Pemanfaatan Algoritma Pencocokan String pada Citra Digital untuk Mengidentifikasi Jenis Batuan Beku

Nurul Utami Amaliah. W
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13517132@std.stei.itb.ac.id

***Abstrak*—Algoritma pencocokan string adalah algoritma yang umum digunakan untuk menyelesaikan berbagai macam perrmasalahan sehari-hari. Salah satu contoh permasalahan yang dapat diselesaikan dengan pencocokan string adalah identifikasi jenis batuan beku. Dari hasil pengujian, algoritma KMP maupun algoritma BM mampu melakukan identifikasi jenis batuan beku.**

Kata Kunci*—string matching*; citra digital; batuan beku; algoritma KMP; algoritma BM

I. PENDAHULUAN

Algoritma pencocokan string adalah algoritma yang umum digunakan untuk menyelesaikan berbagai macam permasalahan sehari-hari. Algoritma pencocokan string sendiri teridiri dari banyak algoritma, contohnya algoritma Knuth-Morris-Pratt atau KMP dan algoritma Boyer-Moore atau BM.

Salah satu contoh permasalahan yang menurut penulis dapat diselesaikan menggunakan penerapan strategi algoritma pencocokan string adalah untuk mengidentifikasi jenis batuan beku yang kerap dibutuhkan oleh orang-orang yang berkecimpung dalam bidang teknik geologi.

Salah satu faktor yang dapat digunakan untuk mengidentifikasi jenis batuan beku adalah indeks warna. Indeks warna di sini dapat diproyeksikan dari bentuk gambar yang kemudian dikonversi menjadi bentuk citra digital yang dipolakan. Hal ini dapat diidentifikasi dengan strategi algoritma pencocokan string. Pada penelitian ini akan dibahas pemanfaatan algoritma pencocokan string pada citra digital untuk mengidentifikasi jenis batuan beku.

II. LANDASAN TEORI

A. Algoritma Pencocokan String

Algoritma pencocokan string (*string matching algorithm*) atau dikenal juga sebagai algoritma pencarian string merupakan algoritma yang digunakan untuk melakukan pencarian suatu string pattern pada suatu string yang lebih panjang yang disebut teks.

Konsep string yang berlaku pada algoritma ini yaitu,

misalkan S adalah suatu string dengan panjang m, S = $X_0X_1…X_{m-1}$. Prefix dari S adalah substring S[0…k], sedangkan suffix dari S adalah substring S[k…m-1], dengan k adalah indeks diantara 0 sampai m-1.

Contoh S = “makan”, maka himpunan seluruh prefix yang mungkin dari S adalah [“m”, “ma”, “mak”, “maka”, “makan”] sedangkan himpunan seluruh suffix yang mungkin dari S adalah [“n”, “an”, “kan”, “akan”, “makan”].

B. Algoritma KMP

KMP merupakan singkatan dari Knuth-Morris-Pratt. Algoritma ini termasuk dalam salah satu algoritma pencocokan string yang 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.

Prinsip algoritma ini mirip dengan algoritma brute force untuk pencarian string namun dengan optimasi pada besar pergeseran yang dilakukan. Pergeseran yang dilakukan lebih besar dan hal ini diharapkan akan menghemat perbandingan yang selanjutnya akan meningkatkan kecepatan pencarian.

Perhitungan penggeseran pada algoritme ini adalah sebagai berikut, bila terjadi ketidakcocokkan 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 v dari pattern akan sama dengan sebagian akhiran u dari sebagian teks. Sehingga kita bisa menggeser pattern agar awalan v tersebut sejajar dengan akhiran dari u.

Dengan kata lain, pencocokkan string akan berjalan secara efisien bila kita mempunyai tabel yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokkan 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] relatif terhadap teks.

