Perbandingan Algoritma Boyer-Moore dan Algoritma Rabin-Karp Terhadap Kode Pos Wilayah Aceh

Dwi Riesky Chandra Wiradhika[1], Yudha Nurdin[2], Fardian[3]

[123]*Jurusan Teknik Elektro dan Komputer, Universitas Syiah Kuala*
Jl. Tgk. Syech Abdurrauf No.7, Darussalam, Banda Aceh, 23111, Aceh, Indonesia
[1]cdwiriesky@mhs.unsyiah.ac.id
[3]Fardian@unsyiah.ac.id
[2]Yudha Nurdin@Unsyiah.ac.id

Abstrak – **Kode Pos adalah serangkaian angka dan/atau huruf yang ditambahkan pada alamat surat untuk mempermudah proses pemilahan surat. Di negara lain, kode pos lebih dikenal dengan sebutan ZIP Code. Kode ini digunakan untuk efisiensi dan mempermudah pengiriman surat maupun paket dari dan ke berbagai wilayah di sebuah negara. ZIP sendiri kepanjangan dari Zone Improvement Plan. ZIP Code atau kode pos ini biasanya terdiri dari beberapa angka yang menunjukkan kode dari sebuah area. Pada penelitian ini, dirancang sebuah aplikasi pencarian dengan proses string matching. String matching sendiri adalah proses pencarian semua kemunculan query yang selanjutnya disebut pattern ke dalam string yang lebih panjang. Perancangan aplikasi kode pos untuk wilayah Aceh berbasis android menggunakan algoritma Boyer-Moore dan Rabin-Karp. Hasil dari pengujian dan perbandingan dari kedua algoritma yang direpresentasikan dalam kompleksitas, yaitu : θ(mn). Dari penelitian yang dilakukan, diperoleh hasil berupa nilai real running time algoritma Boyer-Moore memiliki rata-rata penemuan string : 5,53 ms dan algortima Rabin-Karp memiliki rata-rata penemuan string : 6,96 ms.**

Kata Kunci *— **Kode Pos, String Matching, Algoritma, Boyer-Moore, Rabin-Karp***

I. PENDAHULUAN

. Layanan Pos adalah bagian dari sistem Pos yaitu sebuah metode yang digunakan untuk mengirimkan informasi atau suatu objek, di mana untuk dokumen tertulis biasanya dikirimkan dengan amplop tertutup atau berupa paket untuk benda-benda yang lain, pengirimannya mampu menjangkau seluruh wilayah di dunia. Pada dasarnya, sistem pelayanan pos bisa dilakukan oleh *public* ataupun *private*. Namun, sejak pertengahan abad ke 19, sistem per-pos-an secara umum menjadi ranah yang harus dikuasai negara (monopoli) dengan biaya pada artikel prabayar. Bukti dari pembayaran dilihat dari sebuah prangko tempel yang biasa direkatkan di sudut kanan atas, tetapi ongkos permeter juga dikenakan untuk pengiriman massal [1].

Di Indonesia, layanan Pos masih digunakan dalam kegiatan pengiriman paket dan lain lain di berbagai daerah. Layanan Pos Indonesia menggunakan kode pos sebagai kode inisial setiap daerah. Kode pos merupakan serangkaian angka dan/atau huruf yang ditambahkan pada alamat surat untuk mempermudah proses pemilahan surat. Di negara lain, kode pos lebih dikenal dengan sebutan *ZIP Code*. Kode ini digunakan untuk efisiensi dan mempermudah pengiriman surat maupun paket dari dan ke berbagai wilayah di sebuah negara. *ZIP* sendiri kepanjangan dari *Zone Improvement Plan*. *ZIP Code* atau kode pos ini biasanya terdiri dari beberapa angka yang menunjukkan kode dari sebuah area. Umumnya, kode pos terdiri dari lima angka. Namun, ada juga yang memiliki angka tambahan untuk lokasi yang lebih detail [1].

Di Aceh sendiri terdapat 23 Kabupaten / Kota, 289 Kecamatan, dan 6497 Desa, yang masing - masing memiliki kode pos tersendiri. Untuk Aceh, *range* kode pos nya adalah 23111 – 24794. Dan masing – masing Kabupaten / Kota dan Desa juga memiliki *range* kode tersendiri [2].

Untuk mempermudah pencarian kode pos yang amat banyak tersebut, maka penulis mencoba merancang sebuah aplikasi pencarian dengan algoritma *string matching*. Menurut Rosaria (2015:1) algoritma string matching adalah sebuah algoritma yang digunakan dalam pencocokkan suatu pola kata tertentu terhadap suatu kalimat atau teks panjang [3].

Algoritma *string matching* sendiri dapat dilakukan dengan beberapa cara tertentu, antara lain menggunakan algoritma Boyer-Moore dan algoritma Rabin-Karp. Cara kerja algoritma Boyer-Moore menggunakan metode pencocokan *string* dari kanan ke kiri yaitu men-scan karakter *pattern* dari kanan ke kiri dimulai dari karakter paling kanan [4]. Sedangkan algoritma Rabin-Karp menggunakan *hashing* untuk menemukan sebuah *substring* dalam sebuah teks. *Hashing* adalah metode yang menggunakan fungsi hash untuk mengubah suatu jenis data menjadi beberapa bilangan bulat sederhana. Algoritma Rabin-Karp tidak bertujuan menemukan string yang cocok dengan *string* masukan, melainkan menemukan pola (*pattern*) yang sekiranya sesuai dengan teks masukan. Algoritma Rabin-Karp menghasilkan efisiensi waktu yang baik dalam

mendeteksi string yang memiliki lebih dari satu pola [5]. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi [6].

