The Use of Knuth Morris Pratt and Deep-First Search Algorithms on File-Searching in ITB File Transfer Protocol

Baharudin Afif Suryanugraha 13511021
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
baharudin.afif@students.itb.ac.id

***Abstract*—Searching file in File Transfer Protocol Server (FTP Server) is not a simple task to do. In fact not only problem in finding the correct file, speed retrieval is also an important aspect. In consequence of this complexity, sometimes FTP Server lacks of this feature, like majority FTP Server in Institut Teknologi Bandung (ITB). In this paper, present a technique to find file in FTP Server with effective and efficient way. Actually the main problem on file searching in FTP Server are schema in extraction of FTP Server content and the choice of pattern matching algorithms. Extraction scheme that used in this paper is using "crawler" to observe every directory in FTP Server, and create a database contain of file and directory name (one line one file name). After that, read every line in database and compare with the keyword using Knuth Morris Pratt algorithm.**

***Index Terms*—Crawler, File Searching, FTP Server, Knutt Morris Prattt.**

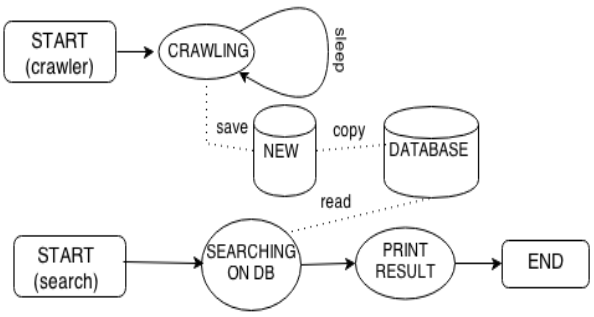
I. INTRODUCTION

File Transfer Protocol (FTP) is a populer network service beside Hyper Text Transfer Protocol (HTTP) and Mailing Service (Simple Mail Transfer Protocol – Post Office Protocol v.3/Internet Message Access Protocol). In Institut Teknologi Bandung (ITB), a lot of students using this service every day. It happen because there is a regulation that limit every students in this University on using Internet service and unlimit access to local computer network. Because of that, nowaday ITB have so many kind network service that implement locally in ITB computer network. One of them is File Transfer Protocol Server (FTP Server) that contain many kind of file, like e-book, mobile or computer application, operating sytem, repository, game, music, video, and etc. Because in ITB, majority FTP Server implement locally in ITB computer network, students can get a hold with all files as much as possible without limitation.

Today, more than 30 FTP Server active every day in ITB and every FTP Server have more than one hundred thousand files. With that size, it's a benefit for students because they can get majority file that they need, especially for courses. But the fact is all FTP Server are manage individually, no one of them have information related content of FTP Server or directory, and there is no file-searching facility. It's make students have to find file they need by opening one by one directory in FTP Server. And of course it's make students frustrated on finding appropiate file.

Actually, implement file searching facility in FTP Server is not difficult. But finding the correct file searching schema and effective pattern matching algorithm are difficult and crucial. Because schema and pattern matching algorithm we used influece the file speed retrieval, and we must make sure that speed retrieval is not consume user time too much.

Transversing every directory in FTP Server every time user running file-searching facility is not a wise way, because iterating every directory (and its content) in FTP Server need at least 100ms for every directory. But it's imposible finding the correct file without iterating every directory. Solution related to this problem is using a database of file or directory name (one line one file/directory-name) that FTP Server have, and it's a task that crawler supposed to be done. After database created, the user can do several file-searching.



