ISSN : 2407 - 3911

IMPLEMENTASI BOYER-MOORE PADA APLIKASI PENCARIAN RUMUS MATEMATIKA DAN FISIKA

Halim Agung, Yogyawan
Program Studi Teknik Informatika
Universitas Bunda Mulia
Jl. Lodan Raya No.2 Jakarta
hagung@bundamulia.ac.id, joshuayogyawan@gmail.com

Abstrak

Sulitnya mencari dan menghafalkan rumus matematika dan fisika menjadi kendala bagi sebagian pelajar SMA terutama dalam menjawab soal yang ada. Dibutuhkan ketelitian untuk menentukan apa saja yang tercantum dalam soal tersebut. Penelitian ini meneliti bagaimana cara melakukan pendeteksian kata kunci yang terdapat pada soal matematika dan fisika sekaligus dicocokkan dengan menggunakan metode pencocokan kata. Algoritma yang digunakan untuk melakukan pencocokan kata adalah algoritma Boyer-Moore. Setiap data kata kunci yang ada di basis data akan diambil berdasarkan fisika atau matematika, kemudian dipecah per kata. Setelah data kata kunci sudah dipecah menjadi per kata, tiap kata kunci akan melakukan pencocokkan ke soal yang dimasukkan pengguna dengan metode pencocokkan kata. Setelah dicocokkan dan hasil pencocokkan berhasil, aplikasi akan menampilkan daftar rumus yang memiliki kata kunci cocok dengan yang ada di soal. Pengujian metode pencocokan kata dengan algoritma Boyer-Moore pada soal matematika dan fisika yang sebanyak 30 kali membuktikan bawah aplikasi dapat menampilkan daftar rumus yang sesuai dalam melakukan pencocokkan pola kata kunci di soal baik matematika maupun fisika tingkat SMA.

Kata kunci :

Pencocokan Kata Kunci, Boyer-Moore, Rumus, Matematika, Fisika

Abstract

Difficult to find and memorize math and physics becomes an obstacle for the majority of high school students, especially in answering the questions that exist. It takes precision to determine what is stated in the question. This study investigated how to make the detection of keywords contained in the math and physics as well matched using string matching method. The algorithm used to perform string matching algorithm is the Boyer-Moore. Each keyword data in the database will be taken based on the physics or mathematics, then broken down per word. Once the data has been broken down into key words per word, every keyword will perform matching to the matter that the user entered with a string matching method. Once matched and the matching result is successful, the application will display a list of formulas that have keywords match the one in question. Testing method of string matching algorithms Boyer-Moore in math and physics as much as 30 times to prove under the application can display a list of the appropriate formula in performing the pattern matching keywords in a matter of both mathematics and physics high school level.

Keywords :

String Matching, Boyer-Moore, Formula, Math, Physics.

I. PENDAHULUAN

Zaman Era Globalisasi ini teknologi semakin berkembang pesat begitupun bidang pendidikan dari zaman ke zaman. Banyak sumbangan dari ahli-ahli pendidikan di masa lalu. Terutama rumus-rumus yang mereka ciptakan dari tiap masalah yang belum terpecahkan saat itu. Mereka memecahkan masalah tersebut dan membuat rumus ciptaan mereka. Tiap bidang walau berbeda-beda namun memiliki hubungan yang mendasar. Sebagai contoh matematika, bidang tersebut sudah lama dan kuno namun berkembang dan terjadi disekitar aktifitas kita. Matematika adalah bidang yang selalu dikaitkan dengan berhitung, bahkan bidang tersebut memiliki

segudang rumus yang saling berkaitan. Rumus-rumus dalam tiap bidang memiliki keunikan dan fungsional yang berbeda-beda tergantung masalah yang ada. Murid kadang susah untuk menghafal rumus-rumus yang rumit dan banyak. Mereka sering harus melihat buku pelajaran mereka lagi untuk mencari rumus yang bisa digunakan untuk masalah yang ada.

Rumusan masalah yang akan dibahas pada penelitian ini adalah Bagaimana penelitian ini dapat membantu para siswa SMA untuk mengatasi kesulitan dalam menjawab soal-soal Matematika dan Fisika dan dalam mencari dan menghafal rumus matematika dan fisika. Oleh karena itu, penelitian ini menerapkan algoritma *Boyer-Moore* pada aplikasi rumus matematika dan fisika tingkat SMA berbasis Android dengan pendeteksian kata kunci.

Tujuan penelitian ini untuk melakukan suatu penelitian terhadap algoritma *Boyer-Moore* dengan dibuatnya aplikasi pencarian rumus matematika dan fisika tingkat SMA berbasis Android yang dapat menampilkan *list* rumus yang berhubungan dengan setiap kata kunci dalam soal yang dimasukkan oleh user. Dengan adanya penelitian ini diharapkan dapat membantu para siswa SMA dalam meningkatkan hasil belajar mereka di sekolah.

