**Laporan Tugas Kecil III**

IF2211 – Strategi Algoritma

Ekstraksi Informasi dan Artikel Berita

Algoritma Pencocokan String

Disusun oleh

Stephen Thajeb – 13518150 – K3



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

1. **Penjelasan Singkat Algoritma Pencocokan String**
2. **Algoritma Knut-Morris-Prath**

Algoritma ini melakukan pencocokan string dengan terlebih dahulu melakukan preprocessing dengan bantuan table lps (longest proper prefix which is also suffix) table. Tabel tersebut dapat diperoleh dari text pada string diuji. Dengan table tersebut, akan diketahui berapa karakter yang harus diskip saat terjadi kondisi tidak match antara string diuji dengan pattern. Worst case complexity untuk algoritma KMP adalah O(n)

1. **Algoritma Boyer Moore**

Sama seperti algoritma KMP, algoritma BM juga melakukan preprcossing yang sering disebut dengan instilah bad character heuristic. Tabel tersebut diperoleh dari pemrosesan string pattern. Tabel tersebut berisi pasangan key and value di mana key adalah semua karakter unik pada string pattern serta semua karakter lain yang tidak terdapat pada pattern sedangkan value menandakan seberapa banyak suatu shifting yang harus dilakukan ketika terjadi mismatch. Algoritma ini melakukan pencocokan string dari paling kanan dan menggeser pattern ke kiri

1. **Regular Expression**

Regular Expression adalah sekuense dari simbol yang merepresentasikan suatu format string tertentu. Berikut ditampilkan beberapa metakarakter dan flag pada regex beserta dengan penjelasannya

|  |  |
| --- | --- |
| **\** | **Awal baris** |
| **$** | **Akhir baris** |
| **\b** | **Batas kata** |
| **\B** | **Batas bukan kata** |
| **\G** | **Akhir match sebelumnya** |
| **\Z** | **Akhir dari input tapi untuk final terminator jika ada sebelumnya** |
| **\z** | **Akhir dari input final** |
| **x?** | **x muncul satu kali atau tidak sama sekali** |
| **x\*** | **x muncul nol atua banyak kali** |
| **x+** | **x muncul satu atau banyak kali** |
| **x{n}** | **X muncul tepat n kali** |
| **X{n,m}** | **X muncul antara n sampai m kali** |
| **.** | **Semua karakter** |
| **\d** | **Digit[0-9]** |
| **\s** | **Whitespace character** |
| **\w** | **Word character** |
| **\D** | **Non digit character** |
| **\S** | **Non whitespace character** |
| **\W** | **Non word character** |

