Pemanfaatan Algoritma Pencocokan Pola dalam Melakukan Verifikasi Kesamaan Karya Digital Metode Pencarian Submatriks

Aufa Fadhlurohman and 13518009[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13518009@std.stei.itb.ac.id

Abstrak*—Plagiarisme karya adalah hal yang cukup sering terjadi di masyarakat saat ini. Kemudahan mendapatkan informasi membuat orang-orang terkadang tidak sadar akan kepemilikan setiap informasi dan karya yang ada di jaringan internet. Salah satu permasalahan plagiarisme yang sering terjadi adalah pencurian karya ilustrasi digital. Masih banyak pihak yang belum sadar dan peduli mengenai kepemilikan suatu karya. Hasilnya, muncul kasus kasus pencurian karya yang mendatangkan kerugian bagi seniman atau kreator yang telah membuat karya dengan ide dan tenaganya sendiri. Terkadang pihak yang memakai suatu karya orang lain tanpa hak mengakui karya tersebut adalah miliknya. Mereka juga seringkali menghapus watermark kepemilikan karya dengan cara memotongnya. Untuk itu, diperlukan suatu program praktis yang dapat membantu melakukan verifikasi pencurian karya tersebut dengan membandingkan karya yang dicurigai dengan yang dimiliki seorang pembuat karya. Gambar digital dibangun oleh kode-kode yang terstruktur. Dengan demikian, salah satu cara verifikasi yang dapat dilakukan adalah dengan melakukan pencarian pola submatriks dari mastriks suatu karya asli, dimana matriks merupakan hasil ekstraksi dari gambar digital.

Kata Kunci*—Pattern Matching, Ilustrasi, Pencurian Karya, Plagiarisme, Pencocokan Pola

I. PENDAHULUAN

Perkembangan internet yang semakin pesat menciptakan kemudahan dalam mengakses jalur komunikasi dan informasi. Kemudahan yang tercipta ini tentunya memberikan banyak manfaat bagi manusia dan kehidupannya, terutama dalam membangun dunia. Kemudahan akses informasi ini tidak lepas dari pesatnya perkembangan internet. Setiap orang dapat melakukan apapun lewat media global yang satu ini, mulai dari memberikan informasi kegiatannya kepada kerabat, tanya jawab suatu persoalan, membagikan pengalaman, hingga melakukan jual beli. Dengan adanya internet, pola kehidupan seakan semakin berubah, dan setiap manusia dituntut untuk terus dapat beradaptasi untuk terus melanjutkan eksistensinya.

Salah satu yang merasakan pengaruh dari perkembangan informasi ini adalah para seniman dan kreator. Sebelumnya, mungkin mereka melakukan pemasaran karyanya lewat pameran secara langsung, dan setelah perkembangan informasi ini harus menyesuaikan dengan memasang karyanya di internet untuk mencari setiap orang yang tertarik. Namun, terdapat beberapa pihak yang belum sadar dan peduli mengenai kepemilikan, sekali pun di dalam media internet. Akhirnya terdapat pihak-pihak yang mengambil karya orang lain, melakukan modifikasi sehingga tidak terdapat tanda kepemilikan pada karya tersebut, dan mengakui menjadi suatu karya miliknya. Hal demikian membuat para seniman atau pencipta karya sering dirugikan dan merasa kurang dihargai. Perlakuan yang sama ini pun sering dirasakan oleh para tim media suatu lembaga. Perilaku yang demikian ini bahkan dapat dikategorikan sebagai pencurian karya digital yang terdapat peraturannya dalam peraturan yang berlaku di Negara maupun Internasional.

Terkadang pencurian karya ini diketahui oleh sang pemilik dan pembuat karya. Dan diperlukan metode untuk melakukan verifikasi yang menyatakan suatu karya adalah karya aslinya. Salah satunya adalah dengan membandingkannya secara langsung. Dan untuk memperkuat argumen verifikasi, jika suatu karya dalam bentuk digital, dapat dilakukan beberapa metode pengecekan. Salah satu metode yang paling sederhana adalah dengan melakukan pengecekan kesamaan piksel pada kedua gambar. Setiap gambar digital dibangun oleh piksel-piksel yang dasarnya adalah untaian Kode ASCII. Metode ini merupakan pencarian submatriks tertentu dalam sebuah matriks.

Setiap piksel dalam sebuah gambar digital berwarna dibangun atas 3 buah Kode ASCII. Ketiga kode tersebut merujuk pada warna dasar yang membangun warna campuran pada piksel itu sendiri. Warna dasar pada tampilan digital biasanya terdiri dari warna RGB (*red, green, blue*), sehingga satu byte untuk menyimpan tingkat warna merah, satu byte untuk menyimpan tingkat warna hijau, dan satu byte lainnya menyimpan tingkat warna biru.

