http://ejournal.seaninstitute.or.id/index.php/InfoSains
Jurnal Info Sains : Informatikan dan Sains, Volume 10, No 01 Maret 2020
E-ISSN.2797-7889, P-ISSN.2089-3329

Brute Force Algorithm Implementation Of Dictionary Search

[1]Charles Barutu, [2]Naufal abdi
[1,2] Teknik Informatika, STMIK Pontianak, Kalimantan
Email: charlesbarutu12@gmail.com[1], naufalabdi67@gmail.com[2]

Keywords

Dictionary
Algoritma Brute Force
Searching

Abstract. In the manual dictionary the more words are accommodated, the heavier the dictionary. Dictionaries that are generally book-shaped are difficult to carry anywhere because they are thick and heavy. With this mobile-based computer dictionary application, users no longer need to carry heavy dictionaries and the search process will also be faster and easier. This mobile-based kmputer dictionary application is created using the java programming language with the NetBeans 7.0 editor and using an RMS (Record Management System) based database. So it can be installed on various types of mobile phones that already support Java. String Matching is one of the algorithms used to speed up the search process for the desired word. String matching algorithms have been used frequently before as examples in the process of matching strings based on the equation of text data namely Brute Force. Because this Brute Force algorithm can be used to perform string or text searches. Brute force algorithm is an algorithm to match a pattern with all text between 0 and n-m to find the presence of a pattern in text.

1. INTRODUCTION

With the development of computer technology, the author was inspired to develop a dictionary application of computer terms that can be used as a quick and precise learning tool, especially in the search for terms in the computer world. Dictionary applications that are basically only used manually can be redirected with a digital dictionary application that can be used on mobiledevices. So with this mobile-based computer dictionary application, users no longer need to carry heavy dictionaries and the search process will also be faster and easier. This mobile-based computer dictionary application was created using the java programming language with the NetBeans 7.0 editor and using an RMS-based database (Record Management System). So that this mobile-based computer dictionary application can be installed on various types of mobile phones that already support java[1].

String matching is an important part of a string searchprocess ina document. The result of searching for a string in a document depends on the technique or way of matching the string used. To know the correct contents of the document according to the needs of information, a method of searching strings (string searching) the contents of a gooddocument [2] . String Matching is one of the algorithms used to speed up the search process for the desired word. String matching is divided into two, namely exact matching and heuristic or statistical matching [3]. String matching algorithms have been used frequently before such as the example in the process of matching strings based on the equation of text data i.e. Brute Force. In this case, the brute force algorithm is selected because this algorithm can be used to perform string or text searches [4], [5]. Brute force algorithm is an algorithm to match a pattern with all text between 0 and n-m to find the presence of a pattern in text. Based on the direction of the search, this algorithm is classified as an algorithm that reads strings from left to right .

2. METHOD

The methods used by the author in writing, data collection and information are required, namely:

1. Literature studies.
2. That is to learn from books related to computer terms. As well as looking for journals related to the Brute Force algorithm and its application.
3. Learn about Brute Force algorithms and their application
4. Design a mobile-based computer dictionary application program using the Java programming language with the Netbeans 7.0 editor.
5. Testing the program and its implementation
6. Draw conclusions and write the report in the form of a skrpsi.

2.1 String Matching

String Matching is the process of searching for all occurrences of a query hereinafter referred to as a pattern into a longer string or String Matching text formulated as follows:

x = x[0…m-1] (2.1)
y = y[0…n-1] (2.2)
where:
x is pattern
m is the length of the pattern
y is text
n is the length of the text

Both strings consist of a set of characters called alphabets denoted by ∑ (sigma) and have a size of σ (tao). String matching is divided into two, namely exact matching and heuristic or statistical matching [6].

2.2. Brute Force Algorithm

Brute force algorithm is an algorithm to match a pattern with all text between 0 and n-m to find the presence of a pattern in text. Inside the matching string,there are text terms and patterns. Text is a word that is searched for and matched with a pattern. While pattern is a word that is inputted to match In detail, the steps that this algorithm performs when matching strings are [7]:

1. The brute force algorithm starts matching patterns from the beginning of the text.
2. From left to right, this algorithm will match the character per character pattern to the corresponding text, until one of the following conditions is met:
   a) Characters in the pattern and in the compared text do not match.
   b) All characters in the pattern match. Then the algorithm will tell you the discovery in this position.
3. The algorithm then continues to shift the pattern one to the right, and repeats step 2 until the pattern is at the end of the text [8].

Brute force algorithms also have advantages and disadvantages. The advantages of brute force algorithm are:

1. Brute force algorithms can be used to solve most problems.
2. Brute force algorithm is simple and easy to understand
3. Brute force algorithms generate decent algorithms for some important issues such as searching, sorting, string matching, ormatrix multiplication.
4. Brute force algorithm generates algorithmsbaku(standart)for computational tasks of summing/multiplication n numbers, specifying minimum or maximum elements in the table(list) [9]–[11].

While the disadvantages of brute force algorithm are as follows:

a) Brute force algorithms rarely produce algorithms that are mangkus (effective).
b) Some brute force algorithms are slow, so it's unacceptable.
c) Not as constructive/creative as other problem solving techniques.

