semanTIK, Vol.3, No.1, Jan-Jun 2017, pp. 115-126
ISSN : 2502-8928 (Online)

ANALISIS PERBANDINGAN *ALGORITMA BOYER MOORE* DAN *ALGORITMA KNUTH MORRIS PRATT* PADA APLIKASI TRIPELKA *FOODSHOP* KENDARI BERBASIS ANDROID

Andi Maghfirah Parenrengi[*1], Rizal Adi Saputra[2], LM Tajidun[3]
[*1,2,3]Jurusan Teknik Informatika, Fakultas Teknik Universitas Halu Oleo, Kendari
e-mail: [*1]**maghfirahandi@gmail.com**,[2]rizaladisaputra@gmail.com,[3]moeh.tajidun@yahoo.com

Abstrak

Kemajuan yang sangat pesat dibidang teknologi terutama teknologi informasi komputer mempengaruhi perkembangan dunia bisnis. Tripelka adalah komunitas pengusaha makanan *online* yang sangat membutuhkan teknologi internet untuk mempercepat penyampaian informasi. Berdasarkan hal itu maka dibuatlah aplikasi pencarian menu makanan berbasis android.

Untuk mempersingkat proses penyajian data pada aplikasi ini maka diterapkan pencocokan string pada pencarian menu-menu makanan yang kita inginkan. Ada berbagai jenis algoritma *string matching* yang umum di gunakan, anatara lain : *Algoritma Boyer Moore*, *Algoritma Brute Force* dan *Algoritma Knuth Morris Pratt.*

Dalam penelitian ini dilakukan analisis perbandingan antara Algoritma *Boyer Moore* dan Algortima *Knuth Morris Pratt*, untuk menentukan algoritma yang paling baik digunakan dalam aplikasi Tripelka Foodshop Kendari. Parameter yang digunakan untuk membandingkan kedua algoritma tersebut adalah waktu pencarian dan tingkat keakurasian data yang ditampilkan.

Hasil dari penelitian ini menunjukkan bahwa Algoritma *Boyer Moore* dan Algoritma *Knuth Morris Pratt* memiliki tingkat keakurasian yang sama tetapi Algoritma *Boyer Moore* adalah algoritma menggunakan waktu pencarian yang lebih cepat dibandingkan Algoritma *Knuth Morris Pratt*.

Kata kunci—Android, Pencocokan String, Algoritma *Boyer-Moore*, Algoritma *Knuth Morris Pratt*

Abstract

An immense improvement in the field of computer technology, especially information technology affects the development of the business world. Tripelka is food online community of entrepreneurs that desperately need internet technology to accelerate the delivery of information. Based on that then made a search application android-based food menu.

To shorten the process of presenting the data in this application is applied to the search string matching food menus we wanted. There are different types of string matching algorithms commonly used, among other things: Algorithm Boyer Moore, Brute Force algorithm and Knuth-Morris-Pratt algorithm.

In this research, comparative analysis between Boyer Moore algorithm and algorithms Knuth Morris Pratt, to determine which algorithm is best used in applications Tripelka Foodshop Kendari. The parameters used to compare the two algorithms are the search time and the level of accuracy of data displayed.

The result of this research shows that Boyer Moore Algorithm and Knuth Morris Pratt Algorithm has the same accurate precise. But, Boyer Moore Algorithm is more faster than Knuth Morris Pratt Algorithm.

Keywords*—Android, Match String, Boyer-Moore Algorithm, Knuth Morris Pratt Algorithm*

1. PENDAHULUAN

Kemajuan yang sangat pesat dibidang teknologi terutama teknologi informasi komputer mendorong munculnya penyajian informasi untuk kebutuhan informasi. Sehubungan dengan itu kemajuan bisnis juga dipengaruhi oleh teknologi informasi, dimana teknologi informasi sangat berperan pada proses pemasaran bisnis yang sedang berjalan. Hal ini memungkinkan banyaknya pengusaha yang bergerak dalam bisnis *online*, dengan memanfaatkan kemajuan teknologi.

Tripelka (Komunitas Kuliner Kendari) adalah wadah para pengusaha kuliner yang berada di Kota Kendari. Komunitas yang beranggotakan kurang lebih 47 usaha ini lebih dari setengahnya adalah pengusaha kuliner *online*. Hal ini menunjukkan bahwa besarnya pengaruh internet untuk perkembangan bisnis anggota Tripelka. Akses internet diperlukan untuk beberapa hal, antara lain : penjualan, promosi produk, komunikasi, dan sebagainya. Menurut Colin Combe pada bukunya yang berjudul *e-bussiness management and strategy* menyatakan bahwa sistem penjualan telah beralih dari *e-bussiness* menjadi *m-bussiness* sehingga aplikasi ini akan dibuat berbasis android agar mempercepat proses penyampaian informasi kepada konsumen.

Untuk mempersingkat proses penyajian data pada aplikasi ini maka diterapkan pencocokan string pada pencarian menu-menu makanan yang kita inginkan. Ada berbagai jenis algoritma *string matching* yang umum di gunakan, anatara lain : *Algoritma Boyer Moore, Algoritma Brute Force* dan *Algoritma Knuth Morris Pratt*.

Dalam penelitian ini penulis akan membandingkan antara *Algoritma Boyer Moore* dan *Algoritma Knuth Morris Patt*. Berdasarkan penelitian [1] disimpulkan bahwa ketepatan *Algoritma Boyer Moore* dalam mencari kata-kata tersembunyi pada permainan *word search puzzle* berjumlah 100% dan *Algoritma Boyer Moore* lebih cepat dibanding dengan *Algortima Brute Force*. Sedangkan berdasarkan penelitian [2] menyatakan bahwa *Algoritma Knuth Morris Patt* merupakan pengembangan dari *Algoritma Brute Force*. Selanjutnya penelitian dilakukan oleh [3] dengan hasil pengujian pada perangkat lunak diperoleh presentase 97,14% pada aplikasi *game* hanacaraka sehingga dianggap sangat tepat untuk membandingkan kedua algoritma ini. Metode penyelesaian yang berbeda dari kedua metode ini juga dianggap akan menghasilkan perbedaan.