Untuk melakukan perbandingan nilai RGB pada piksel-piksel gambar, diperlukan algoritma pencocokan nilai dari tiap piksel. Algoritma *pattern matching* atau pencocokan pola dapat dilakukan untuk melakukan perbandingan ini. Terdapat beberapa algoritma yang umum digunakan untuk membandingkan pola. Pada penulisan kali ini, akan digunakan beberapa algoritma untuk mencoba dan membandingkan hasilnya untuk mendapatkan yang terbaik. Algoritma tersebut antara lain: Brute Force, Knuth Morris Pratt, dan Boyer Moore. Setiap algoritma memiliki cara dan kondisi efisiennya tersendiri. Terlebih gambar digital biasanya memiliki jumlah piksel yang sangat banyak dan keberagaman piksel yang

berbeda sehingga diperlukan algoritma yang cocok dan cukup cepat dalam melakukan komputasi.

II. LANDASAN TEORI

2.1 Pencocokan Pola

Pencocokan pola (*pattern matching*) adalah sebuah teknik untuk mencari sebuah pola dalam suatu ekspresi tertentu. Biasanya digunakan untuk melakukan pencarian suatu substring dalam sebuah string. Namun, algoritma ini dapat melakukan pencocokan pula pada setiap sesuatu yang memiliki pola yang terstruktur seperti string, salah satunya adalah kode ASCII. Terdapat beberapa definisi umum yang biasanya digunakan dalam pembahasan pencocokan pola ini, yaitu:

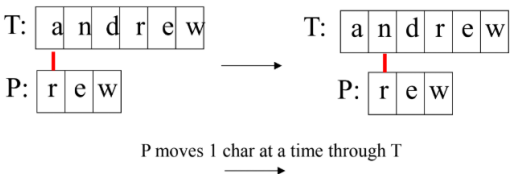
1. T : Teks, yaitu (long) string yang panjangnya *n* karakter.
2. P : Pattern, yaitu *string* dengan panjang *m* karakter yang akan dicari dalam teks [1].

Selain itu, dalam sebuah string, sering digunakan beberapa istilah yaitu prefiks dan sufiks. Prefix dari sebuah string S adalah sebuah substring S[*0..k*], sedangkan sufiks adalah sebuah substring S[*k..m-1*] dengan *m* adalah panjang string dan *k* adalah sembarang indeks antara *0* dan *m-1* [1].

2.2 Algorima Brute Force

Brute Force adalah pemecahan suatu persoalan dengan melakukan traversal setiap kemungkinan ruang pencarian untuk mendapatkan solusi yang diinginkan [2]. Brute force merupakan algoritma konvensional yang melakukan perbandingan pattern secara satu-persatu. Algoritma ini sederhana untuk diimplementasikan, dan akan menghasilkan solusi jika solusi ada. Namun algoritma ini tergolong cukup boros dari segi waktu pemrosesan.

Dalam melakukan pencocokan, algoritma brute force mengecek setiap posisi di teks T untuk melihat pola P pada posisi mula [1]. Berikut adalah ilustrasi sederhana pencocokan pada algoritma ini :



Gambar 2.2.1 Ilustrasi Pergeseran pencocokan pada Algoritma Brute Force
Sumber : Slide Kuliah Strategi Algortima 2020 (Dr. Rinaldi Munir)

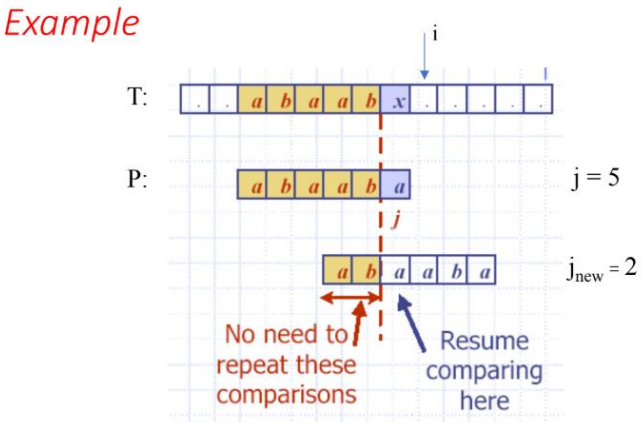
Teks: NOBODY NOTICED HIM
Pattern: NOT

```
  NOBODY NOTICED HIM
1 NOT
2  NOT
3   NOT
4    NOT
5     NOT
6      NOT
7       NOT
8        NOT
```

Gambar 2.2.2 Ilustrasi Tahap Pencocokan pada Algoritma Brute Force
Sumber : Slide Kuliah Strategi Algortima 2020 (Dr. Rinaldi Munir)

2.3 Algorima Knuth Morris Pratt

Algoritma Knuth Morris Pratt (KMP) dikembangkan oleh D. E Knuth, J. H. Morris, dan V. R. Pratt. Algoritma ini membandingkan pola dari kiri ke kanan seperti algoritma brute force, namun dengan mekanisme pergeseran yang lebih baik karena disesuaikan dengan kondisi. Prinsip dasarnya adalah jika terjadi ketidakcocokan antara pola dan teks pada P[j], maka geser berdasar prefix terpanjang dari P[0..j] yang juga merupakan sebuah sufiks dari P[1..j-1][2].