Picture 1.1 Scenario for File Searching and Crawling

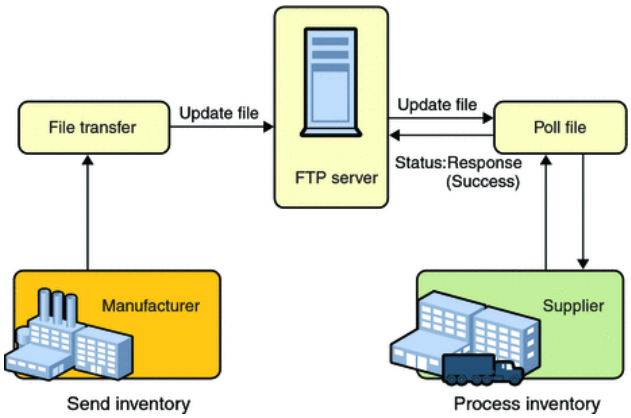
When user insert keyword and excetue file-searching, A program with Knuth Morris Pratt algorithm will try to use ™™™keyword/pattern to check wether there is/are pattern exist in spesific line and print it out. If there is a file/directory modification in FTP Server, crawler won't automatically crawling FTP Server, because crawling need time. Instead of crawling FTP Server every time file or directory modified, crawling every several time (by

using timeout or sleep) seems more wisely.

II. BASIC THEORIES

2.1. File Transfer Protocol

File Transfer Protocol known as FTP, is an internet protocol run in application layer of OSI model. FTP is a standart protocol for file transfer between two or more machine in computer network. Computer or machine that act as file provider is known as FTP Server and the one that consume or donwload file is known as FTP Client. FTP have an unsecure file transfer, because they just transfer the data without doing some encryption first. FTP using text mode (in ASCII) to transfer data. And because there is no encryption, we can see what username and password by analyze protocol using sniffer tools.



Pictures 2.1 FTP Schema

FTP using TCP/IP protocol and running on port 20 and 21. Port 20 use for data transfer and port 21 for interact with FTP client. Before establish a connection, FTP Server trying to listen to any machine that trying to connect to FTP Server IP Address on port 21. After connection establish, the role of this port changed from acting as listener to acting as control port for sending response to FTP Client and accepting request from FTP Client. Once connection establish, FTP Server will open port 20, and trying to connect with FTP Client, because all data transfering (downloading and uploading) will running on this port. FTP Server can create an authorization related with directory that FTP Client can access, number of user that accessing directory at the same, limit download and upload speed for every user, and etc.

Nowadays, there are many kind of file transfer protocol beside FTP. For example SFTP or Secure Shell FTP that also known as FTPS (FTP over SSL). SFTP is a modification of FTP. In all aspect SFTP is the same with FTP except security aspect. Because SFTP based on Secure Shell or using SSL that make FTP Server encrypt the data before transfering into FTP Client. Therefore there is no machine/computer can retrieve username and password from protocol analyzing or sniffing tools.

2.2 String Matching Algorithm

String Matching Algorithm is an algorithm that we can use to determine whether there is a pattern inside of a text or group of code.

2.3.1 Knuth-Morris-Pratt Algorithm

Knuth-Morris-Pratt (also known as KMP) algorithm is an String Matching algorithm that have better algorithm complexity than other string matching algorithm, like brute force algorithm. KMP Algorithm check the pattern from the left side to the right side, like what brute force done. But here, KMP use smarter way in shifting the pattern. If missmatch found in pattern P at P[j], brute force will shifting one character to the right. But KMP won't do the same thing, firstly KMP will count n, the largest prefix of P[1..j-1] that is a suffix of P[1..j-1]. After finding the value of n, KMP will shift the pattern to the right n times. For example, we have pattern "abbaabbaab", and then we count the largest part of pattern that have prefix of P[1..j-1] equal with suffix [1..j-1], string "abbaab" is the answer. It happen because we can see the largest prefix here: "abbaabbaab" and largest suffix here : "abbaabbaab". After that, KMP will count the number of character in the largest suffix or prefix, largest suffix is "abbaab" and it have 6 character. And then KMP will shift the pattern as length as largest suffix or six character.

