MAKALAH

METODE PENCARIAN STRING – RABIN KARP

MATA KULIAH PEMROGRAMAN BERORIENTASI OBJEK I

Disusun oleh:

Nama : Muhammad Fadillah Arsa (140810170005)
Dimas Satria Prakoso (140810170007)
Muhammad Afif (140810170045)
Muhammad Ariq Farhansyah Mutyara (140810170053)

Kelas : A – Teknik Informatika 2017

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2018

KATA PENGANTAR

Puji syukur penyusun panjatkan kehadirat Tuhan Yang Maha Esa atas karunia-Nya penyusun dapat menyelesaikan penyusunan makalah berjudul ‘Metode Pencarian String – Rabin Karp’ ini dengan baik. Penyusun juga mengucapkan terima kasih kepada semua pihak yang telah mendukung penyelesaian tugas ini.

Penyelesaian penyusunan makalah “Metode Pencarian String – Rabin Karp” ini secara khusus ditujukan memenuhi tugas ke-3 mata kuliah Pemrograman Berorientasi Objek I Prodi Teknik Informatika FMIPA Unpad 2018.

Penyusun berharap makalah ini dapat menambah pengetahuan dan wawasan mengenai salah satu metode yang dapat digunakan dalam proses pencarian string dalam bahasa pemrograman java. Penyusun pun menyadari bahwa dalam penyusunannya, makalah ini masih memiliki kekurangan, oleh sebab itu penyusun mengharapkan adanya kritik, saran dan tanggapan yang bersifat membangun demi perbaikan makalah ini di kemudian hari.

Semoga makalah ini dapat berguna dan dapat dipahami oleh seluruh kalangan yang membacanya. Mohon maaf apabila terdapat kesalahan kata-kata yang kurang berkenan.

Atas perhatian Saudara/Saudari, penyusun mengucapkan terima kasih.

Jatinangor, 2 Oktober 2018

Penyusun

DAFTAR ISI

MAKALAH

METODE PENCARIAN STRING – RABIN KARP

MATA KULIAH PEMROGRAMAN BERORIENTASI OBJEK I

A. ALGORITMA RABIN KARP

*Catatan tentang istilah yang nanti kita akan pakai:
String keseluruhan = Text String
Substring yang dicari = Pattern String

Algoritma Rabin-Karp dalam pencarian string, secara garis besar ialah mencari Pattern String pada Text String dengan menggunakan *Hahsing*. Proses perncarian akan di-*looping* sehingga Pattern akan dibandingkan/ di-*compare* dengan tiap substring pada Text string. Yang membedakan metode ini dengan metode pencarian string lainnya ialah pembandingnya, dimana yang dibandingkan disini bukanlah character melainkan *Hash Value* yang didapat dengan proses *Hashing*.

Hash Value adalah suatu nilai yang merepresentasikan substring tersebut melalui perhitungan matematika. Proses untuk mencari Hash Value dari suatu substring dinamakan Hashing. Hash Value ini yang nanti mempermudah kita untuk meng-compare dan membuat teknik Rabin-Karp ini akan lebih cepat dibandingkan dengan linear search yang pembandingnya hanya nilai individual character (nilai numerik character ASCII).

Yang pertama kita butuhkan dalam teknik Rabin-Karp ini selain Text dan Pattern nya adalah panjang dari pattern dan text nya itu sendiri. Kemudian kita mencari Hash Value dari pattern dan Hash Value dari subtring pertama text.

*Catatan: pattern akan dibandingkan dengan kesuluruhan substring pada Text yang panjangnya sama dengan panjang pattern.

