Perbandingan Algoritma Knuth-Morris-Pratt (KMP) dan Brute Force dalam Permainan *Word Search Puzzle*

Taufan Fajarama Putrawansyah Ruslanali - 13520031
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520031@std.stei.itb.ac.id

***Abstrak*—*Word search puzzle* adalah salah satu permainan puzzle untuk mengasah otak dalam mencari kata-kata tersembunyi dari susunan huruf berbentuk grid/matriks. Pencarian kata tersebut dapat dilakukan dengan berbagai cara dan dapat menggunakan algoritma untuk mengoptimasi pencariannya. Algoritma yang digunakan pada makalah ini adalah *pattern matching* menggunakan algoritma *brute force* dan algoritma Knuth-Morris-Pratt (KMP). Terdapat perbedaan yang cukup signifikan di mana algoritma KMP lebih cepat menyelesaikan *word search puzzle* daripada algoritma *brute force*. Algoritma KMP juga memakai perbanding huruf yang lebih efisien daripada algoritma *brute force* sehingga algoritma KMP dapat dianggap sebagai algoritma yang lebih mangkus daripada algoritma *brute force* untuk permasalahan ini.**

Kata Kunci*—*Brute Force, Knuth-Morris-Pratt (KMP), Pattern Matching, Word Search Puzzle.

I. Pendahuluan

Word Search Puzzle adalah sebuah *game* atau permainan yang didesain menyerupai puzzle yang berisikan berbagai macam huruf. Tugas pemain adalah menemukan kumpulan huruf-huruf yang dapat membentuk suatu kata. Susunan huruf-huruf tersebut dapat berupa horizontal, vertikal, atau diagonal. Permainan ini dapat dimainkan sendiri maupun bersama-sama dengan teman, keluarga, dan orang lainnya. Jika dimainkan sendiri, pemain akan menyelesaikan permainan ketika semua kata ditemukan. Ketika bermain bersama orang lain, pemain yang menemukan kata terbanyak yang akan menjadi pemenangnya. Banyak kata, huruf, besar tabel, dan level kesulitan ditentukan oleh masing-masing pemain.

Gambar 1. Word search puzzle (sumber: play.google.com)

Salah satu cara untuk mencari kata dalam word search puzzle adalah dengan menggunakan algoritma *brute force*. Algoritma yang terbilang mudah ini seringkali digunakan orang untuk menyelesaikan suatu persoalan. Manusia biasanya secara alamiah akan menggunakan algoritma *brute force* untuk menyelesaikan persoalan baru yang ditemuinya karena algoritma *brute force* bekerja secara naif. Selain algoritma *brute force*, algoritma Knuth-Morris-Pratt (KMP) juga dapat digunakan dalam menyelesaikan word search puzzle. Algoritma KMP melakukan *pattern matching* seperti *brute force*, tetapi dengan pergeseran yang lebih efisien. Melalui makalah ini, penulis akan membahas perbandingan algoritma *brute force* dan algoritma KMP dalam permainan word search puzzle.

II. Landasan Teori

A. *Pattern Matching*

Pattern matching adalah proses memeriksa apakah suatu pola (*pattern*) terdapat pada suatu teks. Terdapat beberapa algoritma untuk melakukan *pattern matching*, algoritma yang umum digunakan adalah algoritma *brute force* dan algoritma Knuth-Morris-Pratt yang memiliki kesamaan pencarian pola dari kiri ke kanan teks. *Pattern matching* terdiri atas 2 komponen berikut:

1. Sebuah teks (*text*), yaitu sebuah (*long*) *string* yang panjangnya n karakter
2. Sebuah pola (*pattern*), yaitu sebuah *string* dengan panjang m karakter (m < n) yang akan dicari di dalam teks

Pattern matching akan mencari lokasi pertama di dalam teks yang bersesuaian dengan pola (*pattern*) yang diberikan.

Contoh:

Teks: Di mana-mana banyak orang ber**jual**an bakso

Pattern: **jual**

B. *Algoritma Brute Force*

Algoritma *brute force* adalah pendekatan yang lempang (*straight forward*) untuk memecahkan suatu persoalan.