II. KAJIAN LITERATUR

II.1 Penelitian Terdahulu

Berdasarkan hasil telaah terhadap penelitian yang berkaitan dengan algoritma *Boyer-Moore*, terdapat beberapa hasil penelitian yang dapat dijadikan dasar acuan kajian ini, antara lain:

1. (, A. D. (2006). Konsep Kecerdasan Buatan., Hansun, 2014), dalam papernya yang berjudul "Implementasi Algoritma *Boyer-Moore* pada Aplikasi Kamus Kedokteran Berbasis Android" yang menerapkan Algoritma *Boyer Moore* pada aplikasi kamus kedokteran menyatakan bahwa dengan algoritma *Boyer-Moore* berhasil dilakukan dengan persentase sebesar 100% dari responden menyatakan bahwa proses pencarian istilah pada aplikasi Kamus Kedokteran dapat memberikan penjelasan atau hasil yang sesuai dengan yang diharapkan.
2. (Rahmanita, 2014), dalam papernya yang berjudul "Pencarian *String* Menggunakan Algoritma Boyer Moore Pada Dokumen" yang menerapkan Algoritma *Boyer Moore* pada dokumen menyatakan bahwa Algoritma *Boyer Moore* mempunyai keunggulan dalam waktu menemukan *pattern* yang akan dicari dalam ukuran *file* yang lebih besar dan efektifitas Algoritma *Boyer Moore* tergantung pada panjang kata yang dicari.
3. (Utomo, Harjo, Handoko, 2008), dalam papernya yang berjudul "Perbandingan Algoritma *String Matching Brute Force, Knuth Morris Pratt, Boyer Moore* dan *Karp Rabin* Pada Teks Alkitab Bahasa Indonesia" yang melakukan perbandingan pada algoritma *String Searching Brute Force, Knuth-Morris-Pratt, Boyer Moore*, dan *Karp Rabin* menyatakan bahwa pada algoritma *Boyer Moore* membuktikan semakin panjang polanya maka waktu pencarian juga semakin singkat dan untuk proses pencarian pola dalam teks pada Alkitab (lebih dari 26 alphabet) algoritma *Boyer Moore* yang paling cepat dibandingkan dengan *Karp Rabin, Knuth-Morris-Pratt,* dan *Brute Force.*
4. (Ramadhansyah, 2013), dalam papernya yang berjudul "Perancangan Aplikasi Kamus Bahasa Gayo Dengan Menggunakan Metode *Boyer Moore*" yang melakukan teknik pencarian kosakata dengan menggunakan kamus yang dilakukan oleh masarakat Gayo adalah dengan mencari urutan abjad dari suatu kata yang dicari, sehingga proses translasi yang dilakukan memakan waktu yang lama dan dengan menggunakan metode *Boyer Moore* maka proses pencarian translasi kata pada *database* dapat berjalan lebih cepat dan akurat.
5. {Hakim, Juliana, 2016), dalam papernya yang berjudul "Implementasi Algoritma *Boyer Moore* Pada Web *E-Katalog* Flora dan Fauna Pulau Jawa dan Sumatera" yang menyatakan bahwa penerapan algoritma *Boyer Moore* memberikan kemudahan dalam pencarian informasi secara efisien mengenai flora dan fauna Indonesia yaitu khusus Jawa dan Sumatera.

II.2 Metode Pencarian

Pencarian atau pelacakan merupakan salah satu teknik untuk menyelesaikan permasalahan AI (*Artificial Intelligence*). Keberhasilan suatu sistem salah satunya ditentukan oleh kesuksesan dalam pencarian atau pencocokan (Arhami, 2006).

II.3 Algoritma Boyer Moore

Algoritma *Boyer Moore* dianggap sebagai algoritma pencocokan *string* yang paling efisien digunakan dalam berbagai aplikasi sehari-hari. Algoritma tersebut dikembangkan oleh Bob Boyer dan J. Stroher Moore pada tahun 1977.

Pada proses pencarian *string* algoritma *Boyer-Moore* membaca karakter-karakter dari pola (*pattern*) dari kanan ke kiri. Dalam kasus dimana jumlah karakter pada pattern lebih sedikit daripada jumlah karakter pada teks maka algoritma tersebut menggunakan 2 buah fungsi precomputed. Dua buah fungsi pengubah ini disebut *good-suffix shift*. Aturan pada *good-suffix shift* bertujuan untuk menangani kasus dimana terdapat pengulangan karakter pada *pattern*. (Aulia, 2008).

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

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

Precomputation dari algoritma *Boyer-Moore* terdiri dari *bad-character preprocessing* dan *good-suffix preprocessing*. Prinsip dasar yang pertama dari algoritma *Boyer-Moore* adalah melakukan perbandingan antara *pattern* yang dicari dengan teks. Perbandingan *pattern* dengan teks dilakukan dari arah kanan ke kiri.

Perbandingan dimulai dengan membandingkan antara karakter paling kanan dari *pattern* dengan teks. Jika terjadi kecocokkan, maka perbandingan akan dilanjutkan dengan karakter yang disebelah kiri dari yang dibandingkan sampai ke karakter pertama dari *pattern*. Jika terjadi ketidakcocokkan maka akan dilakukan pergeseran yang ditentukan oleh 2 fungsi pergeseran yaitu *bad-character shift* dan *good-suffix shift*. Aturan dari *bad-character shift* dibutuhkan untuk menghindari pengulangan perbandingan yang gagal dari suatu karakter dalam teks dengan *pattern*. Aturan dari *good-suffix shift* dibutuhkan untuk menangani kasus yang di dalamnya terdapat pengulangan karakter pada *pattern*.

