BAB II
TINJAUAN PUSTAKA

Dalam bab ini berisi dasar teori mengenai penelitian yang telah diambil yaitu, "Penerapan Algoritma *Knuth-Morris-Pratt* Pada Aplikasi Pengenal Nominal Uang Untuk Tuna Netra Berbasis *Android*". Tinjauan pustaka ini diperlukan sebagai acuan pada saat melakukan penelitian agar bisa tetap berada dalam aturan yang berlaku. Dalam bab ini juga akan dijelaskan beberapa penelitian terdahulu yang berkaitan dengan penelitian ini, Hal ini bertujuan untuk mendapatkan gambaran mengenai proses dalam pembuatan rancangan aplikasi dalam penelitian ini. Dengan melakukan tinjauan pustaka pada penelitian sebelumnya diharapkan dapat memberikan pembelajaran agar bisa mencapai tujuan yang telah dijelaskan pada bab pendahuluan sebelumnya.

2.1 Penelitian Terdahulu

Dalam penelitian ini penulis mempertimbangkan penelitian terdahulu sebagai acuan, ini sangat penting karena dalam sebuah penelitian pasti ada hubungan dari penelitian yang sejenis, dengan adanya acuan dari penelitian terdahulu ini diharapkan bisa memberikan pembelajaran dalam melakukan penelitian untuk penerapan algoritma *Knuth-Morris-Pratt* ini.

Ada beberapa algoritma yang bisa digunakan dalam menyelesaikan penelitian ini, dari beberapa algoritma yang telah penulis bandingkan dari penelitian terdahulu membuktikan algoritma KMP atau *Knuth-Morris-Pratt* lebih unggul dari algoritma - algoritma lain untuk pencocokan string. Berikut ini adalah beberapa penelitian - penelitian terdahulu yang telah penulis review :

Penelitian pertama adalah penelitian yang dilakukan oleh (Subaeki, 2017), dengan judul "Optimalisasi Perbandingan Algoritma *Brute Force* Dan *Knuth-Morris-Pratt* Untuk Meningkatkan Kecepatan Pencarian Data Pada Aplikasi *Mobile* Tentang Hewan *Vertebrata*." dalam penelitian ini membahas tentang perbandingan dari algoritma *Brute Force* dan *Algoritma Knuth-Morris-Pratt* untuk pencarian data, hasil dari penelitian ini menunjukkan Algoritma KMP lebih unggul dari segi perhitungan kompleksitas waktu.

Penelitian kedua adalah penelitian dari (Fazira, 2019), dengan judul "Perbandingan Algoritma *Knuth-Morris-Pratt* dan *Boyer Moore* dengan Metode Perbandingan Eksponensial Pada Aplikasi Kamus Bahasa Indonesia - Jerman Berbasis *Android*" pada penelitian ini membahas tentang perbandingan antara algoritma *Knuth-Morris-Pratt* dan algoritma *Boyer*

Moore pada aplikasi kamus indonesia - jerman, dalam penelitian ini menghasilkan bahwa algoritma Knuth-Morris-Pratt lebih cepat dari algoritma *Boyer Moore* dalam hal kecepatan pencarian data.

Penelitian ketiga adalah penelitian dari (Sadiah, 2017), dengan judul "Implementasi Algoritma *Knuth-Morris-Pratt* Pada Fungsi Pencarian Judul Tugas Akhir Repository" pada penelitian ini membahas tentang implementasi algoritma Knuth-Morris-Pratt untuk pencarian judul repository tugas akhir, dari penelitian ini memberikan hasil rata-rata dari performa Knuth-Morris-Pratt dalam mencari kata di form pencarian adalah sebesar 0.0138 detik, ini menunjukan bahwa algoritma ini cukup optimal dan cepat dalam fungsi pencarian.