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

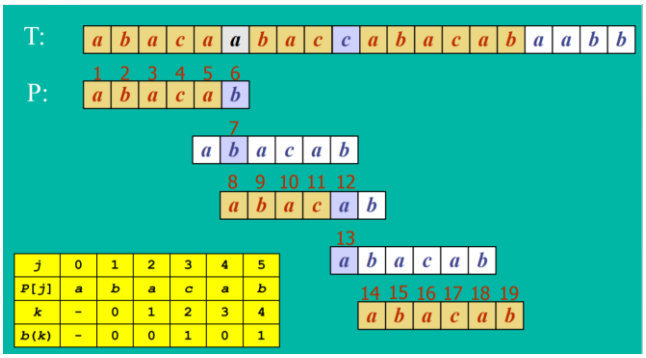
1. Algoritme Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.

2. Dari kiri ke kanan, algoritme ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:

   a. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).

   b. Semua karakter di pattern cocok. Kemudian algoritme akan memberitahukan penemuan di posisi ini.

3. Algoritme kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.



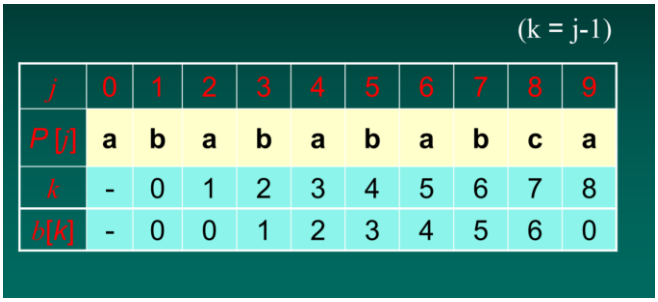
Gambar 1. Contoh Cara Kerja Algoritma KMP

Algoritma KMP membutuhkan *border function* atau fungsi pinggiran. Fungsi ini terdefinisi sebagai ukuran prefix terpanjang dari pattern P [0 … k] yang juga suffix [1 … k]

Contoh :

P = abababababca

j = 0123456789



Gambar 2. Contoh *Border Function*

Kompleksitas waktu algoritma KMP adalah O (m + n), diperoleh dari kompleksitas waktu menghitung fungsi pinggiran yaitu O (m), dan kompleksitas waktu pencarian string yaitu O (n).

Keuntungan penggunaan algoritma KMP adalah tidak perlu dilakukan backward pada teks, membuat algoritma ini sangat baik untuk digunakan untuk memproses file yang ukurannya sangat besar, khususnya yang dibaca dari perangkat eksternal maupun melalui *network stream*.

Kekurangan algoritma KMP yaitu kinerjanya semakin kurang optimal seiring meningkatnya ukuran huruf. Hal ini karena kemungkinan terjadinya *mismatch* juga meningkat. Tapi, KMP tetap optimal jika *mismatch* ditemukan di akhir proses pencarian.

C. Algoritma BM

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977.

Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritma ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Prinsip algoritma ini didasarkan pada dua teknik, yaitu:

1. *Teknik looking-glass*

   yaitu mencari pattern pada teks dengan bergeser secara backward sepanjang P, dimulai dari belakang.

2. Teknik character-jump

   yaitu teknik ketika terjadi mismatch, karakter pada P [j] tidak sama dengan karakter pada T [i].

Misalnya ada sebuah usaha pencocokan yang terjadi pada teks [i … i + n - 1], dan anggap ketidakcocokan pertama terjadi di antara teks [i + j] dan pattern [j], dengan 0 < j < n. Berarti, teks [i + j + 1 … i + n - 1] = pattern [j + 1 … n - 1] dan a = teks [i + j] tidak sama dengan b = pattern [j]. Jika u adalah akhiran dari pattern sebelum b dan v adalah sebuah awalan dari pattern, maka penggeseran-penggeseran yang mungkin adalah:

1. Penggeseran *good-suffix* yang terdiri dari menyejajarkan potongan teks [i + j + 1 … i + n - 1] = pattern [j + 1 … n - 1] dengan kemunculannya paling kanan di pattern yang didahului oleh karakter yang berbeda dengan pattern [j]. Jika tidak ada potongan seperti itu, maka algoritma akan menyejajarkan akhiran v dari teks [i + j + 1 … i + n - 1] dengan awalan dari pattern yang sama.