Secara sistematis, langkah-langkah yang dilakukan algoritma *Boyer-Moore* pada saat mencocokkan *string* adalah (Chiquita, Christabella, 2012) :

1. Buat tabel pergeseran *string* yang dicari (S) dengan pendekatan *Match Heuristic* (MH) dan *Occurence Heuristic* (OH), untuk menentukan jumlah pergeseran yang akan dilakukan jika mendapat karakter tidak cocok pada proses pencocokkan dengan *string* (T).
2. Jika dalam proses perbandingan terjadi ketidakcocokkan antara pasangan karakter pada S dan karakter pada T, pergeseran dilakukan dengan memilih salah satu nilai pergeseran dari dua tabel analisa *string* yang memiliki nilai pergeseran paling besar.
3. Dua kemungkinan penyelesaian dalam melakukan pergeseran S, jika sebelumnya belum ada karakter yang cocok adalah dengan melihat nilai pergeseran hanya pada tabel *Occurence Heuristic*, jika karakter yang tidak cocok tiak ada pada S, maka pergeseran adalah sebanyak jumlah karakter pada S, dan jika karakter yang tidak cocok ada pada S, maka banyaknya pergeseran bergantung pada nilai tabel.
4. Jika karakter pada teks yang sedang dibandingkan cocok dengan karakterpada S, maka posisi karakter pada S dan T diturunkan sebanyak 1 posisi, kemudian dilanjutkan dengan pencocokkan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokkan karakter S dan T, maka dipilih nilai pergeseran terbesar dari dua tabel analis *pattern*, yaitu nilai dari tabel *Match Heuristic* dan tabel *Occurence Heuristic* dikurangi dengan jumlah karakter yang telah cocok.
5. Jika semua karakter telah cocok, artinya S telah ditemukan di dalam T, selanjutnya geser *pattern* sebanyak 1 karakter.
6. Lanjutkan sampai akhir *string* T.

Kelebihan dari algoritma *Boyer-Moore* (Utomo, 2008) ini semakin panjang pola yang dicari maka waktu pencarian semakin singkat.

Sedangkan kekurangan algoritma *Boyer-Moore* adalah lebih lambat untuk *pattern* yang pendek dan tidak bagus untuk pencarian *binary string*. (Aulia, 2008).

III. METODOLOGI PENELITIAN

Dalam pembuatan aplikasi terdapat beberapa kebutuhan fungsional dan non-fungsional dari hasil analisis yang dilakukan.

Untuk kebutuhan fungsional yang dibutuhkan untuk aplikasi ini adalah sebagai berikut :

a. Aplikasi ini harus memiliki fitur-fitur yang tidak rumit, mudah dipelajari dan digunakan oleh pelajar SMA

b. Aplikasi ini harus dapat melakukan pencarian rumus tertentu dengan cepat dalam hal ini matematika dan fisika tingkat SMA

c. Aplikasi ini dapat diakses 24 jam secara *online* pada smartphone berbasis android

Sedangkan untuk kebutuhan non-fungsional melibatkan perangkat keras, perangkat lunak dan pengguna. Untuk spesifikasi kebutuhan perangkat keras adalah sebagai berikut :

a. *Processor* : ARMv6 800MHZ

b. RAM : 843 Mb

c. Memori (*Internal Storage*) : 3,69 Gb

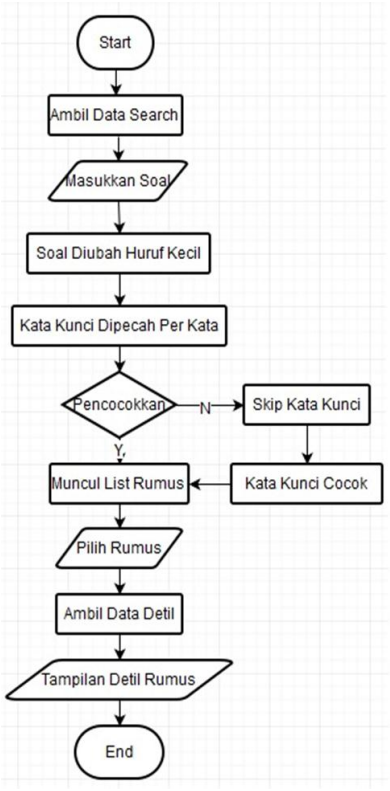
d. Memori (*External Storage*) : 14,83 Gb

Sedangkan untuk kebutuhan *software* yang mendukung pembuatan aplikasi ini adalah Android Studio, Bluestacks, MySQL.

Peneliti menggunakan beberapa tahapan yang harus dilakukan untuk menggambarkan sistem sebelum membuat aplikasi yang menggunakan algoritma *Boyer-Moore*. Metode yang digunakan dalam penelitian ini adalah metode *Waterfall* (Bassil, 2012).

III.1 Flowchart

Perancangan *flowchart* bertujuan untuk menggambarkan aliran proses dalam sistem. *Flowchart* yang dibuat adalah *Flowchart* Pencarian *Text* yang ditampilkan pada gambar 3.



