VOLUME 10 - NOMOR 3 - DESEMBER 2017

ISSN 1979-4835

JURNAL
Techno-Socio Ekonomika

Jurnal Ilmu-Ilmu Ekonomi-Sosial dan Teknologi

JURNAL	VOLUME	NO	HALAMAN	BANDUNG	ISSN
USB--YPKP	10	3	224 - 333	DESEMBER 2017	1979-4835

ISSN 1979-4835

OPTIMALISASI PERBANDINGAN ALGORITMA *BRUTE FORCE* DAN *KNUTH-MORRIS-PRATT* UNTUK MENINGKATKAN KECEPATAN PENCARIAN DATA PADA APLIKASI MOBILE TENTANG HEWAN VERTEBRATA

Beki Subaeki [1], Asep Muhammad Indra Purnama, [2]

Abstrak

Penggunaan kamus secara cetak belakangan mulai ditinggalkan semenjak kehadiran teknologi yang bisa menjawab permasalahan yang komplek. Semisal kehadiran *Handphone* belakangan menjadi kebutuhan wajib bagi tiap pengguna. Ini dikarenakan kemudahan dan efesiensi dalam menggunakan teknologi tersebut. Terlebih *Handphone* sudah ditanam SO android yang sifatnya open source. Tidak heran para pengembang menciptakan aplikasi yang saling bersaing meraih perhatian masyarakat dengan menciptakan segala manfaat dan fitur fitur yang ditawarkan. Pada penelitian ini metode yang digunakan dalam proses penelitian adalah dengan model *prototype*. Model ini perancang dan pengguna bertemu untuk mendefinisikan secara objektif keseluruhan perangkat lunak, mengidentifikasi kebutuhan yang diketahui, kemudian dilakukan perancangan berupa prototipe. Dari hasil penelitian maka didapatkan untuk penerpan algoritma KMP dalam *method* kontruksi pencocokan algoritma menggunakan algoritma KMPNext. Dengan parameter *text* dan *pattern* dalam bentuk *string* yang kemudian dilakukan pencocokan pada dua *variable failure*. Proses selanjutnya ialah mencari kesamaan dari *text* dan *pattern* yang kemudian akan diketahui di index berapa *text* dan *pattern* tersebut memiliki kesamaan. Untuk implementasi algoritma *Method* ini berfungsi untuk mencari kesamaan antara *text* dan *pattern*. Pemeriksaan kecocokan dilakukan perkarakter dari *text* dan *pattern*. *Method* ini mengembalikan nilai iterasi a dan b untuk kemudian di proses pada *method* konstruktor untuk dicari *index* dimana terdapat kesamaan *string*. Terakhir kesimpulan yang dapat ditarik adalah dalam algoritma *Knuth-morris-pratt* dan *Brute Force* tidak ada yang efisien atau pada segi waktu. Dari hasil perhitungan aplikasi *Brute Force* lebih cepat dibandingkan dengan *Knuth-morris-pratt*. Sedangkan KMP unggul dari perhitungan kompleksitas waktu $O(n^3)$ dari algoritma *Brute Force* $O(n^2)$.

Kata kunci : *Android*, *Knuth-Morris-Prath*, *Brute Force*, Hewan *Vertebrata*.

I. PENDAHULUAN

1.1 Latar Belakang Masalah

Pada zaman modern seperti sekarang ini, kamus cetak sudah mulai banyak yang meninggalkannya karena kebanyakan orang sudah beralih ke kamus digital yang bisa di *install* di *headphone* yang berbasis android.[1] Selain lebih mudah penggunaannya juga lebih cepat dibandingkan dengan kamus cetak yang harus mencarianya secara manual.

Dalam upaya untuk memperkenalkan keanekaragaman jenis hewan *vertebrata* (tulang belakang) dengan memanfaatkan teknologi komputer, maka dihadirkan suatu Aplikasi *mobile* berbentuk kamus hewan *vertebrata* sebagai media penambah wawasan dalam konteks nama hewan *vertebrata* (tulang belakang) tersebut.

Pada Aplikasi *mobile* kamus *vertebrata* diperlukan metode Algoritma pencocokkan *string* digunakan untuk menangani pencarian data pada sebuah aplikasi. Pencarian pada umumnya berbentuk teks dimana proses pencarian dilakukkan dengan mencocokkan data yang dimasukkan oleh pengguna dengan data yang sudah tersimpan pada aplikasi. Setiap teknologi yang digunakan pada aplikasi memiliki jenis penyimpanan yang berbeda-beda untuk setiap datanya, tapi teks tetap merupakan bentuk utama penyimpanan data.[2]

Salah satu algoritma yang digunakan pada Aplikasi kamus *vertebrata* berbasis android yaitu algoritma Knuth-Morris-Pratt (KMP) dan Brute Force (BrF), terdapat dua metode yang menentukan algoritma manakah yang bagus.

Algoritma yang bagus adalah algoritma yang mangkus atau efesien diukur dari berapa jumlah waktu dan ruang memori yang dibutuhkan untuk menjalankannya. Sedakankan algoritma yang mangkus adalah algoritma yang meminimumkan kebutuhan waktu dan ruang. Disetiap

algoritma memiliki kelebihan dan kekurangannya dalam mengimplementasikan suatu algoritma dengan cara yang berbeda-beda, namun algoritma dapat dianalisis, efesiesi, dan kompleksitasnya.[3]
Algoritma merupakan urutan langkah-langkah yang dinyatakan dengan efesien dan tidak rancu untuk memecahkan suatu masalah dalam rentang waktu tertentu. Artinya, setiap langkah harus dapat dikerjakan dan mempunyai efek tertentu.[4] Pada umumnya beberapa algoritma diciptakan untuk meningkatkan kinerja suatu aplikasi yang menggunakannya dalam menangani pekerjaan tertentu, karena algoritma adalah urutan langkah-langkah untuk memecahkan suatu masalah.[4]

1.2 Rumusan Masalah

1. Bagaimana perbandingan kompleksitas Algoritma *Knuth Morris Pratt* dan *Brute Force* dalam segi efesien untuk pencarian data pada Aplikasi *Mobile* mengenai hewan *vertebrata*?
2. Bagaimana implementasi Algoritma *Knuth Morris Pratt* dan *Brute Force* pada Aplikasi android untuk mengenali teks pada hewan *vertebrata* ?

