Aplikasi Algoritma String Matching pada Pencarian Karakter Film The Lord Of The Rings

Menggunakan Algoritma Knuth–Morris–Pratt

Mohammad Afif Akromi 13519110
Program Studi Teknik
Informatika Sekolah Teknik
Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha
10 Bandung 13519110@std.stei.itb.ac.id

***Abstract*—Film The Lord of The Rings merupakan film keluaran tahun 2003. Film tersebut dibagi menjadi 3 bagian (trilogy). Pada film tersebut terdapat banyak sekali karakter-karakter yang diperankan actor/aktris. Film ini juga pernah dibuat versi game versi PC console. Mustahil untuk menghafal semua nama karakter dalam film ini. Oleh karena itu, kami membuat website yang dapat mencari nama-nama karakter yg ada alam film ini.**

Keywords—LOTR, Trilogy, Adventure, KMP, String Matching

I. PENDAHULUAN

The Lord of the Rings adalah seri film berisi tiga film petualangan fantasi yang disutradarai Peter Jackson. Film tersebut berdasarkan novel The Lord of the Rings oleh J. R. R. Tolkien. Film-film tersebut berjudul The Fellowship of the Ring (2001), The Two Towers (2002) dan The Return of the King (2003). Film-film tersebut didistribusikan oleh New Line Cinema.

Sering dianggap proyek film terbesar dan paling ambisius, dengan anggaran total $281 juta (beberapa sumber berkata $310 juta-$330 juta),[3] proyek ini memerlukan delapan tahun, dengan syuting semua film dilakukan bersamaan di Selandia Baru, negara asal Jackson.[4] Setiap film dalam seri memiliki edisi diperpanjang spesial yang dirilis di DVD setahun setelah dirilis di bioskop. Meskipun mengikuti cerita utama bukunya, ada bagian novel yang dihilangkan dan ada tambahan yang tidak ada di bahan sumber.

Terletak di Dunia Tengah, film ini menceritakan cerita hobbit Frodo Baggins (Elijah Wood) ketika dia dan Sembilan Pembawa Cincin melakukan perjalanan untuk menghancukan One Ring, dan dengan itu menghancurkan pembuatnya, Dark Lord Sauron. Para Pembawa Cincin terbelah dan Frodo meneruskan perjalanan dengan teman setianya Sam (Sean Astin) dan makhluk pengkhianat Gollum (Andy Serkis). Sementara itu, Aragorn (Viggo Mortensen), putra mahkota Gondor, dan penyihir Gandalf (Ian McKellen) mempersatukan Orang-orang bebas Dunia Tengah dalam Perang Cincin.

Di dalam kisah The Lord of The Rings terdapat banyak sekali karakter dan ras. Diantaranya:

1. Ras Isengard

Isengard, juga dikenal sebagai Angrenost ('Benteng Besi') di Sindarin, adalah salah satu dari tiga benteng utama Gondor, dan di dalamnya terdapat salah satu wilayah Palantiri. Namun, di paruh kedua Zaman Ketiga, benteng tersebut menjadi milik Saruman, menjadi wilayah pribadinya dan rumahnya sampai kekalahannya dalam Perang Cincin.

Isengard dibangun pada Zaman Kedua di sekitar menara Orthanc oleh orang Numenor, di pengasingan pada masa kekuasaan mereka di Nan Curunir (Lembah Penyihir) di Pegunungan Berkabut. Lokasinya berada di sudut barat laut Kerajaan Gondor, menjaga Fords of Isen dari serangan musuh ke Calenardhon dan, bersama dengan benteng Helm's Deep di selatan, melindungi Celah Rohan.

Isengard terdiri dari dinding batu hitam melingkar yang mengelilingi dataran luas, di tengahnya terdapat Menara Orthanc. Isengard hanya memiliki satu gerbang, yang menghadap ke selatan.