Gambar 3. Flowchart Pencarian *Text*

Flowchart diagram yang terlihat pada Gambar 3 adalah *flowchart* untuk menu *search* baik matematika ataupun fisika sekaligus menjelaskan saat aplikasi dijalankan hingga muncul detil rumus. Penjelasan secara detailnya adalah sebagai berikut :

a. *User* menjalankan aplikasi. Kegiatan ini diwakili oleh simbol terminator yang bertuliskan "START" yang berarti *user* mengakses menu *search*.

b. Saat *user* mengakses menu *search*, sistem akan mengambil data dari tabel detil berupa id rumus, nama rumus, kata kunci, bab, dan pelajaran. Kegiatan tersebut bertuliskan "Ambil Data Detil".

c. Saat proses mengambil data sudah selesai, kemudian *user* akan memasukkan soal. Kegiatan tersebut bertuliskan "Masukkan Soal".

d. Setelah *user* memasukkan soal dan menekan *button search*, maka sistem akan langsung mengeksekusi dengan mengubah huruf-huruf soal yang dimasukkan *user* dengan huruf kecil

(lowercase). Kegiatan tersebut bertuliskan "Soal Diubah Kata Kecil".

e. Langsung setelah soal diubah menjadi huruf kecil, sistem juga mengubah data kata kunci menjadi per kata yang nantinya digunakan untuk dicocokkan ke soal. Kegiatan tersebut bertuliskan "Kata Kunci Dipecah Per Kata".

f. Setelah itu barulah sistem akan mencocokkan (kegiatan tersebut bertuliskan "Pencocokkan") kata kunci yang sudah menjadi per kata ke dalam soal yang sudah diubah menjadi huruf kecil. Bila ada kecocokkan maka akan muncul di list rumus (kegiatan tersebut bertuliskan "Muncul List Rumus"). Bila tidak ("Skip Kata Kunci"), sistem akan mencari lagi kata kunci yang hingga cocok ("Kata Kunci Cocok").

g. Setelah semua proses pencocokkan sudah selesai, hasil pencocokkan berupa *list* rumus yang data kuncinya cocok dengan apa yang ada di soal. *User* hanya tinggal memilih rumus apa yang ingin dilihat. Kegiatan tersebut bertuliskan "Pilih Rumus".

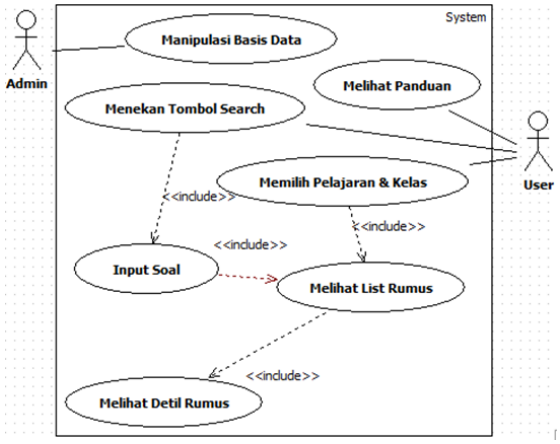
h. Setelah *user* memilih salah satu rumus di *list* rumus ("Pilih Rumus"), sistem akan langsung menampilkan halaman detil rumus. Halaman detil ini masih harus mengambil data dari *database* berupa tabel detil ("Ambil Data Detil").

i. Terakhir ialah tampilan halaman detil beserta detil rumus hasil pilihan *user*. Kegiatan tersebut bertuliskan "Tampilan Detil Rumus".

j. Setelah sudah di halaman detil dan tampil detil rumus, maka jalan sistem sudah selesai. Kegiatan tersebut bertuliskan "END".

III.2 Use Case Diagram

Dalam aplikasi pencarian matematika dan fisika tingkat SMA ini, *user* dapat melihat panduan untuk mengetahui tata cara memakai aplikasi ini, user bisa mengakses menu *search*, atau *user* bisa mengakses menu rumus secara manual. Menu *search* adalah proses dimana *user* memasukkan soal, kemudian hasil dari *search* akan berupa *list* rumus. Dan *user* bisa melihat halaman detil hasil dari aktifitas menu *search* dan menu rumus. Sementara aktor admin bertugas untuk memanipulasi data dengan memasukkan data rumus, mengubah, ataupun menghapusnya. Secara umum, fungsionalitas aplikasi ini dapat digambarkan pada diagram usecase seperti pada gambar 4.