2. Penggeseran *bad-character* yang terdiri dari menyejajarkan teks [i + j] dengan kemunculan paling kanan karakter tersebut di pattern. Bila karakter tersebut tidak ada di pattern, maka pattern akan disejajarkan dengan teks [i + n + 1].

Secara sistematis, langkah-langkah yang dilakukan algoritme Boyer-Moore pada saat mencocokkan string adalah:

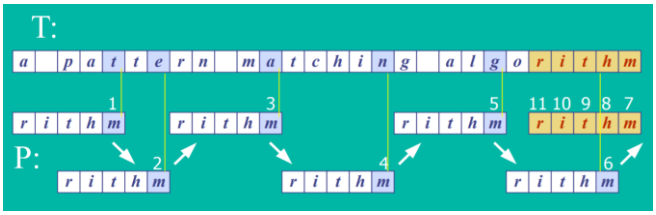
1. Algoritme Boyer-Moore mulai mencocokkan pattern pada awal teks.

2. Dari kanan ke kiri, algoritme ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:

   a. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).

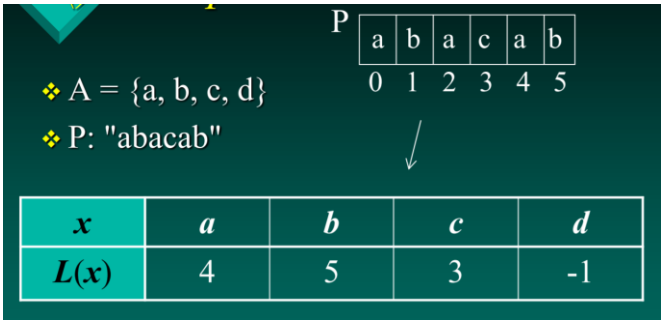
b. Semua karakter di pattern cocok. Kemudian algoritme akan memberitahukan penemuan di posisi ini.

3. Algoritma kemudian menggeser pattern dengan memaksimalkan nilai penggeseran *good-suffix* dan penggeseran *bad-character*, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.



Gambar 3. Contoh Cara Kerja Algoritma BM

Algoritma Boyer Moore membutuhkan last occurrence function yang memperoses lebih awal pattern P dan alphabet A untuk mencatat kemunculan terakhir L(). L() memetakan memetakan semua huruf di A menjadi integer. L(x) didefinisikan sebagai indeks i terbesar dengan P [i] = x, atau -1 jika indeks tersebut tidak ada.



Gambar 4. Contoh Last Occurrence Function

Kompleksitas waktu algoritma Boyer-Moore pada kasus terburuk O (nm + A). Tapi algoritma Boyer-Moore akan lebih cepat di ukuran alphabet (A) yang besar, dan lambat ketika alphabet nya kecil.

D. Citra Digital

Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses sampling.

Citra digital merupakan representatif dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantisasi. Sampling menyatakan besarnya kotak-kotak yang disusun dalam baris dan kolom. Dengan kata lain, sampling pada citra menyatakan besar kecilnya ukuran pixel (titik) pada citra, dan kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (grayscale) sesuai dengan jurnlah bit biner yang digunakan oleh mesin, dengan kata lain kuantisasi pada citra menyatakan jumlah warna yang ada pada citra. (Basuki, 2005:4).

Beberapa jenis citra digital yang sering digunakan antara lain:

1. Citra biner (monokrom)
2. Citra grayscale (skala keabuan)
3. Citra warna (true color)

1	1	1	1	1	1	1
1	0	0	0	0	0	1
1	1	1	0	1	1	1
1	1	1	0	1	1	1
1	1	1	0	1	1	1
1	0	0	0	0	0	1
1	1	1	1	1	1	1

Gambar 5.Contoh Citra Biner

E. Batuan Beku

Batuan beku atau batuan igneus (dari Bahasa Latin: ignis, "api") adalah jenis batuan yang terbentuk dari magma yang mendingin dan mengeras, dengan atau tanpa proses kristalisasi, baik di bawah permukaan sebagai batuan intrusif (plutonik) maupun di atas permukaan sebagai batuan ekstrusif (vulkanik). Magma ini dapat berasal dari batuan setengah cair ataupun batuan yang sudah ada, baik di mantel ataupun kerak bumi. Umumnya, proses pelelehan terjadi oleh salah satu dari proses-proses berikut: kenaikan temperatur, penurunan tekanan, atau perubahan komposisi. Lebih dari 700 tipe batuan beku telah berhasil dideskripsikan, sebagian besar terbentuk di bawah permukaan kerak bumi.