Sungai Angren (atau Isen) dimulai di Methedras di belakang Isengard, yang juga membentuk dinding utaranya. Tiga sisi lainnya dijaga oleh tembok besar, yang dikenal sebagai Cincin Isengard, yang hanya ditembus oleh aliran masuk sungai Angren di timur laut melalui portcullis, dan gerbang Isengard di selatan, di kedua pantai sungai. Gerbang itu dikatakan bisa terbuka tanpa suara.

2. Ras Mordor

Mordor adalah dataran vulkanik hitam yang terletak di tenggara Middle-earth di sebelah timur Gondor, Ithilien, dan sungai besar Anduin. Mordor dipilih oleh Sauron untuk menjadi wilayahnya karena pegunungan yang mengelilinginya di tiga sisi, menciptakan benteng alami melawan musuh-musuhnya.

Mordor dilindungi dari tiga sisi oleh pegunungan besar, disusun secara kasar dalam bentuk persegi panjang: Pegunungan Ash ('Ered Lithui') di utara, dan Ephel Dúath di barat dan selatan. Di sudut barat laut Mordor, lembah Udûn yang dalam adalah salah satu dari sedikit pintu masuk pasukan besar, dan di sanalah Sauron membangun Gerbang Hitam Mordor. Di depan Gerbang Hitam terdapat Dagorlad atau Medan Pertempuran. Benteng utama Sauron di Barad-dûr berada di kaki bukit Ered Lithui. Di sebelah barat daya Barad-dûr terdapat dataran tinggi Gorgoroth dan Gunung Doom yang gersang; di sebelah timur terbentang dataran Lithlad. Geografi Mordor sangat bagus untuk pertahanan melawan musuh yang menyerang di semua lini, karena pegunungan yang hampir tidak dapat diukur mempertahankan Mordor di tiga sisi, sementara tanah Gorgoroth dan Núrn yang rusak dan bergerigi akan sangat menghalangi pasukan yang berhasil menerobos.

3. Ras Man

Ras Manusia adalah ras makhluk penting kedua yang diciptakan oleh Dewa Tertinggi, Ilúvatar, karena mereka terbangun pada awal Zaman Pertama, pada saat terbitnya Matahari pertama (FA 1), sementara para elf terbangun tiga Zaman sebelum mereka . Mereka disebut "Afterborn" (Quenya Atani, Sindarin: Edain) oleh para Elf.

Pria memikul Karunia Pria, yaitu kefanaan, dan karena itu mereka menua dan mati ketika waktunya tiba, dan rentan terhadap penyakit dan penyakit. Peri itu abadi, dalam arti mereka tidak rentan terhadap penuaan dan penyakit. Bahkan jika tubuh mereka dibantai, roh mereka akan tetap terikat pada dunia selama itu berlangsung, dan pergi ke Aula Mando untuk menunggu sampai mereka dilepaskan atau dunia berakhir.

4. Ras Elf

Para Elf, yang menyebut diri mereka Quendi, dan yang dalam pengetahuan biasanya disebut sebagai

Eldar (adj. Eldarin), adalah yang pertama dan tertua dari Bani Ilúvatar, dan dianggap paling adil dan paling bijaksana dari semua ras Arda diberi sapience.

Beberapa, yang kemudian dikenal sebagai Calaquendi (Peri Cahaya), dibawa oleh Valar dari Middle-earth ke Valinor melintasi Laut, di mana mereka diajar oleh Ainur. Tapi setelah Silmarils dicuri oleh Melkor, beberapa Peri kembali ke Dunia Tengah, di mana mereka tinggal sampai akhir Zaman Ketiga.

Peri tidak tunduk pada usia, dan kebal terhadap penyakit. Mereka bisa dibunuh hanya dengan kekerasan atau dengan sangat putus asa.

II. LANDASAN TEORI

A. Pattern Matching

Pattern Matching adalah suatu algoritma yang biasanya digunakan untuk melakukan pencocokan pada suatu string/teks. Biasanya terdapat Teks (T) string yang memiliki Panjang N karakter dan suatu Pattern (P) yang memiliki Panjang M dengan asumsi M tidak lebih panjang dari N. Selanjutnya, Pattern Matching akan mengeluarkan lokasi dimana ditemukannya Pattern (P) didalam Teks (T).