Biasanya didasarkan pada pernyataan pada persoalan (*problem statement*) dan definisi konsep yang dilibatkan. Contoh berdasarkan pernyataan persoalan yaitu mencari elemen terbesar atau terkecil dan pencarian beruntun (*sequential search*). Contoh berdasarkan definisi konsep yang terlibat, yaitu menghitung a^n (a > 0, n bilangan bulat tak-negatif), menghitung n! (n bilangan bulat tak-negatif), mengalikan dua buah matriks, tes bilangan prima, algoritma pengurutan *brute force* (*bubble sort* & *selection sort*), dan mengevaluasi polinom.

Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, dan jelas (*obvious way*). Algoritma brute force memiliki karakteristik:

1. Tidak "cerdas" dan tidak mangkus karena membutuhkan jumlah komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Sesuai namanya "force" mengindikasikan "tenaga" dibandingkan "otak". Dapat dikatakan juga algoritma naif.

2. Lebih cocok untuk persoalan berukuran kecil karena sederhana dan implementasinya mudah. Sering digunakan sebagai basis pembanding dengan algoritma lain yang lebih mangkus.

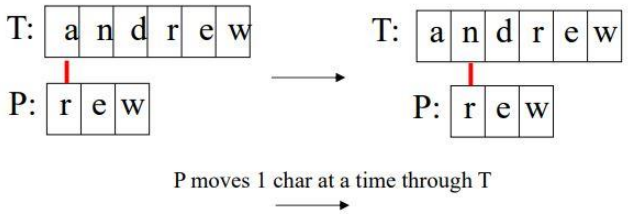
3. Hampir semua persoalan dapat diselesaikan. Bahkan, ada persoalan yang hanya dapat diselesaikan dengan brute force, yaitu mencari elemen terbesar di dalam senarai (*list*).

Penerapan algoritma *brute force* dalam *pattern matching* dengan teks T[1..n] dan *pattern* P[1..m] sebagai berikut:

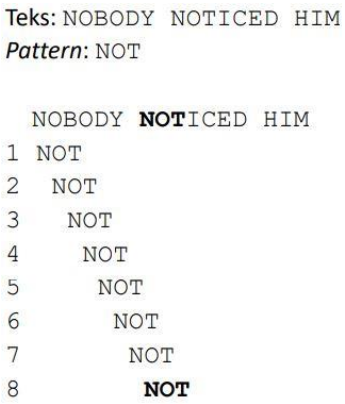
1. Mula-mula *pattern* P dicocokan pada awal teks T

2. Dengan bergerak dari kiri ke kanan, bandingkan setiap karakter di dalam *pattern* P dengan karakter yang bersesesuaian di dalam teks T sampai semua karakter yang dibandingkan cocok atau sama (pencarian berhasil), atau dijumpai sebuah ketidakcocokan karakter (pencarian belum berhasil)

3. Bila *pattern* P belum ditemukan kecocokannya dan teks T belum habis, geser *pattern* P satu karakter ke kanan dan ulangi langkah 2.



Gambar 2. Pergeseran karakter algoritma brute force (sumber: informatika.stei.itb.ac.id)



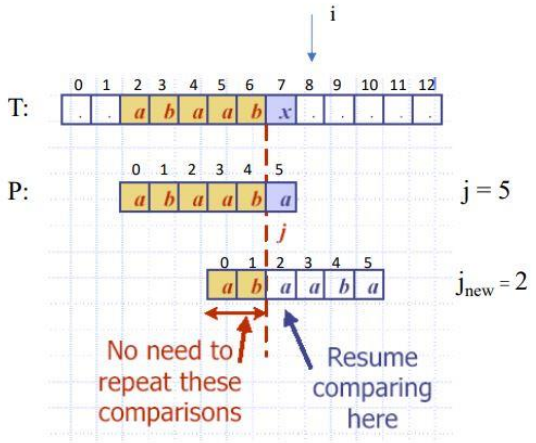
Gambar 3. Contoh penerapan algoritma brute force (sumber: informatika.stei.itb.ac.id)

C. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) melakukan *pattern matching* dari awal teks ke akhir teks seperti algoritma *brute force*. Akan tetapi, algoritma KMP melakukan pergeseran karakter yang lebih "cerdas" daripada algoritma *brute force*.