Hal ini mendorong penulis untuk membangun aplikasi yang dapat memudahkan konsumen mendapatkan informasi mengenai berbagai hal tentang Tripelka khususnya pada pencarian menu-menu makanan yang tersedia, sehingga kelak aplikasi ini dapat berfungsi secara baik untuk menunjang kemajuan bisnis para pengusaha yang tergabung dalam Komunitas Kuliner Kendari atau yang biasa disebut Tripelka.

2. METODE PENELITIAN

2.1 Komunitas Kuliner Kendari

Komunitas Kuliner Kendari atau yang biasa disebut Tripelka adalah salah satu komunitas yang dikukuhkan pada tanggal 27 Januari 2016 oleh walikota Kendari. Komunitas ini terbentuk dengan jumlah anggota 47 usaha kuliner Kota Kendari. Tripelka mempunyai visi "Menjadikan Kota Kendari sebagai tujuan wisata kuliner nusantara serta menjadikan Komunitas Kuliner Kendari sebagai *ONE STEP CLOSER* untuk semua informasi mengenai kuliner Kota Kendari. Berdasarkan visi tersebut, Tripelka mempunyai beberapa tujuan, antara lain ; memajukan kuliner Kendari, mewadahi semua produk-produk kuliner Kendari baik *online* maupun *offline*, media pembelajaran *entrepreneurship* untuk anggota komunitas, dan melaksanakan *event-event* kuliner di Kota Kendari [4].

2.2. Android

Android adalah suatu sistem operasi yang didesain sebagai *platform open source* untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi . Android menyediakan *platform* yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri. Android menyediakan semua *tools* dan *framework* untuk mengembangkan aplikasi dengan mudah dan cepat. Dengan adanya Android SDK (*Software Developement Kit*) pengembang aplikasi dapat memulai pembuatan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java [5].

2.3 *Ionic Framework*

Ionic Framework merupakan sebuah *framework* HTML5 yang membantu dalam mengembangkan aplikasi *mobile* dengan teknologi *web* seperti HTML, CSS dan *Javascript*. *Ionic Framework* adalah *platform* yang menargetkan seorang *Programmer Web* agar bisa membuat aplikasi *mobile* dengan teknologi *web*. Artinya, *Programmer web* yang ingin menjadi *Programmer Mobile* tidak perlu belajar *Java* atau *Objective C* atau C# untuk membuat versi aplikasi dari layanan webnya.

IonicFramework terdiri dari sekumpulan teknologi yang dikembangkan untuk membangun aplikasi *mobile hybrid* yang *powerful*, cepat, mudah dan juga memiliki tampilan yang menarik. *Ionic Framework* menggunakan *AngularJS* sebagai *framework* berbasis *web* dan menggunakan *Cordova* untuk membangun aplikasi *mobile*.

Beberapa kelebihan *Ionic Framework* dibandingkan dengan *framework* lain yaitu :

1. *Ionic Framework* menggunakan lisensi *Open Source*

   IonicFrameworkplatform menggunakan lisensi *open source*, dimana pengguna dapat membuat aplikasi *free* ataupun komersial dengan *Ionic*.

2. Menggunakan teknologi *web* terbaru

   Ionic Framework memanfaatkan *AngularJS* untuk implementasi *logic* nya. *AngularJS* menawarkan performa dan respon cepat seperti aplikasi *native*.

3. Target Hanya untuk Android 4 dan ios 7 ke atas

   Bagi yang suka dengan hal baru tanpa memikirkan kompabilitas dengan versi mobile OS lama, maka *ionic framework* adalah *platform* yang paling pas.

4. Berbasis *Apache Cordova*/*Phonegap*

   Ionic Framework hanya menyediakan *framework* nya, untuk membungkusnya menjadi aplikasi Android atau iOS anda tetap pakai *phonegap*. Artinya bagi para *programmer phonegap* dengan *platform* lain, keahliannya tetap bisa dipakai [6].

2.4 *Apache Cordova/Phonegap*

Phonegap adalah suatu *framework open source* untuk pengembangan aplikasi *hybrid* dalam konteks *mobile* untuk mengembangkan dan membuat aplikasi *mobile* untuk berbagai banyak *platform* seperti Android, Blackberry, iOS, dan *Windows Phone* menggunakan bahasa format HTML5 & CSS3 serta bahasa pemrograman *Javascript*.

Phonegap adalah nama lama, sementara nama baru dari *phonegap* ini adalah *Apache Cordova* karena banyak kontribusi dari perusahaan–perusahaan besar dan lembaga lain terhadap *Phonegap*, lalu *phonegap* diserahkan ke *Apache SoftwareFoundation*.

Mengembangkan suatu aplikasi dengan *Apache Cordova* memiliki banyak kemudahan diantaranya pengembang / *developer* hanya menggunakan satu bahasa pemrograman ditambah bahasa *formatting* untuk mengembangkan aplikasi dan merilisnya ke berbagai *platform*. Inilah yang disebut dengan *Hybrid App*, suatu aplikasi yang dikembangkan dengan bahasa yang sama tapi bisa didistribusikan ke berbagai *platform*. Beda halnya dengan *Native App* yaitu suatu aplikasi yang dikembangkan sesuai bahasa yang diterima oleh platform yang akan disasar oleh *developer*.

Apache Cordova juga memiliki banyak *plugin* yang akan mendukung pengembangan aplikasi dan mempermudah *developer*, yakni *plugin* untuk mengakses kamera *smartphone*, menggunakan akselerometer, mengakses sitem panggilan *smartphone*, dan masih banyak lagi lainnya.

Adapun kelebihan dan kekurangan Apache Cordova.

Kelebihan.