CONTOH:
Diberikan suatu String/Teks (T) dan Pattern (P)

T = Salah Upload Makalah Matkul Strategi Algoritma
P = Strategi

Di zaman sekarang ini Pattern Matching sudah banyak sekali dimanfaatkan pada beberapa aplikasi IT. Contohnya adalah pada Search Engine Google, Search Box Microsoft Word yg sedang saya pakai sekarang ini, dan Message Search based Text pada aplikasi chatting Whatsapp dan masih banyak lagi implementasinya.

Algoritma untuk mencari Pattern Matching sendiri sudah banyak ditemukan. Namun, disini kita akan membahas tiga algoritma pencarian Pattern Matching itu sendiri yaitu algoritma Brute Force, Knuth Morris-Pratt, dan Booyer-Moore. Alasan dari pembahasan ketiga algoritma tersebut karena ketiga algoritma tersebut yang menjadi bahan ajar pada mata kuliah IF2211 Strategi Algoritma 2021 dan sangat berkaitan dengan makalah yang sedang saya kerjakan.

B. Algoritma String Matching

Algoritma string matching (pencocokan string) adalah algoritma untuk melakukan pencarian semua kemunculan string pendek (pattern) yang muncul dalam teks. Pettern yaitu string dengan panjang m karakter (m < n). Teks (text) yaitu long string yang panjangnya n karakter. Contoh implementasi String matching adala pencocokan string pada Microsoft word, editor atau pencocokan website dengan memasukkan kata kunci sebagaimana yang telah diterapkan pada search engine seperti yahoo, tau Google.

C. Algoritma KMP (Knuth-Morris-Pratt)

Algortima knuth-Morris-Pratt dikembang secara terpisah oleh Donald E-Knuth pada tahun 1967 James H. Morris bersama Vaughan R.Pratt pada tahun 1966, namun keduanya mempublikasikannya pada tahun 1977.

Jika kita lihat kembali algoritma straightforward (brute force) lebih mendalam, maka dapat kita ketahui bahwa mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian string. Dalam algoritma Knuth-Morris-Pratt kita dapat memutuskan keberhasilan dan kegagalan setiap karakter yang dibandingkan. Algoritma KMP membangun sebuah mesin automata yang status-statusnya adalah status dari string yang kita cari. Dan setiap status memiliki fungsi berhasil dan gagal. Berhasil artinya status mendekati akan bergerak mendekati ke status akhir dan gagal apabila status status menjadi semakin menjauh dengan status terakhir. Secara sitematis langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan sebuah string adalah sebagai berikut:

1. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkkan string karakter per karakter pattern denga teks yang bersesuaian, sampai salah satu kondisi terpenuhi:
    - Karakter di pattern dan teks yang dibandingkan tidak cocok (mismatch).
    - Semua di pattern cocok, kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma ini menggeser pattern pada table next, lalu mengulangi langkah 2 sampai pattern berada pada ujung teks.

Berikut adalah pseudo code algoritma Knuth-Morris-Pratt pada fase pra pencarian:

```
procedure KMP (
    input P: array[0..n-1] of char,
    input n: integer,
    input/output nextKMP: array[0..n] of integer
)

    DEKLARASI
    i,j: integer

    ALGORITMA
    i = 0
    j = nextKMP[0] - 1

    while (i < n) {
        while (j > -1 and not(P[i] = P[j])) {
            j = nextKMP[j]
        }

        i += 1;
        j += j+1;

        if (P[i] = P[j]) {
            nextKMP[i] = nextKMP[j]
        } else {
            nextKMP[i] = j
        }
```

Dan berikut ini adalah pseudo code algoritma Knuth-Morris-Pratt pada fase pencarian:

```
procedure KMPProccess(
    input m,n: integer,
    input P: array[0..n-1] of char,
    input T: array[0..m-1] of char,
    output found: array[0..m-1] of boolean
)

    DEKLARASI
    i, j, next: integer
    nextKMP: array[0..n] of integer

    ALGORITMA
    KMP(n, P, nextKMP)
    i=0

    while (i < m-n) {
        j = 0

        while (j < n and T[i+j] = P[j]) {
            j += 1
        }

        if (j >= n) {
            found[i] = true
        }
        next = j - nextKMP[j]
        i = i + next
    }
```