Algoritma KMP melakukan pergeseran lebih jauh sesuai dengan informasi yang disimpan, tidak seperti pada algoritma *brute force* di mana pergeseran dilakukan setiap satu karakter, sehingga waktu pencarian dapat dikurangi secara signifikan

Algoritma KMP menggunakan properti degenerasi (pola yang memiliki sub-pola yang sama muncul lebih dari sekali dalam pola) dari pola dan meningkatkan kompleksitas kasus terburuk menjadi O(n). Ide dasar di balik algoritma KMP adalah setiap kali kita mendeteksi ketidakcocokan (setelah beberapa kecocokan), kita sudah mengetahui beberapa karakter untuk pergeseran selanjutnya. Informasi ini digunakan untuk menghindari pencocokan karakter yang kita ketahui akan tetap cocok apabila diurutkan dari awal pola.

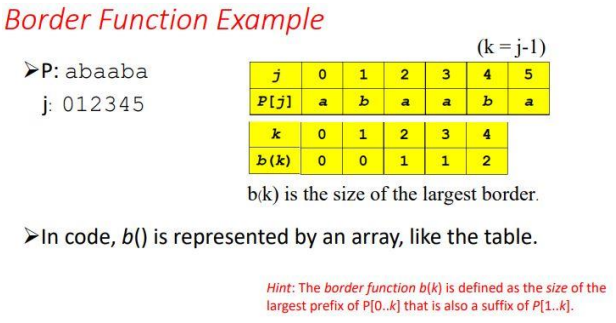


Gambar 4. Pergeseran karakter algoritma KMP (sumber: informatika.stei.itb.ac.id)

Dalam algoritma KMP, ada beberapa definisi yang digunakan:

1. Misalkan A adalah alfabet dan $x = x_1x_2...x_k$, $k \in N$, adalah string yang panjangnya k yang dibentuk dari karakter-karakter di dalam alfabet A.
2. Awalan (prefix) dari *x* adalah upa-string (substring) *u* dengan $u = x_1x_2...x_{k-1}$, $k \in \{1, 2, \ldots, k-1\}$ dengan kata lain, *x* diawali dengan *u*.
3. Akhiran (suffix) dari *x* adalah upa-string (substring) *v* dengan $v = x_{k-b}x_{k-b+1}...x_k$, $k \in \{1, 2, \ldots, k-1\}$ dengan kata lain, *x* diakhiri dengan *v*.
4. Pinggiran (border) dari *x* adalah upa-string (substring) *r* sedemikian sehingga $r = x_1x_2...x_{k-1}$ dan $u = x_{k-b}x_{k-b+1}...x_k$, $k \in \{1, 2, \ldots, k-1\}$ dengan kata lain, pinggiran dari *x* adalah upastring yang keduanya awalan dan juga akhiran sebenarnya dari *x*

Algoritma KMP melakukan proses awal terhadap *pattern* P dengan menghitung fungsi pinggiran yang mengindikasikan pergeseran terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum *pattern matching*. Sebagai contoh apabila j adalah posisi terjadi ketidakcocokan karakter dan k adalah posisi sebelum terjadi ketidakcocokan (k = j-1), maka fungsi pinggiran (*border function*) yang diberikan seperti pada Gambar 5.



Gambar 5. Contoh hasil perhitungan fungsi pinggiran algoritma KMP (sumber: informatika.stei.itb.ac.id)

III. IMPLEMENTASI

Implementasi *pattern matching* algoritma *brute force* dan algoritma KMP pada makalah ini menggunakan bahasa Java.

A. *Pattern Matching Algoritma Brute Force*

Fungsi akan mengembalikan index di mana *pattern* ditemukan atau mengembalikan -1 jika *pattern* tidak ditemukan.

```
public static int brute(String text,String pattern) {
  int n = text.length();        // n is length of text
  int m = pattern.length(); // m is length of pattern
  int j;
  for(int i=0; i <= (n-m); i++) {
    j = 0;
    while ((j < m) &&
(text.charAt(i+j)==pattern.charAt(j))) {
      j++;
    }
    if (j == m)
      return i; // match at i
  }
  return -1; // no match
} // end of brute()
```