Klasifikasi pada batuan beku dapat dilakukan terhadap aspek yang berbeda-beda. Dalam makalah ini hanya dijelaskan lebih lanjut mengenai klasifikasi berdasarakan indeks warnanya.

Klasifikasi berdasarkan indeks warna

Menurut (S.J. Shand, 1943), yaitu:

1. Batuan leukokratik, apabila mengandung kurang dari 30% mineral mafik.
2. Batuan mesokratik, apabila mengandung 30% - 60% mineral mafik.
3. Batuan melanokratik, apabila mengandung lebih dari 60% mineral mafik.

Sedangkan menurut S.J. Ellis (1948) juga membagi batuan beku berdasarkan indeks warnanya sebagai berikut:

1. Holofelsik, untuk batuan beku dengan indeks warna kurang dari 10%.
2. Felsik, untuk batuan beku dengan indeks warna 10% sampai 40%.
3. Mafelsik, untuk batuan beku dengan indeks warna 40% sampai 70%.
4. Mafik, untuk batuan beku dengan indeks warna lebih dari 70%.

III. Implementasi

Dalam implementasi terfokus pada string matching pada "teks" berupa pola dari citra biner. Konversi dari gambar menjadi bentuk citra biner tidak akan dibahas lebih lanjut. Algoritma yang digunakan dalam implementasi adalah algoritma KMP dan algoritma BM dengan pertimbangan meruapakan algoritma yang paling optimal dalam melakukan pencarian string dan dapat menghemat waktu pencarian.

Implementasi algoritma akan dilakukan dalam bahasa python dengan pertimbangan python memiliki syntax yang cenderung ramah untuk dibaca dan dipahami.

Daftar jenis batuan beku dan acuan pola citra digital yang disimpan dalam file txt akan dibaca dan dicatat dalam list dictionary yang terdiri dari key yaitu pola citra digitalnya dan val yaitu nama jenis batuan beku.

Rancangan program akan menerima pola citra digital yang akan diidentifikasi. Selanjutnya program akan memberi keluaran sesuai hasil pencocokan string pada pola masukan dan pola pada dictionary. Jika terdapat pola yang cocok akan mengeluarkan nama jenis batuan tersebut . Sedangkan jika terdapat pada dictionary, akan mengeluarkan pesan kesalahan.

A. Algoritma KMP

```
1.     dictionary =[]
2.     answerlist =[]
3.
4.     def kmpMatch(pattern, text):
5.         global n_kmp
6.         n_kmp = 0
7.         n = len(text)
8.         m = len(pattern)
9.         fail = computeFail(pattern)
10.        i = 0
11.        j = 0
12.        while i < n and j < m:
13.            if pattern[j] == text[i]:
14.                i+=1
15.                j+=1
16.                n_kmp +=1
17.            elif j > 0:
18.                j = fail[j - 1]
19.            else:
20.                i+=1
21.        return n_kmp
22.
23.    def computeFail(pattern):
24.        fail = [len(pattern)]
25.        m = len(pattern)
26.        fail[0] = 0
27.        i = 1
28.        j = 0
29.        while i < m:
30.            if pattern[j] == pattern[i]:
31.                fail.append(j + 1)
32.                i+=1
33.                j+=1
34.            elif j > 0:
35.                j = fail[j - 1]
36.            else:
37.                fail.append(0)
38.                i+=1
39.        return fail
40.
41.    def readfile(file):
42.        with open(file) as f:
43.            for line in f:
44.                (key, val) = line.split('\n ')
45.                dictionary.append([key,val])
46.
47.    readfile("jenisbatu.txt")
48.    question = line.split('\n ')
49.    for idx, key in enumerate(dictionary):
50.
   kmpmatching = kmpMatch(question, dictionary)
51.        try:
52.
     if (kmpmatching/len(dictionary > 0.75):
53.
   answerlist.append([key[0],key[1]])
54.        except ZeroDivisionError:
55.            pass
56.
57.    if (len(answerlist) >= 1):
58.        for question, answer in answerlist:
59.            print("Jenis batuan beku yang
   mungkin: ")
60.            print(answer)
61.    else:
62.        print("Jenis batu tidak dapat
   teridentifikasi")
```

