Tugas ke :
Mata Kuliah : Analisis Algoritma

Tugas Pattern Matching

Disusun Oleh :

NAMA : Moch Achmar
STB : 222362

KELAS : K

UNIVERS1TAS DIPA MAKASSAR

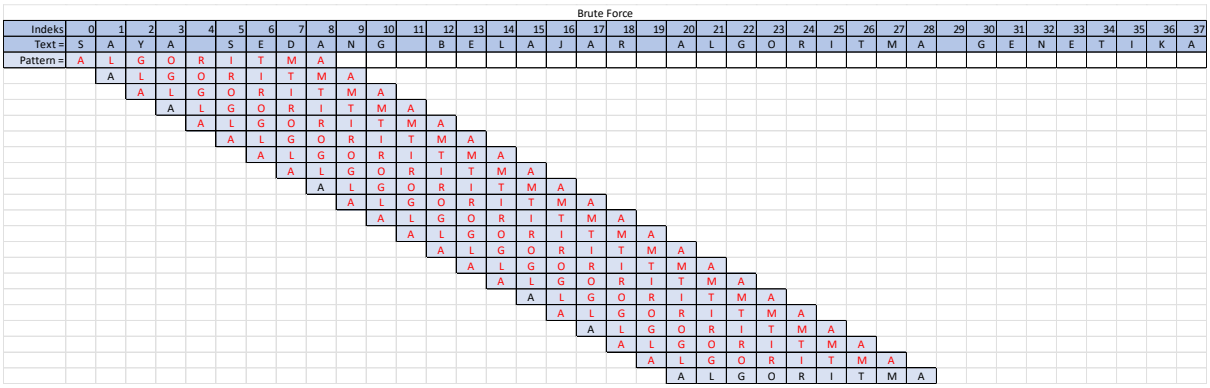
MAKASSAR

2023

1. Lakukan analisis algoritma pattern matching untuk string sebagai berikut:
   Text = SAYA SEDANG BELAJAR ALGORITMA GENETIKA
   Pattern = ALGORITMA
   Selesaikan:
   a. Brute force
   b. Knuth-Morris-Pratt
   c. Booyer Moore
   d. Rabin-Karp
2. Penyelesaian yang sama seperti No. 1
   T = MOCH ACHMAR
   P = ACHMAR
3. Selesaikan analisis pola string berikut:
   T = ABACADABABCABA
   P = ABA
   Selesaikan:
   a. Brute force
   b. Knuth-Morris-Pratt

   Jawab:

1. a. Brute force



   Kompleksitas kasus tergantung pada panjang teks dan panjang pattern yang dicari. Kompleksitas waktu pada algoritma pencocokan pola brute force diatas adalah kasus rata-rata (average case) yaitu O(m+n) di mana n adalah panjang teks dan m adalah panjang pattern (m < n).

   Jumlah pergeseran yang dilakukan dalam algoritma ini adalah (n - m + 1). Setiap pergeseran memerlukan perbandingan karakter antara pola dan teks pada posisi yang bersesuaian. Jumlah perbandingan dalam setiap pergeseran adalah sebanyak n kali perbandingan.

Pseudocode:
procedure PencocokanString(input P : string, T : string, n, m : integer, output idx : integer)
{ Masukan: pattern P yang panjangnya m dan teks T yang panjangnya n. Teks T direpresentasikan sebagai string (array of character)
Keluaran: lokasi awal kecocokan (idx)
}