Before starting pettern matching, KMP save all the number of shifting in every character of pattern have, from the spesified character in pattern if missmatch happen. All the number is store in a table that produced by Border Function. For the example, above pattern "abbaabbaab", Border Function will produce a table look like this:

pattern	b(k)	Index / k
a	0	1
b	0	2
b	0	3
a	1	4
a	1	5
b	2	6
b	3	7
a	4	8
a	5	9
b	6	10

Table 2.3.1 Border Function result for pattern "abbaabbaab"

Where "k" denotes the character number or index of character in spesified pattern and b(k) is the result of Border Function where index equal to "k". As we see that the largest result of border function is located at character

10, but KMP will not use this value to shift, because KMP algorithm only use character in the first index untill j-1, where j is the length of string pattern.

Below are prosedure of Border Function Algorithm and KMP Algorithm :

```
procedure BorderFunction(input m : integer, P : array[1..m]) of char, output b: array[1..m] of integer)
{ Counting the value of b[1..m] for pattern P[1..m]}
```

```
DECLARATION

    k,q : integer
```

```
ALGORITHM

    b[1] ← 0
    q ← 2
    k ← 0
    for q←2 to m do
        while ((k>0) and (P[q] ≠ P[k+1]) do
            k←b[k]
        endwhile
        if P[q] = P[k+1] then
            k←k+1;
        endif
        b[q]=k
    endfor
```

```
Function KMP(input text : array[0..n-1] of char, input pattern : array[0..m-1] of char) -> integer
```

```
DECLARATION

{assumtion:
  1. n is known as text length
  2. m is  known as pattern length }
    index,i,j : integer
    border : array [0..m-1] of integer
```

```
ALGORITHM

    index ← -1
    i ← 0
    j ← 0
    BorderFunction(m,pattern, border)
    while(i<n) do
        if pattern[j]=text[i]
            if j=m-1 {reacing the end of pattern}
                index ← i-m +1
                break
            i← i+1
            j←j+1
        else if j>0
            J←border[j-1]
        else
            i←i+1
        endif
    endwhile
    -> index
```

2.3 Deep-First Search

Deep-First Search also known as DFS is one of tree transversing algorithm, beside Bread-First Search (BFS) and Interative-Deep Search (IDS). DFS idea is transversing a node untill its reach the leaf node, and then its move transversing neighbour node, they do it recursively untill all node are known. To do this, DFS algorithm will keep track of all node that have known also the unknown one, and keep the track of how we got to the neighbour node. Below is the algorithm that can we do to trasversing a tree systematically.

```
procedure DFS(input G : graph)
{ This procedure will inisialize status on every node, and taking an action whether a node have visited or not}
```

```
DECLARATION

    Col  : array of integer
    Pred : array of integer
    U : integer
```

```
ALGORITHM

    for each U on V do
        {Set all node as unknown node}
        Col[U] ← 0
        {node status (have been viseted or not) }
        Pred[U]← -1;
    {set predecessor to unknown}
    endfor
    for each U on V do
        if Col[U] = 0 then
            DFSTransverse(U, Col, P)
            {trying to transverse node}
        endif
    endfor
```

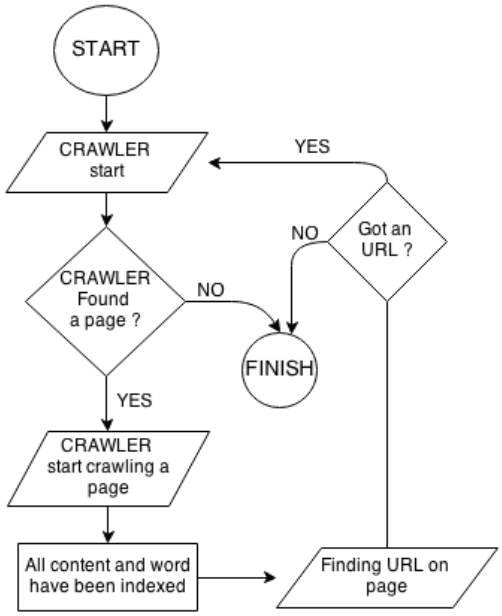
```
procedure DFSTransverse(input U :integer, input/output C : array of integer, P : array of integer)
{This procedure will transverse the child of a node and change the status of a node}
```

```
DECLARATION

    V : integer
```

```
ALGORITHM
    C[U] ← 1;
    for each V in BrachfOf(U)
        if C[V]=0 then
            P[V] = U
            DFSTransverse(U,C,P)
    endif
    C[U] ← 1
```