1.3 Tujuan Penelitian

Batasan-batasan masalah dalam pembuatan bahan ajar sistem pencernaan ini mencakup

1. Membandingkan kompleksitas algoritma *Knuth-morris-pratt* dan *Brute force* dalam segi waktu untuk pencarian data pada Aplikasi *mobile* mengenai hewan *vertebrata.*
2. Menampilkan informasi mengenai hewan *vertebrata.*
3. Implementasi *Knuth-morris-pratt* dan *Brute Force* pada android.

1.4 Batasan Masalah

Adapun tujuan penulis dalam pembuatan bahan ajar ini adalah sebagai berikut :

1. Data dan Informasi yang disediakan mengenai hewan *vertebrata* berupa teks dan gambar.
2. Mengimplementasikan algoritma *Knuth morris pratt* dan *Brute force* untuk pencarian data.
3. Terdapat jumlah 167 data dari hewan *vertebrata.*
4. Dapat dioperasikan pada *smartphone* berbasis sistem operasi Android versi 4.1.2.
5. Menggunakan *tools* Java SE *Development Kit* versi 1.7, Android *Software Development Kit* (SDK), IDE Eclipse Juno, *Android Development Tools* (ADT) dan *Device Platform*

II. Landasan Teori

2.1 Android

Android adalah sistem operasi yang berbasis *Linux* untuk telepon seluler seperti telepon pintar dan komputer tablet. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, *Google Inc.* membeli *Android Inc.*, pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

2.2 OOP *(Objec Oriented Programming)*

Objek Oriented Program atau pemrograman berorientasi objek adalah konsep pemrograman yang difokuskan pada penciptaan kelas yang merupakan abstraksi/*blueprint*/*prototype* dari suatu objek. Kelas ini harus mengandung sifat (data) dan tingkah laku (method) umum yang dimiliki oleh objek-objek yang kelak

akan dibuat (diinstansiasi). Data dan method merupakan anggota (*member*) dari suatu kelas.

Ada enam keuntungan yang diperoleh bila menggunakan *Objek Oriented Program* (OOP) diantaranya:

1. Alami (*Natural*)
2. Dapat diandalkan
3. Dapat digunakan kembali
4. Mudah untuk di maintain (*Maintainable*)
5. Dapat diperluas
6. Efisiensi waktu

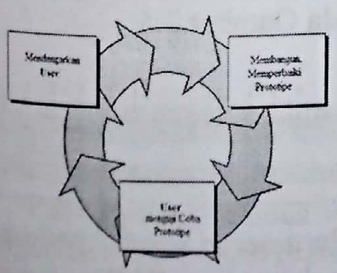
2.3 UML (*United Modified Language*)

UML (*Unified Model Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. *UML* merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modelling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan interatif, yaitu identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi.

III. Metode Penelitian

3.1. Metodologi Penelitian

Guna mendapatkan data yang diperlukan untuk membantu dalam penelitian yang akan dilakukan, maka digunakan metodologi sebagai berikut:



Gambar 3.1. Paradigma *Prototyping* [7]

Dengan model *prototype* ini perancang dan pengguna bertemu untuk mendefinisikan secara objektif keseluruhan perangkat lunak, mengidentifikasi kebutuhan yang diketahui, dan area lebih besar dimana definisi lebih jauh merupakan keharusan kemudian dilakukan perancangan kilat berupa maket atau prototipe sistem untuk kemudian dievaluasi pengguna untuk menyaring kebutuhan pengembangan perangkat lunak.

3.2. Analisis Kebutuhan *Non-fungsional*

Kebutuhan *non-fungsional* yang dibutuhkan untuk membangun system ini terdiri dari tiga hal, yaitu: kebutuhan perangkat keras, kebutuhan perangkat lunak dan analisis pengguna dari aplikasi.

3.3. Analisis Kebutuhan Fungsional

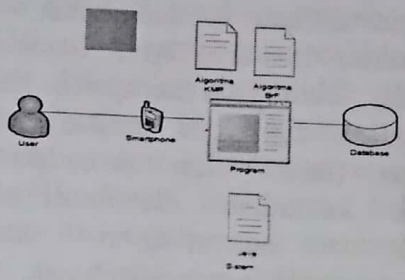
Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah system dan menjelaskan kebutuhan yang diperlukan system agar system dapat berjalan dengan baik.

Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Tahap-tahap pemodelan dalam analisis tersebut antara lain deskripsi global dari aplikasi, arsitektur system yang akan dibangun, arsitektur aplikasi dan juga arsitektur algoritma yang akan digunakan.

3.4. Arsitektur Sistem

Aplikasi yang akan dibuat merupakan aplikasi yang berjalan pada perangkat *smartphone* android, terdapat satu buah aktor yaitu *user*, terdapat *smartphone* yang dapat menyimpan semua data hewan *vertebrata* yang berada di *database*.

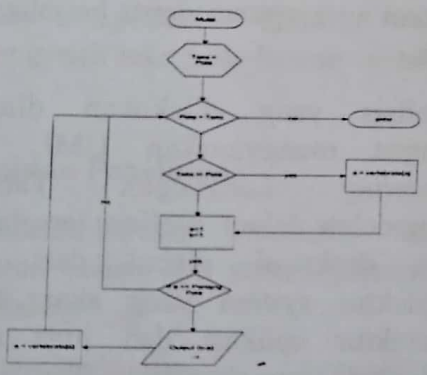
User sebagai anak-anak ataupun dewasa informasi tentang hewan *vertebrata* menggunakan *smartphone* android, *user* yang menjalankan aplikasi yang kemudian apliaksi tersebut memunculkan fitur-fitur yang ada didalamnya dengan mengakses data dari *database*.