Algoritma Boyer-Moore memiliki efisiensi waktu yang cukup baik dalam memproses sebuah *pattern* yang berukuran panjang, karena lompatan karakternya yang cukup besar. Algoritma Rabin-Karp memiliki beberapa kelebihan dalam melakukan pola pencocokkan string, yaitu: tidak banyak menggunakan kapasitas memory, proses dalam mencari pola *pattern*-nya yang tidak terlalu rumit. Karena kedua algoritma ini memiliki kompleksitas algoritma yang sama, namun penulis akan membandingkan nilai *real running time* dari kedua algoritma ini.

Dalam hal ini, penulis ingin melakukan pengujian nilai *real running time* dari kedua algoritma, dimana nilai *real running time* akan menjadi parameter pengujian. Sehingga, didapatlah algoritma mana yang lebih efisien untuk digunakan. Aplikasi ini nantinya dapat digunakan untuk para pengembang aplikasi, guna menguji efisiensi untuk sebuah algoritma.

II. TINJAUAN PUSTAKA

A. Android

Android adalah sistem operasi untuk perangkat mobile berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka [7].

B. Android Studio

Android Studio merupakan *IDE* untuk Android *Development* yang diperkenalkan google pada kegiatan Google I/O 2013. Android Studio ialah pengembangan dari *Eclipde IDE*, serta terbuat bersumber pada *IDE Java* terkenal, yakni *InteliJ IDEa*. Android Studio ialah *IDE* resmi untuk pengembangan aplikasi Android.

Menurut Imaduddin Al Fikri, Darlis Herumurti, dan Ridho Rahman H, dalam Jurnalnya mengatakan bahwa "Android Studio ialah suatu *Integrated Development Environment* (*IDE*) khusus untuk membangun aplikasi yang berjalan pada platform android. Android studio ini berbasis pada *IntelliJ IDEA*, suatu *IDE* untuk bahasa pemrograman *Java*. Bahasa pemrograman utama yang digunakan yakni *Java*, sedangkan untuk membuat tampilan maupun layout, digunakan bahasa *XML*. Android studio pula terintegrasi dengan Android *Software Development Kit (SDK)* untuk *deploy* ke piranti android" [8].

C. Mesin Pencari (Search Engine)

Mesin pencari (*search engine*) adalah salah satu program komputer yang di rancang khusus untuk membantu seseorang menemukan file yang disimpan dalam komputer, misalnya dalam sebuah *web server* umum di *web* atau komputer sendiri. Mesin pencari memungkinkan kita untuk meminta content media dengan kriteria yang spesifik (biasanya berisikan frase atau kata yang kita inginkan) dan memperoleh daftar *file* yang memenuhi kriteria tersebut. Mesin pencari biasanya menggunakan indeks untuk mencari *file* setelah pengguna memasukan kriteria pencarian. Mesin pencari yang akan dibahas adalah mesin pencari khusus yang digunakan untuk mencari informasi di dalam database lokal. Untuk memudahkan dan mempercepat pencarian, mesin pencari mempunyai metode pencarian tertentu yang sering disebut algoritma.

Adapun struktur umum sebuah mesin pencari adalah sebagai berikut:

- Kotak teks pencari, kotak ini digunakan sebagai tempat memasukan kata kunci yang akan dijadikan acuan dilakukan pencarian.
- Tombol pencari, tombol ini yang akan menjalankan perintah pencarian [7].

D. Algoritma String Matching

Algoritma pencocokan *string* merupakan operasi penting yang umumnya terdapat di banyak aplikasi. Algoritma ini merupakan masalah penting dalam pengolahan teks dan umumnya digunakan untuk menemukan satu pola dimensi pada teks. Algoritma pencocokan *string* (*String Matching*) juga disebut sebagai algoritma pencarian *string* adalah kelas yang dominan dari algoritma *string* yang bertujuan untuk menemukan satu atau semua kejadian dari suatu *string* dalam kelompok yang lebih besar dari teks [9]. Pencocokan *string* atau *string matching* merupakan bagian penting dari sebuah proses pencarian *string* (*string searching*) dalam sebuah dokumen [4]. Hasil dari pencarian sebuah *string* dalam dokumen tergantung dari teknik atau cara pencocokan *string* yang digunakan. Pencocokan string (string matching) secara garis besar dapat dibedakan menjadi dua, yaitu *Exact string matching* dan *Inexact string matching.*

1) Exact string matching: merupakan pencocokan string secara tepat dengan susunan karakter dalam string yang dicocokkan memiliki jumlah maupun urutan karakter dalam string yang sama. Contohnya, kata open akan menunjukkan kecocokan hanya dengan kata open.

2) Inexact string matching atau *Fuzzy string matching*: merupakan pencocokan string secara samar, maksudnya pencocokan string dimana string yang dicocokkan memiliki kemiripan dimana keduanya memiliki susunan karakter yang berbeda (mungkin jumlah atau urutannya) tetapi string-string tersebut memiliki kemiripan baik kemiripan tekstual/penulisan (*approximate* string matching) atau kemiripan ucapan (*phonetic string matching*).

E. Algoritma Boyer-Moore

Ide utama dari algoritma Boyer-Moore adalah dengan melakukan pencocokan dari paling kanan *string* yang dicari. Dengan menggunakan algoritma ini, secara rata-rata proses pencarian akan lebih cepat dibandingkan dengan proses pencarian lainnya. Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokkan karakter dari kanan, dan

bukan dari kiri, maka akan lebih banyak informasi yang didapat [4].