Gambar 2.3.1 Ilustrasi Tahap Pencocokan pada Algoritma Knuth Morris Pratt
Sumber : Slide Kuliah Strategi Algortima 2020 (Dr. Rinaldi Munir)

Pada contoh di atas, terjadi *mismatch* pada P[j] dengan j = 5. Untuk menghitung jumlah pergeseran, cari prefix terpanjang yang juga merupakan sufiks dari P[0..j-1] = **ab**a**ab** yaitu ab dengan panjang 2. Dan pengeseran untuk pencocokan selanjutnya sebanyak panjang substring dikurang panjang ab yaitu 5-2 = 3. Nilai dari prefiks terbesar yang juga sufiks dikomputasikan sebelum melakukan pencocokan dan setiap pola memiliki tabel yang menyimpan nilai-nilai itu. Fungsi untuk mendapatkan nilai dari perhitungan tadi disebut fungsi pinggiran KMP atau dikenal dengan *failure function*. Berikut adalah algoritma untuk mencari nilai dari fungsi pinggiran [4] :

Algorithm 1 COMPUTE-$\pi(w)$

```
1: π[1] ← 0
2: k ← 0
3: for i ← 2 to n do
4:     while k > 0 and w[k + 1] ≠ w[i] do
5:         k ← π[k]
6:     end while
7:     if w[k + 1] = w[i] then
8:         k ← k + 1
9:     end if
10:    π[i] ← k
11: end for
```

Sumber :https://www.researchgate.net/publication/226397506_Validating_the_Knuth-Morris-Pratt_Failure_Function_Fast_and_Online/

Biasanya hasil perhitungan fungsi pinggiran disimpan dalam suatu larik untuk nantinya dipanggil saat pencocokan dengan

teks. Berikut adalah hasil komputasi dari fungsi pinggiran untuk pola “abaaba” :

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a
k	-	0	1	2	3	4
b(k)	-	0	0	1	1	2

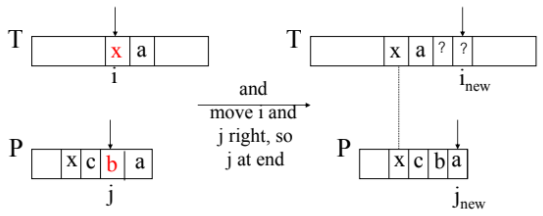
Gambar 2.3.2 Tabel Border Function Knuth Morris Pratt
Sumber : Slide Kuliah Strategi Algortima 2020 (Dr. Rinaldi Munir)

Algoritma pencarian KMP memiliki kompleksitas waktu algoritma $O(m+n)$. Dan algoritma ini memiliki performa yang sangat baik pencarian pola dalam keragaman alfabet yang rendah karena akan banyak pergeseran dalam jumlah yang banyak.

2.4 Algoritma Boyer Moore

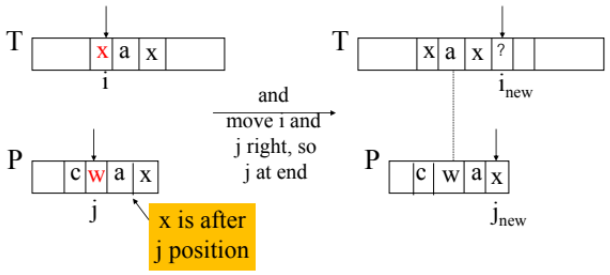
Algoritma Boyer Moore adalah salah satu algoritma yang memiliki performa paling cepat dalam masalah pencocokan umum. Algoritma ini mencocokkan karakter dari yang paling kanan dengan arah ke kiri. Jika terjadi ketidakcocokkan, digunakan sebuah fungsi yang telah dihitung sebelumnya untuk menentukan banyaknya pergeseran ke kanan. Algoritma Boyer Moore menerapkan dua teknik. Teknik pertama adalah *looking-glass* yaitu menemukan pola dalam teks secara mundur, dimulai dari akhir pola. Dan yang kedua adalah teknik *character-jump* yang merupakan tiga kasus pergeseran Ketika terjadi ketidakcocokkan antara pola dan teks.Berikut adalah ketiga kasus yang dapat terjadi :