Gambar 3.2 Arsitektur *system*

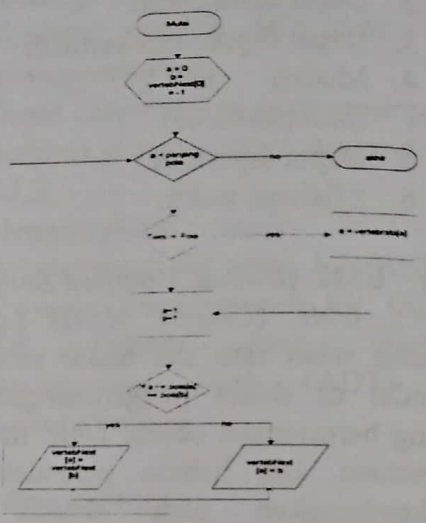
3.4.1 Perancangan Algoritma *KMP*

Pada Aplikasi kamus *vertebrata* yang akan dibangun menggunakan sebuah algoritma pencocokan *string* dalam fitur pencariannya. Algoritma yang digunakan yaitu algoritma pencarian kata *knuth-morris-prath* (KMP). Secara umum pencocokan yang akan dilakukan pada Aplikasi kamus *vertebrata* seperti pada Gambar 3.3.



Gambar 3.3 Perancangan algoritma *kmpnext*

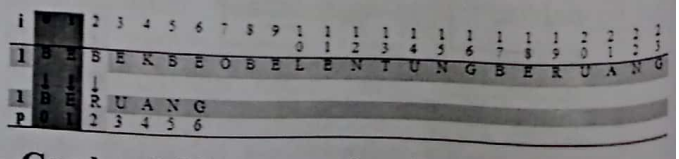
Pada Gambar 3.3 pertama-tama mulai bila a dan b sama-sama nol bila a lebih kecil dari teks maka akan dilanjutkan ke a lebih besar dari -1 dan pola a tidak sama dengan teks b, bila tidak sama maka akan di lanjut ke a sama dengan *vertebNext*[a] bila tidak maka akan dilanjut pertambahan a tambah 1 dan b tambah 1 juga, jika a lebih besar sama dengan panjang pola keluar b-a dan a sama dengan *vertebNext*[a] lalu mengulang lagi ke b lebih kecil dari teks dan bila b lebih kecil dari teks maka akan berakhir.



Gambar 3.4 Perancangan algoritma *kmpsearch*

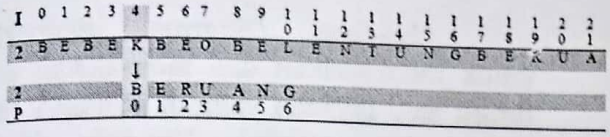
Pada Gambar 3.4 pertama-tama mulai a sama dengan 0, b sama dengan *vertebNext*[0] sama dengan 1, pengulangan a lebih kecil dengan panjang pola bila benar maka b lebih besar dari min 1 dan pola [a] tidak sama dengan [b] bila tidak sama maka benar b sama dengan *vertebNext* [0] lanjut ke a ditambah 1 dan b ditambah 1, melakukan pengulangan jika pola[a] sama dengan pola[b] bila benar *vertebNext*[a] sama dengan *vertebNext* [0], bila tidak sama maka *vertebNext* [a] = b, mengulang kembali ke a lebih besar dari pola bila i-nya lebih besar dari pola maka akan berakhir.

Terdapat skema pencarian *string knuth-morris-pratt* pada Gambar 3.5



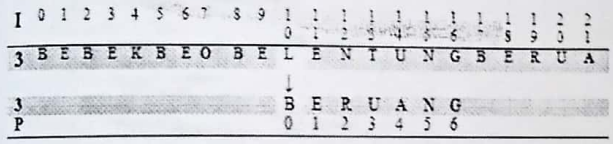
Gambar 3.5 Pergeseran kondisi pertama dengan algoritma KMP

Kondisi 1. Pada kondisi pertama Teks(B) dengan Pola(R) tidak sesuai maka akan di geser. Pada penggeseran panjang pola 6 – 2 = 4, di mulai pada indeks ke 4.



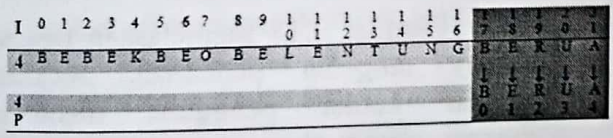
Gambar 3.6 Pergeseran kondisi kedua dengan algoritma KMP

Kondisi 2. Pada kondisi kedua Teks(K) dengan Pola(B) tidak sesuai maka akan di geser kembali. Pada penggeseran panjang pola 6 – 0 = 6, di mulai pada indeks ke 10.



Gambar 3.7 Pergeseran kondisi ketiga dengan algoritma KMP

Kondisi 3. Pada kondisi kedua Teks(L) dengan Pola(B) tidak sesuai maka akan di geser kembali. Pada penggeseran panjang pola 6 – 0 = 6, di mulai pada indeks ke 17.

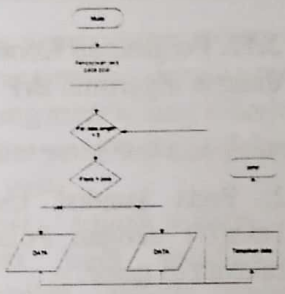


Gambar 3.8 Pergeseran kondisi keempat dengan algoritma KMP

Kondisi 4. Pada kondisi kelima ini karakter (B) pada pola dengan Teks Beruang sesuai maka pencocokan string telah selesai ditemukan.