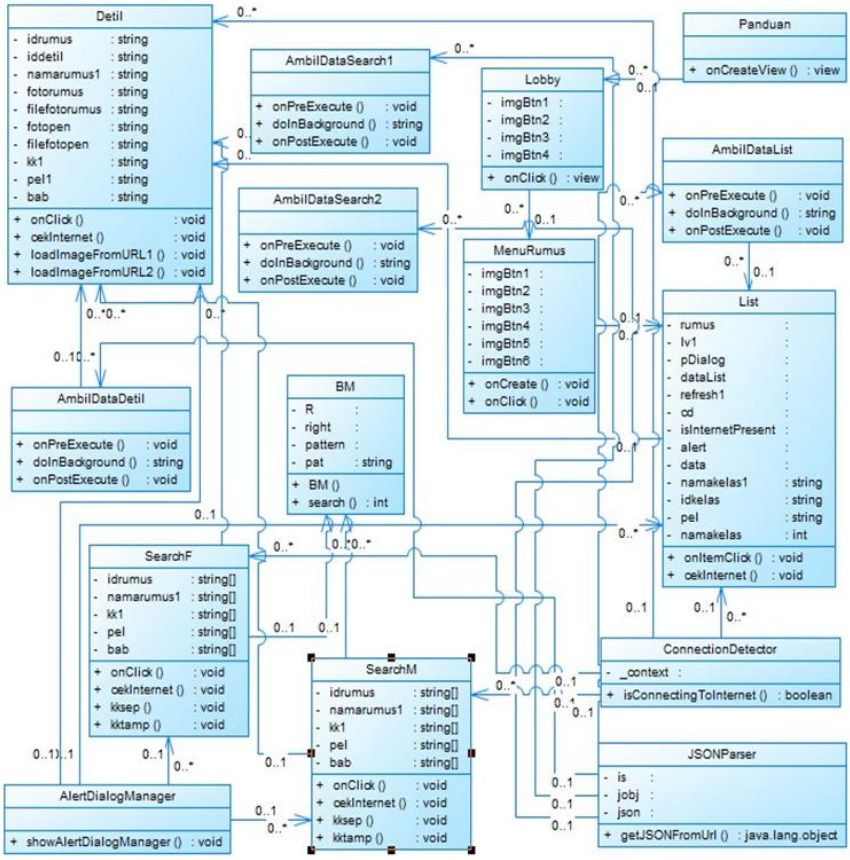


Gambar 4. *Use Case Diagram* Aplikasi

III.3 Class Diagram

Desain aplikasi ini dibuat berdasarkan class diagram seperti pada gambar 5, dengan penjelasan sebagai berikut:

1. Kelas *lobby* memiliki hubungan dengan kelas panduan, menu rumus, *search* matematika, dan *search* fisika. Karena kelas lobby adalah menu utama dari aplikasi ini. Memiliki 4 tombol gambar untuk akses ke panduan, 2 *search*, serta menu rumus.
2. Kelas panduan hanyalah kelas biasa berisikan teks panduan bagi *user*.
3. Kelas menu rumus adalah menu cari rumus secara manual, memiliki hubungan dengan kelas list. Kelas menu rumus memiliki 6 tombol gambar dengan masing-masing matematika dan fisika tingkat SMA.
4. Kelas menu *list* memiliki hubungan dengan JSONParser, Detil, AlertDialogManager, ConnectionDetector, dan AmbilDataList. Karena menu *list* ini sudah harus memakai koneksi internet
5. JSONParser memiliki hubungan dengan semua kelas yang memiliki permintaan mengambil data dari database. Seperti AmbilData, List, Detil, Search Matematika, dan Search Fisika. JSONParser adalah kelas jembatan antara MySQL dengan Android.

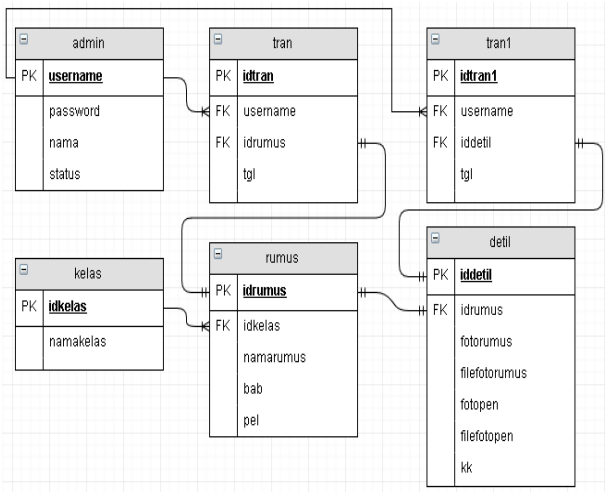


Gambar 5. *Class Diagram* Aplikasi

6. Kelas semacam AmbilData adalah kelas mengambil data dari *database* dengan 3 void. Karena *database* yang digunakan adalah MySQL dan Android tidak mengenal bahasanya, maka dibutuhkan kelas dengan nama JSONParser. onPreExecute() adalah proses mengirim URL ke *database*. doInBackground() adalah proses dibelakang layar dengan mengambil data hasil respon *database* lewat URL. onPostExecute() adalah proses menampung data ke dalam variabel tertentu.
7. Kelas SearchF dan SearchM adalah kelas *search* dengan tujuan yang sama namun beda pelajaran. Memiliki hubungan dengan AmbilDataSearch1, AmbilDataSearch2, BM, serta Detil.
8. BM adalah kelas algoritma *Boyer-Moore* yang berperan penting dalam aplikasi ini untuk bagian *search* rumus. BM memiliki hubungan dengan kelas SearchF dan SearchM.
9. Detil adalah kelas terakhir dan memiliki hubungan dengan SearchF, SearchM, dan List. Detil berperan penting dalam menampilkan detil rumus hasil dari ketiga kelas tersebut.
10. AlertDialogManager dan ConnectionDetector adalah kelas pendeteksi koneksi internet sekaligus validasi apakah terhubung atau tidak. Bila tidak maka sistem akan menampilkan notifikasi berupa tidak ada koneksi internet.

III.4 *Entity Relationship Diagram* (ERD)

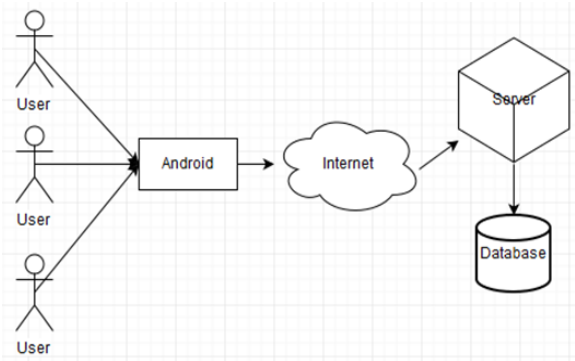
ERD pada gambar 6 menampilkan implementasi basis data yang digunakan pada aplikasi ini. Aplikasi ini menggunakan 6 tabel dengan dua tabel (rumus dan detil) digunakan untuk menyimpan data utama pada aplikasi ini.