Penelitian keempat adalah penelitian dari Maulana dan (Maulana, 2019) dengan judul "Penerapan Algoritma *Knuth-Morris-Pratt* pada Fungsi Pencarian Dokumen untuk Sistem Informasi Administrasi Sekolah Berbasis Website" penelitian ini membahas tentang implementasi algoritma *Knuth-Morris-Pratt* untuk pencarian dokumen sistem informasi administrasi sekolah, hasil dari penelitian ini memberikan kesimpulan bahwa algoritma *Knuth-Morris-Pratt* bisa mengendalikan dokumen secara efektif serta efisien dan memiliki hasil yang tepat untuk pencarian dokumen.

Penelitian kelima adalah penelitian dari (Zainab et al., n.d.) dengan judul "Perbandingan Algoritma *Horspool* dan Algoritma *Knuth-Morris-Pratt* Pada Aplikasi Kamus Farmasi Berbasis *Android*" penelitian ini membahas tentang perbandingan antara algoritma Horspool dan algoritma Knuth-Morris-Pratt, dalam penelitian ini memberikan hasil bahwa algoritma Knuth-Morris-Pratt memiliki kecepatan yang lebih baik dalam melakukan pencarian dibanding dengan algoritma Horspool, dengan rata - rata waktu yang dibutuhkan adalah 7.96 ms, jauh dibandingkan dengan algoritma Horspool yaitu 53.87 ms.

Tabel 2. 1 Tabel Penelitian Terdahulu

No.	Penulis	Judul	Hasil
1.	(Subaeki, 2017)	Optimalisasi Perbandingan Algoritma *Brute Force* Dan *Knuth-Morris-Pratt* Untuk Meningkatkan Kecepatan Pencarian	Hasil dari penelitian ini menunjukkan bahwa *Algoritma Knuth-Morris-Pratt* unggul dari segi perhitungan kompleksitas waktu.

		Data Pada Aplikasi *Mobile* Tentang Hewan *Vertebrata.*	
2.	(Fazira, 2019)	Perbandingan Algoritma *Knuth-Morris-Pratt* dan *Boyer Moore* dengan Metode Perbandingan Eksponensial Pada Aplikasi Kamus Bahasa Indonesia - Jerman Berbasis *Android.*	Dalam penelitian ini menghasilkan bahwa algoritma Knuth-Morris-Pratt lebih cepat dari algoritma Boyer Moore dalam hal kecepatan pencarian data.
3.	(Sadiah, 2017)	Implementasi Algoritma *Knuth-Morris-Pratt* Pada Fungsi Pencarian Judul Tugas Akhir Repository.	Dari penelitian ini memberikan hasil rata-rata dari performa Knuth-Morris-Pratt dalam mencari kata di form pencarian adalah sebesar 0.0138 detik, ini menunjukan bahwa algoritma ini cukup optimal dan cepat dalam fungsi pencarian.
4.	(Maulana, 2019)	Penerapan Algoritma Knuth-Morris-Pratt pada Fungsi Pencarian Dokumen untuk Sistem Informasi Administrasi	Hasil dari penelitian ini memberikan kesimpulan bahwa algoritma *Knuth-Morris-Pratt* bisa mengendalikan dokumen secara efektif serta efisien dan memiliki hasil yang tepat untuk pencarian dokumen.

		Sekolah Berbasis Website	
5.	(Zainab et al., n.d.,2020)	Perbandingan Algoritma Horspool dan Algoritma Knuth-Morris Pratt Pada Aplikasi Kamus Farmasi Berbasis Android	Dalam penelitian ini memberikan hasil perbandingan dengan menyatakan bahwasannya algoritma KMP memberikan kecepatan yang lebih baik dalam melakukan pencarian dibanding dengan algoritma Horspool, dengan rata - rata waktu yang dibutuhkan adalah 7.96 ms, jauh dibandingkan dengan algoritma Horspool yaitu 53.87 ms.