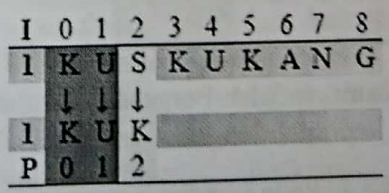
3.4.2. Perancangan Algoritma *Brute Force*

Pada Aplikasi kamus *vertebrata*, selain algoritma *knuth-morris-pratt* (KMP) yang akan dibangun menggunakan sebuah algoritma pencocokan *string* dalam fitur pencariannya. Algoritma yang digunakan yaitu algoritma pencarian kata *Brute Force (BrF)*. Hal ini dilakukan untuk mengetahui bagaimana kinerja pencarian dalam Aplikasi kamus *vertebrata* setelah menggunakan algoritma tersebut dengan tanpa menggunakan metode khusus. Secara umum pencocokan yang akan dilakukan pada Aplikasi kamus *vertebrata* seperti pada Gambar 3.10.



Gambar 3.10 *Flowchart* arsitektural algoritma BrF

Pada gambar 3.10 menjelaskan dimana algoritma dimulai ketika pengguna memasukkan *pattern* kemudian system akan mengambil teks yang ada di dalam penyimpanan data *vertebrata* yang kemudian akan dicocokan teks *pattern*-nya bila kondisi cocok dan tidak sama dengan nol itu salah maka *pattern* akan langsung berakhir tapi jika kondisinya benar maka system akan di cek dan dicocokan bila cocok maka data akan disimpan dan bila salah akan berakhir. Terdapat skema pencarian *string Brute Force* pada Gambar 3.11.



Gambar 3.11. Pergeseran Kondisi pertama dengan algoritma BrF

Kondisi 1. Pada kondisi pertama karakter Teks(K) dengan Pola(K) sesuai, karakter Teks(U) dengan Pola(U) sesuai, pada karakter Teks(S) dengan Pola(K) tidak sesuai maka akan di geser. Pada penggeseran bruteforce semua teks akan di coba satu persatu sampe sesuai dengan pola.

I	0	1	2	3	4	5	6	7	8
2	K	U	S	K	U	K	A	N	G
		↓							
2		K	U	K					
P		0	1	2					

Gambar 3.12. Pergeseran Kondisi kedua dengan algoritma BrF

Kondisi 2. Pada kondisi kedua pada karakter Teks(U) dengan Pola(K) tidak sesuai maka akan di geser.

I	0	1	2	3	4	5	6	7	8
3	K	U	S	K	U	K	A	N	G
			↓						
3			K	U	K				
P			0	1	2				

Gambar 3.13. Pergeseran Kondisi ketiga dengan algoritma BrF

Kondisi 3. Pada kondisi ketiga pada karakter Teks(S) dengan Pola(K) tidak sesuai maka akan di geser.

I	0	1	2	3	4	5	6	7	8
4	K	U	S	K	U	K	A	N	G
				↓	↓	↓			
4				K	U	K			
P				0	1	2			

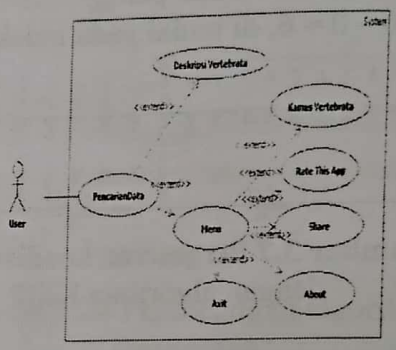
Gambar 3.14. Pergeseran Kondisi keempat dengan algoritma BrF

Kondisi 4. Pada kondisi keempat pada karakter Teks(K,U,K) dengan Pola(K,U,K) sesuai maka pencarian teks selesai.

3.5. Perancangan Model Fungsional

3.5.1 *Use-case Diagram*

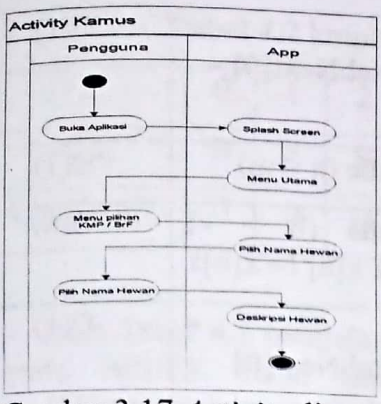
Use-case Diagram digunakan untuk memodelkan dan menyatakan unit fungsi atau layanan yang di sediakan oleh system ke pemakai. Use-case dibuat berdasar keperluan actor, merupakan "apa" yang dikerjakan *system*, bukan "bagaimana" system mengerjakannya. *Use-case* diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*.



Gambar 3.16 *Use-case* aplikasi

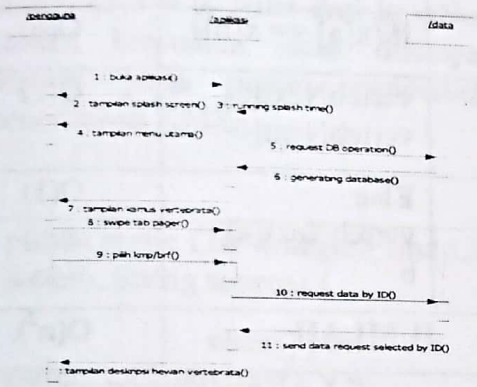
3.5.2 *Activity Diagram*

Gambar 3.17 menjelaskan tentang *activity diagram* pencarian. Aktifitas dimulai ketika pengguna membuka aplikasi, sistem akan menampilkan *splash screen* dalam beberapa waktu kemudian akan masuk ke menu utama. Dalam menu utama terdapat beberapa pilihan *button* dan penggun akan memilih button yang bernama kamus. Sistem akan menampilkan list daftar nama hewan *vertebrata* yang bisa dipilih oleh pengguna melalui fitur *tab view*. Pengguna kemudian memilih nama hewan *vertebrata* yang kemudian akan dideskripsikan oleh aplikasi melalui sebuah tampilan yang memberikan beberapa informasi hewan vertebrata tersebut kepada pengguna.