B. *Fungsi Pinggiran Algoritma KMP*

Fungsi akan mengembalikan hasil perhitungan fungsi pinggiran KMP.

```
public static int[] computeFail(String pattern) {
  int fail[] = new int[pattern.length()];
  fail[0] = 0;
  int m = pattern.length();
  int j = 0;
  int i = 1;
  while (i < m) {
    if (pattern.charAt(j) == pattern.charAt(i)) {
    // j+1 chars match
      fail[i] = j + 1;
      i++;
      j++;
    } else if (j > 0) { // j follows matching prefix
        j = fail[j-1];
    } else { // no match
        fail[i] = 0;
        i++;
    }
  }
  return fail;
}
```

C. *Pattern Matching Algoritma KMP*

Fungsi akan mengembalikan index di mana *pattern* ditemukan atau mengembalikan -1 jika *pattern* tidak ditemukan.

```
public static int kmpMatch(String text, String
pattern) {
  int n = text.length();
  int m = pattern.length();
  int fail[] = computeFail(pattern);
  int i = 0;
  int j = 0;
  while (i < n) {
    if (pattern.charAt(j) == text.charAt(i)) {
      if (j == m-1) {
        return (i - m + 1); // match
      }
    i++;
    j++;
    } else if (j > 0) {
        j = fail[j-1];
    } else {
        i++;
    }
  }
  return -1; // no match
}
```

IV. PENGUJIAN

Berikut adalah hasil pengujian algoritma *brute force* (atas) dan algoritma KMP (bawah) dengan beberapa data uji:

A. *Word Search Puzzle Ukuran 15x15 Test1*

```
N I E T S M B Y H P N S Y R B
D Q U N X E B E R Z M I E A W
P D C N Z J K O N F H E V H J
R L A G E R S R J S B A Y C L
E H X D Q I B R E D R B S W E
T T V L T F W V A I J U A X D
Z W L T K S E N A E E C U M U
E B O B Q P C S V G J Q S T R
L C H Y X I L O T Q Z Z A S T
S H O L N B W E M I G F G G S
F X C G W V Q E I E V E E V X
T X L G F J W Z R D D A E W L
I P A R A D E M J B L E L N N
X P Q H V L A A G F N L X K T
Q Q E I V N S P A L T E R Y V

ALCOHOL
BAVARIA
BEER
BREW
DANCING
FESTIVAL
GERMAN
LAGER
PARADE
PRETZELS
PROSIT
SAUSAGE
SPALTER
STEIN
STRUDEL
```

Gambar 6. Data uji 15x15 (sumber: arsip pribadi)

```
Waktu eksekusi program (dalam nanosecond): 636049200
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 1456

Waktu eksekusi program (dalam nanosecond): 424032800
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 1164
```

Gambar 7. Hasil uji 15x15 (sumber: arsip pribadi)

B. *Word Search Puzzle Ukuran 15x15 Test2*

```
W E M E Q R S S R E L I O Q V
F L A M E S F W Q W Z Y S A B
W C A P I T A L S W R E K T O
O G Q Z G V E Z K M R S N U V
W K I N G S L A D B S D I C C
S L I V E D E Z A R J T G N D
E N S W N O L S O G A U H J G
U R I D T B P T H F V O T J D
S D Z U R V A U L E A G S Q D
O I X U G N M Y V T L P R I C
S U I I E N E S T J A Z E Q A
M N D S I R E N V N N E G S R
S J B C S B P P O E C U N T N
C B S F W Z X A C T H F A E B
X V M I I Q W H Q N E O R J I

AVALANCHE
BRUINS
CAPITALS
DEVILS
FLAMES
FLYERS
JETS
KINGS
KNIGHTS
MAPLE LEAFS
OILERS
PENGUINS
RANGERS
SABRES
SENATORS
```

Gambar 8. Data uji 15x15 (sumber: arsip pribadi)

```
Waktu eksekusi program (dalam nanosecond): 618592100
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 1476

Waktu eksekusi program (dalam nanosecond): 412394733
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 1180
```

Gambar 9. Hasil uji 15x15 (sumber: arsip pribadi)