Kemudian kita akan membuat suatu looping yang jumlah iterasinya merupakan jumlah substring seluruhnya dari Text. Dalam looping ini kita

membuat 2 kondisi. Kondisi pertama ialah ketika hash value pada substring bernilai sama dengan hash value pada pattern. Jika demikian, baru kira bandingan character pattern dengan character substring satu per satu untuk memastikan jika pattern sesuai dengan substring. Jika iya, maka kondisi 1 berakhir dan menge-print nilai dari indeks dari substring terhadap Text. Kondisi kedua ialah ketika hash value substring sebelumnya tidak sama dengan hash value pattern maka langkah yang kita lakukan selanjutnya ialah mencari hash value untuk substring selanjutnya. Hash Value ini yang nanti kita akan bandingkan kembali dengan hash Value pattern. Begitu seterunya hingga looping selesai.

Output yang ingin dicapai nanti ialah program dapat memberi tahu posisi-posisi indeks pada keseluruhan substring text yang bersesuaian dengan pattern. Tentunya dengan proses pencarian yang lebih cepat dan efisien.

Pencarian String metode Rabin-Karp ini dikenal juga dengan *Fingerprint Search* karena menggunakan jumlah informasil yang minim untuk merepresentasikan suatu pola (memiliki potensi jumlah informasi yang bersar). Algoritma ini cukup baik sebab fingerprint (hash value pattern) dapat dikomputasikan dan dibandingkan dengan lebih mudah dan efisien.

B. SEJARAH ALGORITMA RABIN KARP

Algoritma Rabin Karp adalah algoritma pencarian string yang diciptakan oleh Miachel O. Rabin dan Richard M. Karp pada tahun 1987 yang penggunaannya menggunakan metode hashing untuk mencari set pattern string pada suatu teks.

C. CONTOH PERHITUNGAN MANUAL

Text = AAAAAAAAAAAAAAAAAABC

Pattern = ABC

Misal nilai AAA=1

AAB=2

ABC=3

AAAAAAAAAAAAAAAAAABC

compare

(AAA) 1!=3(ABC)

AAAAAAAAAAAAAAAAAABC

compare

(AAA) 1!=3(ABC)

AAAAAAAAAAAAAAAAAABC

compare

(AAA) 1!=3(ABC)

compare

(AAB) 2!=3(ABC)

compare

(ABC) 3==3(ABC)

->nilai hash sama, compare tiap elemen satu satu

D. CONTOH PERHITUNGAN MANUAL DENGAN DATA ASLI

Text =	ASCABC
Pattern=	ABC

R=256

Q= 997

M=3

N=7

RM= R^{M-1} % Q=256^{3-1} % 997=731

Hash total = (R * Hash sebelumnya (hash karakter 1 sampai hash karakter 3) + indexASCII) % Q

i	0	1	2
	A/65	**B/66**	**C/67**
0	A%997	=65	
1	A	B%997	(65*256+66)%997
2	A	B	C%997 (754*256+67)%997

Nilai Hash “A ”=65

Nilai Hash”AB “=754

Nilai Hash”ABC“=670

i	0	1	2	3	4	5	6
	A/65	**S/83**	**C/67**	**A/65**	**B/66**	**C/67**	**D/68**
0	A%997		=65				
1	A	S%997	(65*256+83)%997			=771	
2	A	S	C%997	(771*256+67)%997		=37	
3		S	C	A%997	(256*(37 - 65 * 731) + (65)) % 997=124		
4			C	A	B%997	(256*(124 - 83 * 731) + (66) % 997=878	
5				A	B	C%997	(256*(878 - 67 * 731) + (67) %997= 670