Berikut *Pseudocode* langkah-langkah pencarian Algoritma Boyer-Moore:

- *Pseudocode Bad-Character Shift (Occurance Heuristic)*
- *Pseudocode Good-Suffix Shift (Match Heuristic)*
- *Pseudocode perhitungan Suffix*
- *Pseudocode pencarian Boyer-Moore*

```
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
```

Gambar *3 Pseudocode Bad-Character Shift*

```
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
```

Gambar 4 *Pseudocode Good-Suffix Shift*

```
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
```

Gambar 5 *Pseudocode* Perhitungan *Suffix*

```
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;
       shift := bmGs[0]
    else
       bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
       bmGsShift:= BmGs[j]
       shift:= max(bmBcShift, bmGsShift)
    endif
    i:= i+shift
endwhile
```

Gambar 6 *Pseudocode* pencarian *Boyer-Moore*

F. Algoritma Rabin-Karp

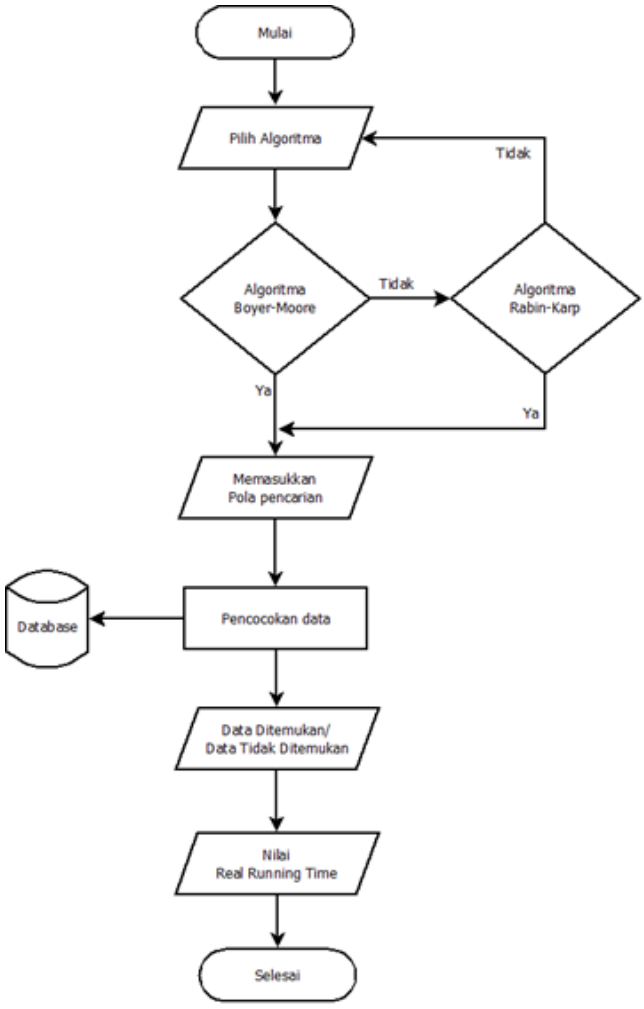
Algoritma Rabin-Karp diciptakan oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 yang menggunakan fungsi *hashing* untuk menemukan pattern di dalam *string* teks [11]. Fungsi *hashing* yaitu menyediakan metode sederhana untuk menghindari perbandingan jumlah karakter yang kuadratik di dalam banyak kasus atau situasi. Dari pada melakukan pemeriksaan terhadap setiap posisi dari teks ketika terjadi pencocokan pola, akan lebih baik dan efisien

melakukan pemeriksaan hanya jika teks yang sedang proses memiliki kemiripan seperti pada *pattern*.

Untuk melakukan pengecekan kemiripan antara dua kata ini digunakan fungsi *hash*. Fungsi *hash* menyimpan bentuk string dalam bentuk lain yaitu enumerasi sehingga suatu *string* tertentu akan memiliki nilai enumerasinya yang unik. Gambar 7 merupakan*Pseudocode* dari algoritma Rabin-Karp.

```
function Rabin Karp(string s[1..n], string sub[1..m])
    hsub := hash(sub[1..m])
                for i from 1 to n-m+1
                    if hs = hsub
                        if s[i..i+m-1] = sub
                            return i
                    hs := hash(s[i+1..i+m])
                return not found
```

Gambar 7 Pseudocode Rabin-Karp



Gambar 8 Diagram Alir Pencarian String

G. Penelitian yang Relevan

Penelitian tentang algoritma Boyer-Moore pada tahun 2012 dengan judul "Studi Perbandingan Implementasi Algoritma Boyer-Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore Dalam Pencarian String" oleh Vina Sagita dan Maria Irmina Prasetiyowati dari Program Studi Teknik Informatika, Universitas Multimedia Nusantara, Tangerang. Penelitian ini membandingkan algoritma Boyer-Moore untuk mengetahui performa terutama di bidang waktu terhadap algoritma tersebut dengan membuat aplikasi *prototyping* menggunakan *Microsoft Visual Studio* yang mendukung pencarian algoritma Boyer-Moore [6].

Selanjutnya penelitian tentang Algoritma Boyer-Moore pada tahun 2014 dengan judul "Implementasi Algoritma Boyer-Moore Pada Aplikasi Kedokteran Berbasis Android" oleh Kencana Wulan Argakusumah dan Seng Hansun dari Program Studi Teknik Informatika, Universitas Multimedia Nusantara, Tangerang. Penelitian ini menguji performa kecepatan algoritma Boyer-Moore yang diimplementasikan ke dalam Aplikasi Kamus Kedokteran [10].

Kemudian Penelitian Algoritma Boyer-Moore Pada tahun 2016 dengan judul "Penerapan String Matching dengan Algoritma Boyer-Moore Pada Aplikasi Font Italic Untuk Deteksi Kata Asing" oleh Rohmat Indra Borman dan Agus Pratama pada Jurusan Teknik Informatika STMIK Teknokrat, Bandarlampung. Penelitian ini menerapkan algoritma Boyer-Moore untuk melakukan *String Matching* dalam menemukan kata asing. Pengujian dilakukan dengan cara memasukkan *file* yang ingin di periksa ke dalam aplikasi *Font Italic* yang telah dirancang untuk mengetahui jumlah kata asing dan juga waktu pencariannya [4].