Pola masukan diperiksa menggunakan algoritma KMP dan dihitung banyak bit yang cocok dengan di dictionary, Kemudian dibandingkan dengan panjang pola masukan. Jika perbandingan kecocokannya lebih dari 75% akan dicatat sebagai kandidat kemungkinan jenis batuan beku tersebut.

B. Algoritma BM

```
1.     dictionary =[]
2.     answerlist =[]
3.
4.     def boyermooreMatch(pattern, text):
5.         global n_same
6.         global i
7.         n_pattern = len(pattern)
8.         n_text = len(text)
9.         n_same = 0
10.        wordlist = []
11.        for k in range(1024):
12.            wordlist.append(n_pattern)
13.        for k in range(n_pattern - 1):
14.
   wordlist[ord(pattern[k])] = n_pattern - k - 1
15.        while k < n_text:
16.            j = n_pattern - 1
17.            i = k
18.
     while j >= 0 and text[i].lower() == pattern[
   j].lower():
19.                j -= 1
20.                i -= 1
```

```
                n_same += 1
            if j == -1:
                i += 1
            k += wordlist[ord(text[k][0])]
        return n_same

    def readfile(file):
        with open(file) as f:
            for line in f:
                (key, val) = line.split('\n ')
                dictionary.append([key,val])

    readfile("jenisbatu.txt")
    question = line.split('\n ')
    for idx, key in enumerate(dictionary):
    bmmatching = boyermooreMatch(question, diction
    ary)
        try:
      if (bmmatching/len(dictionary > 0.75):
    answerlist.append([key[0],key[1]])
        except ZeroDivisionError:
            pass

    if (len(answerlist) >= 1):
        for question, answer in answerlist:
            print("Jenis batuan beku yang
    mungkin: ")
            print(answer)
    else:
        print("Jenis batu tidak dapat
    teridentifikasi")
```

Pola masukan diperiksa menggunakan algoritma BM dan dihitung banyak bit yang cocok dengan di dictionary, Kemudian dibandingkan dengan panjang pola masukan. Jika perbandingan kecocokannya lebih dari 75% akan dicatat sebagai kandidat kemungkinan jenis batuan beku tersebut.

IV. Pengujian dan Pembahasan

A. *Skenario Pengujian*

Dalam pengujian performansi dari algoritma, script pengujian akan diuji pada computer dengan spesifikasi sebagai berikut:

- Processor: 3.4 GHz Intel Core i5-8250U
- RAM: 8 GB
- Tipe system: 64-bit Operating System, x64-based processor

Setiap algoritma akan diuji dengan beberapa kasus pengujian.

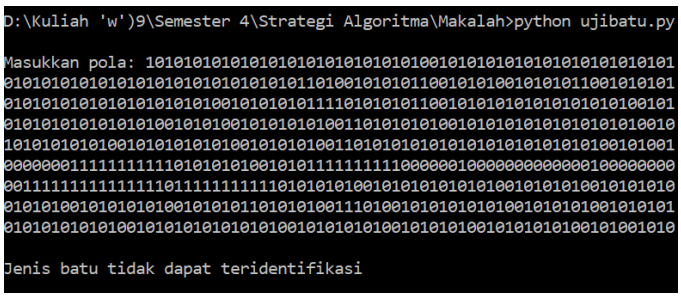
1. Kasus pengujian untuk batu yang tidak dapat teridentifikasi
2. Kasus pengujian untuk batu yang dapat teridentifikasi