2.2 Android

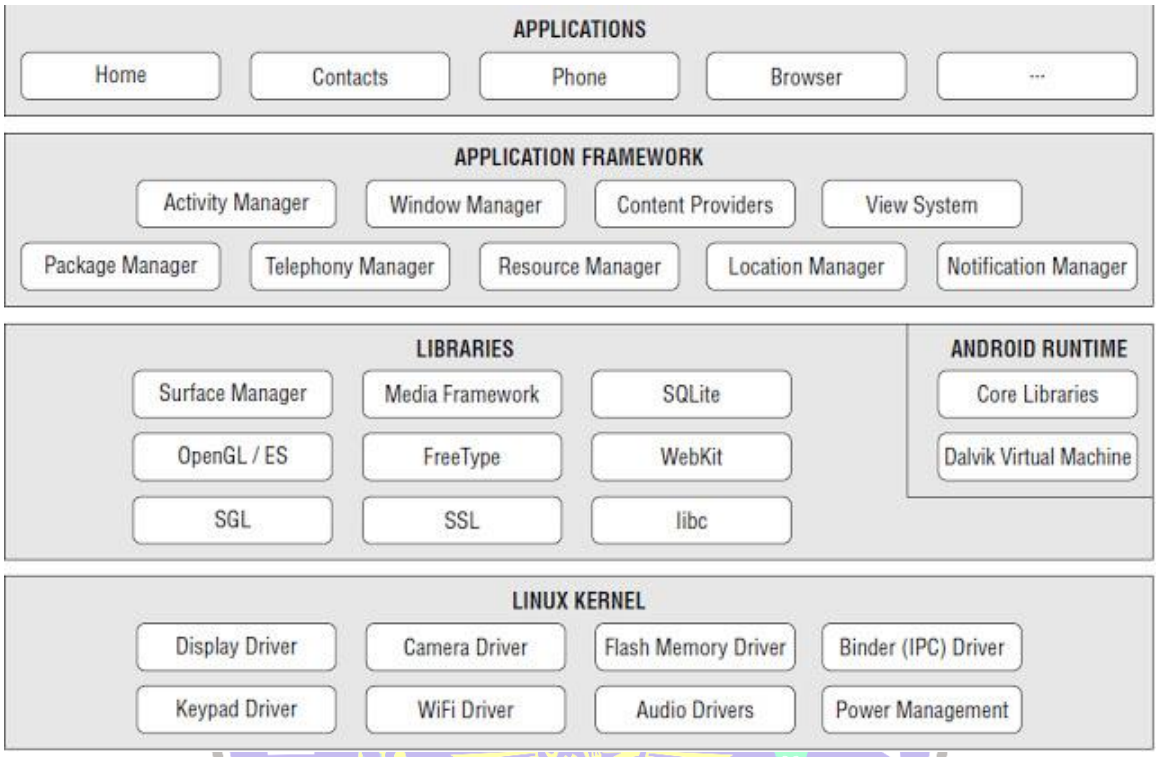
Sistem operasi Android merupakan sistem operasi yang biasa di sematkan pada perangkat *smartphone* sistem operasi ini berbasis *Linux* dan saat ini dikembangkan oleh *Google*, sistem ini dulunya dikembangkan oleh perusahaan rintisan dengan nama yang sama yaitu *Android* lalu diambil alih oleh *Google* pada tahun 2005, lalu pada tahun 2007 *Google* mengembangkan sistem operasi *Android* dengan merilis *SDK* (*System Development Kit*).(Seng Hansun, Marcel Bonar Kristanda, 2018).

Android biasa disematkan pada ponsel pintar selain itu sistem operasi *android* juga terdapat pada jam tangan pintar serta tv digital. Sistem operasi *Android* bersifat terbuka dan bisa dikembangkan oleh siapa saja, namun sekarang *Android* telah dikembangkan oleh *Google Inc* dan akan update setiap tahunnya. Awalnya sistem operasi *Android* untuk ponsel pintar dan juga tablet layar sentuh. Namun pada saat ini *Android* dikembangkan juga pada perangkat lain seperti televisi, jam tangan pintar, dan kamera digital. Android sampai saat penelitian ini dibuat semakin berkembang pesat dan membuat para pengembang aplikasi tertarik untuk mengembangkan aplikasi di sistem operasi *Android*. Sampai saat ini sudah ada total jutaan aplikasi yang telah terindeks di *Google Play Store*. (Agus Wahadyo, 2013)