Penelitian tentang Algoritma Rabin-Karp pada tahun 2012 dengan judul "Analisis Dan Implementasi Algoritma Rabin-Karp Dan Algoritma Stemming Nazief-Adriani Pada Sistem Pendeteksi Plagiat Dokumen Teks Berbahasa Indonesia" oleh Yosef Agung Wicaksono, Suyanto, dan Suyanto dari Fakultas Teknik Informatika, Universitas Telkom. Penelitian ini menggunakan algoritma Rabin-Karp dan algoritma Stemming Nazief-Adriani untuk mendeteksi plagiat dengan cara memasukkan dokumen yang akan diperiksa ke dalam system yang sudah dirancang [5].

Kemudian penelitian tentang Algoritma Rabin-Karp pada tahun 2017 dengan judul "Analisis Algoritma Rabin-Karp Pada Kamus Umum Berbasis Android" oleh Herriyance, Handrizal, dan Siti Dara Fadilla dari Program Studi S1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara. Penelitian ini dilakukan untuk mengetahui kecepatan algoritma Rabin-Karp dalam menemukan sebuah kata yang telah diimplementasikan di dalam aplikasi kamus umum berbasis android [11].

III. METODOLOGI PENELITIAN

A. Metode Penelitian

Penelitian ini menggunakan algoritma Boyer-Moore dan algoritma Rabin-Karp yaitu dengan menjalankan aplikasi

e-ISSN: 2252-7036
Vol.6 No.1 2021: 1-10

yang akan di rancang menggunakan kedua algoritma tersebut untuk mendeteksi perbandingan dari nilai real running time. Diagram alir pencarian string dapat dilihat pada gambar 8.

Pada tahap awal, terapkan kedua algoritma ini ke dalam aplikasi yang akan di rancang. Kemudian, dibuat *database* yang berisikan kode pos yang ada di wilayah Aceh untuk menjadi data sampel pengujian. Uji coba kedua Algoritma tersebut dilakukan dengan menggunakan data dari kode pos wilayah Aceh yang sudah diinput ke dalam *database*.

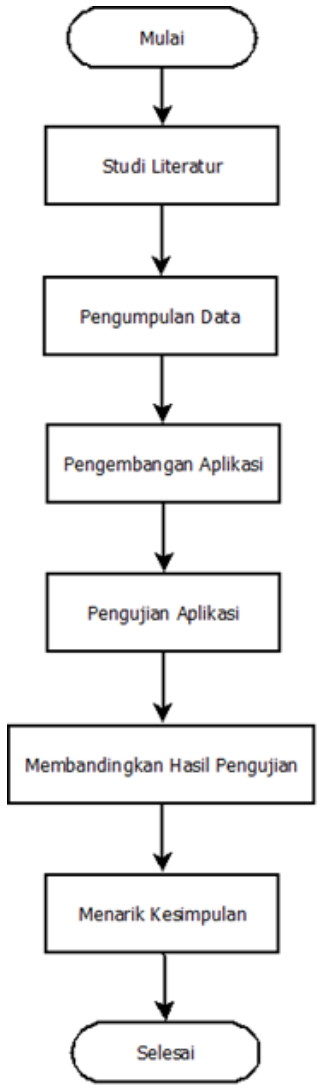
Percobaan pertama dilakukan untuk algoritma Boyer-Moore. Algoritma ini kemudian di uji dengan beberapa pencarian (*string matching*) terhadap kode pos yang sudah ada di dalam database. Setelah itu input kode pos yang akan di uji. Jika pencarian berhasil, berarti data yang diinput ada di dalam database dan kemudian akan muncul halaman yang menampilkan Nama Desa dan Kode Pos serta nilai *real running time*. Jika pencarian gagal, berarti data yang diinput tidak ada di dalam database, dan hasilnya akan keluar nilai real running time.

Percobaan kedua dilakukan untuk algoritma Rabin-Karp. Algoritma ini kemudian di uji dengan beberapa pencarian (*string matching*) terhadap kode pos yang sudah ada di dalam *database*. Setelah itu input kode pos yang akan di uji. Jika pencarian berhasil, berarti data yang diinput ada di dalam *database* dan kemudian akan muncul halaman yang menampilkan Nama Desa dan nilai *real running time*. Jika pencarian gagal, berarti data yang diinput tidak ada di dalam *database*, dan hasilnya akan keluar nilai *real running time*.

Angka / nilai dari setiap *real running time* dari kedua algoritma yang diuji kemudian dikumpulkan untuk dijadikan nilai perbandingan. Nilai ini akan menjadi parameter dalam pengujian untuk menentukan algoritma yang lebih efisien berdasarkan hasil nilai *real running time*.

B. Alur Penelitian

Alur penelitian ini digambarkan seperti pada gambar 9. Penelitian ini dimulai dengan melakukan studi literatur, dilanjutkan dengan tahap pengumpulan data. Berdasarkan data yang telah terkumpul, maka dilakukan pengembangan aplikasi. Setelah tahap pengembangan aplikasi selesai, aplikasi diuji dan hasil pengujian kemudian akan dibandingkan serta dilakukan analisa lebih lanjut untuk dapat ditarik kesimpulan.



Gambar 9 Diagram Alir Penelitian

C. Desain Model

1) Tampilan Halaman Utama/Home Screen: Halaman Utama/Home Screen merupakan halaman yang pertama kali muncul ketika aplikasi dibuka. Pada halaman ini terdapat 2 tombol yaitu Mulai dan Keluar Aplikasi. Tampilan Halaman Utama / Home Screen dapat dilihat pada Gambar 10.