3. RESULTS AND DISCUSSION

Use brute force algorithm for pattern search in text:

Text = access backup case
Pattern = case
Then the settlement is as follows :

Step – 1																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern	c	a	s	e														
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 1 found a mismatch between the pattern and the text, slide the pattern as much as one Step right towards the next index,

Step – 2																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern		c	a	s	e													
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 2 found a match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 3																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern		c	a	s	e													
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 3 also found a mismatch between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 4																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern			c	a	s	e												
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 4 found a match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 5

Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern			c	a	s	e												

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 5 found a mismatch between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 6																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern				c	a	s	e											

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 6 also found a mismatch between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 7																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern					c	a	s	e										

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 7 also found a mismatch between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 8																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern						c	a	s	e									

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 8 also found a mismatch between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 9																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern							c	a	s	e								

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 9 also found a mismatch between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 10																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern								c	a	s	e							

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern									c	a	s	e						

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14
					15	16	17	18						

In Step - 11 also found a mismatch between the pattern and the text, slide the pattern as much as one Step right towards the next index,

Step – 12																	
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern										c	a	s	e					

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14
					15	16	17	18						

In Step - 12 found a match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 13																	
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern										c	a	s	e					

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						16	17	18							

In Step - 13 there is no match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 14																	
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern											c	a	s	e				

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14
					15	16	17	18						

In Step - 14 there is no match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 15																	
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern												c	a	s	e			

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14
					15	16	17	18						

In Step - 15 there is no match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 16																	
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern													c	a	s	e		

Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14
					15	16	17	18						

In Step - 16 there is no match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step – 17																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern														c	a	s	e	
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 17 there is no match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step - 18																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern															c	a	s	e
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 18 found a match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step - 19																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern															c	a	s	e
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 19 found again the match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step - 20																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern															c	a	s	e
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In Step - 20 found again the match between the pattern and the text, slide the pattern as much as one Step right to the next index,

Step - 21																		
Text	a	c	c	e	s	s		b	a	c	k	u	p		c	a	s	e
Pattern															c	a	s	e
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

In this step, you find a match between each character in the pattern and the character in the text. Then the search is complete because the pattern has been found, it has reached the end of the text and the search stops at the index to – 18. Then the algorithm displays the data.

4. CONCLUSION

Matching text using Brute Force algorithm will place and search all possible words searched by tryingeach pattern position against the text, then the process of matching each

character and text at that position. The application of Gross Force algorithm can perform string matching and provide the desired result in word search in Java Mobile Based Computer Dictionary Application.

REFERENCE

[1] J. Agape Sianturi, I. N. Piarsa, and I. K. Adi Purnawan, "Aplikasi Pencarian dan Penyewaan Rumah Kost Berbasis Web dan Android," *J. Ilm. Merpati (Menara Penelit. Akad. Teknol. Informasi)*, 2018, doi: 10.24843/jim.2018.v06.i03.p06.

[2] N. Afif, "IMPLEMENTASI ALGORITMA BRUTE FORCE DALAM PERANCANGAN APLIKASI PENELUSURAN SKRIPSI," *Inform. Sains dan Teknol.*, vol. 3, 2018.

[3] C. R. Gunawan, A. Ihsan, and M. Munawir, "Optimasi Penyelesaian Permainan Rubik's Cube Menggunakan Algoritma IDA* dan Brute Force," *J. Infomedia*, vol. 3, no. 1, 2018, doi: 10.30811/jim.v3i1.627.

[4] S. S, "Implementasi Algoritma Brute Force Dalam Pencarian Kebudayaan Di Indonesia Berbasis Mobile Application," *Buffer Inform.*, vol. 4, no. 2, 2018, doi: 10.25134/buffer.v4i2.1472.

[5] A. S. Sumi, P. Purnawansyah, and L. Syafie, "Analisa Penerapan Algoritma Brute Force Dalam Pencocokan String," *Pros. SAKTI (Seminar Ilmu Komput. dan Teknol. Informasi)*, vol. 3, no. 2, 2018.

[6] R. Manivannan and S. K. Srivatsa, "Semi automatic method for string matching," *Inf. Technol. J.*, vol. 10, no. 1, 2011, doi: 10.3923/itj.2011.195.200.

[7] T. Zebua and N. Silalahi, "Aplikasi Saran Buku Bacaan Bagi Pengunjung Perpustakaan AMIK STIEKOM Sumatera Utara Berdasarkan Algoritma Brute Force," *Jurasik (Jurnal Ris. Sist. Inf. dan Tek. Inform.*, vol. 3, 2018, doi: 10.30645/jurasik.v3i0.67.

[8] Y. Yulianto, R. Ramadiani, and A. H. Kridalaksana, "Penerapan Formula Haversine Pada Sistem Informasi Geografis Pencarian Jarak Terdekat Lokasi Lapangan Futsal," *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 13, no. 1, 2018, doi: 10.30872/jim.v13i1.1027.

[9] A. Rasool, A. Tiwari, G. Singla, and N. Khare, "String Matching Methodologies: A Comparative Analysis," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 2, 2012.

[10] D. W. T. PUTRA and J. J. PUTRA, "PERANCANGAN SISTEM INFORMASI PENCARIAN LOWONGAN PEKERJAAN," *J. TEKNOIF*, vol. 6, no. 1, 2018, doi: 10.21063/jtif.2018.v6.1.48-54.

[11] Y. Rohmiyati, "Model Perilaku Pencarian Informasi Generasi Milenial," *Anuva*, vol. 2, no. 4, 2018, doi: 10.14710/anuva.2.4.387-392.

[12] D. Ratnasari, D. B. Qur'ani, and A. Apriani, "Sistem Informasi Pencarian Tempat Kos Berbasis Android," *J. Inf.*, vol. 3, no. 1, 2018, doi: 10.25139/ojsinf.v3i1.657.