Gambar 3.17 *Activity* diagram Pencarian

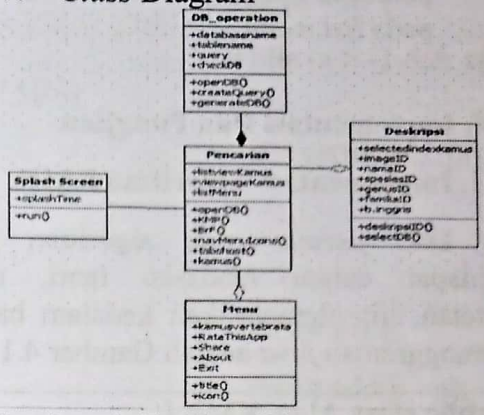
3.5.3 *Sequence Diagram*



Gambar 3.18 *Sequence diagram* pencarian

Gambar 3.18 mendeskripsikan tentang *sequence diagram*. dalam *sequence diagram* ini terdapat 3 objek entitas yang saling berhubungan yaitu pengguna, aplikasi, database. Pengguna merepresentasikan objek yang menjalankan aplikasi, aplikasi merepresentasikan *interface* sistem, database merepresentasikan objek yang berkaitan dengan data dalam *database*. Sedangkan algo_kmp dan algo_brf merupakan repsentasi dari proses pencarian string dengan algoritma *Khunt_Morris_Pratt* dan algoritma *BruteForce*. Dimulai dengan penguna yang memberikan *message* kepada aplikasi yang kemudian direspon dengan menampilkan *layout* yang dipesan oleh pengguna. Aplikasi dapat mengirim pesan untuk mengakses data kepada objek *database*.

3.5.4 *Class* Diagram



Gambar 3.19 *Class* Diagram

Dalam perancangan system ini terdapat 5 class yang masing-masing memiliki fungsi yang mendukung kinerja system. *Class-class* yang terdapat dalam *class* diagram yaitu:

a. *Class splash_screen*, merupakan *class* yang menjalankan fungsi tampilan *splash screen activity*, merupakan aktifitas awal yang dijalankan aplikasi. Memiliki 1 atribut *splash time* dan 1 *method run()* yang mempunyai fungsi untuk mengatur berapa lama aplikasi melakukan *loading data*.
b. *Class DB_operation*, merupakan *class* yang berfungsi untuk memungkinkan sistem dapat mengakses data dari *database*.
c. *Class* pencarian, merupakan *class* yang berfungsi untuk menampilkan daftar nama *vertebrata* beserta b.inggris yang terdapat dalam *database* dalam sebuah *list*.
d. *Class* deskripsi, merupakan *class* yang menampilkan deskripsi dari vertebrata. *Class* ini adalah generalisasi dari *class* kamus. *Class* kamus akan mengirimkan nomer identitas hewan yang ada di *database* kemudian *class* deskripsi akan menampilkan deskripsi dari *vertebrata* tersebut.
e. *Class Menu*, Merupakan class yang menampilkan *Menu* dari kamus *vertebrata*. *Class* ini adalah suatu info yang dapat melihat *profile*

pembuat dan *download* Aplikasi pada *list menu.*

IV. Implementasi Dan Pungjian

4.1. Implementasi Algoritma KMP

Dari *pseude code* algoritma yang terdapat dalam landasan teori, maka setelah diimplemetasikan kedalam bahasa pemoggraman *java* adalah Gambar 4.1.

```
public class Algo_KMP {
    private static void preKmp(char[] x, int[] vertebNext) {
        int a, b, m = (x.length - 1);
        a = 0;
        b = vertebNext[0] = -1;
        while (a < m) {
            while (b > -1 && x[a] != x[b])
                b = vertebNext[0];
            a++;
            b++;
            if (x[a] == x[b])
                vertebNext[a] = vertebNext[0];
            else
                vertebNext[a] = b;
```

Gambar 4.1 Implementasi algoritma kmp*Next*

Tabel 4.1 Perhitungan Big-O Algoritma *kmpNext*

Baris Intruksi	Operasi	**Pengulangan**
4	a = 0	O(1)
5	b = vertebNext[0] = -1	O(1)
6	**while** (a < m)	O(1)
7	**while** (b > -1 && x[a] != x[b])	O(n)
8	b = vertebNext[0]	O(1)
9	a++	O(1)
10	b++	O(1)
11	**if** (x[a] == x[b])	O(n)
12	vertebNext[a] = vertebNext[0]	O(1)
13	**Else** vertebNext[a] = b	O(1)
JUMLAH		**$O(n^2)$**

Berdasarkan hasil perhitungan performasi algoritma *kmpNext* menggunakan notasi Big-O didapat kompleksitas waktu dengan $O(n^2)$ dengan n adalah ukuran input. Yang mempengaruhi nilai kompleksitas waktu dari algoritma *kmpNext* adalah nilai n karena n berpangkat dua.

Dari Tabel 4.1 pengulangan operasi-operasi dasar pada algoritma kmpNext, didapat kompleksitas waktu sebagai berikut :

$$T(n) = \sum ti$$

$$T(n) = 1+1+1+1+1+1+1+1+2n$$

$$T(n) = 8 + 2n$$

$$T(n) = 2n + 8$$

Tabel 4.2 kmpNext

a	0	1	2	3	4	5
x[a]	B	E	B	E	K	
kmpNext[a]	-1	0	-1	-1	2	.0

Pada Tabel 4.1 menjelaskan pattern (x) yaitu BEBEK telah mempunyai nilai KMPNext setelah melakukan perhitungan menggunakan prefix KMP dengan nilai B = -1, E = 0, B = -1, E = -1, K = 2 dan indeks ke 5 = 0. Nilai dari kmpNext pada pattern kemudian akan disimpan dan dipakai pada proses pencarian atau pencocokan KMP.

```
public static List<Integer> findAll(String pattern, String source) {
        char[] ptrn = pattern.toCharArray(), y = source.toCharArray();
        char[] x = new char[ptrn.length + 1];
        System.arraycopy(ptrn, 0, x, 0, ptrn.length);
        int a, b, m = ptrn.length, n = y.length;
        List<Integer> result = new ArrayList<Integer>();

        int[] vertebNext = new int[x.length];

        /* Preprocessing */
        preKmp(x, vertebNext);

        /* Searching */
        a = b = 0;
        while (a < n) {
            while (a > -1 && x[a] != y[b])
                a = vertebNext[a];
            a++;
            b++;
            if (a >= m) {
                result.add(b - a);
                a = vertebNext[a];
            }
        }
        return result;
```