Gambar 10 Tampilan Halaman Utama/Home Screen

e-ISSN: 2252-7036
Vol.6 No.1 2021: 1-10

2) Tampilan Halaman Pilih Algoritma: Halaman Pilih Algoritma merupakan halaman yang muncul ketika kita menekan tombol Mulai pada Halaman Utama/*Home Screen*. Halaman ini difungsikan untuk memilih algoritma yang ingin digunakan untuk pengujian. Pada halaman ini terdapat 3 tombol, yaitu Tombol Algoritma Boyer-Moore untuk menggunakan metode algoritma Boyer-Moore dalam proses pencarian, Tombol Algoritma Rabin-Karp untuk menggunakan metode algoritma Rabin-Karp dalam proses pencarian, dan Tombol Kembali untuk kembali ke Halaman Utama/*Home Screen*. Tampilan Halaman Pilih Algoritma dapat dilihat pada Gambar 11.

Gambar 11 Tampilan Halaman Pilih Algoritma

Gambar 12 Tampilan Halaman Pencarian Algoritma Boyer-Moore

3) Tampilan Halaman Pencarian Algoritma Boyer-Moore: Halaman Pencarian Algoritma Boyer-Moore merupakan halaman yang muncul ketika Tombol Algoritma Boyer-Moore pada halaman Pilih Algoritma ditekan. Halaman ini difungsikan untuk melakukan pencarian *string* dalam database dengan menggunakan Algoritma Boyer-Moore. Pada halaman ini terlihat simbol *search*/pencarian untuk memasukkan pola pencarian, waktu untuk menampilkan nilai *real running time* ketika sudah dilakukan pencarian data, data berupa (nilai *real running time*, Kecamatan, Kode Pos, dan Nama Desa/Gampong), dan hanya terdapat Tombol Cari untuk melakukan pencarian. Halaman Pencarian algoritma Boyer-Moore dapat dilihat pada Gambar 12.

4) Tampilan Halaman Pencarian Algoritma Rabin-Karp: Halaman Pencarian Algoritma Rabin-Karp merupakan halaman yang muncul ketika Tombol Algoritma Rabin-Karp pada halaman Pilih Algoritma di pilih. Halaman ini difungsikan untuk melakukan pencarian string dalam database dengan menggunakan Algoritma Rabin-Karp. Pada halaman ini terlihat simbol *search*/pencarian untuk memasukkan pola pencarian, waktu untuk menampilkan nilai *real running time* ketika sudah dilakukan pencarian data, data berupa (nilai *real running time*, Kecamatan, Kode Pos, dan nama Desa/Gampong), dan hanya terdapat Tombol Cari untuk melakukan pencarian. Halaman Pencarian Algoritma Rabin-Karp dapat dilihat pada Gambar 13.

Gambar 13 Tampilan Halaman Pencarian Algoritma Rabin-Karp

e-ISSN: 2252-7036
Vol.6 No.1 2021: 1-10

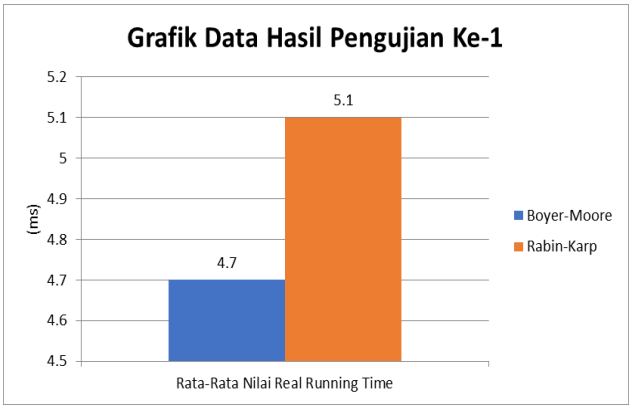
IV. HASIL DAN PEMBAHASAN

A. *Analisa Data Hasil Pengujian Ke-1*

Adapun hasil dari pengujian 1 dapat dilihat pada Tabel I.

TABLE I
ANALISA DATA HASIL PENGUJIAN KE-1

No.	Pola	Jumlah Desa yang Ditemukan	Hasil	Nilai *Real Running Time*	
				Boyer-Moore	Robin-Karp
1.	Ban	1	Cocok	8	8
2.	Pin	1	Cocok	7	6
3.	Doi	1	Cocok	6	5
4.	Jan	0	Tidak ditemukan	5	8
5.	Kut	1	Cocok	5	3
6.	Ge	3	Cocok	3	7
7.	Lam	2	Cocok	4	3
8.	Lam	22	Cocok	4	5
9.	Lak	1	Cocok	2	3
10.	Ate	4	Cocok	3	3
Rata-rata Nilai *Real Running Time*				4,7	5,1



Gambar 14 Grafik Perbandingan Data Hasil Pengujian ke-1

Dari table tersebut dapat dilihat bahwa pengujian dengan menggunakan pola yang sama dapat menghasilkan perbandingan antara hasil nilai Real Running Time menggunakan algoritma Boyer-Moore dan algoritma Rabin-Karp.

Pada data dan grafik hasil pengujian ke-1 di atas dapat dilihat perbandingan hasil pengujian Algoritma Boyer-Moore dan Algoritma Rabin-Karp dengan menggunakan 10 pola yang sama, yakni dengan pola "Ban, Pin, Doi, Jan, Kut, ge, lam, Lam, Lak, dan Ate". Pada pengujian menggunakan algoritma Boyer-Moore menggunakan pola Ban mendapatkan nilai Real Running Time sebesar 8 ms. Pengujian dengan pola Pin dengan nilai Real Running Time 7 ms, pengujian dengan pola Doi mendapatkan nilai Real Running Time sebesar 6 ms, pengujian menggunakan pola Jan mendapatkan nilai Real Running Time sebesar 5 ms, pengujian menggunakan pola Kut mendapatkan nilai Real *Running Time* sebesar 5 ms, pengujian menggunakan pola Ge mendapatkan nilai Real Running Time sebesar 3 ms, pengujian menggunakan pola lam mendapatkan nilai Real Running Time sebesar 4 ms, pengujian menggunakan pola Lam mendapatkan nilai Real Running Time sebesar 4 ms, pengujian menggunakan pola Lak mendapatkan nilai Real Running Time sebesar 2 ms, pengujian menggunakan pola Ate mendapatkan nilai Real Running Time sebesar 3 ms. Dari hasil pengujian menggunakan Algoritma Boyer-Moore diperoleh nilai Real Running Time rata-rata sebesar 4,7 ms. Pencarian pola terlama pada pola pencarian Ban dengan nilai Real Running Time sebesar 8 ms, dan pencarian Pola tercepat pada pola pencarian Lak dengan nilai Real Running Time 2 ms.