Nilai Hash “A”=65

Nilai Hash”AS”=771

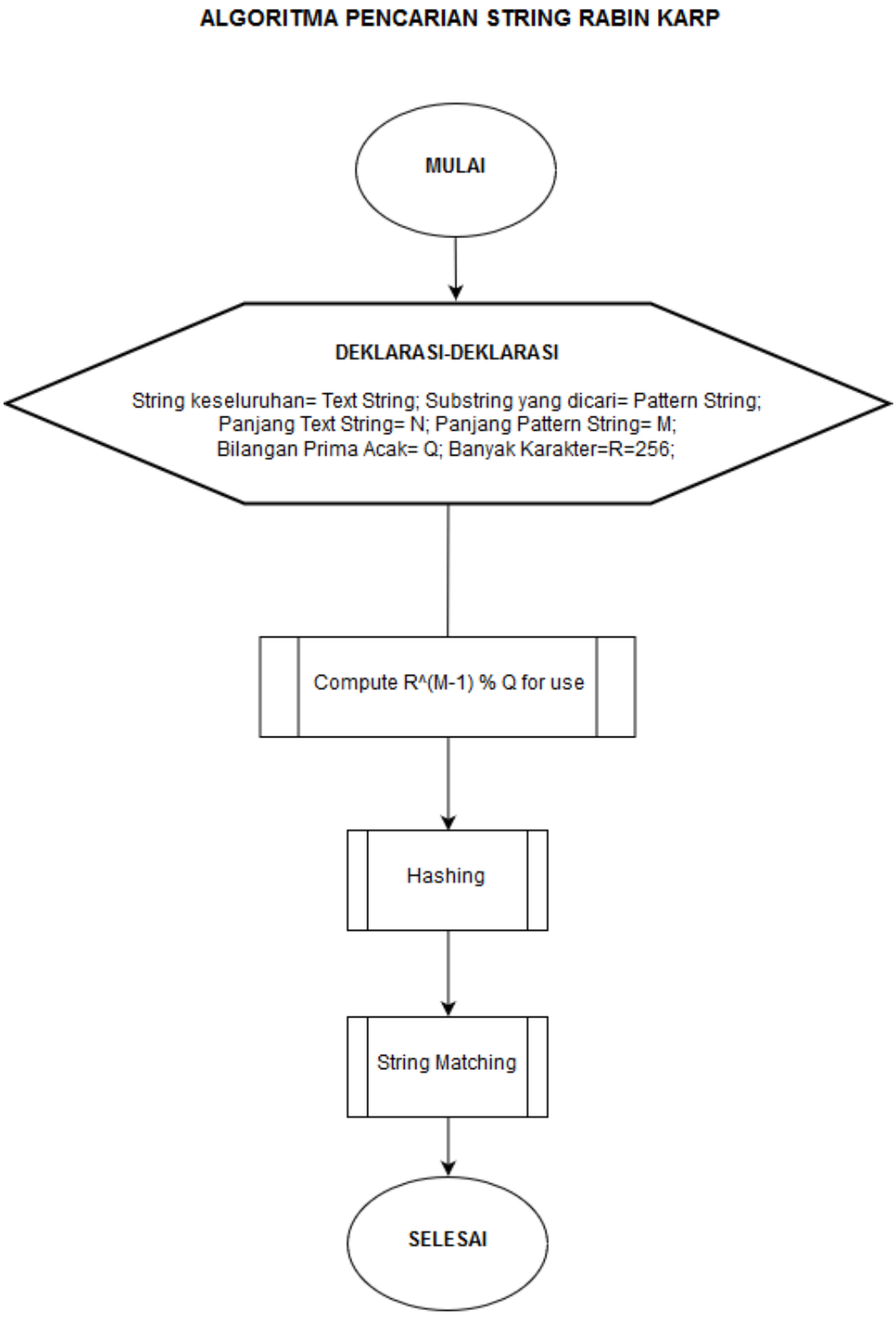
Nilai Hash”ASC”=37