C. *Word Search Puzzle Ukuran 15x15 Test3*

```
E T P Y H F Z A S I V G E X E
B J H L I M E H C K Q A T L L
X R S N A A U X R B M X D I E
Q A H Y S U G A R A P S A D M
H F A B F Z N M T W F G O I L
F E M J K Q M I E D A J N S F
A M R P D F D S U V X T F A U
C E O I G H F P O Q M A J G U
P R C N R E D C E A E D E E A
A A K E F U A A V R Y L T C G
R L V O R D A S Z G S M R O L
I D C F O P P V Y I M I X A H
S Z L A X Q S S O M K R A D H
Q T W Y E L O R J E G Z P N Q
P F O R E S T F F J E V I L O

ASPARAGUS
AVOCADO
DARK MOSS
EMERALD
FOREST
HARLEQUIN
JADE
LIME
MINT
OLIVE
PARIS
PERSIAN
PINE
SAGE
SHAMROCK
```

Gambar 10. Data uji 15x15 (sumber: arsip pribadi)

```
Waktu eksekusi program (dalam nanosecond): 579878500
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 1469

Waktu eksekusi program (dalam nanosecond): 386585666
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 1175
```

Gambar 11. Hasil uji 15x15 (sumber: arsip pribadi)

D. Word Search Puzzle Ukuran 20x22 Test1

```
C A M Y R J N G H D Q X F Y R T Q V Q J F F
G V B M N L R I C E S I Z T O G H V E S I P
T F B W Q D I U X H T J N V H B D F S N O R
D E T N U P G I O E Y L O O K I J S I C P E
F Z W A M Y P M R J K Y I Y H E N S Z Z K S
U V G I E V U Z I R H Y T I E A H K J A Y E
M N Y I K C Q G O J N X S F J G S V E F B R
Y K B H Z J A W J E N W E S T N A H D R I V
B U O E V Z T F Q K A S G O I G R C A F L I
K Q C S K E A A T T F M N M A N C S P T D N
M X T S N N P R K U T N O H C I Y R E V A G
Y Q O Q Q G O C D N O Y C L M T N W T G Y P
K Y I F S N W W Z N P K A X I R A I O N U E
I E R M Y I N X N Z R Q V M Z O R V N O R X
T V M F O R D U Y T E M O W O S X D A R T S
V R P Z L A C O M I V Y M K E R Y A L G D N
L U K T P L R G D T A V T I C R T V N I Z S
U S H R M C A C N Q Z A Y E D A G I U R U Z
C X Y I E E Z Z U O U J S S P A H A Z E X B
G X F B B D D W F L R Y G T X B I W C E V Y

ADLIB
AMORTIZE
CAGEY
CONGESTION
DARTS
DECLARING
EMPLOY
FINISH
GIZZARD
GREW
NETWORK
NOTEPAD
OUTFACE
PRESERVING
PUNTED
SORTING
SURVEY
THINKER
UNBEKNOWN
WHACKO
```

Gambar 12. Data uji 20x22 (sumber: arsip pribadi)

```
Waktu eksekusi program (dalam nanosecond): 1425639800
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 4826

Waktu eksekusi program (dalam nanosecond): 950426533
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 3860
```

Gambar 13. Hasil uji 20x22 (sumber: arsip pribadi)

E. Word Search Puzzle Ukuran 20x22 Test2

```
F I U T W S M P O I N T L D P Y F S X H E D
J N X M U E P I K H S X W I V O X B E F D M
U F L L J L G W W E Y S Y C M U Z L W J O W
G C G F P R L G S U E V L E M N G O W Y G J
N T G Q L A T F N Q L K M N P G Q W N F E F
I M T U B H G B U I P J R S S S O Q J U J X
T V X A I C N E V X G L S U D T W H T J R B
N W E D K A N Q J N J G F S Y E N Z N A Q P
A Q O X I C T Z O Z B B E Z C R V M D G G V
H C H D I F E W B Q M V V P I H Y U T U E M
C C G N Z T M L D E D I D M Z S N M O A M K
J M G N M G Q K D I C M P P P G V G W R I U
G F L I X P R I M I C A M O E F W N E G G L
W N U R G D C E O N N M U O O Y Z D L A R M
V R I I S A H U X E C T N R E Z L S E L A H
T F Q D T H S Z L B I U M E T F N A D E T B
F G C E D B T E U N M H R S T L U D A V I B
W X D D M U D P G V P M D E L G N A M M N Q
I J B D A I P H C J T W I T C H U C V J G J
J X X T B F O X D K B J J S H H V U B X E L

ADULTS
ANGLED
CENSUS
CHANTING
CHARLES
DUNGEON
EMIGRATING
GALE
IMPANELED
JAGUAR
MEDICATED
PEGGING
POINT
PUDDING
SEQUENCING
SPOUTING
TOWELED
TWITCH
VICIOUS
YOUNGSTER
```