Gambar 6. ERD Aplikasi

III.5 Arsitektur Sistem

Diagram arsitektur sistem menjelaskan interaksi sistem dengan pengguna. Dalam kasus ini pengguna berinteraksi dengan Android. Android terhubung ke internet. Internet digunakan untuk mengakses *server* dimana *server* memiliki *database* dari sistem. Arsitektur nya ditunjukkan pada gambar 7.



Gambar 7. Arsitektur Sistem

III.6 Pseudocode Boyer-Moore

Algoritma *Boyer-Moore* untuk mendeteksi apakah ada kata kunci yang terdapat pada soal yang dimasukkan *user*. Berikut *pseudocode* algoritma *Boyer-Moore* seperti terlihat pada gambar 1, gambar 2 dan gambar 3.

```
Algoritma Boyer-Moore fase pra-pencocokkan, preBmBc.
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 8. *Pseudocode Algoritma Boyer-Moore preBmBc*

```
Algoritma Boyer-Moore fase pra-pencocokkan, preSuffixes.
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
```

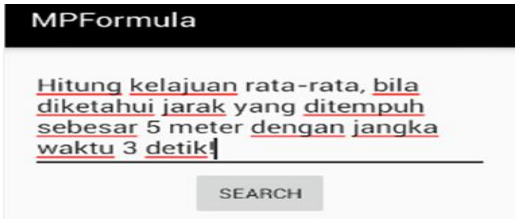
Gambar 9. *Pseudocode Algoritma Boyer-Moore preSuffix*

```
Algoritma Boyer-Moore fase pra-pencocokkan, preBmGs.
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
Algoritma:
        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] = 1 + 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 10. *Pseudocode Algoritma Boyer-Moore preBmGs*

IV. HASIL PENELITIAN DAN PEMBAHASAN

Pada Aplikasi yang sudah mengimplementasikan algoritma *Boyer-Moore*, Pengguna dapat menjalankan aplikasi dan memilih menu *search* yang sesuai dengan keinginan *user*. Pengguna memasukkan soal seperti pada gambar 11:



Gambar 11. Tampilan *Input* Soal

Kemudian setelah selesai memasukkan soal, *user* menekan tombol *search*. Kemudian sistem akan mengubah data kata kunci yang diambil dari *database* menjadi per kata melalui fungsi pada gambar 12.

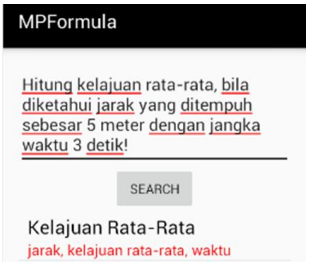
```
public void kksep()
{   for(int x=0; x<dataList.size(); x++) {
      String kk2 = kk1[x].toString().replace(",", "");
      b = 1;
      int flg=0;
      huruf = "";
      String allkk = kk2 + " ";
      //Log.e("allkk", String.valueOf(allkk.length()));
      for (i = 0; i < allkk.length(); i++) {
        String cek = allkk.substring(b - 1, b);
        //bukan spasi
        if (!cek.equals(" ")) {
          huruf += cek;
          //Log.e("bc", huruf);  }
        else
        {  flg++;
          //Log.e("All", String.valueOf(flag));
          kata1[index] = huruf;
          huruf = "";
          index++; }
        b++; }
flag[x] = flg; }}
```

Gambar 12. Kode Program Pemisah Kata

Setelah kata kunci dipisah per kata melalui fungsi kksep(). Selanjutnya, sistem memasukkan setiap kata yang sudah dipisah per kata dimasukkan ke dalam variabel tamp. Selanjutnya masuk ke fungsi onClick() dimana kalimat yang dimasukkan pengguna diubah menjadi huruf kecil. Selanjutnya, sistem akan mengambil hasil tampungan kata kunci beserta dengan kalimat yang dimasukkan pengguna untuk dicocokkan menggunakan algoritma *Boyer-Moore*. Kode Program dari Algoritma *Boyer Moore* dapat dilihat pada gambar 13.

```
public class BM {
   private int R;    // the radix
   private int[] right;    // the bad-character skip array
   private char[] pattern;  // store the pattern as a
character array
   String pat;     // or as a string
   public BM(String pat) {
      this.R = 256; this.pat = pat;
      Log.e("pat",pat);
      // position of rightmost occurrence of c in the pattern
      right = new int[R];
      for (int c = 0; c < R; c++)
         right[c] = -1;
      for (int j = 0; j < pat.length(); j++)
         right[pat.charAt(j)] = j;  }
   public BM(char[] pattern, int R) {
      this.R = R;
      this.pattern = new char[pattern.length];
      for (int j = 0; j < pattern.length; j++)
         this.pattern[j] = pattern[j];
      // position of rightmost occurrence of c in the pattern
      right = new int[R];
      for (int c = 0; c < R; c++)
         right[c] = -1;
      for (int j = 0; j < pattern.length; j++)
         right[pattern[j]] = j; }
   public int search(String txt) {
      int M = pat.length();
      int N = txt.length();
      int skip;
      for (int i = 0; i <= N - M; i += skip) {
         skip = 0;
         for (int j = M-1; j >= 0; j--) {
            if (pat.charAt(j) != txt.charAt(i+j)) {
               skip = Math.max(1, j - right[txt.charAt(i+j)]);
               break;}  }
         if (skip == 0) return i;    // found  }
      return N;                    // not found  }
   public int search(char[] text) {
      int M = pattern.length;
      int N = text.length;
      int skip;
      for (int i = 0; i <= N - M; i += skip) {
         skip = 0;
         for (int j = M-1; j >= 0; j--) {
            if (pattern[j] != text[i+j]) {
               skip = Math.max(1, j - right[text[i+j]]);
               break; }  }
         if (skip == 0) return i;    // found  }
      return N;        // not found  } }
```