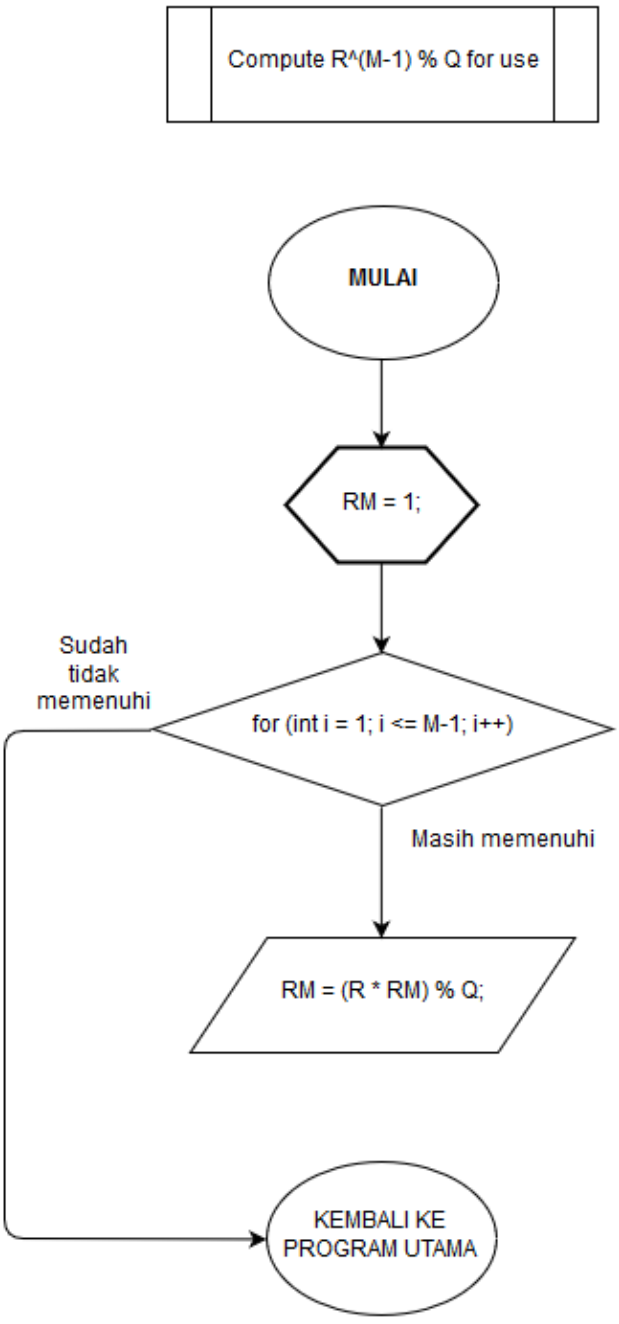
Nilai Hash”SCA”=124

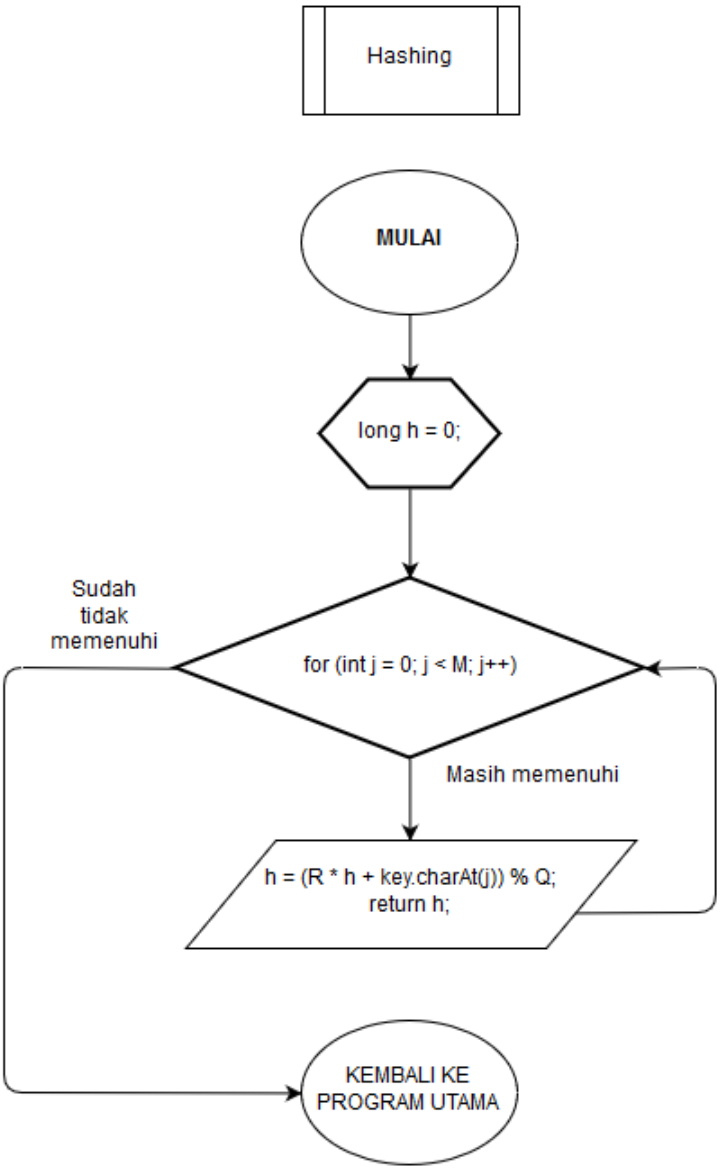
Nilai Hash “CAB”=878