1. Kemudahan dalam membangun, mengembangkan dan merilis suatu aplikasi
2. Hanya menggunakan HTML5, CSS3 dan *Javascript* untuk pengembangan, *developer* tidak perlu mempelajari bahasa pemrograman yang terpisah, seperti *Java* untuk Android, *Objective-C* untuk iOS, C# untuk *Windows Phone*
3. Banyak terdapat *library* yang memungkinkan *developer* membuat aplikasi lebih cepat dan mudah seperti yang diinginkan

4. Kemudahan dalam membuat dan mendesain *User Interface* / Antarmuka Pengguna
5. Sama seperti poin diatas, mengatur gambar seperti ikon untuk aplikasi sangat mudah
6. Walaupun dikembangkan dengan bahasa *web standard*, pemasangan aplikasi tidak berbeda dari pemasangan aplikasi *native*, karena aplikasi dengan *cordova* juga dirilis dengan *package installer* sesuai target *platform*nya.

Kekurangan.

1. Adanya kemungkinan *reverse-engineering*, yakni memungkinkan pengeditan *source code* oleh pengguna
2. Karena banyaknya *plugin* yang dikembangkan oleh kontributor berbeda, bisa jadi nanti ada *plugin* yang pengembangannya terhenti
3. Dokumentasi masih cukup minim.
4. Kemungkinan jalannya *Hybrid app* yang lebih lama dibandingkan *Native app* [7].

2.5 *File.json*

File.js merupakan *file javascript* yang dibuat menjadi satu kesatuan, dari suatu *javascript* yang begitu panjang dapat dibuat menjadi satu *file* dengan *format*.js. Suatu *javascript* yang mengandung *file.js*, memang umumnya sulit untuk di *edit*, karena harus membongkar terlebih dahulu *file.js* tersebut, kemudian baru dapat meng*editscript* yang terkandung dalam *file.js* tersebut.

File.js biasanya diimpor pada bagian kepala HTML. Selain itu dapat membandingkan fungsi-fungsi, yang melakukan validitas *form*, membuat menu-menu *drop-down* atau membuka dan menutup *Windows* [8].

2.6 *Javascript*

Javascript merupakan modifikasi dari bahasa C++ dengan pola penulisan yang lebih sederhana. Interpreter bahasa ini sudah disediakan ASP ataupun *internet explorer*. Secara khusus ada beberapa hal yang penting dalam *javascript* antara lain.

a. Menggunakan blok awal "{" dan blok akhir :}".
b. *Automatic conversion* dalam pengoperasian tipe data yang berbeda.
c. *Sensitive case*, sehingga *programmer java* harus ekstra hati-hati dalam menggunakan nama variabel, fungsi dan lain-lain.
d. Ekstensi umumnya menggunakan "*.js".
e. Setiap *statement* dapat diakhiri dengan tanda "," sebagaimana C++ tetapi dapat juga tidak.
f. Jika tidak didukung oleh *browser* versi lama, scriptnya dapat disembunyikan diantara tag "<!-- dan "--".
g. Jika program dalam satu baris terlalu panjang dapat disambung dengan karakter "/".

Seperti CSS yang menggunakan tag :
<style></style>
Javascript menggunakan *tag*:
<script></script>

Tag ini boleh diletakkan beberapa kali didalam sebuah dokumen, sama seperti *tag-tag* HTML yang lain.

Kelebihan *Javascript* adalah berinteraksi dengan HTML. Hal ini membolehkan pembuat *web* untuk memasukkan *web* mereka dengan kandungan-kandungan yang dinamik, menukar warna *background*, menukar *banner*, efek *mouse*, menu interaktif dan sebagainya.

2.6 *Array*

Array atau larik adalah sekumpulan variabel yang memiliki tipe data yang sama dan dinyatakan dengan nama yang sama. *Array* dapat juga diartikan sebagai sekelompok data sejenis yang disimpan ke dalam variabel dengan nama yang sama, dengan memberi indeks pada variabel untuk membedakan antara yang satu dengan yang lain.

Array merupakan konsep yang penting dalam pemrograman, karena *array* memungkinkan untuk menyimpan data maupun referensi objek dalam jumlah banyak dan terindeks. *Array* menggunakan indeks *integer* untuk menentukan urutan elemen-elemennya, dimana elemen pertamanya dimulai dari indeks 0, elemen kedua memiliki indeks 1, dan seterusnya.

Array memiliki karakteristik dimana *Array* mempunyai batasan dari pemesanan alokasi memori (bersifat statis). Selain itu, *Array* memiliki tipe data yang sama (bersifat homogen) dan dapat diakses secara acak.

Bentuk umum dari *Array* adalah sebagai berikut.

Tipe_data nama_variabel[indeks];
Contoh : float bil[10];

Indeks *array* dimulai dari nol(0) , sedangkan nomor elemen biasanya dimulai dari satu (1). Nomor elemen dapat dibuat sama dengan nomor indeks untukmempermudah pembuatan program yaitu dengan memberi indeks satu lebih banyak dari jumlah data yang dibutuhkan[9].

A. *Array* 1 Dimensi

Array satu dimensi adalah kumpulan elemen yang memiliki tipe data yang sama dan hanya memiliki satu indeks saja.
Pendeklarasian *Array* 1 Dimensi.
Tipe_data nama_variabel [jumlah_elemen_array];
Contoh : int A[3] = {3,5,6}