2.2.1 Arsitektur Android

Arsitektur sistem operasi *android* pada umumnya dapat digambarkan seperti pada Gambar 2.1, Pada gambar tersebut diperlihatkan bahwa terdapat empat tingkatan atau lapisan dalam

arsitektur *Android* yang dimulai dari *Linux Kernel* sebagai lapisan terbawah, diikuti lapisan *Libraries* dan *Android Runtime*, lapisan *Application Framework*, Hingga lapisan *Applications*. (Seng Hansun, Marcel Bonar Kristanda, 2018).



Gambar 2. 1 Arsitektur Android

Linux kernel yang berada pada lapisan terbawah dari susunan android menyediakan suatu perantara bagi perangkat keras (hardware) dengan lapisan diatasnya. Berdasarkan versi *linux* 2.6, *Linux Kernel* menyediakan preemptive multitasking dan layanan level dasar inti sistem, seperti memory, process, dan power management. Selain itu, Kernel tersebut juga menyediakan suatu tingkatan jaringan dan device drivers untuk perangkat keras, seperti monitor, WI-FI, dan audio. Saat suatu aplikasi *Android* dibangun dengan *Android Studio*, aplikasi tersebut dikompilasikan menjadi suatu intermediate bytecode yang kerap dirujuk sebagai DEX format. Saat aplikasi tersebut dijalankan ke suatu perangkat, *Android Runtime (ART)* menggunakan suatu proses yang disebut dengan *Ahead-of-Time (AOT)* compilation untuk menerjemahkan bytecode tersebut menjadi perintah - perintah dasar yang dapat dieksekusi oleh prosesor perangkat tersebut. Format ini dikenal sebagai *Executable and Linkable Format (ELF).* Setiap kali aplikasi tersebut dijalankan dengan menggunakan versi *ELF*, aplikasi dapat berjalan dengan lebih cepat dan meningkatkan daya tahan baterai. Hal ini sangat berlawanan dengan pendekatan *Just-In-Time (JIT)* compilation yang digunakan oleh versi android yang lebih lama

dimana *bytecode* diterjemahkan dalam suatu mesin virtual (Virtual Machine) lebih dulu setiap kali aplikasi tersebut dijalankan. Selain libraries pengembangan standar Java, lingkungan pengembangan Android juga menambahkan Android Libraries yang diperuntukan khusus bagi pengembangan Android. Application framework libraries dan libraries lain yang mendukung pembuatan elemen antarmuka, gambar grafis dan, dan akses database merupakan contoh libraries yang masuk dalam kategori ini. Application Framework merupakan sekumpulan layanan yang bersama - sama membentuk suatu lingkungan di mana aplikasi - aplikasi Android berjalan dan diatur. *Framework* ini menerapkan konsep saling berbagi dan saling melengkapi yang mana suatu aplikasi mampu membagikan kemampuan serta data - data di dalamnya. Selanjutnya terdapat lapisan paling atas dari susunan arsitektur android adalah Applications. Lapisan ini meliputi seluruh aplikasi bawaan (native applications) yang telah dilengkapi dengan fitur khusus (seperti aplikasi web browser dan aplikasi email), maupun aplikasi - aplikasi pihak ketiga yang diinstal sendiri oleh pengguna setelah membeli perangkat tersebut. (Seng Hansun, Marcel Bonar Kristanda, 2018).

2.3 Android Studio

Android Studio merupakan alat pengembangan perangkat lunak untuk sistem operasi *Android*. *Android Studio* merupakan perangkat lunak resmi dari *Google* dan biasa juga disebut *(Integrated Development Environment)* yang didasarkan pada IntelliJ IDEA. *Android Studio* mendukung untuk pengembangan aplikasi dalam bahasa pemrograman *Java* dan *Kotlin* serta bisa juga untuk pengembangan aplikasi *cross-platform* berbasis *flutter* dengan bahasa pemrograman *Dart*. Agar dapat digunakan untuk pengembangan aplikasi *Android* maka Android Studio membutuhkan kit yang bernama SDK (Software Development Kit) kit ini berfungsi untuk mempermudah dalam memulai pengembangan aplikasi Android. (https://developer.android.com, 2020).