1) Kasus 1
Jika pola P memiliki karakter x, dan terjadi *mismatch* antara pada teks ke-i dengan $T[i] = x$ dan posisi kemunculan x terakhir di kiri *mismatch* pada pola maka geser pola P ke posisi kemunculan x terakhir dalam pola sehingga x *miss* bersanding dengan x “*Last Occurence*”.[1]



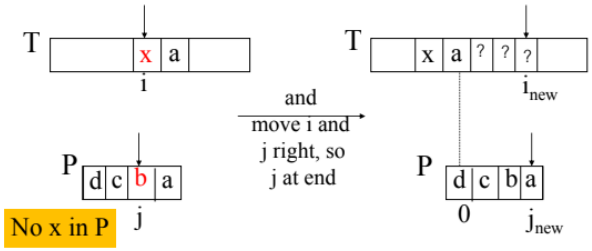
Gambar 2.4.1 Visualisasi Kasus pertama Boyer Moore
Sumber : Slide Kuliah Strategi Algoritma 2020 (Dr. Rinaldi Munir)

2) Kasus 2
Jika pola P memiliki karakter x, dan terjadi *mismatch* antara pada teks ke-i dengan $T[i] = x$ dan posisi kemunculan x terakhir di kanan *mismatch* pada pola maka geser pola P ke kanan 1 karakter ke $T[i+1]$.[1]
Berikut adalah ilustrasi pergeseran untuk kondisi kedua ini:



Gambar 2.4.2 Visualisasi Kasus kedua Boyer Moore
Sumber : Slide Kuliah Strategi Algoritma 2020 (Dr. Rinaldi Munir)

3) Kasus 3
Jika tidak terjadi kasus 1 dan 2 atau dengan kata lian tidak terdapat x pada pola P maka geser pola P untuk selanjutnya membandingkan P[0] dengan T[i+1| dengan i adalah posisi *mismatch* sebelumnya.

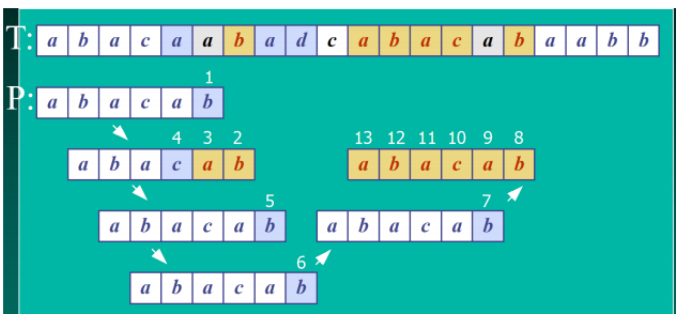


Gambar 2.4.3 Visualisasi Kasus ketiga Boyer Moore
Sumber : Slide Kuliah Strategi Algoritma 2020 (Dr. Rinaldi Munir)

Berikut ini adalah contoh pencocokan string menggunakan algoritma Boyer Moore dengan teks adalah “abacaabad cabacabaabb” dan pola yang ingin dicari “abacab”. Sebelum menyandingkan, algoritma ini akan melakukan pemrosesan awal berupa pembuatan tabel kemunculan terakhir dari tiap karakter yang ada pada pola untuk nantinya digunakan untuk menghitung banyaknya pergeseran yang akan dilakukan. Dari persoalan yang disebutkan, terbentuk tabel kemunculan terakhir seperti berikut:

x	*a*	*b*	*c*	*D*
L(x)	*4*	*5*	*3*	*-1*

Angka -1 pada tabel hasil merupakan tanda bahwa karakter tersebut tidak terdapat pada pola. Dan proses pencocokan akan seperti berikut ini:

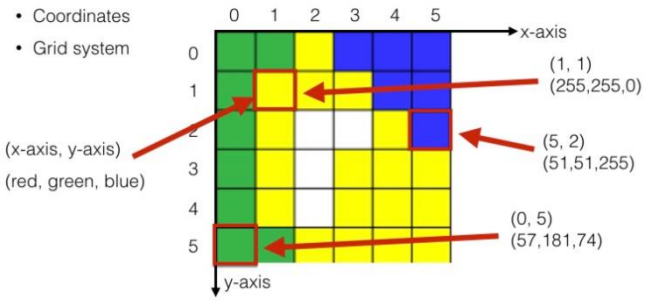


Gambar 2.4.4 Visualisasi Proses Boyer Moore
Sumber : Slide Kuliah Strategi Algoritma 2020 (Dr. Rinaldi Munir)

Algoritma Boyer Moore memiliki kemampuan yang sangat baik dalam mengolah alfabet yang keragamannya besar, dan cenderung lebih cepat dari algoritma Brute Force. Berkas foto memiliki nilai byte warna yang sangat beragam, sehingga diprediksi algoritma ini juga akan bekerja baik pada berkas foto. Algoritma ini memiliki kompleksitas waktu terburuk pada $O(nm + A)$ dengan A keragaman alfabet.