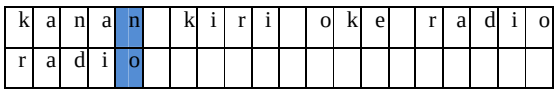
B. *Array* 2 Dimensi

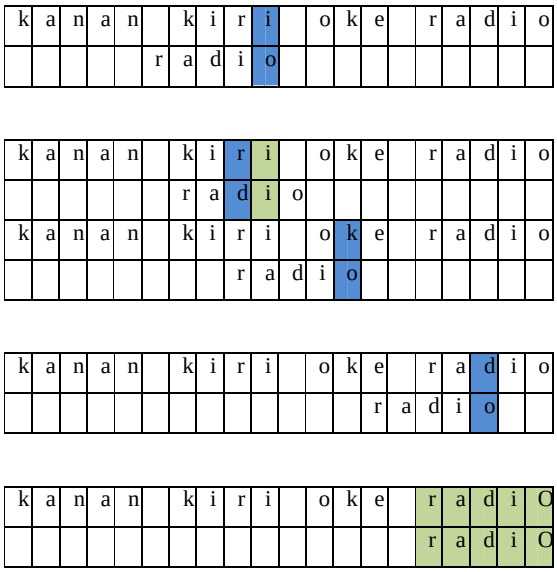
Array dua dimensi adalah *array* yang dapat membantu dalam pemrograman apabila *array* satu dimensi tidak mencukupi dalam menghasilakan suatu solusi.*Array* dua dimensi sering kali digambarkan/dianalogikan sebagai sebuah matriks atau bentuk *grid*. Jika *array* berdimensi satu hanya terdiri dari 1 baris dan banyak kolom, *array* berdimensi dua terdiri dari banyak baris dan banyak kolom yang bertipe sama.
Pendeklarasian Array 1 Dimensi.
Tipe_data nama_variabel [index-1] [index-2];
Contoh : int A[3] [2]= {{4,7,2}, {3,1}}

2.7 Algoritma *Boyer-Moore*

Algoritma *Boyer-Moore* adalah salah satu algoritma untuk mencari suatu *string* di dalam teks, dibuat oleh *R.M Boyer dan J.S Moore*. Ide utama algoritma ini adalah mencari *string* dengan melakukan pembandingan karakter mulai dari karakter paling kanan dari *string* yang dicari [10]. Dengan mengunakan algoritma ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingakan dengan algoritma lainnya. alasan melakukan pencocokan dari kanan (posisi terakhir *string* yang dicari) ditunjukan pada Gambar 1.

Gambar 1 menunjukkanPergeseran *Algoritma Boyer-Moore*





Gambar 1 Pergeseran *Algoritma Boyer-Moore*

Dengan melakukan pembandingan dari posisi paling akhir *string* dapat dilihat bahwa karakter “n” pada *string* “kanan” tidak cocok dengan karakter “o” pada *string* “radio” yang dicari, dan karakter “n” tidak pernah ada dalam *string* “radio” yang dicari sehingga *string* “radio” dapat digeser melewati *string* “kanan” selanjutnya karakter “o” tidak cocok dengan karakter “i” tetapi ada pada *string* “radio” karakter “i” disejajarkan dengan karakter “i” selanjutnya karakter “d” tidak cocok dengan karakter “r” tetapi terdapat pada *string* “radio” sehingga digeser sedemikian rupa agar karakter “r” sejajar.Selanjutnya dimulai lagi pencocokan dari kanan, karakter “k” tidak cocok dengan karakter “o” pada *string* “radio” dan tidak ada dalam *string* “radio” sehingga digeser sepanjang jumlah karakter melewati karakter “k” selanjutnya karakter “d” tidak cocok dengan karakter “o” tetapi ada pada *string* “radio” lalu disejajarkan sehingga teks yang cocok akhirnya ditemukan.

Dalam contoh terlihat bahwa *Algoritma Boyer-Moore* memiliki loncatan karakter yang besar sehingga mempercepat pencarian *string* karena dengan hanya memeriksa sedikit karakter, dapat langsung diketahui bahwa *string* yang dicari tidak ditemukan dan dapat digeser ke posisi berikutnya. Adapun langkah-langkah algoritma Boyer-Moore adalah [11] :

1. Membuat 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 pencocokan dengan *string* (T).

2. Jika dalam proses pembandingan terjadi ketidakcocokan 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 tidak ada pada S maka pegeseran adalah sebanyak jumlah karakter pada S. dan jika karakter yang tidak cocok ada pada S, maka banyaknya pergeseran bergantung dari nilai pada tabel.
4. Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada S, maka posisi karakter pada S dan T diturunkan sebanyak 1 posisi, kemudian lanjutkan dengan pencocokan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokan karakter S dan T, maka pilih nilai pergeseran terbesar dari dua tabel analisa *pattern* yaitu nilai dari tabel match heuristic dan nilai tabel *occurenceheuristic* dikurangi dengan jumlah karakter yang telah cocok.
5. Jika semua karakter telah cocok, artinya S telah ditemukan di dalam T, selanjutnya geser *pattern* sebesar 1 karakter.
6. Lanjutkan sampai akhir *string* T.

2.8 Algoritma *Knuth Morris Pratt*

Algoritma *Knuth-Morris-Pratt* adalah salah satu algoritma pencarian *string*, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977. Jika kita melihat algoritma *Brute Force* lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian [12].

Perhitungan penggeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokan pada saat *pattern* sejajar dengan *teks[i..i + n – 1],* kita bisa menganggap ketidakcocokan pertama terjadi di antara *teks[I + j] dan pattern [j],*dengan *0 < j < n.*berarti, *teks [i..i + j – 1] = pattern [0..j – 1] dan a = teks [I + j]*tidak sama dengan *b = pattern[j].*

Ketika kita menggeser, sangat beralasan bila ada sebuah awalan *u*dari *pattern* akan sama dengan sebagian akhiran *u* dari sebagian teks. Sehingga kita bisa menggeser *pattern* agar awalan *u* tersebut sejajar dengan akhiran dari *u*.dengan kata lain, pencocokan *string* akan berjalan secara efesien bial kita mempunyai table yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokan di karakter ke- *j*dari *pattern.* Tabel itu harus memuat *next[j]*yang merupakan posisi karakter *pattern[j]* setelah digeser, sehingga kita bisa menggeser *pattern* sebesar *j –next[j] relative* terhadap teks.

Secara sistematis, langkah-langkah yang dilakukan Algoritma *Knuth-Morris-Pratt*pada saat mencocokkan *string*:

1. Algoritma *Knuth-Morris-Pratt*mulai mencocokkan *pattern* pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per-karakter di teks yang bersesuaian, sampai salah satu kondisi dipenuhi
3. Algoritma kemudian menggeser *pattern* berdasarkan tabel next, lalu mengulangi langkah 2 sampai *pattern* berada di ujung teks.