2.4 Web Crawler

Web crawler also known as Spider is a program that almost all search engine use to find whether there is a new file or file modification in Internet. For example GoogleBot, a web crawler that google have. All of the content of Internet or Web will eventually be found and spidered. Search engine may run thousands or more isntances web crawling program at the same time, on multiple server that they have. When a web crawler reached a pages, it will loads all the content first and then it will send the content to the database. Once a page load successfully, web crawler will load all the text to the search engine's index, which actually is a big data. Below the schema or procedure involved web crawling.



Picture 2.4.1 Web Crawler Schema

Actually there is a task that Crawler supposed to be done first before starting crawling entire web page, that is looking for file called "robot.txt". They do this because this file contains intructions and regulation for Web Crawler. One of the regulation is limitation related page that Crawler can crawl and Crawler cannot crawl. And all Web Crawler supposed to be follow the rule, and the fact is majority search engine obey these rule for the most part, like google and bing.

III. IMPLEMENTATION

FTP Server contains directories and files. And every directory and file could have the same name, but every directory or filename could not have the same path name or URI (Uniform Resource Identifier). For example, we want to have a directory named "download" with URI value is at ftp://ftp.itb.ac.id/pub/download/, and another directory named "download" with URI value at ftp://ftp.itb.ac.id/download/, that scenario could be implemented on FTP Server, because both of them have different URI value. But we cannot implement if the second "download" directory also have URI value at ftp://ftp.ac.id/pub/download. Because they have same name and URI, if we force doing this, the directory will merge into one directory named "download".

The uniqueness of file or directory have can we use created some searching mechanism. File-searching in FTP Server can be done using Web Crawler to serve database of unique filename and directory name also using KMP algorithm to give fast string matching response to user. Schema that we will use is like pictures 1.1 shown above. First of all Crawler create a database of unique filename and directory name, if crawling done, Crawler will sleep / timeout for several time. After database created, this is the time for KMP algorithm to iterate every line of database and match it's string with pattern. If match found, KMP print out the result in the form of URI.

Crawler, actually is a simple program or bot to seek every pages on website. If we talk about FTP, we will not see pages or site, but what we see is files and directories. And if it's about seeking directory, Deep-First-Search algorithm seems powerful on doing this task. So this paper will create a FTP Crawler using Deep-First-Search algorithm. Below is the modification of DFS algorithm so that its can crawling every node (files or directories) in FTP Server.

```
procedure FTPCrawler(input baseURI, databasePath:
                     string, output db : FILE)
{This procedure is trying to seek all node (file and folder)
in FTP Server, if node not leaf (folder) found, this
procedure will open it and seek it first until reach leaf.
Every time this procedure seek a node (can be file or
directory), this procedure will print out its node URI into
database.}
```

```
DECLARATION

    StreamReader reader {use for receiving response in
stream from FTP Server}
    int i
    string URIvalue
```

```
ALGORITHM

    i=0;
    CreateFTPRequest(baseURI)
    {Trying to connect to FTP Server with address equal to
```

```
baseURI}
    CreateNetworkCredential("anonymous",
"baharudin.afif@s.itb.ac.id")
  {create anonymous ID}
  try
      GetResponseFromServer(reader)
  {trying accepting response from server}
      i++
      if !(reader.EndOfSteam) then
          URIvalue= reader.ReadLine()
          {parse from stream to string}
          if i>2 then
{the first (i=0) and second(i=1) URI that FTP Server
send is shortcut to "reopen this directory" and "go to
parent directory", its can make infinite loop, so its
ignored}
              printToFile(URIvalue, db)
              {print the value of "name" to database}
              if isDirectory(URIvalue) then
                  FTPCrawler(URIvalue,databasePath,db)
                  {DFS}
              endif
          endif
      endif
  catch exception E
      write(E)
  {Print error message to monitor}
```