D. Algoritma Boyer-Moore

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

Algoritma ini dianggap sebagai algoritme yang paling efisien pada aplikasi umum.[1] Tidak seperti algoritme pencarian string yang ditemukan sebelumnya, algoritme Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritme ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Misalnya ada sebuah kecocokan 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[i+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:
    - Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
    - Semua karakter di pattern cocok. Kemudian algoritme akan memberitahukan penemuan di posisi ini.
3. Algoritme kemudian menggeser pattern dengan memaksimalkan nilai penggeseran good-suffix dan penggeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Berikut adalah pseudocode algoritme Boyer-Moore pada fase pra-pencarian:

```
procedure preBmBc(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output bmBc : array[0..n-1] of integer
)
Deklarasi:
  i: integer

Algoritme:
  for (i := 0 to ASIZE-1)
     bmBc[i] := m;
  endfor
  for (i := 0 to m - 2)
     bmBc[P[i]] := m - i - 1;
  endfor
```

```
procedure preSuffixes(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output suff : array[0..n-1] of integer
)

Deklarasi:
  f, g, i: integer

Algoritme:
  suff[n - 1] := n;
  g := n - 1;
  for (i := n - 2 downto 0) {
     if (i > g and (suff[i + n - 1 - f] < i - g))
        suff[i] := suff[i + n - 1 - f];
     else
        if (i < g)
           g := i;
        endif
        f := i;
        while (g >= 0 and P[g] = P[g + n - 1 - f])
           --g;
        endwhile
        suff[i] = f - g;
     endif
  endfor
```

```
procedure preBmGs(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output bmBc : array[0..n-1] of integer
)
Deklarasi:
  i, j: integer
  suff: array [0..RuangAlpabet] of integer

  preSuffixes(x, n, suff);

  for (i := 0 to m-1)
     bmGs[i] := n
  endfor
  j := 0
  for (i := n - 1 downto 0)
     if (suff[i] = i + 1)
        for (j:=j to n - 2 - i)
           if (bmGs[j] = n)
              bmGs[j] := n - 1 - i
           endif
        endfor
     endif
  endfor
  for (i = 0 to n - 2)
     bmGs[n - 1 - suff[i]] := n - 1 - i;
  endfor
```

Dan berikut adalah pseudocode algoritme Boyer-Moore pada fase pencarian:

```
procedure BoyerMooreSearch(
    input m, n : integer,
    input P : array[0..n-1] of char,
    input T : array[0..m-1] of char,
    output ketemu : array[0..m-1] of boolean
)

Deklarasi:
i, j, shift, bmBcShift, bmGsShift: integer
BmBc : array[0..255] of interger
BmGs : array[0..n-1] of interger

Algoritme:
preBmBc(n, P, BmBc)
preBmGs(n, P, BmGs)
i:=0
while (i<= m-n) do
    j:=n-1
    while (j >=0 n and T[i+j] = P[j]) do
       j:=j-1
    endwhile
    if(j < 0) then
       ketemu[i]:=true;
    endif
    bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
    bmGsShift:= BmGs[j]
    shift:= max(bmBcShift, bmGsShift)
    i:= i+shift
```

III. ANALISA DAN PEMBAHASAN

Pada Analisa dan Pembahasan kali ini sebelumnya saya menggunakan kembali website yang saya gunakan sebagai Tubes 3 terakhir kemarin untuk saya gunakan kembali beberapa fungsinya untuk membantu pembuatan tugas makalah kali ini.

Sebelumnya dibutuhkan sebuah database untuk menyimpan informasi yang nantinya akan kita pakai ketika user menginputkan sebuah query dalam kasus ini querry nya merupakan sebuah nama karakter dalam film The Lord of The Rings. Untuk databasenya sendiri utk memudahkan dalam pembuatan makalah saya menggunakan sebuah file yang bereksetensi .txt. Di dalam file .txt tersebut terdapat informasi-informasi berupa nama karakter dan segala atributnya (Ability, HP, Level, Skill).