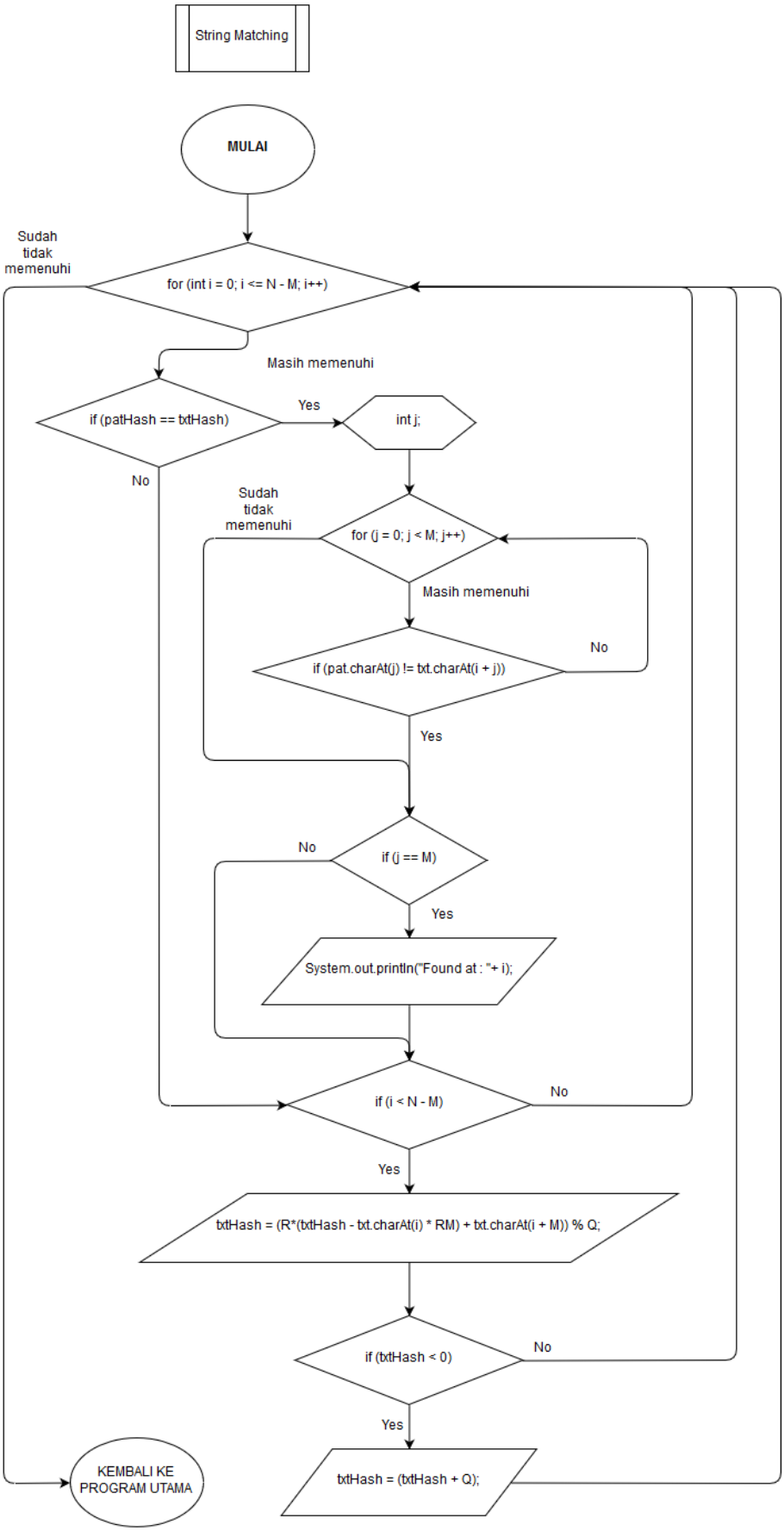
Nilai Hash “ABC”=670, Nilai Hash substring sama dengan pattern