1. **Kode Program**

**2.1 patternMatching.py**

|  |
| --- |
| import re  from nltk import tokenize  from collections import defaultdict  import sys  sys.path.append("../test")  def readFile(filename):      f = open(filename,"r")      text = f.read()      return text  #================================================Using BM Algorithm==================================================================  #Preprocessing  def buildLast(pattern):    last = [-1 for i in range(128)]#Initalize all char    for i in range(len(pattern)):#Revalue char that is in pattern      last[ord(pattern[i])] = i;    return last  #Return the first idx to found pattern, idx starts from 0, return -1 if not found  def bmMatch(casePattern,caseText):      pattern = casePattern.lower()      text = caseText.lower()      patternLength = len(pattern)      textLength = len(text)      table = buildLast(pattern)      if (patternLength > textLength):          return -1      for i in range(textLength-patternLength+1):          numOfSkips = 0          for j in range(patternLength-1,-1,-1):              if (pattern[j] != text[i+j]):                  if text[i+j] in table:                      numOfSkips = table[text[i+j]]                      break                  else:                      numOfSkips = patternLength                      break          if (numOfSkips==0): #pattern is found              return i      return -1  #================================================Using KMP Algorithm==================================================================  #Preprocessing  def borderFunction (pattern):      length = len(pattern)      #Initialize the border table      b = [0 for i in range(length)]      lps = 0 #variable to store previous longest prefix that is also suffix      i = 1  # b[0] is always 0, start calculating from b[1]      while (i < length):          if (pattern[i] == pattern[lps]):              lps+=1 #updating lps              b[i] = lps              i+=1          else:              if (lps!=0):                  lps = b[lps-1] #shorten the lps, since the current lps is not valid              else:                  b[i] = 0                  i +=1      return b  #return the first idx in text where pattern match subtext, idx start from 0  def kmpMatch(casePattern,caseText):      pattern = casePattern.lower()      text = caseText.lower()      patternLength = len(pattern)      textLength = len(text)      b = borderFunction(pattern)      i = 0      j = 0      while (i < textLength):          if (pattern[j] == text[i]):              if (j == patternLength-1):                  return i-patternLength+1              i +=1 # move the next character for both text and pattern              j +=1          elif (j > 0):              j = b[j-1] #move back the patten to left according to border function          else:              i +=1      return -1  # print(kmpMatch("abc","abaabaabbAbc"))    #=========================================================== Using Regex =========================================================================  # #Searching for the keyword and return the last occurance matching index  def regexMatching(pattern,sentence):      #Index version      expression = re.compile(rf"{pattern}",re.IGNORECASE)      # matchIdx = []      for match in expression.finditer(sentence):          return match.span()[0]      return -1  #=========================================================== Extracting ===================================================================  # #Extract information about the number of people that are involve or related to a certain topic or keyword  def extractNumeric(sentence):      expression = re.compile(r"(?:^(?:\d+(?:\.\d+)\*)(?:\,\d+)?(?:[\%]| ratus| ribu| juta| milyar| hundreds? | thousands? | million?s)? )|(?:\d{1,3}(?:\.\d+)\*)(?:\,\d+)?(?:[\%]| ratus| ribu| juta| milyar| hundreds?| thousands?| millions?)?[\. ,]",re.IGNORECASE)      matchIdx = []      for match in expression.finditer(sentence):          matchIdx.append(match.span())      return matchIdx  # #Extract date information  def extractDate(sentence):      expression = re.compile(r"(?:Sen(?:in)?|Sel(?:asa)?|Rabu?|Kam(?:is)?|Jum(?:at)?|Sab(?:tu)?|Minggu|Sun(?:day)?|Mon(?:day)?|Tue(?:sday)|Wed(?:nesday)?|Thur(?:sday)|Fri(?:day)?|Sat(?:urday)?)?(?:, | \()?(?:(?:[\d]{1,2}[\/-][\d]{1,2}[\/-][\d]{4})|(?:\d{1,2}(?:-\d{1,2})? (?:Jan(?:nuari)?|Feb(?:ruari)?|Mar(?:et)?|Apr(?:il)?|Mei|Juni?|Juli?|Agus(?:tus)?|Ags|Sept(?:ember)?|Sep|Okt(?:ober)?|Nov(?:ember)?|Des(?:ember)?) \d{4})|(?: yang lalu| lalu| kemarin))\)?",re.IGNORECASE)      matchIdx = []      for match in expression.finditer(sentence):          matchIdx.append(match.span())      return matchIdx  #Function to return the closest numeric or date data to a keyword  def closestToKeyword (firstIdx,wantedData,pattern):      distance = []      for position in wantedData:          if (wantedData == []):              return []          if position[0] < firstIdx:              distance.append(abs(position[1]-firstIdx))          else:              distance.append(abs(position[0]-(len(pattern)-1+firstIdx)))      return (wantedData[distance.index(min(distance))])  #Extract based on algorithm  def foundIdx(pattern,sentence,algo):      idx = -1      if(algo == "KMP"):          idx = kmpMatch(pattern,sentence)      elif (algo=="BM"):          idx = bmMatch(pattern,sentence)      else:          idx = regexMatching(pattern,sentence)      return idx  #Extrac by input multiple files in the test folder  def extractInfoFromNews(files,pattern,algo):      output = []      for single\_file in files :          news = readFile("test/"+single\_file)          sentences = tokenize.sent\_tokenize(news)          foundNewsDate = 0          for sentence in sentences:              keyword = foundIdx(pattern,sentence,algo)              if (foundNewsDate == 0):                  news\_date = extractDate(sentence)                  if (news\_date!=[]):                      news\_date = sentence[news\_date[0][0]:news\_date[0][1]]                      foundNewsDate =1              print(keyword)              if (keyword != -1):                  # print("Keyword : ",end="")                  # print(keyword)                  date = extractDate(sentence)                  numeric = extractNumeric(sentence)                  # print("Ini kalimat :",end="")                  # print(sentence)                  if (date == [] and foundNewsDate==1):                      date = news\_date                  else:                      dateRange = closestToKeyword(keyword,date,pattern)                      date = sentence[dateRange[0]:dateRange[1]]                  if (numeric != []):                      numericRange = closestToKeyword(keyword,numeric,pattern)                      numeric = sentence[numericRange[0]:numericRange[1]]                  if (numeric == []):                      numeric = "Tidak ditemukan"                  print("Ini tanggal :",end="")                  print(date)                  print("Ini jumlah :",end="")                  print(numeric)                  subSentence = re.split(pattern,sentence,flags=re.IGNORECASE)                  # print(subSentence)                  output.append((date,numeric,subSentence,single\_file))      return output |