Gambar 2 menunjukkanPergeseran Algoritma*Knutt Morris Pratt.*

r	a	m	b	u		r	a	d	i	o
r	a	d	i	o						

r	a	m	b	u		r	a	d	i	o
		r	a	d	i	o				

r	a	m	b	u		r	a	d	i	o
						r	a	d	i	o

Gambar 2 Pergeseran Algoritma*Knutt Morris Pratt*

Pertama-tama karakter “r” pada “radio” dicocokkan dengan karakter “r” pada *string* “rambu” karena ada kecocokan maka *Algoritma Knuth Morris Pratt* akan menyimpan informasi tersebut, begitu juga karakter “a” yang ditemukan kecocokan, tidak akan mengalami pergeseran. Selanjutnya dicocokkan karakter “m” dan “d” Karena tidak cocok, maka *Algoritma Knuth Morris Pratt* akan menggunakan informasi yang telah disimpan pada langkah sebelumnya untuk menentukan pergeseran selanjutnya.

Sehingga pencocokan yang dilakukan bukanlah antara *pattern* ke-1 dan teks ke-3 melainkan antara *pattern* ke-1 dengan teks ke 4 selanjutnya di cocokkan karakter secara berurutan karena tidak ada lagi persamaan hingga ditemukan kecocokan antara *string* dan *pattern*.

2.9 *Unified Process*

Unified Process (UP) merupakan suatumetode pembangunan system secara objekorientedyangdikembangkanoleh *Rational Rose*, bagian dari IBM. Secara luas, UP telah diakuisebagai standar metodologi pembangunan system berorientasi objek.

Versi asli dari UP didefinisikan sangat rumit untuk setiap kegiatan. Namun versi terrbaru dari UP yakni metodologinya lebih sederhana. Ciri utamametode ini adalah menggunakan *use-case driven* dan pendekatan iterative untuk siklus pengembangan perangkat lunak.

Dalam penelitian ini metode pengembangan orientasi objek yang diterapkan adalah *Unified Process* (UP). UP adalah salah satu bentuk *framework*pengembangan perangkat lunak secara *iterative* dan *incremental*. Proses pengembangan perangkat lunak dalam UP dibagi dalam 4 fase dan setiap fase dijalani secara *iterative* dan *incremental* seperti telah disebutkan [13].

Empat fase UP adalah sebagai berikut :

1. *Inception*

Inception bermakna permulaan. *Inception* merupakan tahapan paling pendek dalam fase UP.

2. *Elaboration*

Tujuan dari *elaboration* adalah mengidentifikasi resiko dalam pengembangan perangkat lunak dan membangun dasar arsitektur system. Proses pembangunan arsitektur diantaranya dilakukan dengan penggambaran diagram use case dan diagram konsep dasar.

3. Construction

Fase *construction* adalah fase terpanjang dalam UP. Proses inti dalam fase ini adalah pengkodean perangkat lunak. Tentunya pengkodean pun diawali dengan desain dengan UML (UML akan diuraikan pada subbab selanjutnya).

*4. Transitio*n

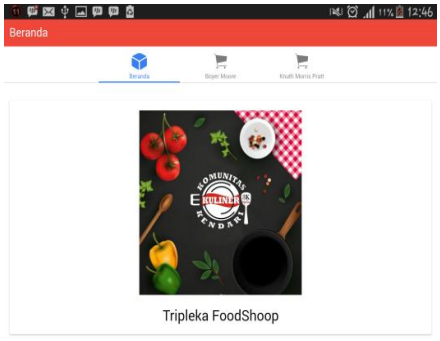
Fase transition adalah fase saat perangkat lunak telah selesai dibuat dan mulai digunakn. Dalam fase transisi akan diperoleh umpan blik dari pengguna.

3. HASIL DAN PEMBAHASAN

3.1. Implementasi Desain Antarmuka

Implementasi *interface* dari perangkat lunak dilakukan berdasarkan perancangan *user interface* yang telah dijelaskan, namun disesuaikan dengan komponen-komponen *view* yang tersedia pada *ionic framework* android. Implementasi antarmuka ditampilkan dalam bentuk *screen shoot* dari *handphone* android.

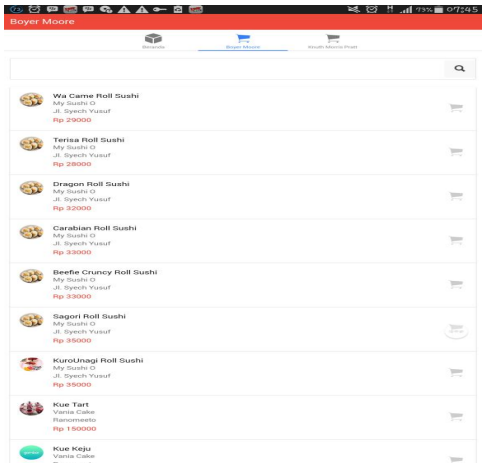
Gambar 3 menunjukkan implementasi dari *interface.*



Gambar 3 Beranda Aplikasi Tripelka Foodshoop Kendari

Beranda yang telah dirancang. *Interface* ini menampilkan menu utama aplikasi Tripelka *Foodshoop* Kendari yaitu menu *Algoritma Boyer Moore dan menu Algoritma Knuth Morris Pratt.* Pada menu ini, user dapat memilih untuk mencari menu makanan menggunkan *Algoritma Boyer Moore* atau *Algoritma Knuth Morris Pratt.*

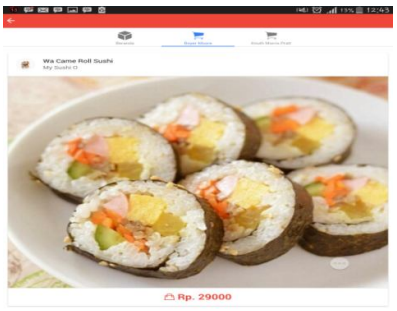
Gambar 4 menunjukkan *interface* Menu Pencarian Menggunakan *Algoritma Boyer Moore.*



Gambar 4 *Interface*Menu Pencarian Menggunakan *Algoritma Boyer Moore*

Interface menu pencarian dari aplikasi menggunakan *Algoritma Boyer Moore. User* cukup memasukkan nama makanan yang ingin dicari di kolom pencarian. Selanjutnya akan ditampilkan menu-menu makanan dan harga makanan yang berada dalam *database* dan waktu pencarian.