Deklarasi
i : integer
ketemu : boolean

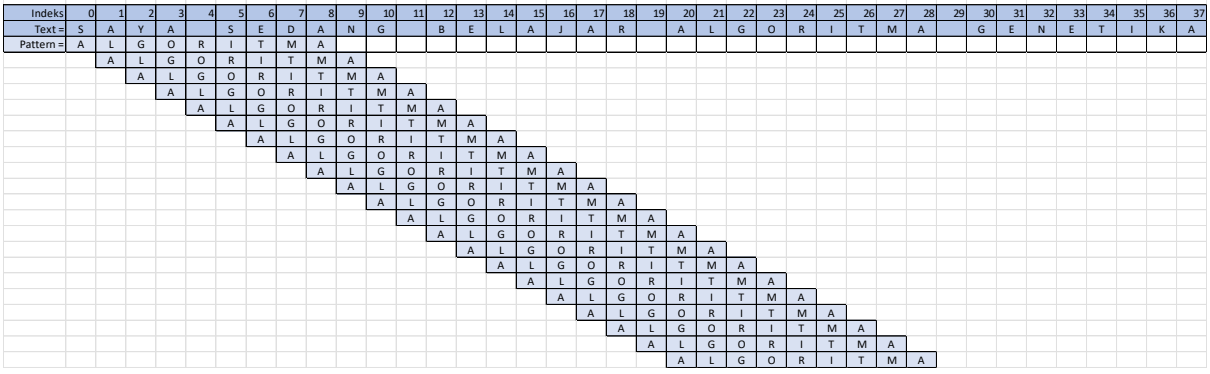
Algoritma:
$i \leftarrow 0$
$ketemu \leftarrow false$
while ($i \leq n - m$) and (not ketemu) do
    $j \leftarrow 1$
    while ($j \leq m$) and ($P_j = T_{i+j}$) do
        $j \leftarrow j + 1$
    endwhile
    { $j > m$ *or* $P_j = T_{i+j}$ }

    if $j = m$ then { kecocokan string ditemukan }
        $ketemu \leftarrow true$
    else
        $i \leftarrow i + 1$ {geser pattern satu karakter ke kanan teks }
    endif
endwhile
{ $i > n - m$ *or* $ketemu$ }
if ketemu then return $i + 1$ else return -1 endif

b. Knuth-Morris-Pratt

Knuth-Morrtis-Pratt (KMP)

Indeks	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													
Boundary =	0	0	0	0	0	0	0	0	1																													



Menghitung fungsi pinggiran : O(m)

Pencarian String : O(n)

Kompleksitas total : O(m+n)

Keuntungan : tidak pernah bergerak “mundur” (mengulang pemeriksaan) seperti Brute Force dan cocok untuk memproses file yang ukuran besar atau streaming

Kerugian : Ketika variasi karakter teks nya beragam, akan lebih sering mismatch seperti Brute Force dan Algoritma ini tidak memperhitungkan apakah karakter yang mismatch

Pseudocode:

```
function computeLPSArray(pattern):
    m = length(pattern)
    lps = array of size m
    len = 0
    i = 1
    lps[0] = 0
    while i < m:
        if pattern[i] = pattern[len]:
            len = len + 1
            lps[i] = len
            i = i + 1
        else:
            if len != 0:
                len = lps[len - 1]
            else:
                lps[i] = 0
                i = i + 1
    return lps


function knuthMorrisPratt(text, pattern):
    n = length(text)
    m = length(pattern)
    lps = computeLPSArray(pattern)
    i = 0
    j = 0
    while i < n:
        if pattern[j] = text[i]:
            i = i + 1
            j = j + 1
            if j = m:
                return i - j
        else:
            if j != 0:
                j = lps[j - 1]
            else:
                i = i + 1
    return -1
```

c. Booyer Moore

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													

Menghitung fungsi L(x) : O(A)

Pencarian String kasus terburuk : O(nm)

Kompleksitas total untuk kasus terburuk: O(nm+A)

Keuntungan : Lebih cepat dibandingkan Brute Force untuk teks dengan variasi karakter yang sangat beragam (A besar) atau karakter dalam Bahasa Inggris

Kerugian : Lambat untuk teks dengan variasi karakter tidak beragam (karakter binary)

Pseudocode:

```
function preprocessBadCharacterTable(pattern):
    m = length(pattern)
    badChar = array of size 256
    for i from 0 to 255:
        badChar[i] = m
    for i from 0 to m - 1:
        badChar[pattern[i]] = m - 1 - i
    return badChar

function boyerMoore(text, pattern):
    n = length(text)
    m = length(pattern)
    badChar = preprocessBadCharacterTable(pattern)
    s = 0
    while s <= n - m:
        j = m - 1
        while j >= 0 and pattern[j] = text[s + j]:
            j = j - 1
        if j < 0:
```

```
            return s
        else:
            s = s + max(1, badChar[text[s + j]] - j)
    return -1
```

d. Rabin-Karp

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													

Pseudocode:

```
function rabinKarp(text, pattern, prime):
    n = length(text)
    m = length(pattern)
    hashPattern = hash(pattern)
    hashText = hash(text[0...m-1])
    for i from 0 to n - m:
        if hashPattern = hashText:
            if text[i...i+m-1] = pattern:
                return i
        if i < n - m:
            hashText = (hashText - text[i] * prime^(m-1)) * prime + text[i+m]
    return -1
```