Pada pengujian menggunakan algoritma Rabin-Karp dengan menggunakan pola Ban mendapatkan nilai Real Running Time sebesar 8 ms. Pengujian dengan pola Pin dengan nilai Real Running Time 6 ms, pengujian dengan pola Doi mendapatkan nilai Real Running Time sebesar 5 ms, pengujian menggunakan pola Jan mendapatkan nilai Real Running Time sebesar 8 ms, pengujian menggunakan pola Kut mendapatkan nilai Real Running Time sebesar 3 ms, pengujian menggunakan pola Ge mendapatkan nilai Real Running Time sebesar 7 ms, pengujian menggunakan pola lam mendapatkan nilai Real Running Time sebesar 3 ms, pengujian menggunakan pola Lam mendapatkan nilai Real Running Time sebesar 5 ms, pengujian menggunakan pola Lak mendapatkan nilai Real Running Time sebesar 3 ms, pengujian menggunakan pola Ate mendapatkan nilai Real Running Time sebesar 3 ms. Dari hasil pengujian menggunakan Algoritma Rabin-Karp didapatkan nilai Real Running Time rata-rata sebesar 5,1 ms. Pencarian pola terlama pada pola pencarian Ban dan Jan dengan nilai Real Running Time sebesar 8 ms, dan pencarian Pola tercepat didapat saat melakukan pencarian pada pola pencarian Kut, lam, Lak dan Ate dengan nilai Real Running Time yang sama yaitu sebesar 3 ms.

Dari hasil pengujian kedua algoritma di atas, hasil pengujian menggunakan Algoritma Boyer-Moore menghasilkan rata-rata nilai Real Running Time sebesar 4,7 ms, sedangkan hasil pengujian menggunakan Algoritma Rabin-Karp menghasilkan rata-rata nilai Real Running Time sebesar 5,1 ms.

B. *Analisa Data Hasil Pengujian Ke-2*

Adapun hasil dari pengujian 2 dapat dilihat pada tabel II. Pada tabel terlihat bahwa pengujian dengan menggunakan pola "Tibang" secara berulang sebanyak 10 kali percobaan terhadap kedua algoritma, dapat menghasilkan perbandingan antara hasil nilai Real Running Time menggunakan Algoritma Boyer-Moore dan Algoritma Rabin-Karp.

Pada data dan grafik hasil pengujian ke-2 di atas dapat dilihat, bahwa pengujian terhadap pola yang sama, yaitu pola "Tibang" yang dilakukan berulang sebanyak 10 kali percobaan terhadap algoritma Boyer-Moore dan algoritma Rabin-Karp mendapatkan hasil yang berbeda. Pada pengujian menggunakan algoritma Boyer-Moore, didapatlah

e-ISSN: 2252-7036
Vol.6 No.1 2021: 1-10

nilai Real Running Time masing masing 6 ms, 8 ms, 10 ms, 7 ms, 5 ms, 7 ms, 7 ms, 5 ms, 5 ms, dan 4 ms dengan rata-rata nilai Real Running Time sebesar 6,4 ms.

Pada pengujian menggunakan algoritma Rabin-Karp, diperoleh nilai Real Running Time masing masing 12 ms, 10 ms, 9 ms, 8 ms, 5 ms, 6 ms, 6 ms, 6 ms, 7 ms, dan 10 ms dengan rata-rata nilai Real Running Time sebesar 7,9 ms.

Dari analisa data hasil pengujian di atas, dapat disimpulkan bahwa pengujian ke-2 menggunakan algoritma Boyer-Moore menghasilkan nilai Real Running Time rata-rata sebesar 6,4 ms yang berarti masih lebih cepat menemukan pola dibandingkan dengan algoritma Rabin-Karp dengan rata-rata nilai Real Running Time sebesar 7,9 ms.

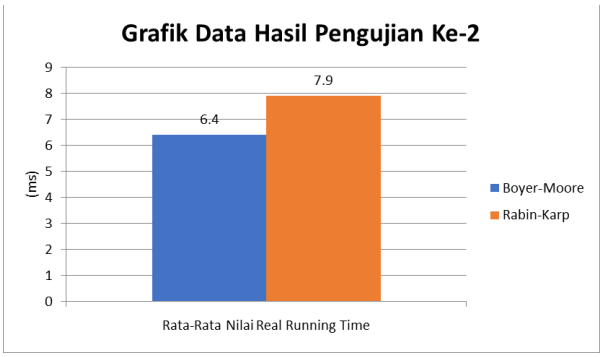
C. Analisa Data Hasil Pengujian Ke-3

Pengujian ke-3 ini menggunakan pola acak dan dilakukan sebanyak 10 kali percobaan pencarian terhadap algoritma Boyer-Moore dan algoritma Rabin-Karp.

Adapun hasil dari pengujian ke-3 dengan menggunakan algoritma Boyer-Moore dapat dilihat pada tabel III. Dari table di atas dapat dilihat data hasil pengujian ke-3 menggunakan algoritma Boyer-Moore. Pengujian dilakukan terhadap pola acak yang dilakukan sebanyak 10 kali percobaan.