Ketika user menginputkan nama karakter pada kolom input. Maka engine akan menjalankan algoritma KMP dan mencarinya pada database (dalam kasus ini file .txt). Ketika string input query user cocok pada salah satu database. Maka engine akan otomatis mengembalikan nama karakter tersebut beserta atributnya. Jika tidak ada nama karakter yang di dapat maka engine akan mengembalikan pesan "Error – [Character not Found]".

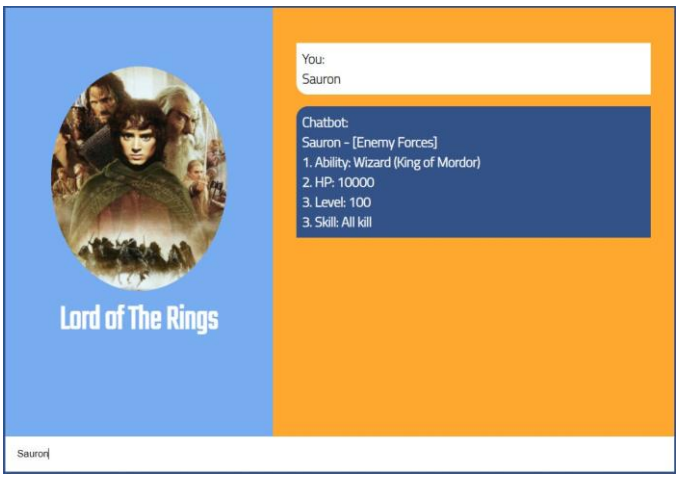
Berikut test case yang sudah saya coba:

1. User menginputkan nama karakter Aragorn.

Terlihat engine akan mengembalikan informasi terkait karakter bernama Aragorn karena engine berhasil mencocokan string input user dengan database. Serta informasi yang didapat antara lain:

Aragorn – [Allied Forces]
1. Ability: Swords, Archer
2. HP: 200
3. Level: 40
4. Skill: Fast Run

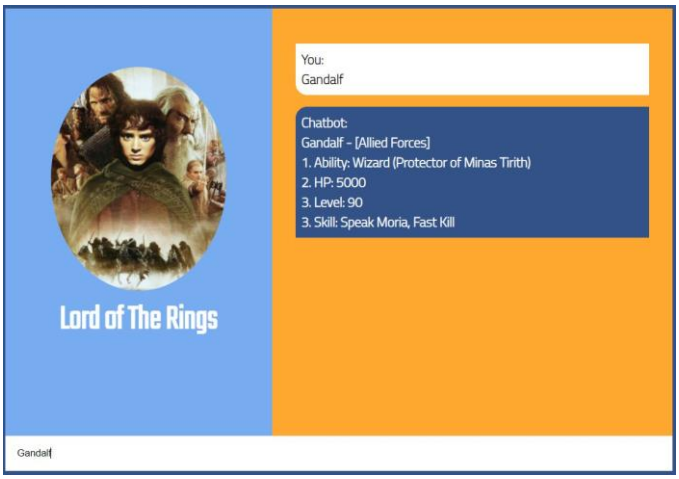
2. User menginputkan nama karakter Sauron



Terlihat engine akan mengembalikan informasi terkait karakter bernama Aragorn karena engine berhasil mencocokan string input user dengan database. Serta informasi yang didapat antara lain:

Sauron – [Enemy Forces]
1. Ability: Wizard (King of Mordor)
2. HP: 10000
3. Level: 100
4. Skill: All Kill