After creating the database, the next step is file-searching problem. A problem to find whether there is a file or directory that have name which match with pattern or keyword. This paper will use KMP algorithm to find matching file or directory, because its one of the best string mathing algorithm. Below is the implementation of KMP algorithm so that it can use to check whether there is matching URI (files/directories) or not. Related with border table of keyword that inserted by user, its can created using Border Function Algorithm above.

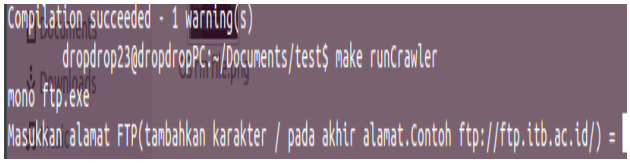
```
procedure SearchOnDB(input keyword : string)
{This procedure will try to read string every line in
database, and then compare it with keyword using KMP
algorithm
KMP() : function to compare two string, if its match it
        will return first match character, if its not match
        it will return -1}
```

```
DECLARATION

  line : string
  i : int
```

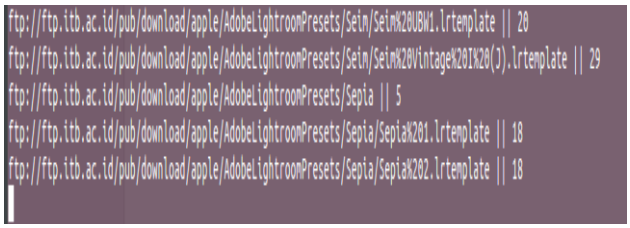
```
ALGORITHM

  try
      while (line=ReadLine()) != null do
          {if database exist and contain URI}
              {Neglect upper case}
              ConvertLineToLower(line)
          {try comparing keyword string and string that exist
in database}
          if KMP(keyword, line) ≠ -1 then
          {match}
              PrintOut(line)
          endif
          endWhile
    catch Exception E
          PrintOut(E)
```

IV. TESTING

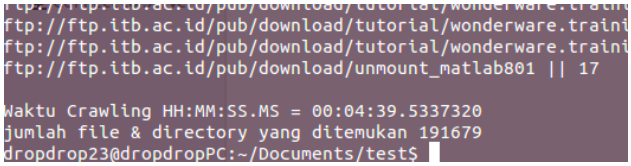
In order to test correctness of this metodology, some scenario are done to the program that using above algortihm. First program is FTPCrawler, that is a proram that implement Crawler to transverse the content of a FTP. And below is the result of FTPCrawler program.



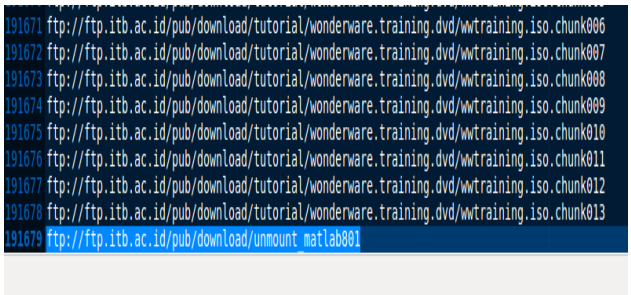
Picture 4.1 FTPCrawler asked for FTPServer address