ambar 4.2 Implementasi algoritma *KMPSearch*

Tabel 4.3 Perhitungan Big-O Algoritma *kmpSearch*

Baris Intruksi	Operasi	Pengulangan
21	a = 0	O(1)
22	b = 0	O(1)
23	**while** (a < n)	O(1)
24	**while** (a > -1 && x[a] != y[b])	O(n)
25	a = vertebNext[a];	O(1)
26	a++	O(1)
27	b++	O(1)
28	**if** (a >= m)	O(1)
29	result.add(b - a);	O(1)
30	a = vertebNext[a];	O(1)
Jumlah		**O(n)**

Berdasarkan hasil perhitungan performasi algoritma *kmpSearch* menggunakan notasi Big-O didapat

kompleksitas waktu dengan O(n) dengan n adalah ukuran input. Yang mempengaruhi nilai kompleksitas waktu dari algoritma *kmpSearch* adalah nilai n.

Dari Tabel 4.1 pengulangan operasi-operasi dasar pada algoritma *kmpSearch*, didapat kompleksitas waktu sebagai berikut :

$T(n) = \sum ti$

T(n) = 1+1+1+1+1+1+1+1+1+n

T(n) = n + 9

Kompleksitas waktu suatu algoritma dapat juga ditentukan dengan membandingkan kompleksitas waktu algoritma tersebut dengan kompleksitas waktu yang presisi untuk suatu algoritma. Kompleksitas seperti ini disebut juga dengan kompleksitas waktu asimptotik yang dinyatakan dengan O-besar (0). Untuk algoritma

kmpNext $T(n) = \sum ti$

T(n) = 1+1+1+1+1+1+1+1+2n

T(n) = 8 + 2n

T(n) = 2n + 8

kmpSearch $T(n) = \sum ti$

T(n) = 1+1+1+1+1+1+1+1+n

T(n) = n + 9

Jadi T(n) = 2n + n + 8 + 9 (Di tambah)

$T(n) = 3n + 17 = O(n^3)$

Jadi, Kompleksitas waktu asimptotiknya adalah $O(n^3)$

TEKS	B	E	B	B	E	B	E	B	E	B	E
Kondisi 1	B	E	B	E	K						
Kondisi 2					B	E	B	E	K		
Kondisi 3						B	E	B	E	K	
Kondisi 4								B	E	B	E

Gambar 4.3 Proses Pencocokan KMP

Kondisi 1 pergeseran pada Gambar 4.4 terjadi karena Teks B dan *Pattern* E tidak sesuai.

Kondisi 2 pergeseran pada Gambar 4.4 terjadi karena Teks E dan *Pattern* B tidak sesuai.

Kondisi 3 pergeseran pada Gambar 4.4 terjadi karena Teks B dan *Pattern* K tidak sesuai.

Kondisi 4 pergeseran pada Gambar 4.4 terjadi karena Teks E dan *Pattern* E sesuai.

4.2 Implementasi algoritma *BruteForce*

Dari *pseude code* algoritma yang terdapat dalam landasan teori, maka setelah diimplemetasikan kedalam bahasa pemoggraman *java* adalah sebagai berikut :

```
private int bruteforce(String text, String pola) {
        int n = text.length();
        int m = pola.length();
        int b;
        int hasil = -1;
        for (int a = 0; a <= (n - m); a++) {
            b = 0;
            while ((a < m) && (text.charAt(a + b) == pola.charAt(b))) {
                b++;
            }
            if (b == m)
                hasil = a;
            else
                hasil = -1;
        }
        return hasil;
```

Gambar 4.4 Implementasi algoritma *Brute Force*

Gambar 4.5 merupakan bagian terakhir dari implementasi algoritma BrF. *Method* ini berfungsi untuk mencari kesamaan antara *text* dan *pattern*. Pemeriksaan kecocokan dilakukan perkarakter dari *text* dan *pattern*. *Method* ini mengembalikan nilai iterasi a dan b untuk kemudian di proses pada *method* konstruktor untuk dicari *index* dimana terdapat kesamaan *string*.

Tabel 4.4 Perhitungan Big-O Algoritma *Brute Force*

Baris Intruksi	Operasi	Pengulangan
4	b = 0	O(1)
5	for (int a = 0; a <= (n - m); a++)	O(n)
6	**b = 0;**	O(1)
7	while ((a < m) && (text.charAt(a + b) == pola.charAt(b)))	O(n)
8	b++;	O(1)
9	if (b == m) hasil = a;	O(1)
10	**Else hasil = -1;**	O(1)
Jumlah		**$O(n^2)$**

Berdasarkan hasil perhitungan performasi algoritma *Brute Force* menggunakan notasi Big-O didapat kompleksitas waktu dengan $O(n^2)$. dengan n adalah ukuran input. Yang mempengaruhi nilai kompleksitas waktu dari algoritma *Brute Force* adalah nilai n karena n berpangkat dua.