**2.2 app.py**

|  |
| --- |
| from flask import Flask, render\_template,request,redirect,url\_for  from patternMatching import \*  app = Flask(\_\_name\_\_)  UPLOAD\_FOLDER = './test'  app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER  @app.route('/',methods=["GET","POST"])  def home():     if request.method == "GET":        return render\_template("index.html")     else:        multipleFiles = request.form.getlist("file")        # print(multipleFiles)        keyword = request.form["keyword"]        algo = request.form["algo"]        output = extractInfoFromNews(multipleFiles,keyword,algo)        print(output)        # print(output)        return render\_template("result.html",output = output,keyword=keyword,algo=algo)  @app.route('/about',methods=["GET"])  def about():     return render\_template("about.html")  if \_\_name\_\_ == '\_\_main\_\_':     app.run(debug=True) |

**2.3 index.html**

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>Info Extractor App</title>      <link rel="stylesheet" href="{{ url\_for('static', filename='css/index.css') }}">    </head>    <body>      <!-- <h1>My Info Extraction App</h1> -->      <div class="main-block">        <h1 align="center">Info Extraction App</h1>        <hr width="450px" size="1.5px">        <form action="/" method="POST">          <h3 align="center">Get the latest update about Corrona Virus !!!</h3>            <div class="block">              <label>Folder : </label>              <input type="file" name="file" multiple="" required/>            </div>              <div class="block">              <label>Keyword : </label>              <input type="text" name="keyword" value="terkonfirmasi positif" required/>            </div>              <div class="block">              <label id="lblAlgo">Algorithm : </label>                <select id="dropDownAlgo" name="algo">                  <option value="BM">Boyer Moore</option>                  <option value="KMP">Knuth Morris Prath</option>                  <option value="Regex">Regular Expression</option>                </select>            </div>              <h4 style="text-align:center">Most Popular Keyword Searched</h4>            <div class="suggestion">              <p>#Terkonfirmasi Positif #Meninggal Dunia #Indonesia</p>              <p>#ODP  #PDP #Sembuh  #Tersebar #Dikondisikan #Corona</p>              <p>#Pengobatan #Penyebab #COVID-19 #Korban #Rumah Sakit</p>          </div>              <div class="block-bottom">                <a href="/about">Perihal</a>                <span id="submit"><input type="submit" id="btn\_submit" value="Search"/></span>            </div>          </form>    </body>    </html> |

**2.4 index.css**

|  |
| --- |
| html, body {      display: flex;      justify-content: center;      height: 100%;  }  label {      display: inline-block;      width: 100px;      text-align: right;      margin-bottom: 20px;    }​  form{      margin-left: 10rem;  }  .main-block {      max-width: 520px;      max-height: 540px;      min-width: 480px;      min-height: 520px;      padding: 10px 10px;      margin: auto;      border-radius: 5px;      border: solid 1px #ccc;      box-shadow: 1px 2px 5px rgba(0,0,0,.31);      background: #ebebeb;      justify-content: center;      margin-top: 30px;  }  .block{      margin-left: 80px;      margin-bottom: 10px;      align-content: center;  }  hr{      margin-bottom: 10px;      background-color: #666;      border-style: solid;  }  #submit{      float: right;  }  .suggestion{      margin-top: 0px;      margin-bottom: 0px;      margin-left: 40px;      color: grey;  }  .block-bottom{      margin-top: 60px;  }  h3{     margin-bottom: 25px;  } |

**2.5 result.html**

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>Result</title>      <link rel="stylesheet" href="{{ url\_for('static', filename='css/result.css') }}">    </head>    <body>      <div class="main-block">        <h1 align="center">Info Extractor App</h1>        <hr width="500px" size="1.5px">        <h3 align="center">Search Result</h3>          <div class="block">            <h4>Keyword : <label class="keyword"><u>{{keyword}}</u></label>            <h4><span>{{output|length}}</span> related information found <span class="right"> <a href="/about">{{algo}} algorithm </a></span></h4>          </div>            {% for i in range(output|length): %}              <p>{{i+1}}. Waktu: <label class="date">{{output[i][0]}}</label></label> <span class="right">Jumlah: <label class="numeric">{{output[i][1]}}</label></span></p>              {% for j in range (output[i][2]|length-1):%}                  <span>{{output[i][2][j]}}</span><span class="keyword">{{keyword}}</span>              {% endfor %}              <span>{{output[i][2][output[i][2]|length-1]}}</span>              <p><i>< {{output[i][3]}} ></i></p>            <p> ---------------------------- </p>          {% endfor %}          <div class="block-bottom">          <a href="/">Search Again</a>        </div>      </div>      </body>      </html> |