TABLE III
ANALISA DATA HASIL PENGUJIAN KE-2

No.	Pola	Jumlah Desa yang Ditemukan	Hasil	Nilai *Real Running Time*	
				Boyer-Moore	Robin-Karp
1.	Tibang	1	Cocok	6	12
2.	Tibang	1	Cocok	8	10
3.	Tibang	1	Cocok	10	9
4.	Tibang	1	Cocok	7	8
5.	Tibang	1	Cocok	5	5
6.	Tibang	1	Cocok	7	6
7.	Tibang	1	Cocok	7	6
8.	Tibang	1	Cocok	5	6
9.	Tibang	1	Cocok	5	7
10.	Tibang	1	Cocok	4	10
Rata-rata Nilai *Real Running Time*				6,4	7,9



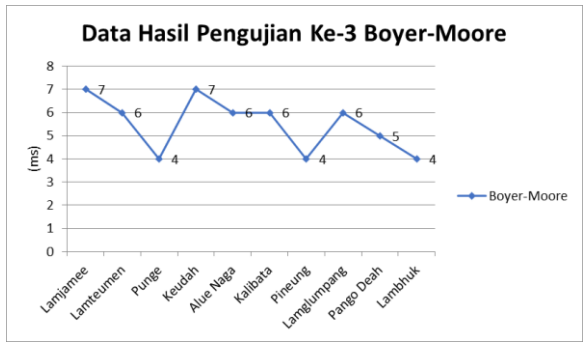
Gambar 15 Grafik Perbandingan Data Hasil Pengujian Ke-2

Pada data dan grafik hasil pengujian ke-3 terhadap algoritma Boyer-Moore di atas dapat dilihat, bahwa pencarian pola secara acak yang dilakukan berulang sebanyak 10 kali percobaan terhadap algoritma Boyer-Moore, diperoleh nilai Real Running Time masing masing pada pola Lamjamee sebesar 7 ms, pola Lamteumen sebesar 6 ms, pola Punge sebesar 4 ms, pola Keudah sebesar 7 ms, pola Alue Naga sebesar 6 ms, pola Kalibata sebesar 6 ms, pola Pineung sebesar 4 ms, pola Lamglumpang sebesar 6 ms, pola Pango Deah sebesar 5 ms, dan pola Lambhuk sebesar 4 ms dengan rata-rata nilai Real Running Time sebesar 5,5 ms.

Adapun hasil dari pengujian ke-3 menggunakan Algoritma Rabin-Karp dapat dilihat pada Tabel IV. Dari table IV dapat dilihat data hasil pengujian ke-3 menggunakan algoritma Rabin-Karp. Pengujian dilakukan terhadap pola acak yang berbeda dengan pola pengujian ke-3 terhadap algoritma Boyer-Moore. Penguji melakan sebanyak 10 kali percobaan pencarian pola. Berikut grafik data hasil pengujian ke-3 menggunakan algoritma Rabin-Karp.

TABLE IIIII
ANALISA DATA HASIL PENGUJIAN KE-3 MENGGUNAKAN ALGORITMA BOYER-MOORE

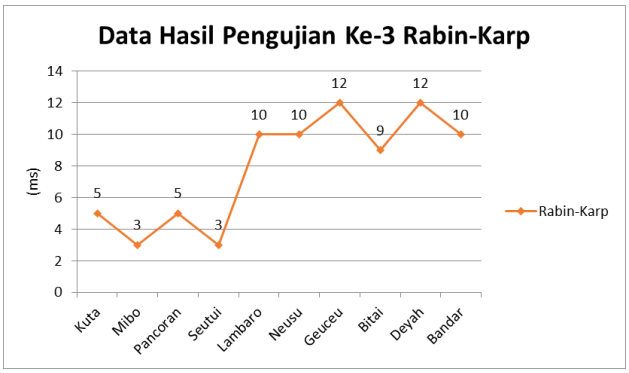
No.	Pola	Jumlah Desa yang Ditemukan	Hasil	Nilai *Real Running Time*
1.	Lamjamee	1	Cocok	7
2.	Lamteumen	2	Cocok	6
3.	Punge	3	Cocok	4
4.	Keudah	1	Cocok	7
5.	Alue Naga	1	Cocok	6
6.	Kalibata	0	Tidak ditemukan	6
7.	Pineung	1	Cocok	4
8.	Lamglumpang	1	Cocok	6
9.	Pango Deah	1	Cocok	5
10.	Lambhuk	1	Cocok	4
Rata-rata Nilai *Real Running Time*				5,5



Gambar 16 Grafik Data Hasil Pengujian Ke-3 Menggunakan Algoritma Boyer-Moore

TABLE IV
ANALISA DATA HASIL PENGUJIAN KE-3 MENGGUNAKAN ALGORITMA RABIN-KARP

No.	Pola	Jumlah Desa yang Ditemukan	Hasil	Nilai *Real Running Time* (ms)
1.	Kuta	1	Cocok	5
2.	Mibo	1	Cocok	3
3.	Pancoran	0	Tidak ditemukan	5
4.	Seutui	1	Cocok	3
5.	Lambaro	1	Cocok	10
6.	Neusu	2	Cocok	10
7.	Geuceu	4	Cocok	12
8.	Bitai	1	Cocok	9
9.	Deyah	1	Cocok	12
10.	Bundar	1	Cocok	10
Rata-rata Nilai *Real Running Time*				7,9



Gambar 17 Grafik Data Hasil Pengujian Ke-3 Menggunakan Algoritma Rabin-Karp

Pada data dan grafik hasil pengujian ke-3 terhadap algoritma Rabin-Karp di atas dapat dilihat, bahwa pencarian pola secara acak yang dilakukan berulang sebanyak 10 kali percobaan terhadap algoritma Rabin-Karp, didapatlah nilai Real Running Time masing masing pola Kuta sebesar 5 ms, pola Mibo sebesar 3 ms, pola Pancoran sebesar 5 ms, pola Seutui sebesar 3 ms, pola Lambaro sebesar 10 ms, pola Neusu sebesar 10 ms, pola Geuceu sebesar 12 ms, pola Bitai sebesar 9 ms, pola Deyah sebesar 12 ms, dan pola Bandar sebesar 10 ms dengan rata-rata nilai Real Running Time sebesar 7,9 ms.