2.5 Konsep Gambar Digital

Gambar digital terdiri dari untaian *byte* kode yang merepresentasikan kadar warna pembangun tiap pikselnya. Dalam satu piksel terdiri dari tiga komponen RGB yaitu kadar merah, kadar hijau, dan kadar biru. Tiap kadar direpresentasikan dengan sebuah char atau 8-bit angka (0-255). Piksel-piksel tersebut memiliki koordinat dalam sumbu-x dan sumbu y sehingga terdapat keteraturan yang membuat tampilan sebuah gambar. Berikut adalah ilustrasi yang menggambarkan konsep gambar digital :

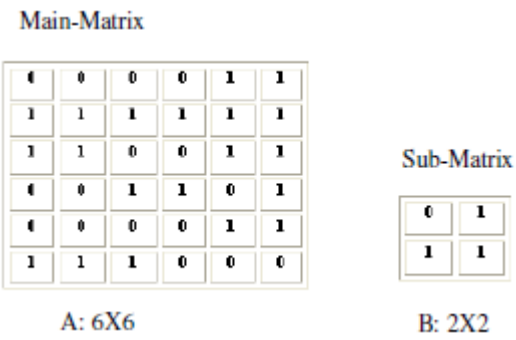


Gambar 2.5.1 Konsep Digital Image

Sumber : https://knowthecode.io/labs/basics-of-digitizing-data/episode-14

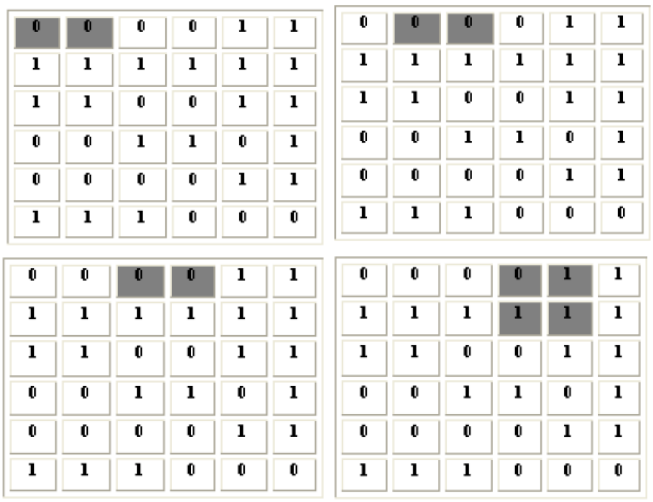
2.6 Pencocokan Submatriks

Persoalan pencarian submatriks dalam matriks dapat dilakukan dengan melakukan pencocokan polanya. Prinsip dasarnya dengan melakukan pencocokan tiap *grid* dari tiap baris, jika terdapat baris yang memiliki kesamaan pola, periksa sepanjang banyak baris submatriks ke bawah, jika sesuai semua maka ditemukan, namun jika tidak sesuai balik ke pencarian terakhir. Berikut contoh proses pencocokan sederhana menggunakan algoritma Brute Force :



Gambar 2.6.1 Contoh pencocokan matriks sederhana

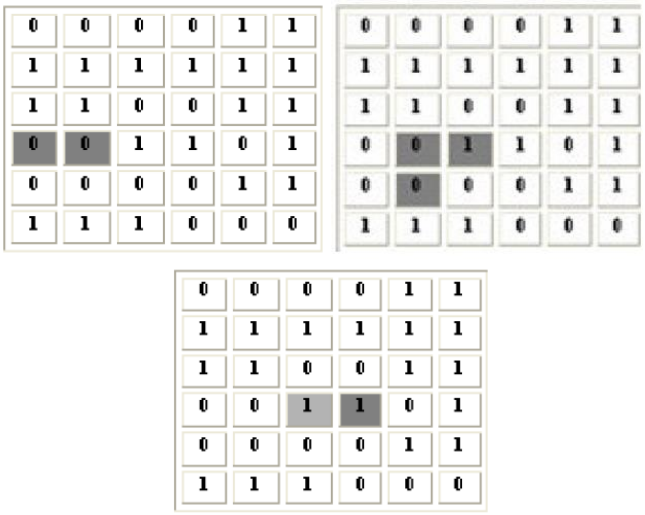
Sumber : Efficient Processing for Binary Submatrix Matching. American Journal of Applied Sciences



Gambar 2.6.2 Contoh pencocokan matriks sederhana : Pola ditemukan

Sumber : Efficient Processing for Binary Submatrix Matching. American Journal of Applied Sciences

Pada kasus di atas ini pola baris pertama ditemukan, lalu dicek baris kedua pada pola dan sama, maka pola ditemukan.