Dari Tabel 4.4 pengulangan operasi-operasi dasar pada algoritma *Brute Force* didapat kompleksitas waktu sebagai berikut :

$T(n) = \sum ti$

$T(n) = 1+1+1+1+1+n+n$

$T(n) = 2n + 5 = O(n^2)$

4.3 Implementasi Perancangan Antarmuka

a. Splash Screen

Gambar 4.5 Hasil Implementasi tampilan *Splash Screen*

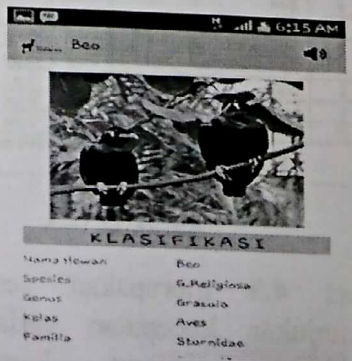
Gambar 4.5 merupakan tampilan *splash screen* dari aplikasi. Menu ini merupakan tampilan awal saat membuka aplikasi dan memberikan waktu untuk sistem untuk mengisi data sebelum masuk ke menu utama.

b. ActionBar *Menu*

Gambar 4.7 Tampilan *ActionBar Menu*

Gambar 4.7 merupakan tampilan actionbar menu. Terdapat *list* yang berisi daftar *menu* yaitu kamus *vertebrata*, *Rate This App*, *Share*, *About*, dan *Exit*.

a. Deskripsi *Vertebrata*



Gambar 4.8 Hasil implementasi tampilan deskripsi *vertebrata*

Gambar 4.8 merupakan tampilan hasil implementasi untuk mendeskripsikan *vertebrata* yang terdapat dalam daftar kamus *vertebrata*. Terdapat gambar dari *vertebrata*, nama *vertebrata*, spesies, genus, familia, ordo, divisi, kingdom, deskripsi, dan bahasa inggris.

4.4 Pengujian algoritma *Knuth-Morris-Pratt* dan *Brute Force*

Pengujian algoritma didasarkan pada sejauh mana penerapan algoritma tersebut mampu mengoptimasi kinerja sistem pada saat pencocokan *string*. Untuk melihat sejauh mana algoritma tersebut mampu mereduksi *time of execution* dari sistem ditempatkan suatu metode perhitungan fungsi waktu yang mana fungsi ini akan diletakkan pada awal maupun akhir dari kode proses pencocokan.

Tabel 4.9 merupakan hasil percobaan pada proses pencocokan *string* dengan beberapa *pattern* yang memiliki panjang karakter beragam yang dijadikan sebagai acuan. Beberapa *parameter* yang digunakan didalam tabel diantaranya adalah *key* dan *pattern*, panjang karakter, waktu eksekusi dengan menggunakan algoritma KMP dan BrF, selisih waktu antara kedua metode tersebut dan untuk mencari rata-rata jumlah percobaan.

Tabel 4.9 Tabel hasil percobaan panjang karakter 0-5

No	*Pattern*	Panjang Karakter	Waktu					
			BrF			KMP		
			ke-I	ke-II	ke-III	ke-I	ke-II	ke-III
1	Cicak	5	48ms	10 ms	33 ms	67 ms	21 ms	67 ms
2	Kiwi	4	12 ms	12 ms	33 ms	28 ms	29 ms	17 ms
3	Bebek	5	31 ms	23 ms	33 ms	33 ms	17 ms	26 ms
4	Beo	3	32 ms	5 ms	34 ms	67 ms	23 ms	16 ms
5	gagak	5	9 ms	12 ms	28 ms	26 ms	26 ms	60 ms
6	N	1	5 ms	5 ms	6 ms	24 ms	23 ms	24 ms
7	As	2	8 ms	8 ms	6 ms	18 ms	30 ms	36 ms
8	Babi	4	8 ms	8 ms	8 ms	18 ms	18 ms	23 ms
9	Lemur	5	10 ms	8 ms	10 ms	21 ms	20 ms	13 ms
10	Maleo	5	9 ms	9 ms	8 ms	16 ms	17 ms	15 ms
Rata – rata			172 ms	100 ms	199 ms	318 ms	224 ms	297 ms

Tabel 4.9 merupakan sesi pertama menunjukan kecepatan variatif dengan percobaan panjang karakter 0-5 penggunaan metode ke-1 BrF (172ms), ke-2 BrF (100ms), dan ke-3 BrF (199ms) terjadi perubahan di percobaan ke-2 menurun dan percobaan ketiga BrF meningkat pesat. Ke-1 KMP (318 ms), ke-2 KMP (224ms), dan ke-3 KMP (297ms) terjadi perubahan dipercobaan ke-2 menurun dan percobaan ke-3 KMP meningkat sedikit. Pada percobaan ini ada algoritma BrF yang efesien dan mangkus dalam segi waktu pada percobaan no 8. dan dalam segi waktu *Brute Force* unggul dalam pencarian teks pada panajang karakter 6-10

Tabel 4.10 Tabel hasil percobaan panjang karakter 6-10

```
double awal = System.currentTimeMillis();
Algo_KMP(); // proses pencocokan
double akhir = System.currentTimeMillis();
double waktuExe = (akhir-awal); // waktu eksekusi
```

No	*Pattern*	Panjang Karakter	Waktu					
			BrF			KMP		
			ke-I	ke-II	ke-III	ke-I	ke-II	ke-III
1	Gurame	6	9ms	14 ms	10 ms	16 ms	15 ms	21 ms
2	Harimau	7	8 ms	8 ms	8 ms	19 ms	19 ms	24 ms
3	Macan Tutul	9	8 ms	10 ms	8 ms	16 ms	16 ms	17 ms
4	Arwana Asia	10	8 ms	14 ms	10 ms	19 ms	29 ms	20 ms
5	carpelai	8	6 ms	9 ms	8 ms	22 ms	19 ms	24 ms
6	Marmot	6	10 ms	14 ms	9 ms	25 ms	20 ms	33 ms
7	Merpati	7	8 ms	10 ms	11 ms	30 ms	20 ms	15 ms
8	Serigala	8	9 ms	8 ms	7 ms	34 ms	17 ms	22 ms
9	Puripang	8	8 ms	10 ms	9 ms	19 ms	17 ms	29 ms
10	Cheetah	9	10 ms	9 ms	9 ms	24 ms	14 ms	34 ms
Rata – rata			84 ms	106 ms	89 ms	224 ms	186 ms	233 ms

Tabel 4.10 merupakan sesi kedua menunjukan kecepatan variatif dengan percobaan panjang karakter 6-10 penggunaan metode ke-1 BrF (84ms), ke-2 BrF (106ms), dan ke-3 BrF (89ms) terjadi perubahan di percobaan ke-2 meningkat pesat dan percobaan ke-3 menurun. Ke-1 KMP (224 ms), ke-2 KMP (186ms), dan ke-3 KMP (223ms) terjadi perubahan dipercobaan ke-2 menurun dan percobaan ke-3 KMP meningkat pesat. Pada percobaan ini ada algoritma BrF yang efesien dan mangkus dalam segi waktu pada percobaan no 2. dan dalam segi waktu *Brute Force* unggul dalam pencarian teks pada panajang karakter 6-10.