Dari analisa data hasil pengujian di atas, dapat disimpulkan bahwa pengujian ke-3 menggunakan algoritma Boyer-Moore menghasilkan nilai Real Running Time rata-rata sebesar 5,5 ms yang berarti masih lebih cepat menemukan pola dibandingkan dengan algoritma Rabin-Karp dengan rata-rata nilai Real Running Time sebesar 7,9 ms.

Dari ketiga hasil pengujian yang dilakukan, diperoleh nilai rata-rata untuk algoritma Boyer-Moore dengan nilai masing-masing 4,7 ms, 6,4 ms, dan 5,5 ms dengan rata-rata penemuan string: 5,53 ms. Sedangkan untuk algoritma Rabin-Karp, diperoleh nilai rata-rata dengan nilai masing-masing 5,1 ms, 7,9 ms, dan 7,9 ms dengan rata-rata penemuan string: 6,96 ms. Dapat diketahui bahwa algoritma Boyer-Moore lebih cepat 21,33 % dibandingkan dengan algoritma Rabin-Karp.

V. KESIMPULAN

Penelitian ini dilakukan untuk mengembangkan aplikasi berbasis android yang dikembangkan menggunakan Android Studio. Aplikasi ini berhasil melakukan pencarian (String Matching) dengan menggunakan algoritma Boyer-Moore dan algoritma Rabin-Karp. Dalam proses evaluasi, dilakukan 3 kali pengujian. Pada pengujian pertama, dilakukan dengan menggunakan algoritma Boyer-Moore mendapatkan rata rata nilai Real Running Time sebesar 4,7 ms sedangkan pengujian Ke-1 dengan menggunakan algoritma Rabin-Karp mendapatkan rata-rata nilai Real Running Time sebesar 5,1 ms. Pada pengujian kedua dengan menggunakan algoritma Boyer-Moore mendapatkan rata-rata nilai Real Running Time sebesar 6,4 ms sedangkan pengujian Ke-2 dengan menggunakan algoritma Rabin-Karp mendapatkan rata-rata nilai Real Running Time sebesar 7,9 ms. Pada pengujian ketiga dengan menggunakan algoritma Boyer-Moore mendapatkan rata-rata nilai Real Running Time sebesar 5,5 ms sedangkan pengujian Ke-3 dengan menggunakan algoritma Rabin-Karp mendapatkan rata-rata nilai Real Running Time sebesar 7,9 ms. Dari penelitian yang dilakukan, diperoleh hasil berupa nilai Real Running Time algoritma Boyer-Moore dengan rata-rata penemuan string : 5,53 ms dan algortima Rabin-Karp dengan rata-rata penemuan string : 6,96 ms. Dapat diketahui bahwa algoritma Boyer-Moore lebih cepat 21,33% dalam menemukan pola dibandingkan dengan algoritma Rabin-Karp.

REFERENSI

[1] S. Wahyuningsih and J. Suryanto, "Evaluasi Pemanfaatan Kode Pos," Bul. Pos dan Telekomun., vol. 9, no. 3, p. 317, 2015, doi: 10.17933/bpostel.2011.090304.

[2] "Kode POS Kota Banda Aceh - urut Desa/Kelurahan - hal 1." https://kodepos.nomor.net/_kodepos.php?_i=desa-kodepos&sby=100000&daerah=Kota&jobs=Banda Aceh (accessed Dec. 27, 2020).

[3] M. Rossaria and B. Susilo, "Implementasi Algoritma Pencocokan String Knuth-Morris-Pratt Dalam Aplikasi Pencarian Dokumen Digital Berbasis Android," J. Rekursif, vol. 3, no. 2, pp. 183–195, 2015.

[4] R. I. Borman, "Penerapan String Matching Dengan Algoritma Boyer Moore Pada Aplikasi Font Italic Untuk Deteksi Kata Asing," J. Teknoinfo, vol. 10, no. 2, p. 39, 2016, doi: 10.33365/jti.v10i2.9.

[5] Y. A. Wicaksono and S. Suyanto, "Analisis Dan Implementasi Algoritma Rabin-Karp Dan Algoritma Stemming Nazief-Adriani Pada Sistem Pendeteksi Plagiat Dokumen," 2012.

[6] V. Sagita and M. I. Prasetiyowati, "Studi Perbandingan Implementasi Algoritma Boyer-Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore dalam Pencarian String," J. Ultim., vol. 5, no. 1, pp. 31–37, 2013, doi: 10.31937/ti.v5i1.311.

[7] "Nazruddin Safaat: Pemrograman Aplikasi Android mobile... - Google Scholar." https://scholar.google.com/scholar?cluster=14350338175945645754&hl=en&oi=scholarr (accessed Dec. 27, 2020).

[8] I. Al Fikri, "Aplikasi Navigasi Berbasis Perangkat Bergerak dengan Menggunakan Platform Wikitude untuk Studi Kasus Lingkungan ITS," J. Tek. ITS, vol. 5, no. 1, pp. 48–51, 2016, doi: 10.12962/j23373539.v5i1.14511.
[9] H. I. Flower, Introduction to the second edition, vol. 7. 2010.
[10] K. W. Argakusumah and S. Hansun, "Implementasi Algoritma Boyer-Moore pada Aplikasi Kamus Kedokteran Berbasis Android," J. Ultim., vol. 6, no. 2, pp. 70–78, 2014, doi: 10.31937/ti.v6i2.340.
[11] Herriyance, Handrizal, and S. D. Fadilla, "Analisis Algoritma Rabin-Karp Pada Kamus," J. Ris. Sist. Inf. dan Tek. Inform., vol. 2, no. 1, pp. 64–74, 2017.