2.4 Bahasa Pemrograman Java

Bahasa pemrograman *Java* merupakan bahasa pemrograman yang dikembangkan oleh perusahaan teknologi *Oracle* bahasa pemrograman ini biasa digunakan untuk pengembangan perangkat lunak khususnya dalam sisi *backend development* serta bisa juga untuk *mobile*. Bahasa pemrograman ini pertama kali dipopulerkan oleh James Gosling disaat masih di perusahaan *Sun Microsystems*. Bahasa pemrograman ini adalah hasil dari pengembangan bahasa terdahulu yaitu bahasa pemrograman C++ karena banyak sekali kesamaan dari segi *syntax*. Kepopuleran bahasa pemrograman *Java* saat ini meningkat karena bahasa ini mudah

dipahami dan tidak jauh beda dengan bahasa pemrograman terdahulu. Bahasa Pemrograman *Java* memiliki berbagai banyak kelebihan salah satunya bisa digunakan multiplatform contohnya aplikasi ponsel,komputer serta website, selain itu bahasa ini juga menganut OOP (*Object Oriented Programming*) dan memiliki banyak library yang lengkap. (Nofriadi, 2020).

Java menjalankan sistemnya di atas suatu mesin yang disebut juga dengan interpreter yang dinamakan dengan Java Virtual Machine (JVM). JVM ini akan membaca file berupa file yang berekstensi .class dari suatu program yang bisa juga direpresentasikan sebagai bahasa mesin. Maka dari itu bahasa pemrograman Java biasa disebut sebagai bahasa pemrograman yang portable karena bisa dijalankan diberbagai sistem operasi termasuk sistem operasi Android. (M. Shalahuddin dan Rosa, 2009)

2.4.1 Percabangan

Percabangan if dalam *Java* kurang lebih sama dengan bahasa pemrograman lain yang mana pada percabangan ini memiliki tugas untuk menjalankan sebuah aksi yang memiliki kondisi dimana jika dalam kondisi yang telah diinisiasikan tidak bisa terpenuhi maka blok kode yang dijalankan akan terjadi *error* begitupun sebaliknya jika dalam sebuah kondisi dapat dipenuhi maka kode akan bisa dijalankan dengan berbagai aksi, percabangan biasa digunakan untuk menjalankan aksi dengan berbagai kondisi contohnya adalah pada proses pencarian jika pencarian belum ketemu maka proses akan terus berjalan namun ketika yang dicari telah ditemukan maka pencarian akan dihentikan.(Sianipar, 2018).

2.4.2 Perulangan

Dalam perulangan ini bertujuan untuk melakukan suatu proses yang berulang dalam suatu program komputer dengan sesuai perintah yang diinginkan. Dalam suatu program tentunya ada beberapa banyak aksi yang perlu dibutuhkan salah satunya adalah kegiatan mengulang dalam hal ini perlu dibutuhkan suatu proses yang dinamakan perulangan atau *looping.* Dengan adanya perulangan pemrogram sangat diuntungkan dikarenakan pemrogram tidak perlu untuk menulis kode program sebanyak perulangan yang diinginkan. Didalam proses perulangan atau *looping* akan mempunyai beberapa bagian yang harus dipenuhi antara lain adalah : (Sianipar, 2018).

1. Inisialisasi

Tahap inisialisasi merupakan tahapan pertama dalam perulangan, biasanya inisialisasi terjadi pada awal dan bertujuan untuk mendefinisikan variabel yang akan digunakan untuk perulangan.

2. Proses

Dalam tahapan proses merupakan tahapan yang digunakan untuk menerapakan algoritma yang mana berisi perulangan dengan mencapai suatu tujuan atau aksi tertentu dalam kode program.

