Ide dibalik algoritma ini adalah pencocokan karakter dimulai dari kanan ke kiri, maka akan lebih banyak informasi yang didapat. Kerangka pemikiran dari penelitian ini dapat penulis gambarkan seperti pada Gambar 2.2 dibawah ini:



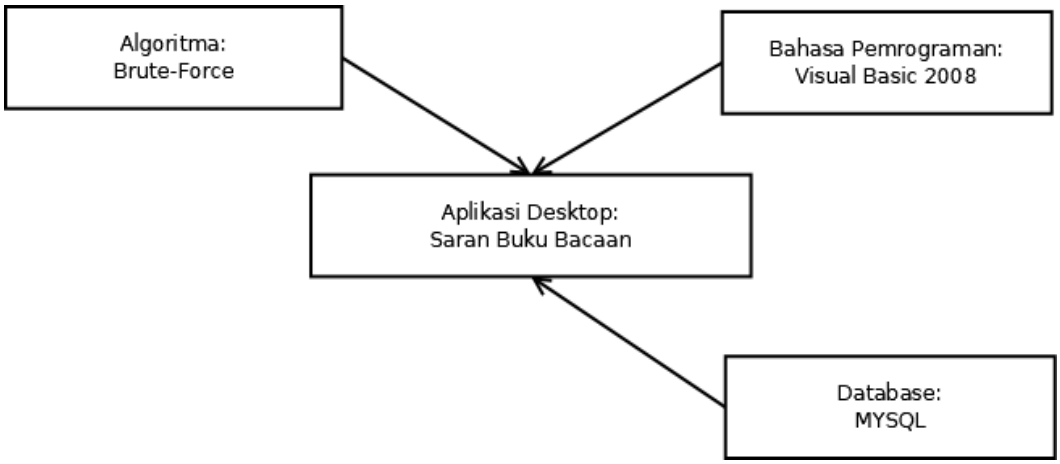
Gambar 2. 2. Kerangka pemikiran penelitian [7]

2. Penelitian [8], setelah diuji coba memiliki hasil pengujian menunjukan rata-rata performa dari Algoritma *Knuth-Morris-Pratt* dalam proses pencarian *string* di *form* pencarian adalah 0.0138 detik. Hal tersebut menunjukan bahwa Algoritma *Knuth-Morris-Pratt* cukup cepat dan optimal dalam menentukan hasil pencarian kata pada aplikasi *repository* tugas akhir. Kerangka pemikiran dari penelitian ini dapat penulis gambarkan seperti pada Gambar 2.3 dibawah ini:



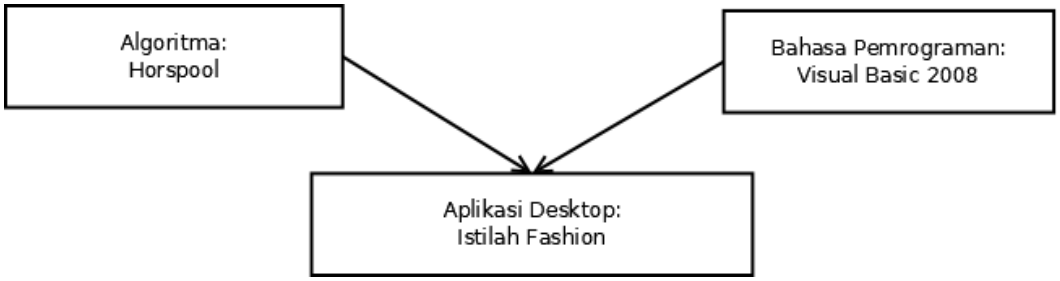
Gambar 2. 3. Kerangka pemikiran penelitian [8]

3. Penelitian [9], Algoritma *Brute-Force* memiliki beberapa kelebihan diantaranya sederhana dan mudah dimengerti, dapat digunakan untuk memecahkan hampir dari sebagian besar masalah, dan cocok untuk proses pencarian, pengurutan, atau pencarian *string*. Kerangka pemikiran dari penelitian ini dapat penulis gambarkan seperti pada Gambar 2.4 dibawah ini:



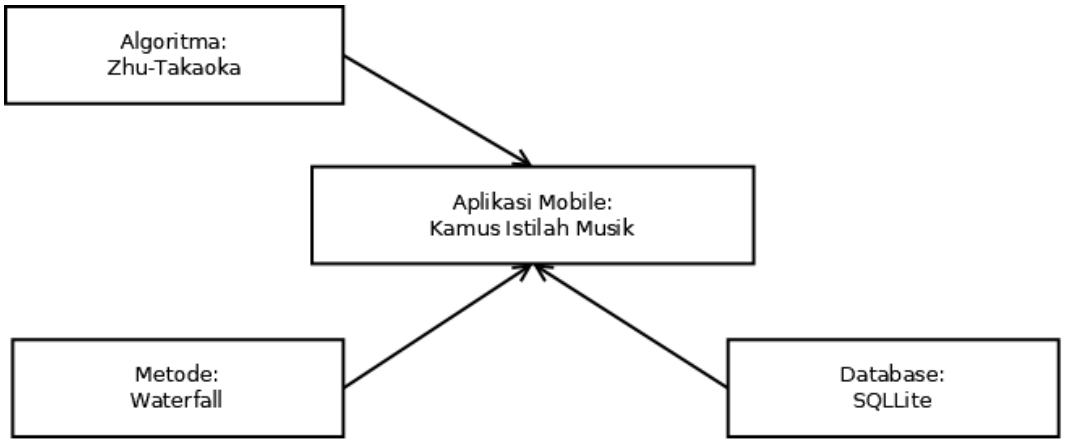
Gambar 2. 4. Kerangka pemikiran penelitian [9]

4. Penelitian [6], Algoritma *Horspool* dapat menemukan kecocokan *string* dengan cepat, dengan menggunakan nilai pergeseran *bad match* Algoritma *Horspool* dapat menemukan kecocokan *string* dalam tahap ketiga atau tahap pergeseran ketiga. Kerangka pemikiran dari penelitian ini dapat penulis gambarkan seperti pada Gambar 2.5 dibawah ini:



Gambar 2. 5. Kerangka pemikiran penelitian [6]

5. Penelitian [10], Algoritma *Zhu-Takaoka* merupakan algoritma modifikasi dari Algortima *Boyer-Moore* dimana pada proses *bad character* pada Algoritma *Boyer-Moore* menggunakan *array* satu dimensi sedangkan pada Algoritma *Zhu-Takaoka* menggunakan *array* dua dimensi dengan kompleksitas waktu O(m+o2) dari tahap *prepocessing* dan kompleksitas waktu O(mn) dari fase pencarian. Kerangka pemikiran dari penelitian ini dapat penulis gambarkan seperti pada Gambar 2.6 dibawah ini:

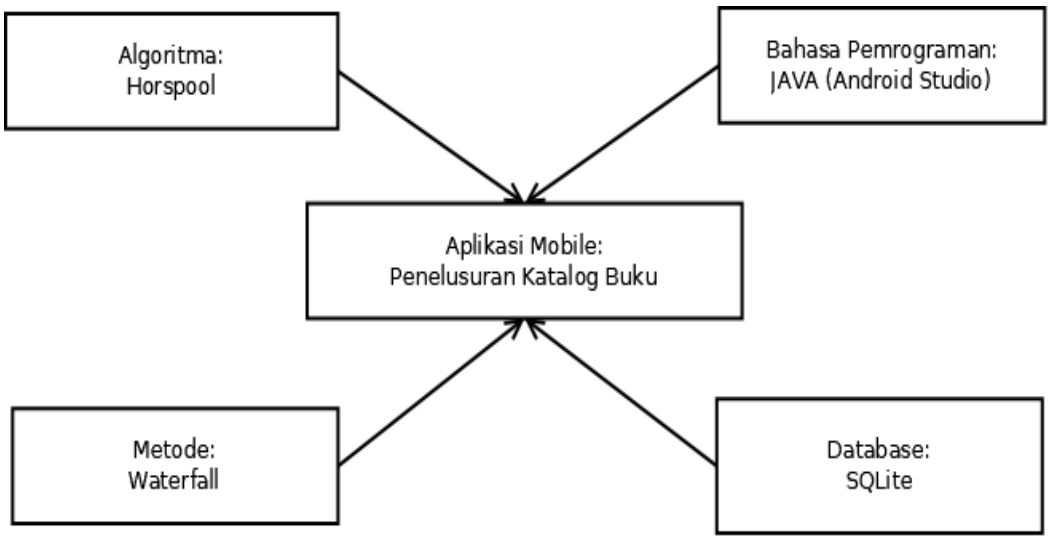


Gambar 2. 6. Kerangka pemikiran penelitian [10]

Berdasarkan beberapa referensi-referensi diatas, juga ada beberapa penelitian seperti [24] yang membandingkan Algoritma *Brute- Force* dan Algoritma *Boyer-Moore* menggunakan Metode Perbandingan Eksponensial (MPE) yang mana dalam perhitungannya, jumlah iterasi dan jumlah huruf dari setiap proses menjadi kriteria pembandingnya dan penelitian [25] yang melakukan perbandingan antara Algoritma *Boyer-Moore* dan Algoritma *Knuth-Morris-Pratt* menggunakan Metode Perbandingan Eksponensial (MPE) yang mana dalam perhitungannya, jumlah memori yang digunakan dan besarnya waktu yang dibutuhkan dari setiap proses pencocokan menjadi kriteria pembandingnya, menyatakan bahwa Algoritma *Boyer-Moore* merupakan algoritma yang tercepat dalam melakukan pencarian. Pada penelitian [10] menggunakan Algoritma *Zhu-Takaoka* yang mana merupakan modifikasi dari Algoritma *Boyer-Moore* menyatakan Algoritma *Zhu-Takaoka* dapan diterapkan dalam perancangan

aplikasi berbasis android dan berjalan tanpa adanya *error*. Pada penelitian [5] dalam penelitiannya membandingkan Algoritma *Horspool* dan Algoritma *Zhu-Takaoka* menyatakan bahwa Algoritma *Horspool* lebih cepat 19.82845% pada uji coba pertama menggunakan 50 *file text* kelipatan 1000 kata dengan *pattern* yang sama dan 15.9442% pada uji coba kedua dengan *file text* 7000 kata. Berdasarkan beberapa penelitian diatas, dapat disimpulkan bahwa Algoritma *Horspool* lebih cepat daripada Algoritma *Zhu-Takaoka.*

Dikarenakan studi kasus penelitian ini akan membahas tentang penelusuran katalog buku pada perpustakaan yang mana akan menerapkan salah satu algoritma *String Matching* diatas, maka dalam penelitian ini akan menerapkan Algoritma *Horspool* sesuai dengan studi literatur diatas, yang mana menurut penelitian [5] pada tahun 2017 dan penelitian [26] pada tahun 2020 menyatakan bahwa Algoritma *Horspool* merupakan algoritma tercepat dalam proses pencarian kata. Metode yang digunakan pada penelitian ini yaitu metode *waterfall*, karena pada penelitian [7], [8] dan [10] menerapkan metode ini sebagai metode pengembangan sistemnya. Media penyimpanan data yang akan digunakan pada penelitian ini adalah *database* SQLite dan untuk membangun aplikasi android dari penelitian ini akan menggunakan *software* Android Studio dengan bahasa pemrograman JAVA. Kerangka pemikiran pada penelitian ini dapat penulis gambarkan seperti pada Gambar 2.7 dibawah ini:



Gambar 2. 7. Kerangka pemikiran penelitian Aplikasi Penelusuran Katalog Buku