Gambar 14. Data uji 20x22 (sumber: arsip pribadi)

```
Waktu eksekusi program (dalam nanosecond): 1395825500
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 4218

Waktu eksekusi program (dalam nanosecond): 930550333
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 3374
```

Gambar 15. Hasil uji 20x22 (sumber: arsip pribadi)

F. Word Search Puzzle Ukuran 20x22 Test3

```
W N I G H T M A R E M T I J Y W Y F I G N W
J Q Z X S Q T I O J Y N N E F J Z R W D O R
J I A B M D D G N I C I L S Q O J I W F Y I
K Q W M Z E Q B I I M Q D D K B U Y O V H H
Y H X A M N S R F S B G C C O Y D A E W H Q
A H D Q S I K D A P K I B M K I Q K X R G B
V E M H I L G V V G Q R I U Z S M M C S D B
J S Z C L C U B C A E N X Z Z C A R N A G E
G N X M A N N C P C L W I C I T A H P M E N
O A N L R I N O O D U E B Z A H S F T R K M
J S F B U O U U Q B S I M M E R S I N G P W
D R O C T R N W W T E C T U R K I S H P G J
Y S C C A T B Q E N F E C O P S E Q F N G W
J A O R N P W R O J U N U K G O G H I K Y D
Q S X E E V I I Y P L F G I M T F N T C V O
G Q A S V C S J K B N O R V P R I W O E B K
X O O R A A S O A K E R B T L H L T Z Y E V
E O L O R C K A A W S A E W S D W M I V V S
G F T B U L X W Y X S H T I J G C K N H Y R
P V A R B Y E E S G Y K A M I M I C S B Q Y

ABRASION
BETA
CARNAGE
COPSE
CORD
DIZZIEST
EMPHATIC
GOOSE
IMMERSING
INCLINED
MIMIC
NATURALISM
NIGHTMARE
RECOUNT
SEETHE
SHINING
SLICING
SOAKER
TURKISH
USEFULNESS
```

Gambar 16. Data uji 20x22 (sumber: arsip pribadi)

```
Waktu eksekusi program (dalam nanosecond): 1257312200
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 5149

Waktu eksekusi program (dalam nanosecond): 838208133
Jumlah total perbandingan huruf untuk menemukan seluruh kata: 3089
```

Gambar 17. Hasil uji 20x22 (sumber: arsip pribadi)

V. KESIMPULAN

Algoritma yang digunakan pada makalah ini adalah *pattern matching* menggunakan algoritma *brute force* dan algoritma Knuth-Morris-Pratt (KMP). Terdapat perbedaan yang cukup signifikan di mana algoritma KMP lebih cepat menyelesaikan *word search puzzle* daripada algoritma *brute force*. Algoritma KMP juga memakai perbanding huruf yang lebih efisien daripada algoritma *brute force* sehingga algoritma KMP dapat dianggap sebagai algoritma yang lebih mangkus daripada algoritma *brute force* untuk permasalahan ini.

UCAPAN TERIMA KASIH

Pertama-tama penulis bersyukur kepada Tuhan Yang Maha Esa sehingga penulis dapat menyelesaikan makalah ini. Penulis ingin mengucapkan terima kasih khususnya kepada dosen K1, Ibu Masayu Leylia Khodra yang sudah menyalurkan ilmunya kepada penulis dan teman-teman kelas K1. Penulis juga ingin berterima kasih kepada seluruh tim pengajar dan tim asisten IF2211 Strategi Algoritma. Penulis harap makalah ini dapat bermanfaat bagi yang membacanya.

REFERENSI

[1] Munir, R. 2021. *Pencocokan String (String/Pattern Matching)*. URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses pada tanggal 23 Mei 2022.

[2] Munir, R. 2021. *Algoritma Brute Force*. URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)-Bag1.pdf. Diakses pada tanggal 23 Mei 2022.

[3] Geeks for Geeks. 2021. *KMP Algorithm for Pattern Searching*. URL: https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/. Diakses pada tanggal 23 Mei 2022.

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, 23 Mei 2022

Taufan Fajarama Putrawansyah R
NIM 13520031