3. Iterasi

Iterasi merupakan suatu proses dan kondisi pertambahan nilai yang terus berjalan di dalam perulangan atau *looping*.

4. Terminasi

Terminasi merupakan suatu kondisi stop atau berhenti pada perulangan, kondisi berhenti ini sangat penting didalam perulangan supaya perulangan bisa berhenti, dan tidak akan menjadi perulangan yang tiada hentinya atau *infinity loop*.

2.4.3 Larik (Array)

Pengertian array sangatlah luas namun pada hakikatnya array merupakan sekumpulan data berupa variabel yang dikumpulkan menjadi satu dengan tipe data yang sama satu sama lain, array biasanya didefinisikan dengan bracket [] di bahasa pemrograman manapun termasuk dalam Java, dalam dunia pemrograman fungsi dari array sangat penting unruk menyimpan data, jika saja array tidak ada maka akan butuh banyak penyimpanan yang akan membuat kode program berat dan tidak efisien. (Sianipar, 2018).

2.5 Algoritma Knuth-Morris-Pratt

Algoritma ini adalah algoritma yang biasa difungsikan untuk pencarian string (*string matching)*, algoritma ini ditemukan oleh D.E Knuth, J.H Morris, dan V.R Pratt. Algoritma ini memungkinkan untuk mencari substring dari sebuah string dengan cara mencocokan masing - masing karakternya. Algoritma ini berbeda dengan algoritma terdahulu yaitu algoritma *Brute Force* dimana pada algoritma *Brute Force* jika saat pencocokan *pattern* karakter yang tidak sesuai, maka karakter akan bergeser ke kanan, namun pada algoritma *Knuth-Morris-Pratt* ini, pada saat sebelum memulai pencocokan karakter, algoritma ini menyimpan segala informasi yang bisa digunakan untuk mengetahui dimana pencocokan selanjutnya akan dilakukan, sehingga tidak perlu adanya pergeseran. (Munir, 2004).

Sebagai contohnya, misalkan kita akan mencari pattern berupa karakter teks sebuah angka ‘10000’ di dalam teks ‘10010000’ maka :

Teks : 10010000

Pattern : 10000

Dapat dilihat dari contoh tersebut bahwa jika dijalankan akan gagal karena teks dan pattern berbeda pada karakter ke-4. Jika kita menggunakan pencarian string Brute Force maka pencocokan setelahnya akan dilakukan mulai pada karakter ke-3 jika menggunakan algoritma KMP hasilnya akan menjadi seperti berikut :

Teks : 10010000

Pattern : 10000

Pencocokkan string akan dimulai berdasarkan karakter dengan tingkat kecocokan tertinggi. Selain daripada itu pencocokan pada string juga langsung memulai dari karakter pattern ke-2 karena pencocokkan karakter pertama telah dilakukan saat pengambilan informasi. (Munir, 2004).

2.5.1 Cara Kerja Algoritma Knuth-Morris-Pratt

Pada prinsipnya cara kerja algoritma knuth-morris-pratt hampir mirip dengan algoritma *brute force* namun ada beberapa peningkatan yang signifikan terutama dari segi waktu pencarian yang lebih cepat dari algoritma terdahulu (Hakim et al., 2019).

Secara sistematik, berikut ini merupakan tahap - tahapan yang dilakukan algoritma Knuth-Morris-Pratt dalam pencocokan string:

1. Algoritma Knuth-Morris-Pratt akan memulai pencocokan string ke pattern pada awal teks.
2. Dalam algoritma ini proses pencocokan karakternya dimulai dari arah kiri ke kanan sampai memenuhi karakter yang dituju sama dengan pattern, lalu algoritma KMP akan mengirim hasil dari pencariannya yang telah dicocokan ke posisi awalnya.
3. Dalam tahap selanjutnya akan terjadi pergeseran pattern berdasarkan tabel next, lalu langkah kedua akan diulangi sampai dengan pattern berada pada ujung teks.