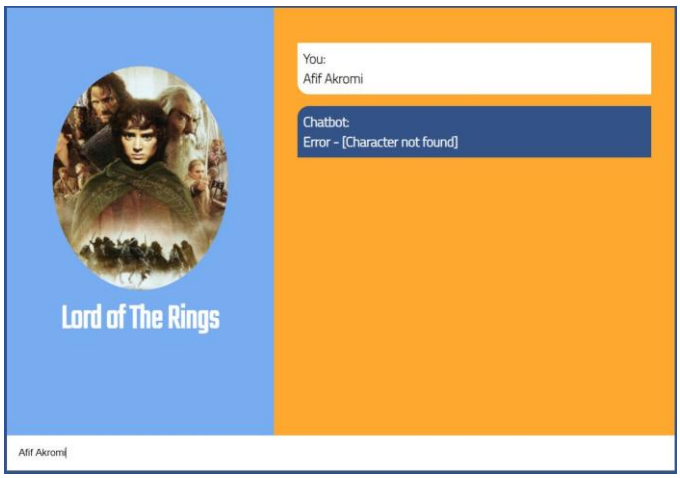
3. User menginputkan nama karakter Gandalf



Terlihat engine akan mengembalikan informasi terkait karakter bernama Aragorn karena engine berhasil mencocokan string input user dengan database. Serta informasi yang didapat antara lain:

Gandalf – [Allied Forces]
1. Ability: Wizard (Protector of Minas Tirith)
2. HP: 5000
3. Level: 90
4. Skill: Speak Moria, Fast Kill

5. User menginputkan nama karakter Afif Akromi



Terlihat engine akan mengoutputkan error jika nama karakter tidak ditemukan di dalam database.

KESIMPULAN

Algoritma KMP (Knuth-Morris-Pratt) sangatlah berguna untuk kasus pengolahan string. Pengaplikasian algoritma ini sudah banyak sekali dan sangat membantu kehidupan manusia antara lain efek yang terbsesar adalah lahirnya search engine Google dan masih banyak lain kegunaan algoritma KMP ini utk memudahkan kehidupan manusia.

Algoritma ini menemukan semua kemunculan dari pattern dengan panjang n di dalam teks dengan panjang m dengan kompleksitas waktu O(m+n). Algoritme ini hanya membutuhkan O(n) ruang dari memory internal jika teks dibaca dari file eksternal. Semua besaran O tersebut tidak tergantung pada besarnya ruang alpabet.

UCAPAN TERIMAKASIH

Penulis tidak lupa mengucapkan terima kasih serta puja dan puji syukur kehadirat Tuhan Yang Maha Esa karena karunia dan berkat rahmatnya penulis dapat menyelesaikan makalah ini dengan tepat waktu dan tanpa kurang apapun.

Tak lupa penulis juga mengucapkan terima kasih kepada seluruh tim pengajar mata kuliah IF2211 Strategi Algoritma 2021 yang telah membimbing penulis dengan sabar dan penuh kasih sayang selama satu semester ini. Penulis tidak tahu apa jadinya jika tidak mendapat bimbingan dari tim pengajar IF2211 Strategi Algoritma 2021. Kemudian penulis juga mengucapkan terima kasih kepada tim asisten mata kuliah IF2211 yang telah mengawal penulis dalam memberikan ilmu melalui tugas besar dan tugas kecil yang ada.

Ucapan terima kasih selanjutnya penulis ucapkan untuk teman-teman penulis yang memberikan dukungan penulis baik langsung maupun tidak langsung dalam segala bentuk dukungan. Terakhir, penulis mengucapkan terima kasih kepada keluarga penulis khususnya orang tua penulis yang sudah memberika support dari awal hingga saat ini yang memberikan dampak kepada proses keberjalanan dalam membuat makalah IF2211 Strategi Algoritma

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

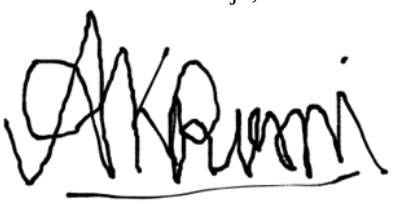
[3] https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/

[4] https://id.wikipedia.org/wiki/The_Lord_of_the_Rings_(seri_film)

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.

Sidoarjo, 19 Mei 2021



Mohammad Afif Akromi 13519110