**2.6 result.css**

|  |
| --- |
| html, body {      display: flex;      justify-content: center;      height: 100%;      /\* overflow: auto; \*/  }  a{   color:blue;  }  h3{      margin-bottom: 25px;   }   hr{      margin-bottom: 10px;      background-color: #666;      border-style: solid;  }  .main-block {      max-width: 680px;      max-height: 1000px;      min-width: 600px;      min-height: 640px;      padding: 10px 20px 10px 20px;      margin: auto;      border-radius: 5px;      border: solid 1px #ccc;      box-shadow: 1px 2px 5px rgba(0,0,0,.31);      background: #ebebeb;      justify-content: center;      margin-top: 30px;      overflow: auto;      /\* vertical-align: middle;    \*/  }  .block{      margin-right: 5px;      margin-left: 10px;      margin-bottom: 10px;      align-content: center;  }  .keyword{      text-emphasis-position: over;      color: blue;  }  .block-bottom{      /\* margin-top: 300px; \*/      position: relative;      bottom: 0px;  }  .right{      float: right;  }  .numeric{      color: red;  }  .date{      color:#9e7827fa;  } |

**2.7 about.html**

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>Output</title>      <link rel="stylesheet" href="{{ url\_for('static', filename='css/about.css') }}">    </head>    <body>      <div class="main-block">        <h1 align="center">Info Extraction App</h1>        <hr width="450px" size="2px">        <div class="block">          <p>            Info Extractor is an app designed to extract information about date and numeric based on a keyword.            It's purpose is to help people to quickly access latest update about Corona Virus.            Input:multiple text files(English or Indonesian,CASE-INSENSITIVE). The algorithm used in processing the extraction :            <p>              <ul>                <li><b><i>Knuth Morris Path (KMP) Algorithm</i></b>                    <p>                      This algorithm used a lps (longest proper prefix which is also suffix) table to preprocess.                      The value of the obtained from the table can determine the position of test string to rollback while a mismatch happened between the test string and the pattern                      Worst case complexity for this algorithm is O(n)                    </p>                </li>                <li><b><i>Boyer Moore (BM) Algorithm</i></b>                  <p>                    This algorithm use a bad heuristic table for preprocessing.                    The bad heuristic contained key and value pair that determined the number of shift need to be done when a mismatch is happening.                    The table is obtained from the pattern and not the text.                    This algorithm match string from to left.                  </p>                </li>                <li><b><i>Regular Expression (Regex)</i></b>                  <p>                    Regular Expression is a sequence of symbol or character or metacharacter that defines a string or a grammar for a string.                    Regular Expression is mainly used to extract the numeric and date information in this app.                  </p>                </li>              </ul>        </div>        <div class="block-bottom">          <a href="/">Home</a>        </div>      </div>    </body>  </html> |

**2.8 about.css**

|  |
| --- |
| html, body {      display: flex;      justify-content: center;      height: 100%;      /\* overflow: auto; \*/  }  .main-block {      max-width: 520px;      max-height: 600px;      min-width: 480px;      min-height: 540px;      padding: 10px 10px;      margin: auto;      border-radius: 5px;      border: solid 1px #ccc;      box-shadow: 1px 2px 5px rgba(0,0,0,.31);      background: #ebebeb;      justify-content: center;      margin-top: 28px;      overflow: auto;  }  .block{      margin-left: 10px;      margin-right: 15px;      margin-bottom: 10px;      align-content: center;      text-align: justify;  }  hr{      margin-bottom: 10px;      background-color: #666;      border-style: solid;  }  .block-bottom{      position: relative;      margin-left: 10px;      bottom: 3%;  }  p{      margin-top: 0px;  } |

1. **Screenshot Input-Ouput Program**

|  |
| --- |
|  |
| **1.Input: berita1.txt** |

|  |
| --- |
|  |
| **2.Input: jabar11042020.txt** |

|  |
| --- |
|  |
| **3.Input: berita1.txt,berita2.txt,berita3.txt** |

|  |
| --- |
|  |
| **4.Input: berita2.txt,berita3.txt** |

|  |
| --- |
|  |
| **5.Input: berita6.txt, berita7.txt, berita8.txt** |

|  |
| --- |
|  |
| **6.Input: jabar11042020.txt** |

|  |
| --- |
|  |
| **7.Input: berita7.txt, berita8.txt, berita9.txt** |

|  |
| --- |
|  |
| **8.Input: berita10.txt** |

|  |
| --- |
|  |
| **9.Input: berita9,berita10.txt** |

|  |
| --- |
|  |
| **10.Input : Nerita9.txt, berita10.txt** |

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1.Program berhasil dikompilasi | √ |  |
| 2.Program berhasil running | √ |  |
| 3.Program dapat menerima input dan output | √ |  |
| 4.Luaran sudah benar untuk semua data uji | √ |  |