2.5.2 Pseudocode Algoritma Knuth-Morris-Pratt

Berikut ini merupakan pseudocode pada tahap pra-pencarian dari algoritma Knuth-Morris-Pratt :

```
procedure preKMP(
    input P: array[0..n-1] of char,
    input n: integer,
    input/output kmpNext: array[0..n] of integer
)
Deklarasi:
i,j: integer

Algoritma
  i:= 0;
  j:= kmpNext[0]:= -1;
  while (i < n) {
     while (j > -1 and not(P[i] = P[j]))
        j:= kmpNext[j];
     i:= i+1;
     j:= j+1;
     if (P[i] = P[j])
        kmpNext[i]:= kmpNext[j];
     else
        kmpNext[i]:= j;
     endif
  endwhile
```

Gambar 2. 2 *Pseudocode* pra Pencarian Algoritma *Knuth-Morris-Pratt*

Pada Gambar 2.2 merupakan contoh pseudocode dalam proses *preKMP*, proses ini merupakan proses awal dalam algoritma KMP *Knuth-Morris-Pratt* dengan tujuan untuk memecah karakter dalam bentuk array, dalam psedocode tersebut menggunakan perulangan while untuk memproses iterasi sehingga kode bisa berjalan sesuai dengan kaidah-kaidah dalam algoritma KMP *Knuth-Morris-Pratt*

Berikut ini merupakan pseudocode pada tahap pencarian dari algoritma Knuth-Morris-Pratt :

```
procedure KMPSearch(
    input m, n: integer,
    input P: array[0..n-1] of char,
    input T: array[0..m-1] of char,
    output ketemu: array[0..m-1] of boolean
)

Deklarasi:
i, j,next: integer
kmpNext: array[0..n] of interger

Algoritme:
preKMP(n, P, kmpNext)
i:=0
while (i<= m-n) do
    j:=0
    while (j < n and T[i+j] = P[j]) do
       j:=j+1
    endwhile
    if(j >= n) then
       ketemu[i]:=true;
    endif
    next:= j - kmpNext[j]
    i:= i+next
endwhile
```

Gambar 2. 3 Pseudocode Tahap Pencarian Algoritma Knuth-Morris-Pratt

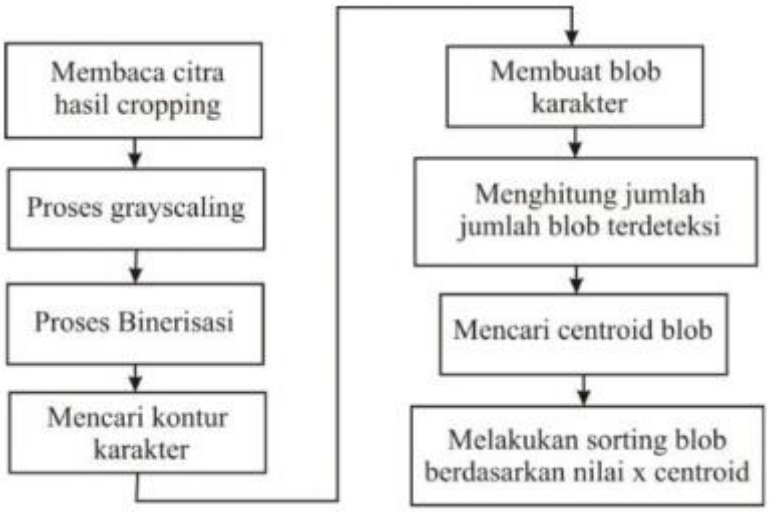
Dalam Gambar 1.3 merupakan contoh pseudocode pada proses pencarian string di dalam algoritma KMP Knuth-Morris-Pratt, pada proses pencarian string ini merupakan lanjutan dari proses sebelumnya yaitu proses preKMP atau proses pemecahan karakter ke dalam array, pada proses pencarian string ini pencariannya dimulai dari kiri ke kanan sesuai dengan kaidah algoritma KMP Knuth-Morris-Pratt dengan menggunakan perulangan while.

