Danyl Fernandes
2020012004 (72)
26-04-2021

AOA Experiment 10

Aim:

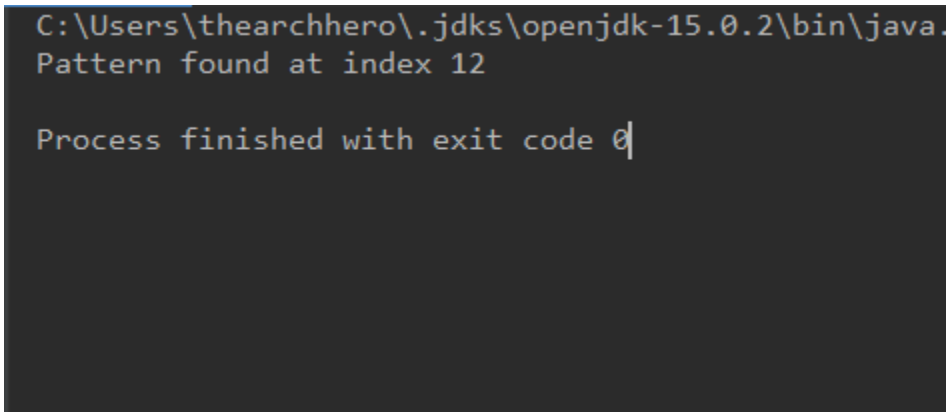
To implement & analyze Rabin-Karp Algorithm:

Implementation:

```
public class RabinKarp {
	public final static int d = 256;

	static void search(String pat, String txt, int q) {
	int M = pat.length();
	int N = txt.length();
	int i, j;
	int p = 0;
	int t = 0;
	int h = 1;

	for (i = 0; i < M - 1; i++)
		h = (h * d) % q;

	for (i = 0; i < M; i++) {
		p = (d * p + pat.charAt(i)) % q;
		t = (d * t + txt.charAt(i)) % q;
	}

	for (i = 0; i <= N - M; i++) {
		if (p == t) {
			for (j = 0; j < M; j++) {
```

```
                if (txt.charAt(i + j) != pat.charAt(j))
                     break;
                }

                if (j == M)
                System.out.println("Pattern found at index " + i);
           }

           if (i < N - M) {
                t = (d * (t - txt.charAt(i) * h) + txt.charAt(i + M)) % q;

                if (t < 0)
                t = (t + q);
           }
     }
     }

     public static void main(String[] args) {
     String txt = "DANYL LOVES LINUX";
     String pat = "LINUX";

     int q = 13;

     search(pat, txt, q);
     }
}
```

Output:


C:\Users\thearchhero\.jdks\openjdk-15.0.2\bin\java.
Pattern found at index 12
Process finished with exit code 0

Daryl Fernandes
2020012004 (72)

Exp 10

Theory:

- Rabin & Karp proposed a string-matching algorithm that performs well in practice & that also generalizes to other algorithms for related problems such as two-dimensional pattern matching.

- This algorithm makes use of elementary number-theoritic notions ~~want to refer~~ such as to the equivalence of two numbers modulo a third number.

- Like the Naive Algorithm, Rabin-Karp Algorithm also slides the pattern one by one. But unlike the naive approach, Rabin-Karp Algo. matches the hash values of the patterns with the hash values of current substrings of the text.

- If there is a match found, only then it begins matching individual characters. As such, Rabin-Karp algo. needs to calculate hash values for following strings
  1) Pattern itself
  2) All the substrings of the text

Daryl Fernandes
20200120004(72)

Algorithm :

```
Rabin_Karp_Matcher (T, P, d, q)
    n = T.length
    m = P.length
    h = d^(m-1) mod q
    p = 0
    t0 = 0
    for i = 1 to m
        p = (dp + P[i]) mod q
        t0 = (dt0 + T[i]) mod q
    for s = 0 to n-m
        if p == ts
            if P[1...m] == T[s+1...s+m]
                print "Pattern occurs with shift" s
        if s < n-m
            ts+1 = (d(ts - T[s+1]h) + T[s+m+1]) mod q
```

Analysis :

- The average & best case running time of the Rabin-Karp algorithm is $O(n+m)$ but its worst-time is $O(nm)$. Worst case of Rabin-Karp algorithm occurs when all characters of pattern & text are same as the hash values of all the substrings of txt[] match with hash value of patt[]. For example patt[] = "AAA" & txt[] = "AAAAAAA".

Danyl Fernandes
2020012004 (72)

Assume

a = 0
b = 1
c = 2
d = 3

Example :

(m=3) Pattern : a a c

Hash : 1 + 1 + 2 = 4

(n=5) TEXT : a a a a c

I1 1 + 1 + 1 = 3

I2 1 + 1 + 1 = 3

I3 1 + 1 + 2 = (4)

↑ Pattern found

Conclusion :

We successfully implemented & analyzed Rabin-Karp Algorithm.