Tabel 4.11 Tabel hasil percobaan panjang karakter 11-25

No	*Pattern*	Panjang Karakter	Waktu					
			BrF			KMP		
			ke-I	ke-II	ke-III	ke-I	ke-II	ke-III
1	Beruang madu	11	8	14	8	19	16	20
2	Burung Perkutut	14	9	9	10	21	20	22
3	Cendrawasih merah	16	11	9	9	21	18	22
4	Hiu karpet bintik	15	6	11	14	23	22	32
5	Ikan cakalang	12	9	9	8	14	27	17
6	Rusa bawean	11	9	11	6	18	18	17
7	Musang galling	12	10	9	9	21	18	15
8	Lurung budeng	12	8	10	9	21	26	15
9	Kumbang tanduk	13	7	9	9	25	13	20
10	Kanguru pohon mantel emas	22	0	10	10	1	20	18
Rata – rata			77	101	92	184	198	198

Tabel 4.11 merupakan sesi ketiga menunjukan kecepatan variatif dengan percobaan panjang karakter 11-25 penggunaan metode ke-1 BrF (77ms), ke-2 BrF (101ms), dan ke-3 BrF (92ms) terjadi perubahan di percobaan ke-2 meningkat pesat dan percobaan ke-3 menurun. Ke-1 KMP (184 ms), ke-2 KMP (198ms), dan ke-3 KMP (198ms) terjadi perubahan dipercobaan ke-2 meningkat dan percobaan ke-3 KMP sama dengan percobaan ke-2. Pada percobaan ini tidak ada algoritma yang efesien dan mangkus dalam segi waktu kecepatan. dan dalam segi waktu *Brute Force* unggul dalam pencarian teks pada panajang karakter 11-25.

Dilihat dari hasil percobaan optimasi menggunakan algoritma KMP dan BrF seperti pada Tabel 4.9, Tabel 4.10, dan Tabel 4.11 algoritma BrF lah yang paling cepat untuk pencarian data. Hal ini akan sangat berpengaruh apabila *device* yang digunakan memiliki spesifikasi yang lebih rendah dan sangat berpengaruh bila ada Aplikasi yang lain lagi berjalan.

V. Penutup

5.1 Kesimpulan

Setelah melalui beberapa proses dan tahapan dengan rancangan pembangunan secara *prototype* maka dapat disimpulkan beberapa hal sebagai berikut:

1. Aplikasi kamus *vertebrata* secara fungsional dapat berjalan sesuai dengan yang diharapkan berdasarkan hasil pengujian *blackbox*.
2. Secara hasil penelitian dalam algoritma *Knuth-morris-pratt* dan *Brute Force* tidak ada yang efisien atau pada segi waktu. Dari hasil perhitungan aplikasi *Brute Force* lebih cepat dibandingkan dengan *Knuth-morris-pratt.* Sedangkan KMP unggul dari perhitungan kompleksitas waktu $O(n^3)$ dari algoritma *Brute Force* $O(n^2)$.
3. Aplikasi hewan *vertebrata* berbasis android ini memberikan informasi yang bermanfaat untuk memberikan pengetahuan tentang klasifikasi dan deskripsi dalam bentuk tulisan dan gambar hewan *vertebrata.*

Daftar Pustaka

[1] Galia, Lami, Dozo, 2006. Atlas *vertebrata* dan *invertebrata*. Solo.

[2] Donald Knuth, James H. Morris, Jr. Vaughan Pratt, 1977. *Fast pattern matching in strings*. SIAM Journal on Computing.

[3] Ekaputri, Gahayu Handari dan Yulie Anneria Sinaga, 2010. *Aplikasi Algoritma Pencarian String Knuth-Morris-Pratt* dan *Brute Force dalam Permainan Word Search*. Jurnal tidak diterbitkan, Bandung.

[6] Munawar. 2005. *Pemodelan Visual dengan UML*. Yogyakarta: Graha

[7] Munir, Rinaldi, 2011. *Algoritma dan Pemrograman*. Bandung: Informatika.

[6] Nazruddin, Safaat H, 2012. *ANDROID Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Bandung: Informatika.

[7] Pressman, Roger S, 2002. *Rekayasa Perangkat Lunak: pendekatan praktisi*. Diterjemahkan oleh: LN Harnaningrum. Yogyakarta: Andi.

[8] Sadiman, Arief S, dkk., 2011. Media Pendidikan Pengertian, Pengembangan dan Pemanfaatannya. Jakarta: Rajawali Pers.

[9] Safaat, Nazruddin. 2011. *Android: Pemrograman Aplikasi Mobile Smarthphone dan Tablet PC Berbasis Android*. Bandung:Informatika

[10] Soleh, Moch. Yusuf, 2010. *Implementasi Algoritma KMP dan Boyer-Moore dalam Aplikasi Search Engine Sederhana*. Jurnal tidak diterbitkan, Bandung.

Penulis :
Beki Subaeki M. Kom [1],
Asep Muhammad Indra Purnama, M. Kom [2]
[1] Teknik Informatika, [2] Sistem Informasi
Fakultas Teknik Universitas Sangga Buana YPKP