Gambar 2.6.3 Contoh pencocokan matriks sederhana : Pola tidak ditemukan

Sumber : Efficient Processing for Binary Submatrix Matching. American Journal of Applied Sciences

Sedangkan pada kasus yang satu ini, pola baris pertama ditemukan, namun saat dicek baris kedua pola (submatriks) tidak cocok, maka pencarian sebelumnya dilanjutkan.

Pada contoh ilustrasi sebelumnya merupakan implementasi pencarian dengan algortima brute force. Pada algoritma lainnya akan terdapat perbedaan dalam metode pergeseran sesuai dengan konsep tiap algoritma. Untuk melakukan pemeriksaan tiap barisnya berikut merupakan cuplikan algoritma yang digunakan (KMP hanya contoh) :

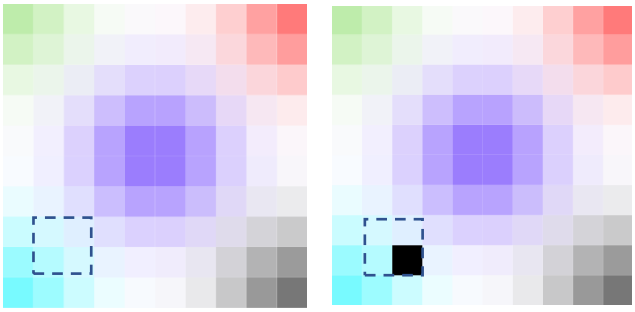
```
def check(pattern, text):
    pos = -1
    k = 0
    i = 0
    while(i < len(text) and k < len(pattern)):
        theKMP = KnuthMorrisPrrat(pattern[k])
        searchRes = theKMP.search(text[i])
        if(searchRes != -1):
            if(k == 0):
```

```
                pos = searchRes
                k = k + 1
            else:
                if(pos == searchRes):
                    k = k + 1
                else:
                    k = 0
        else:
            k = 0
        i = i + 1
    return(k != 0)
```

Gambar 2.6.4 Kode Pencarian SubMatriks
Sumber : Dokumentasi Pribadi Penulis

III. Analisis Permasalahan

Pengujian pencarian submatriks dalam matriks untuk mencari potongan gambar ini memerlukan dua buah masukan gambar. Masukkan diproses menjadi kode-kode RGB untuk tiap piksel yang dimiliki. Sebagai contoh digunakan dua buah gambar yang akan menjadi matriks pengujian pertama ini. Tiap gambar terdiri dari 10 x 10 piksel dan dengan 3 *byte* kode representasi RGB tiap pikselnya membuat matriks pemeriksaan berukuran 30 x 10. Nilai RGB dari tiap piksel diekstrak menggunakan perkakas PIL pada bahasa python. Berikut adalah ilustrasi dari kedua bahan uji yang digunakan untuk pengujian:



Gambar 3.1 Bahan Uji Matriks1.png(kiri) dan Matrix1Edited.png(kanan)
Sumber : Dokumentasi Pribadi Penulis

Pola yang ingin dicari adalah sebuah gambar berukuran 2 x 2 piksel yang ditandai pada kotak bergaris putus-putus pada *Matriks1.png*. Sebelumnya gambar diekstrak dan menghasilkan Matriks berikut :

Pertama, dilakukan pencarian Submatriks1 dalam Matriks1Edited (pola yang dicari diubah) dan saat eksekusi menghasilkan keluaran berikut:

```
$ Algoritma Pattern Matching :

$ 1. BruteForce
False
time : 2.38427734375 ms
$ 2. Knuth Morris Pratt
False
time : 0.416748046875 ms
$ 3. Boyer Moore
False
time : 2.95947265625 ms
```

Gambar 3.3 Hasil pencarian SubMatrix1.png dalam Matrix1Edited.png
Sumber : Dokumentasi Pribadi Penulis

Terlihat pola tidak ditemukan dan bersesuaian dengan yang seharusnya didapatkan (dapat dilihat pada gambar 3.1). Untuk waktu masih sangat singkat karena bahan uji tidak dalam ukuran yang besar.