Picture 4.2 FTPCrawler start crawling content of FTPServer



Picture 4.3 FTPCrawler finish crawling

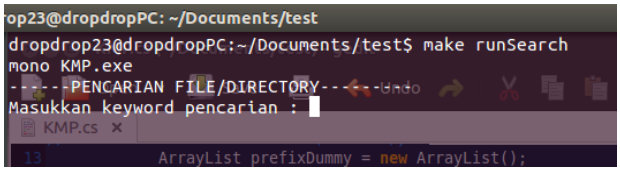


Picture 4.4 Database Content

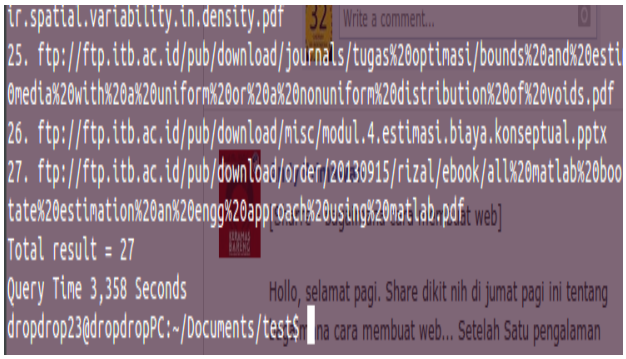
First time, we insert FTPServer address, after that program will start crawling the content of FTPServer.

When its finish, program will print out crawling time, total of file and directory that found. From Picture 4.3, it shows that crawling is very time consuming. Crawling ftp://ftp.itb.ac.id/pub/download/ take around 4 minutes. Therefore this program will not crawling every time file or directory on FTPServer modified, but waiting for a period of time.

The second is FTPSearch program. This program will read every line on database an compare its string with keyword string. Below are the screenshot of program execution.



Picture 4.5 FTPSearch asking for keyword



Pictures 4.6 FTPSearch result for keyword = "STIMA"

27 Result for keyword "STIMA" are found in ftp://ftp.itb.ac.id/pub/download with query time around 3 seconds using KMP algorithm, and thats seems not satisfying enough. The link on that result are clickable, if link clicked, its will open web browser and open directory (if link refer to directory) or downloading a file (if link refer to file).

V. CONCLUSION

We can create a file-searching mechanism in FTP Server using DFS algorithm to collect the data and KMP algorithm to compare file/directory name inside FTP Server with keyword that user inserted. But eventhough KMP is one of the best string matching algorithm, we still have to concern with speed retrieval. Remember that KMP algorithm in the example above used to query about 191679 item from one FTP Server. The fact is ITB have more than 30 FTP Server. Therefore we still have to create ideas related with speed retrieval, like using cache on file-search. And running Crawler to recrawl all FTP Server every timeout passed seems not a wise way. Maybe using log file to save every file or directory operation that user done to the content of FTP Server, and process it to create a limitation to file and directory that observed by FTP Crawler.

REFERENCES

[1] Filetransferplanet.com, *FTP Guides.* December 20, 2013 (00.25 AM) < http://www.filetransferplanet.com/ftp-guides-resources/>

[2] H Cormen, Thomas, "Introduction To Algorithm, 3rd Edition". Cambridge : MIT Press, 2009 , pp.1002-1011.

[3] Indiana University Southeast. *Deep-First Search.* Decmeber 20, 2013 (11.13 AM). < http://homepages.ius.edu/rwisman/C455/html/notes/Chapter22/DFS.htm>

[4] Munir, Rinaldi, "Diktat Kuliah IF2211 Strategi Algoritma". Bandung, 2009, pp. 190–193.

[5] Tanenbaum, "Computer Network, 5th Edition. New Jersey : Prentice Hall, 2011, pp. 47.

[6] The University of Texas. *KMP String Matching Example*. December 20, 2013 (00.35 AM) < https://www.cs.utexas.edu/~moore/best-ideas/string-searching/kpm-example.html>

[7] Siteground, *FTP Tutorial*. December 20, 2013 (00.24 AM)< http://www.siteground.com/tutorials/ftp/>

[8] Wpthemesplanet. *How does a Web Crawler work?*. December 19, 2013 (10.23 PM)< http://www.wpthemesplanet.com/2009/09/how-does-web-crawler-spider-work/ >

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah benar tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan pula hasil plagiasi.

Bandung, 20 Desember 2013

Baharudin Afif S - 13511021