Gambar 13. Kode Program Algoritma *Boyer-Moore*



Gambar 14. Hasil Pencocokkan Menggunakan Algoritma *Boyer-Moore*

Kemudian untuk membuktikan algoritma *Boyer-Moore* dapat berfungsi dengan baik pada aplikasi rumus matematika dan fisika maka dilakukan pengujian. Pengujian algoritma dilakukan dengan melakukan identifikasi kata kunci rumus pada input berupa soal atau beberapa kata untuk mendapatkan kata kunci yang sesuai dan apakah hasil rumus yang ditampilkan sudah sesuai dengan kata kunci. Tabel 1 menampilkan beberapa hasil pengujian dari 30 pengujian yang dilakukan pada penelitian ini.

Tabel 1. Tabel Pengujian Algoritma Boyer Moore

No.	Soal	Kata kunci yang ada pada soal	Hasil *search*	Rumus yang seharusnya	Rumus yang cocok dengan soal	Valid
1	Berapa kecepatan rata-rata sebuah helikopter jika helikopter berangkat dari helipad rumah sakit ke arah timur sejauh 120 km dalam 1 jam.	Kecepatan rata-rata	Kecepatan Rata-Rata (Gerak Lurus) Kecepatan Sesaat (Gerak Lurus) Kecepatan Rata-Rata (Kinematika) Kecepatan Sesaat (Kinematika) Percepatan Rata-Rata (Gerak Lurus) ZXPercepatan Rata-Rata (Kinematika)	Kecepatan Rata-Rata (Gerak Lurus)	Kecepatan Rata-Rata (Gerak Lurus) Kecepatan Rata-Rata (Kinematika)	100%
2	Seorang pelaut melompat ke laut dari geladak kapal yang ketinggiannya 10 m untuk menyelamatkan seorang penumpang yang terjatuh dari geladak. Berapa detik yang diperlukannya untuk sampai di air?	Tinggi, jatuh, detik, air	Persamaan Gerak Jatuh Bebas (Gerak Lurus) Tekanan (Fluida Statis) Tekanan	Persamaan Gerak Jatuh Bebas (Gerak Lurus)	Persamaan Gerak Jatuh Bebas (Gerak Lurus)	100%

No.	Soal	Kata kunci yang ada pada soal	Hasil *search*	Rumus yang seharusnya	Rumus yang cocok dengan soal	Valid
			Hidrostatis (Fluida Statis) Debit (Fluida Dinamis)			
3	Sebuah kipas angin berputar dengan kecepatan 900 rpm (*rotation per minute* = putaran per menit). Berapakah laju titik di ujung baling-baling, jika panjang baling-baling 20 cm?	Berputar, kecepatan, laju, putaran, laju, panjang	Kecepatan Rata-Rata (Gerak Lurus) Kecepatan Sesaat (Gerak Lurus) Kecepatan Linear (Gerak Melingkar Beraturan) Kecepatan Pada Gerak Melingkar Percepatan Sentripetal (Gerak Melingkar Beraturan)	Kecepatan Pada Gerak Melingkar	Kecepatan Pada Gerak Melingkar	100%
4	Sebuah benda yang tingginya 10 cm diletakkan 60 cm di depan cermin cekung yang jarak fokusnya 15 cm. Di manakah letak bayangan?	Tinggi, cermin, cekung, jarak, fokus, bayangan	Perbesaran Linear (Optika Geometris) Cermin Lengkung (Optika Geometris) Pembiasan Cahaya (Optika Geometris)	Cermin Lengkung (Optika Geometris)	Cermin Lengkung (Optika Geometris)	100%
5	Sebuah benda bergetar harmonis dengan persamaan y = 10 sin πt dengan y dalam cm dan t dalam sekon. Tentukanlah amplitudo.	Bergetar, harmonis, persamaan, sin, sekon, amplitudo	Persamaan Gerak Harmonik Sederhana	Persamaan Gerak Harmonik Sederhana	Persamaan Gerak Harmonik Sederhana	100%
6	Berapa usaha yang	Usaha,	Frekuensi Pegas, Periode Bandul Usaha (Usaha & Energi)	Teorema Usaha-	Teorema	100%