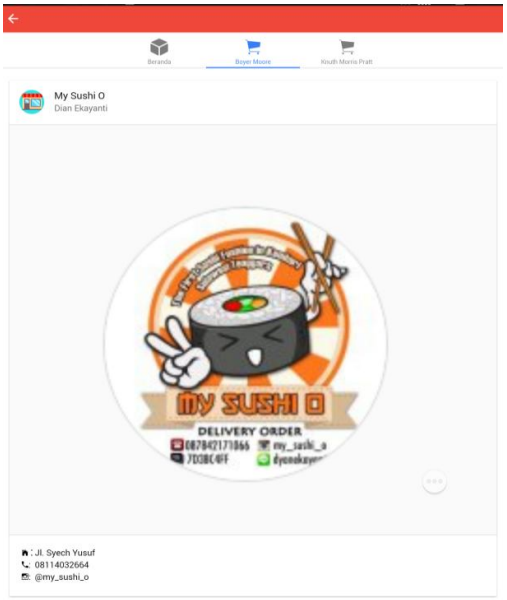
Gambar 5menunjukkan *interface* dari detail makanan



Gambar 5 *Interface* Menu Detail Makanan Pada *Algoritma Boyer Moore* dan *Knuth Morris Pratt*

Interface dari detail makanan yang menampilkan gambar makanan dan harga makanan.

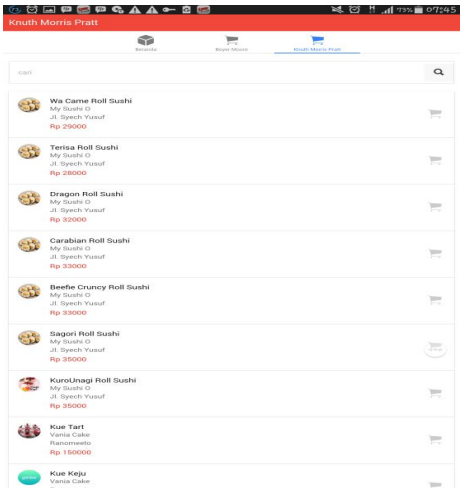
Gambar 6 merupakan *interface* dari detail toko



Gambar 6 *Interface* Detail Toko Pada *Algoritma Boyer Moore* dan *Knuth Morris Pratt*

Menampilkan gambar toko, alamat toko, contact person dan akun instagram penjual.

Gambar 7 menunjukkan *interface* menu pencarian



Gambar 7 *Interface* Menu Pencarian Menggunakan *Algoritma Boyer Moore* dan *Knuth Morris Pratt*

*Interface*Menu Pencarian dari aplikasi menggunakan *Algoritma Knuth Morris Pratt.User* cukup memasukkan nama makanan yang ingin dicari di kolom pencarian. Selanjutnya akan ditampilkan menu-menu makanan dan harga makanan yang berada dalam *database* dan waktu pencarian.

3.2. Uji Coba Aplikasi

Pengujian ini dilakukan untuk memeriksa ketepatan menu makanan yang ditampilkan dengan menu makanan yang dicari oleh *user*. Pengujian ini dilakukan dengan memasukkan contoh-contoh huruf atau kata yang akan dicari kedalam *form* aplikasi untuk menguji performa dari aplikasi. Tabel 1 menunjukkan Waktu Pencarian Menggunakan *Algoritma Boyer Moore*.

Tabel 1 Waktu Pencarian Menggunakan *Algoritma Boyer Moore*

No.	Kata yang dicari	Algoritma BM										Rata-rata (ms)	Standar deviasi
1	a	1.53	0.82	0.71	0.71	1.76	1.44	0.72	1.64	1.48	0.88	1.17	0.43
2	c	1.44	0.75	0.78	1.49	0.84	0.75	0.79	1.58	2.11	1.29	1.18	0.47
3	bu	0.81	0.61	0.58	0.63	1.97	1.46	1.53	0.81	0.6	0.77	0.98	0.50
4	na	0.89	0.71	0.69	0.65	0.55	0.98	0.56	0.62	1.09	0.72	0.75	0.18
5	ayam	0.67	0.58	0.74	0.58	0.35	0.45	1.86	1.41	0.45	0.54	0.76	0.48
6	nasi	0.6	0.45	0.45	0.56	0.4	0.54	0.54	0.48	0.49	0.69	0.52	0.08
7	kue	1.26	0.56	1.94	0.56	1.92	0.84	2.91	0.52	1.01	0.44	1.20	0.81
8	pie	2.29	2.28	0.62	3.04	0.72	0.68	0.57	1.27	0.44	0.47	1.24	0.95
9	pizza	3.84	0.47	1.12	0.48	0.65	0.56	0.31	0.53	2.13	0.36	1.04	1.12
10	roti	0.69	0.71	0.42	0.44	0.49	0.46	0.45	0.47	0.66	0.91	0.57	0.16
11	keju	2.63	1.31	0.49	0.68	1.32	0.68	3.34	1.67	1.06	1.39	1.46	0.90
12	coklat	1.44	2.7	1.49	3.19	0.6	0.46	1.47	0.46	0.35	0.66	1.28	0.99
13	bubur	1.26	2.02	0.61	1.23	1.4	1.37	1.01	1.93	1.23	0.89	1.29	0.43
14	nasi goreng pisang goreng coklat	1.38	1.22	0.5	0.37	2.03	0.6	1.43	0.48	1.02	0.35	0.94	0.57
15	Nasi ayam	0.46	0.42	0.36	0.74	0.27	1.49	1.3	0.42	0.43	1.4	0.73	0.48
16	Sandwich	3.07	2.22	0.94	0.77	0.49	0.45	0.94	2.73	1.79	1.76	1.52	0.94
17	Roll	0.41	0.86	1.88	0.63	0.66	1.32	0.39	0.99	0.84	0.85	0.88	0.44
18	cumi	1.9	1.81	0.41	0.52	1.9	1.74	2.21	2.17	1.52	0.65	1.43	0.70
19	bayam	1.57	0.64	0.56	0.95	0.73	0.97	2.66	0.69	0.52	2.23	1.15	0.75
20		0.36	1.58	0.4	0.45	0.73	0.4	1.39	0.34	1.4	0.44	0.75	0.50