2.6 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) adalah proses klasifikasi pola optik yang terkandung dalam citra digital yang sesuai dengan alfanumerik atau karakter lainnya. Pengenalan karakter dicapai melalui langkah-langkah penting seperti segmentasi, ekstraksi fitur dan klasifikasi. OCR telah mendapatkan perhatian yang meningkat baik dalam penelitian akademis maupun dalam industri karena banyak keunggulannya. Salah satu nya yaitu replikasi fungsi manusia dalam membaca dokumen yang mencakup berbagai bentuk teks.(Gunawan et al., 2014) Sebagai sebuah aplikasi yang kompleks OCR memiliki sejumlah tahapan antara lain adalah:

2.6.1 Tahap Pra Proses

Sebelum ekstraksi fitur dilakukan maka terlebih dahulu citra tulisan di olah sedemikian rupa agar citra yang dihasilkan nantinya lebih bersih dan ringan serta minim *noise* tahapan ini bisa di saksikan dalam Gambar 1.4. Lalu pada citra hasil capture dari kamera akan di konversikan menjadi citra grayscale. Selanjutnya lanjut ke tahap binerisasi dengan metode yang digunakan adalah adaptive thresholding. ini dilakukan dengan tujuan untuk mengurangi noise dalam latar belakang gambar sebelum tahap ekstraksi karakter fitur.(Firdaus, 2020) Selanjutnya dalam tahap segmentasi dilakukan untuk bisa mendapat potongan dari karakter terpisah dari hasil tahap sebelumnya guna untuk diproses lebih lanjut. ada beberapa tahap dalam melakukan proses segementasi, pada tahap pertama deteksi karakter berdasar kontur, lalu menjumlah hasil potongan, kemudian mencari centroid dari blob selanjutnya melakukan penyortiran blob agar bisa menjaga posisi tetap sesuai dengan aslinya urutannya. (Apriyanti & Widodo, 2016).



Gambar 2. 4 Tahapan Pra Proses

2.6.2 Tahap Ekstraksi Fitur

Dalam tahap ekstraksi fitur ini bertujuan sebagai pengambilan dari fitur dalam citra hasil dari pencarian blob karakter di dalam tahap sebelumnya. Dari citra yang telah diproses pada tahap pra proses citra akan di ekstraksi berdasarkan fiturnya kemudian intesitasnya dan kontur yang didapatkan. Sebelum tahapan ini dilakukan ukuran dari blob karakter akan terlebih dahulu di normalkan sesuai dari dataset yang telah di hasilkan sehingga ukurannya harus sama lalu citra tersebut kemudian akan dibinerisasi sebagai inputan jaringan syaraf tiruan perambatan balik.(Apriyanti & Widodo, 2016).

2.6.3 Tahap Klasifikasi dan Pengenalan

Tahap klasifikasi dan pengenalan merupakan sebuah tahap pengidentifikasian yang mana dalam melakukan pengidentifikasian ini diperlukan metode salah satunya adalah jaringan syaraf tiruan perambatan balik dalam sistem *OCR multilayer model* meliputi 3 jenis layer, yang pertama adalah *layer input* yang kedua *hidden layer* dan yang terakhir adalah *layer output* yang semuanya memiliki sejumlah unit node. Rancangan ini bisa dilihat dalam Gambar 2.1.(Apriyanti & Widodo, 2016).

2.7 Google Mobile Vision

Google Mobile Vision merupakan library yang disediakan Google untuk menemukan sebuah objek. Google Mobile Vision menyediakan kerangka kerja untuk menemukan objek dalam foto maupun video. API Pengenalan Teks mengenali teks dalam bahasa berbasis Latin apa pun. Ini juga mewakili struktur teks yang dikenali, termasuk paragraf dan baris. Pengenalan Teks dapat mengotomatiskan entri data yang terlalu lama untuk kartu kredit, tanda terima, dan kartu nama, serta membantu mengatur foto, menerjemahkan dokumen, atau meningkatkan aksesibilitas. Aplikasi bahkan dapat melacak objek nyata, seperti membaca angka di kereta.(Google, 2021).