E. FLOWCHART


ALGORITMA PENCARIAN STRING RABIN KARP
MULAI
DEKLARASI-DEKLARASI
String keseluruhan= Text String; Substring yang dicari= Pattern String;
Panjang Text String= N; Panjang Pattern String= M;
Bilangan Prima Acak= Q; Banyak Karakter=R=256;
Compute R^(M-1) % Q for use
Hashing
String Matching
SELESAI


Compute R^(M-1) % Q for use
MULAI
RM = 1;
Sudah tidak memenuhi
for (int i = 1; i <= M-1; i++)
Masih memenuhi
RM = (R * RM) % Q;
KEMBALI KE PROGRAM UTAMA


Hashing
MULAI
long h = 0;
Sudah
tidak
memenuhi
for (int j = 0; j < M; j++)
Masih memenuhi
h = (R * h + key.charAt(j)) % Q;
return h;
KEMBALI KE
PROGRAM UTAMA


String Matching
MULAI
Sudah tidak memenuhi
for (int i = 0; i <= N - M; i++)
Masih memenuhi
if (patHash == txtHash)
Yes
int j;
No
Sudah tidak memenuhi
for (j = 0; j < M; j++)
Masih memenuhi
if (pat.charAt(j) != txt.charAt(i + j))
No
Yes
No
if (j == M)
Yes
System.out.println("Found at : "+ i);
if (i < N - M)
No
Yes
txtHash = (R*(txtHash - txt.charAt(i) * RM) + txt.charAt(i + M)) % Q;
if (txtHash < 0)
No
Yes
txtHash = (txtHash + Q);
KEMBALI KE PROGRAM UTAMA

F. SOURCE CODE PROGRAM

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.util.Random;
import java.math.BigInteger;

public class RabinKarp {
    private String pat; // Pattern
    private long patHash; // Hash Value for Pattern
    private long txtHash; // Hash Value for Text
    private int M; // Pattern Lenght
    private int N;
    private long Q; // Large Prime Number
    private int R; // Number of Character from input
    private long RM;

    public RabinKarp(String txt, String pat) {
        this.pat = pat;
        R = 256;
        M = pat.length();
        N = txt.length();
        Q = longRandomPrime();
        RM = 1;
        for (int i = 1; i <= M-1; i++)
            RM = (R * RM) % Q;
        patHash = hash(pat, M);
        txtHash = hash(txt, M);
        search(txt);
    }
    // Hash Function
    private long hash(String key, int M) {
        long h = 0;
        for (int j = 0; j < M; j++)
            h = (R * h + key.charAt(j)) % Q;
        return h;
    }

```

```
    private void search(String txt) {
        for (int i = 0; i <= N - M; i++) {
            if (patHash == txtHash) {
                int j;
                for (j = 0; j < M; j++) {
                    if (pat.charAt(j) != txt.charAt(i + j))
                        break;
                }

                if (j == M)
                    System.out.println("Found at : "+ i);
            }

            if (i < N - M) {
                txtHash = (R*(txtHash - txt.charAt(i) * RM) + txt.charAt(i + M)) % Q;

                if (txtHash < 0)
                    txtHash = (txtHash + Q);
            }
        }
    }

    private static long longRandomPrime() {
        BigInteger prime = BigInteger.probablePrime(31, new Random());
        return prime.longValue();
    }