Selanjutnya akan dilakukan pencarian untuk pola Submatriks1 dalam Matriks1. Dan menghasilkan keluaran sebagai berikut :

```
$ Algoritma Pattern Matching :

$ 1. BruteForce
True
time : 0.0 ms
$ 2. Knuth Morris Pratt
True
time : 15.622314453125 ms
$ 3. Boyer Moore
True
time : 0.0 ms
```

Gambar 3.4 Hasil pencarian SubMatrix1.png dalam Matrix1.png
Sumber : Dokumentasi Pribadi Penulis

Pola Submatriks1 ditemukan (seperti pada ilustrasi pada gambar 3.1 kiri) dan KMP mendapatkan waktu eksekusi yang paling lambat. Hal ini kemungkinan dikarenakan komputasi fungsi pinggiran serta gambar/matriks yang cenderung memiliki ruang alfabet yang banyak. Ketiga algoritma berhasil mendapatkan pola submatriks yang dicari.

```
Matrix :
190 236 171 210 242 196 232 248 226 246 251 245 250 249 251 252 247 251 253 236 238 254 207 208 255 161 161 255 122 122
210 242 196 222 244 214 235 245 235 241 242 248 241 237 253 242 235 252 246 232 243 251 215 220 254 185 186 255 157 157
232 248 226 235 245 235 235 237 245 228 222 251 220 209 253 220 208 253 231 217 248 243 221 237 251 214 219 254 203 204
246 251 245 241 242 248 228 222 251 205 190 253 184 163 254 184 162 254 204 188 252 231 216 247 246 231 242 253 235 237
249 250 252 242 239 254 220 209 253 184 163 254 155 125 253 155 125 253 184 162 254 220 208 253 243 236 252 251 246 250
248 251 255 240 239 255 219 209 254 184 163 254 155 125 253 155 125 253 184 163 254 219 208 253 239 235 252 248 246 250
234 252 255 233 243 255 224 223 254 203 190 254 184 163 254 184 163 254 203 189 252 224 216 247 233 231 242 235 235 237
202 252 255 213 249 255 223 239 255 224 223 254 219 209 254 219 208 253 224 216 247 223 219 235 212 211 216 202 202 203
155 251 255 181 251 255 213 249 255 233 243 255 241 239 255 240 236 252 232 230 241 211 210 215 179 179 180 155 155 155
120 250 255 157 252 255 204 252 255 235 253 255 247 250 254 246 246 250 233 233 235 200 200 201 153 153 153 119 119 119

SubMatrix :
213 249 255 223 239 255
181 251 255 213 249 255
```

Gambar 3.2 Hasil Ekstraksi Bahan Uji Matriks1.png dan SubMatriks1.png
Sumber : Dokumentasi Pribadi Penulis

Gambar 3.5 Bahan Uji Matriks2.png(kiri) dan SubMatrix2.png(kanan)
Sumber : Dokumentasi Pribadi Penulis

Pengujian selanjutkan akan memperlihatkan algoritma yang memiliki tingkat efisiensi terbaik pada kondisi data yang sangat banyak. Ukuran gambar digital pada karya fotografi tersebut adalah 2250 x 4000 piksel yang artinya memiliki matriks karakter atau byte dengan ukuran 6750 x 4000 data byte. Dan untuk submatriks memiliki ukuran 666 x 620 piksel atau 1998 x 620 data byte. Ukuran yang sangat besar ini diperkirakan akan membuat efisiensi tiap algoritma terlihat. Dan setelah pengujian didapatkan :

```
$ Algoritma Pattern Matching :

$ 1. BruteForce
True
time : 5770.973876953125 ms
$ 2. Knuth Morris Pratt
True
time : 11932.41552734375 ms
$ 3. Boyer Moore
True
time : 1989.260009765625 ms
```

Gambar 3.5 Hasil pencarian SubMatrix2.png dalam Matrix2.png
Sumber : Dokumentasi Pribadi Penulis

Dari hasil terlihat bahwa Algoritma Boyer Moore memiliki performa yang paling baik unutk kasus pencocokan karya fotograsi ini. Hal ini dikarenaka foto tersusun atas piksel yang sangat beragam dan jumlah macam byte yang mendirikannya pasti sangat beragam. Algoritma Brute Force memiliki performa kedua terbaik. Sedangkat selisih kesuanya dengan Algoritma Knuth Morris Pratt cukup terlihat. Kemungkinan hal ini dikarenakan ukuran submatriks yang juga tergolong besar serta keragamannya yang besar. Untuk submatriks yang besar tersebut Algoritma KMP selalu melakukan perhitungan nilai fungsi pinggiran. Algoritma KMP dan Bm juga kemungkinan besar dipengaruhi oleh alokasi larik untuk tiap fungsinya yang selalu dilakukan.