Berdasarkan pencarian pada 20 kata dan huruf yang dimasukkan menggunakan *Algoritma Boyer Moore* diperoleh rata-rata waktu pencarian selama 1.04 ms. Perbedaan *Algoritma Boyer Moore* dan perintah *select * form* adalah *Algoritma Boyer Moore* mencoba kecocokan karakter dari arah sebelah kanan kata yang dimasukkan kemudian menentukan jumlah pergeseran apabila tidak menemukan kecocokan sampai ditemukan kecocokan sesuai kata yang dimasukkan sedangkan perintah *select * form* menampilkan kata yang sama persis dengan kata yang dimasukkan misal : kata yang tersedia nasi ayam,, apabila kita memasukkan kata “na” pada *Algoritma Boyer Moore* algoritma ini akan menampilkan data nasi ayam karena data ini memuat kata “na” sedangkan pada parintah *select * form* tidak akan ditampilkan.

Tabel 2 menunjukkanWaktu Pencarian Menggunakan *Algoritma Knuth Morris*

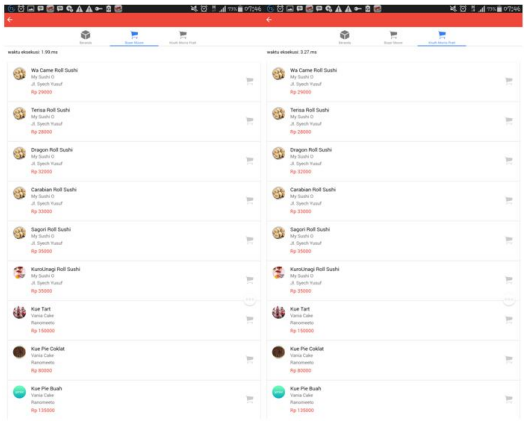
Tabel 2 Waktu Pencarian Menggunakan *Algoritma Knuth Morris*

No.	Kata yang dicari	Algoritma KMP										Rata-rata (ms)	Standar deviasi
1	a	5.43	4.15	1.97	2.56	2.27	3.82	1.43	2	3.55	2.03	2.92	1.26
2	c	3.76	3.48	1.52	2.67	1.41	2.18	3.73	3.15	2.71	1.86	2.65	0.88
3	bu	1.74	1.57	1.95	1.86	1.9	2.93	1.4	2.16	2.17	2.14	1.99	0.42
4	na	2.77	1.21	2.29	1.72	1.5	2.11	2.64	2.02	1.93	1.4	1.96	0.52
5	ayam	1.89	2.33	2.75	2.48	2.37	2.49	1.41	2.13	1.5	1.51	2.08	0.48
6	nasi	1.76	1.94	2.67	2.62	2.32	1.3	2.32	2.95	1.83	1.62	2.13	0.52
7	kue	2.55	2.19	1.57	1.61	1.52	2.24	2.3	1.33	1.52	1.54	1.84	0.43
8	pie	3.73	3.12	1.6	1.68	2.74	1.18	2.18	1.81	1.73	1.31	2.12	0.83
9	pizza	1.94	3.57	3.43	3.15	3.9	1.79	3.18	3.73	1.34	1.32	2.73	1.02
10	roti	1.73	1.74	1.96	1.92	2.33	2.17	2.27	2	2.43	1.8	2.03	0.25
11	keju	2.81	3.26	3.06	2.24	2.97	3.28	4.21	2.52	3.1	2.45	2.99	0.55
12	coklat	3.13	2.27	2.13	1.82	2.89	2.93	2.7	1.37	2.47	2.05	2.38	0.55
13	bubur	3.78	2.28	2	1.16	1.77	2.82	2.78	2.66	3.18	2.74	2.52	0.74
14	nasi goreng pisang goreng coklat	1.77	1.77	1.79	1.46	3.13	4.02	2.25	3.86	2.94	1.5	2.45	0.97
15	Nasi ayam	2.88	1.32	1.31	4.14	2.72	5.07	1.7	1.6	1.66	2.6	2.5	1.27
16	Sandwich	5.72	1.62	2.59	1.29	1.57	1.91	3.09	1.07	2.18	3.11	2.41	1.36
17	Roll	3.25	2.14	1.27	1.82	3.29	2.34	4.39	1.88	1.84	1.19	2.34	1.07
18	cumi	3.17	3.78	3.3	3.09	2.41	2.36	1.42	1.79	3.44	2.62	2.74	0.75
19	bayam	3.45	3.05	2.37	2.49	1.88	1.59	2.74	3.41	1.95	3.46	2.64	0.70
20		2.77	1.5	2.02	3.19	1.47	1.83	2.43	2.32	1.57	3.96	2.31	0.81