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Rabin Karp Algorithm\n");
        System.out.print("Enter Text : ");
        String text = br.readLine();
        System.out.print("Enter Pattern : ");
        String pattern = br.readLine();
        System.out.println("Results : \n");
        RabinKarp rk = new RabinKarp(text, pattern);
    }
}
```

//PSEUDO CODE

```
function RabinKarpSet(string s[1..n], set of string subs, m):
    set hsubs := emptySet
    foreach sub in subs
        insert hash(sub[1..m]) into hsubs
    hs := hash(s[1..m])
    for i from 1 to n-m+1
        if hs ∈ hsubs and s[i..i+m-1] ∈ subs
            return i
        hs := hash(s[i+1..i+m])
    return not found
```

G. SCREENSHOT EKSEKUSI PROGRAM

```
C:\Users\M Fadillah Arsa\Downloads>javac RabinKarp.java

C:\Users\M Fadillah Arsa\Downloads>java RabinKarp
Rabin Karp Algorithm

Enter Text : Bahasa Pemrograman Java
Enter Pattern : Java
Results :

Found at : 19
```

```
C:\Users\M Fadillah Arsa\Downloads>javac RabinKarp.java

C:\Users\M Fadillah Arsa\Downloads>java RabinKarp
Rabin Karp Algorithm

Enter Text : buku itu ternyata ada di bawah rak buku
Enter Pattern : buku
Results :

Found at : 0
Found at : 35
```

H. KOMPLEKSITAS WAKTU ALGORITMA PENCOCOKAN RABIN KARP

Pertama-tama, apa yang pra-pemrosesan dan apa runtime online, tergantung pada apa yang tetap konstan dan apa yang bervariasi. Sebagai contoh, jika kita diberi beberapa teks tetap dan diminta untuk mencocokkan berbagai string kecil dengan teks itu, pra-pemrosesan ialah akan melakukan *Hashing* text.

Dalam hal ini, mari kita katakan kita hanya memiliki satu string untuk dibandingkan dengan satu teks, untuk menghindari kebingungan tentang pra-pemrosesan. Operasi yang perlu kami lakukan adalah

1. Hash untuk string yang akan dicari (p) = O (m)
2. Hash untuk setiap substring berukuran m dalam teks (T) (dengan asumsi hash bergulir) = O (n − m + 1)
3. Jumlah perbandingan hash (satu per setiap substring berukuran m dalam T = O (n − m + 1)
4. Jika ada r cocok dalam hash, maka kita membandingkan masing-masing substring T dengan p (setiap perbandingan menjadi O (m)) = O (rm)

Namun dalam kasus terburuk, jumlah kecocokan dalam hash dapat sebesar n − m + 1. Jadi kompleksitas kasus terburuk adalah O (m (n − m + 1)).

REFERENSI

Addanki, Rajesh. (2011). *Karp-Rabin*. USA: Indiana State University. Diakses dari http://cs.indstate.edu/~raddanki/abstract.pdf

Sedgewick, Robert. (2011). *Algorithms – Fourth Edition*. USA: Princeton University.

Noprisson, Handrie. (2013). *Implementasi Algoritma Rabin-Karp untuk Menentukan Keterkaitan Antara Publikasi Penelitian Dosen Tahun 2013*. Indonesia: Universitas Bengkulu. Diakses dari https://www.academia.edu/5977583/Implementasi_Algoritma_Rabin-Karp_-_Paper_-_UNIB_Rev_1

Stack Excange – Computer Science. *Time Complexity of Rabin-Karp matching algorithm*. Diakses pada 2 Oktober 2018 dari https://cs.stackexchange.com/questions/10258/time-complexity-of-rabin-karp-matching-algorithm