Dan terakhir gambar *Matriks2.png* diedit dengan memasukkan gambar *Submatriks2.png* yang memiliki ukuran 2 x 2 piksel. Ternyata didapatkan nilai yang mirip dengan sebelumnya namun waktu eksekusi Algoritma KMP yang lebih cepat. Hal ini kemungkinan karena dengan submatriks yang kecil, perhitungan fungsi pinggiran KMP menjadi sangat minim, dan perilakunya menjadi mirip dengan Algoritma Brute Force.

```
$ Algoritma Pattern Matching :

$ 1. BruteForce
True
time : 5734.95166015625 ms
$ 2. Knuth Morris Pratt
True
time : 6081.050048828125 ms
$ 3. Boyer Moore
True
time : 2559.417236328125 ms
```

Gambar 3.6 Hasil pencarian SubMatrix1.png dalam Matrix2.png
Sumber : Dokumentasi Pribadi Penulis

Dengan menggunakan teknik pencocokan pola pada penyelesaian pencarian submatriks ini, suatu gambar digital dapat secara sangat sederhana dicocokkan dengan gambar lainnya. Hal ini dapat menjadi dasar, pengembangan harus selalu dilanjutkan karena persoalan ini adalah persoalan yang menarik. Mungkin masih terdapat berbagai algoritma lain yang dapat melakukan pencarian submatriks ini. Namun ternyata algoritma pencocokan pola dapat melakukannya secara sederhana.

IV. KESIMPULAN

Algoritma pencocokan pola tidak hanya dapat dimanfaatkan untuk pencarian pola pada string. Metode pencarian submatriks menggunakan algoritma pencocokan pola adalah sebuah hal yang dapat dilakukan. Dengan metode ini, dapat dicari hasil dari potongan gambar dan dapat dilakukan pengecekan apakah potongan gambar tersebut adalah bagian identik dari suatu gambar. Hal ini dapat membantu secara sederhana proses pengecekan pada kasus-kasus pengambilan karya digital tanpa izin. Proses untuk melakukan hal demikian dimulai dengan melakukan konversi gambar menjadi nilai dari kadar RGB tiap pikselnya. Selanjutnya cari submatriks menggunakan algoritma pencocokan pola seperti brute force, knuth morris pratt, dan boyer moore. Dari hasil pemrosesan gambar yang merupakan sebuah karya fotografi, algoritma boyer moore cenderung memiliki kecepatan yang paling naik disbanding yang lain. Hal ini dikarenakan karya fotografi terdiri dari piksel-piksel yang sangat beragam dan jumlah variasi char dalam byte warna sangat luas. Namun metode pengecekan yang demikian masih tergolong sangat sederhana, diperlukan algoritma pembacaan yang leboh fungsional seperti penentuan persen kemiripan. Penyelesaian untuk persoalan ini harus terus dikembangkan.

V. UCAPAN TERIMA KASIH

Puji syukur kehadirat Allah SWT, karena atas limpahan Rahmat dan Karunia-Nya, sehingga penulis dapat

menyelesaikan pembuatan makalah ini. Terima kasih penulis ucapkan kepada Bapak Dr. Ir. Rinaldi Munir, M. T. sebagai dosen pengampu dalam mata kuliah IF2211 Strategi Algoritma ini. Terima kasih juga penulis ucapkan kepada kedua Orang Tua penulis yang selalu memberikan dukungan moral dalam setiap aktivitas penulis.

LINK VIDEO

http://bit.ly/13518009StrategiAlgoritmaVideo

LINK REPOSITORI KODE

http://bit.ly/13518009StrategiAlgoritmaRepo

REFERENSI

[1] Munir, Rinaldi. Slide Kuliah Strategi Algoritma, Bandung: Informatika, 2020.
[2] Steven Halim, Felix Halim. Competitive Programming 3. Singapore, 2013.
[3] Hondro, Rivalri & Hsb, Zumrotul & Gienam, Suginam & Sianturi, Ronda. (2016). Implementasi Algoritma Knuth Morris Pratt Pada Aplikasi Penerjemahan Bahasa Mandailing-Indonesia. Jurnal Riset Komputer (JURIKOM). 3. 49-53.
[4] Gawrychowski, Pawel & Jeż, Artur & Jeż, Łukasz. (2010). Validating the Knuth-Morris-Pratt Failure Function, Fast and Online. Theory of Computing Systems. 54. 132-143. 10.1007/978-3-642-13182-0_13.
[5] Sleit, Azzam & Almobaideen, Wesam & Qatawneh, Mohammad & Saadeh, Heba. (2009). Efficient Processing for Binary Submatrix Matching. American Journal of Applied Sciences. 6. 10.3844/ajas.2009.79.89.

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.

Depok, 2 April 2020

Aufa Fadhlurohman
13518009