B. *Kriteria Pengujian*

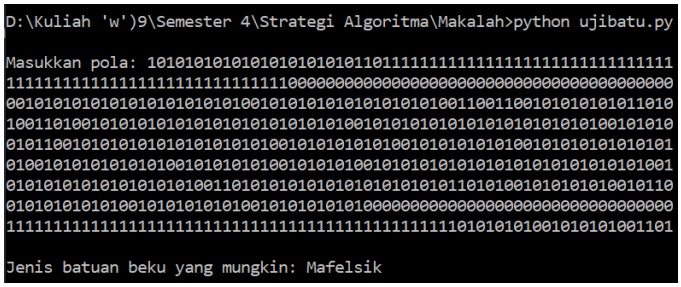
Parameter keberhasilan dalam pengujian ini hanya berdasarkan kesuksesan program dalam mengidentifikasi dan menghasilkan keluaran yang sesuai dengan ekpektasi dari hasil implementasi.

C. *Hasil Pengujian*

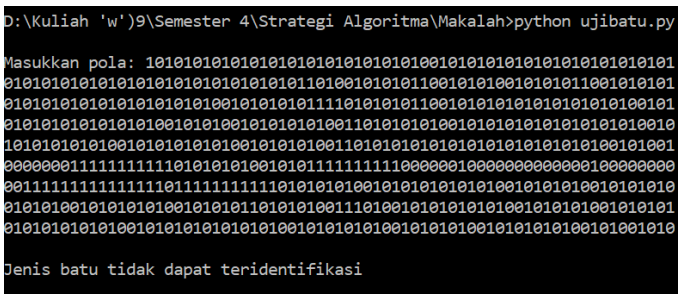
1. Kasus Uji 1 – Jenis batu tidak teridentifikasi (menggunakan algoritma KMP)



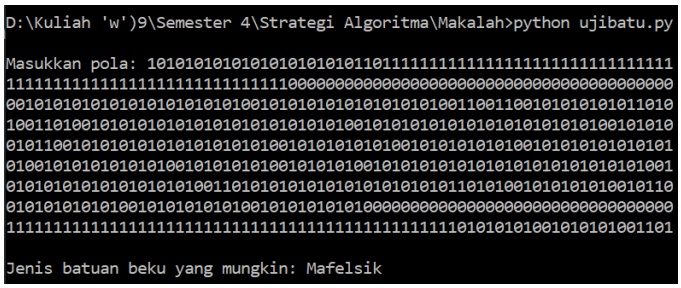
2. Kasus Uji 2 – Jenis batu teridentifikasi (menggunakan algoritma KMP)



3. Kasus Uji 3 – Jenis batu tidak teridentifikasi (menggunakan algoritma BM)



4. Kasus Uji 4 – Jenis batu teridentifikasi (menggunakan algoritma BM)



Hasil di atas menunjukkan baik algoritma KMP maupun BM mampu melakukan pencocokan string untuk mengidentifikasi jenis batuan beku.

DAFTAR PUSTAKA

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf diakses pada 25 April 2019 pukul 06.00 WIB

[2] https://www.kajianpustaka.com/2016/04/pengolahan-citra-digital.html diakses pada 25 April 2019 pukul 10.00 WIB

[3] Lecroq, Thierry Charras, Christian. 2001. Handbook of Exact String Matching Algorithm. ISBN 0-9543006-4-5

[4] Knuth, Donald E. Morris, James H. Pratt, Vaughan R. 1977. Fast Pattern Matching in Strings, SIAM Journal of Computing Vol 6 No.2. K. Elissa, “Title of paper if known,” unpublished.

[5] Boyer, Robert Moore, J. 1977. A Fast String Searching Algorithm. Comm. ACM 20: 762–772

[6] Prothero, Donald R.; Schwab, Fred (2004).Sedimentary geology : an introduction to sedimentary rocks and stratigraphy (2nd ed.). New York: Freeman. p. 12. ISBN 978-0-7167-3905-0.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2019

Nurul Utami Amaliah. W
13517132