2. a. Brute force

	0	1	2	3	4	5	6	7	8	9	10
Text =	M	O	C	H		A	C	H	M	A	R
Pattern =	A	C	H	M	A	R					
		A	C	H	M	A	R				
			A	C	H	M	A	R			
				A	C	H	M	A	R		
					A	C	H	M	A	R	
						A	C	H	M	A	R

Pseudocode:

```
function bruteForce(text, pattern):
    n = length(text)
    m = length(pattern)
    for i from 0 to n - m:
        j = 0
        while j < m and text[i + j] = pattern[j]:
            j = j + 1
        if j = m:
            return i
    return -1
```

b. Knuth-Morris-Pratt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													

Pseudocode:

```
function computeLPSArray(pattern):
    m = length(pattern)
    lps = array of size m
    len = 0
    i = 1
    lps[0] = 0
    while i < m:
        if pattern[i] = pattern[len]:
            len = len + 1
            lps[i] = len
            i = i + 1
        else:
            if len != 0:
                len = lps[len - 1]
            else:
                lps[i] = 0
                i = i + 1
    return lps


function knuthMorrisPratt(text, pattern):
    n = length(text)
    m = length(pattern)
    lps = computeLPSArray(pattern)
    i = 0
    j = 0
```

```
    while i < n:
        if pattern[j] = text[i]:
            i = i + 1
            j = j + 1
            if j = m:
                return i - j
        else:
            if j != 0:
                j = lps[j - 1]
            else:
                i = i + 1
    return -1
```

c. Booyer Moore

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													

Pseudocode:

```
function preprocessBadCharacterTable(pattern):
    m = length(pattern)
    badChar = array of size 256
    for i from 0 to 255:
        badChar[i] = m
    for i from 0 to m - 2:
        badChar[pattern[i]] = m - 1 - i
    return badChar


function boyerMoore(text, pattern):
    n = length(text)
    m = length(pattern)
    badChar = preprocessBadCharacterTable(pattern)
    s = 0
    while s <= n - m:
```

```
    j = m - 1
    while j >= 0 and pattern[j] = text[s + j]:
        j = j - 1
    if j < 0:
        return s
    else:
        s = s + max(1, badChar[text[s + j]] - j)
  return -1
```

d. Rabin-Karp

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													

Pseudocode:

```
function rabinKarp(text, pattern, prime):
  n = length(text)
  m = length(pattern)
  hashPattern = hash(pattern)
  hashText = hash(text[0:m])
  for i from 0 to n - m:
    if hashPattern = hashText:
      if text[i:i+m] = pattern:
        return i
    if i < n - m:
      hashText = (hashText - text[i] * prime^(m-1)) * prime + text[i+m]
  return -1

function hash(str, prime):
  hashValue = 0
  n = length(str)
  for i from 0 to n - 1:
    hashValue = (hashValue * prime + str[i]) % prime
  return hashValue
```

3. a. Brute force

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Text =	A	B	A	C	A	D	A	B	A	B	C	A	B	A
Pattern =	A	B	A											

Pseudocode:

```
function bruteForce(text, pattern):
  n = length(text)
  m = length(pattern)
  for i from 0 to n - m:
    j = 0
```

```
        while j < m and text[i + j] = pattern[j]:
            j = j + 1
        if j = m:
            return i
    return -1
```

b. Knuth-Morris-Pratt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Text =	S	A	Y	A		S	E	D	A	N	G		B	E	L	A	J	A	R		A	L	G	O	R	I	T	M	A		G	E	N	E	T	I	K	A
Pattern =	A	L	G	O	R	I	T	M	A																													

Pseudocode:

```
function computeLPSArray(pattern):
    m = length(pattern)
    lps = array of size m
    len = 0
    i = 1
    lps[0] = 0
    while i < m:
        if pattern[i] = pattern[len]:
            len = len + 1
            lps[i] = len
            i = i + 1
        else:
            if len != 0:
                len = lps[len - 1]
            else:
                lps[i] = 0
                i = i + 1
    return lps

function knuthMorrisPratt(text, pattern):
    n = length(text)
    m = length(pattern)
    lps = computeLPSArray(pattern)
    i = 0
    j = 0
    while i < n:
        if pattern[j] = text[i]:
            i = i + 1
            j = j + 1
            if j = m:
                return i - j
        else:
            if j != 0:
                j = lps[j - 1]
            else:
                i = i + 1
    return -1
```