No.	Soal	Kata kunci yang ada pada soal	Hasil *search*	Rumus yang seharusnya	Rumus yang cocok dengan soal	Valid
	dilakukan untuk menggerakkan sebuah mobil bermassa 2.000 kg dari keadaan diam hingga mencapai kelajuan 25,0 m/s?	bermassa, kg, kelajuan, m/s		Energi (Usaha & Energi)	Usaha-Energi (Usaha & Energi)	
			Energi Kinetik (Usaha & Energi) Teorema Usaha-Energi (Usaha & Energi) Daya (Usaha & Energi) Energi Potensial (Usaha & Energi)			
7	Sebuah bejana berbentuk silinder diisi fluida 200 cm^3 dengan massa jenis kg/m^3. Luas alas bejana 20 cm^2. Hitunglah: a. tekanan pada dasar bejana, b. gaya yang bekerja pada dasar bejana.	Fluida, massa jenis, kg/m^3, luas, tekanan, gaya	Tekanan (Fluida Statis)	Tekanan (Fluida Statis)	Tekanan (Fluida Statis)	100%
			Tekanan Hidrostatis (Fluida Statis) Hukum Archimedes (Fluida Statis) Tegangan Permukaan (Fluida Statis) Kapilaritas (Fluida Statis) Hukum Stokes (Fluida Statis)	Tekanan Hidrostatis (Fluida Statis)	Tekanan Hidrostatis (Fluida Statis)	
8	Pada keadaan normal (t = 0^oC, P = 1 atm), berapakah volume 4 gram gas oksigen O_2?	Atm, volume, gas	Hukum Boyle-Gay Lussac (Teori Kinetik Gas)	Persamaan Umum Gas Ideal (Teori Kinetik Gas)	Persamaan Umum Gas Ideal (Teori Kinetik Gas)	100%
			Persamaan Umum Gas Ideal (Teori Kinetik Gas) Tekanan Gas pada Ruang Tertutup (Teori Kinetik Gas) Kecepatan Efektif Gas Ideal (Teori Kinetik Gas)			
9	Nilai yang memenuhi persamaan: 2^{x+1} =	Nilai, persamaan	Persamaan Eksponensial	Persamaan Eksponensial	Persamaan Eksponensial	100%

No.	Soal	Kata kunci yang ada pada soal	Hasil *search*	Rumus yang seharusnya	Rumus yang cocok dengan soal	Valid
	8 adalah...		Pertidaksamaan Eksponensial Persamaan Logaritma Pertidaksamaan Logaritma			
10	Himpunan penyelesaian pertidaksamaan x^2 < 9 adalah...	pertidaksamaan	Pertidaksamaan Kuadrat, Ptk (Pertidaksamaan Nonlinear) Pertidaksamaan Pecahan, PtP (Pertidaksamaan Nonlinear) Pertidaksamaan Irasional, PtI (Pertidaksamaan Nonlinear) Aturan Turunan (Turunan Fungsi Trigonometri)	Pertidaksamaan Kuadrat, Ptk (Pertidaksamaan Nonlinear)	Pertidaksamaan Kuadrat, Ptk (Pertidaksamaan Nonlinear)	100%

V. KESIMPULAN DAN SARAN

Dari hasil penelitian diperoleh kesimpulan bahwa metode *string matching* dengan algoritma *boyer moore* dapat diterapkan pada aplikasi pencarian rumus matemtika dan fisika tingkat SMA yang disertai dengan pengujian algoritma sebanyak 30 kali dimana aplikasi dapat mencocokkan pola kata kunci pada soal serta menampilkan list atau kumpulan rumus matematika dan fisika tingkat SMA yang cocok dan sesuai dengan soal yang dikerjakan.

Untuk penelitian berikutnya yang dapat dikembangkan adalah dengan membuat penelitian yang membuat aplikasi yang dapat menjawab soal matematika dan fisika dengan disertai tahapan penyelesaian soal dan jawaban.

REFERENSI

Argakusumah, K.W., & Hansun, S. (2014). Implementasi Algoritma Boyer-Moore pada Aplikasi Kamus Kedokteran Berbasis Android. Jurnal ULTIMATICS Vol.6. No.2.

Arhami, A. D. (2006). Konsep Kecerdasan Buatan. Yogyakarta: Andi.

Aulia, R. (2008). Analisis Algoritma Knuth Morris Pratt dan Algoritma Boyer Moore dalam Proses Pencarian String.

Bassil, Youssef, (2012), *A Simulation Model for the Waterfall Software Development Life Cycle*, *International Journal of Engineering & Technology* (iJET), Vol.2. No.5.

Chiquita, Christabella. (2012). Penerapan Algoritma Boyer-Moore Dynamic Programming untuk Layanan Auto-Complete dan Auto Correct. Bandung.

Rahmanita, E. (2014). Pencarian String Menggunakan Algoritma Boyer Moore Pada Dokumen. Jurnal Ilmiah NERO Vol.1. No.1. hal 15-26.

Ramadhansyah. (2013). Perancangan Aplikasi Kamus Bahasa Gayo Dengan Menggunakan Metode Boyer Moore. Jurnal Pelita Informatika Budi Darma Vol.4. No.3. hal 118-122.

Utomo, D., Harjo, E.W., & Handoko. (2008). Perbandingan Algoritma String Matching Brute Force, Knuth Morris Pratt, Boyer Moore dan Karp Rabin Pada Teks Alkitab Bahasa Indonesia. Jurnal Ilmiah Elektroteknika Vol.7. No.1. hal 1-13.