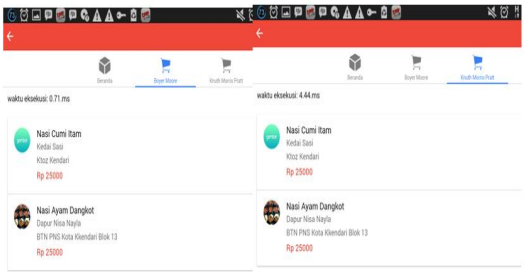
Tabel 2 menunjukkan berdasarkan pencarian pada 20 kata dan huruf yang dimasukkan menggunakan *Algoritma Knuth Morris Pratt* diperoleh rata-rata waktu pencarian selama 2.38 ms. Perbedaan *Algoritma Boyer Moore* dan perintah *select * form* adalah *Algoritma Boyer Moore* mencoba kecocokan karakter dari arah sebelah kanan kata yang dimasukkan kemudian menentukan jumlah pergeseran apabila tidak menemukan kecocokan sampai ditemukan kecocokan sesuai kata yang dimasukkan sedangkan perintah *select * form* menampilkan kata yang sama persis dengan kata yang dimasukkan misal : kata yang tersedia nasi ayam,, apabila kita memasukkan kata “na” pada *Algoritma Boyer Moore* algoritma ini akan menampilkan data nasi ayam karena data ini memuat kata “na” sedangkan pada parintah *select * form* tidak akan ditampilkan.

Dari hasil uji coba terhadap kedua algoritma, urutan pengujian tidak berpengaruh terhadap waktu pencarian, karena waktu yang ditunjukkan pada kedua algoritma tersebut apabila diubah urutan pengujiannya tetap berbeda-beda.

Contoh penerapan pada aplikasi ditunjukkan pada Gambar 10. Gambar 8 menunjukkan tampilan menu makanan yang mempunyai huruf yang sama pada dengan huruf yang dicari, beserta waktu pencarian. Gambar 9 menunjukkan tampilan menu makanan

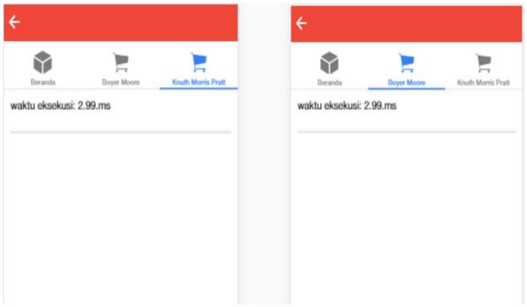


Gambar 8 *Interface* Hasil Pencarian Huruf A Pada Algoritma*Boyer-Moore* danAlgoritma *Knuth Morris Pratt*



Gambar 9 *Interface* Hasil Pencarian Kata Nasi Pada Algoritma*Boyer-Moore* danAlgoritma *Knuth Morris Pratt*

Gambar 9 menunjukkan tampilan menu makanan yang mempunyai kesamaan kata dengan kata yang dicari, beserta waktu pencarian.



Gambar 10 *Interface* Hasil Pencarian Kata Yang Tak Ditemukan Pada Algoritma*Boyer-Moore* danAlgoritma *Knuth Morris Pratt*

Gambar 10 menampilkan waktu pencarian apabila tidak ditemukan kata yang cocok dengan yang akan dicari.

4. KESIMPULAN

Dari hasil pengembangan aplikasi Tripelka *Foodshop* Kendari menggunakan smartphone berbasis android dapat diambil beberapa kesimpulan, yaitu :

1. Aplikasi Tripelka *Foodshoop* Kendari ini telah diterapkan pada *smartphone* berbasis android.
2. Aplikasi ini dilengkapi dengan *Algoritma Boyer Moore* dan *Algoritma Knuth Morris Pratt* yang memudahkan user untuk mencari menu makanan yang diinginkan dengan mudah dan cepat.
3. Dari hasi uji coba, *Algortima Knuth Morris Pratt* lebih banyak melakukan pergeseran sebelum menemukan kata yang cocok dibandingkan *Algoritma Boyer Moore.*
4. Dari hasil pengujian dapat disimpulkan bahwa jumlah data yang ditampilkan pada *Algoritma Boyer Moore* dan *Knuth Morris Pratt* adalah sama dan tidak ada data hilang atau tidak tertampil dari *database*.
5. Berdasarkan rata-rata waktu pencarian menunjukkan 100% dari 20 data uji lebih cepat menggunakan *Algoritma Boyer* Moore dibandingkan menggunakan *Algoritma Knuth Morris Pratt.*

5. SARAN

Saran untuk pengembangan lebih lanjut dari aplikasi Tripelka *Foodshoop* Kendari ini adalah aplikasi ini masih berbentuk *offline*, pengembangan selanjutnya diharapkan berbentuk *online*

DAFTAR PUSTAKA

[1] Antonius, R., 2013, *Algoritma BM pada Permainan Word Search Puzzle*, Yogyakarta, Universitas Kristen Duta Wacana.

[2] Boko, S., 2014, *Efisiensi Penggunaan Algoritma BM untuk Prediksi Perilaku Orang Melalui Interaksi di Twitter*, Semarang, Universitas Dian Huswantoro.

[3] Abi, M., 2015, *Implementasi Algoritma KMP Pada Perancangan Game Hanacaraka*, Semarang, Universitas Negeri Semarang.

[4] Tripelka, 2016, *Komunitas Kuliner Kendari*, Kendari.

[5] Priawan, M., 2013, *Teknologi, Smartphone, Android*.

[6] Moreno, Z., 2015, *AngularJS Deployment Essentials*, Packt Publishing.

[7] Harahap, N.S., 2012, *Pemograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*, Bandung : Informatika.

[8] Kadir, A., 2002, *Dasar Pemrograman Web Dinamis Menggunakan PHP*, Yogyakarta: Andi.

[9] Lanhof, S., 2015, *Algoritma dan Pemrograman*, Jogjakarta, Penerbit Andi.

[10] Sarno, R., 2012, *Semantic Search*, Penerbit Andi.

[11] Chiquita, Christabella., 2011, *Penerapan Algoritma Boyer Moore-Dynamic Programming untuk Layanan Auto-Complete dan Auto-Correct*, Makalah IF3051 Strategi Algoritma.

[12] Munir, R., 2006, *Strategi Algoritmik Departemen Teknik Informatika*, Institut Teknologi Bandung

[13] Process, R. U., 2001, *Best Practices for Software Development Teams. A Rational Software Corporation White Paper*.