Pencarian Berkas dan Direktori pada Sistem Berkas Hierarkis Menggunakan Algoritma *Breadth-First Search* dan *Depth-First Search*

Makalah IF2211 Strategi Algoritma

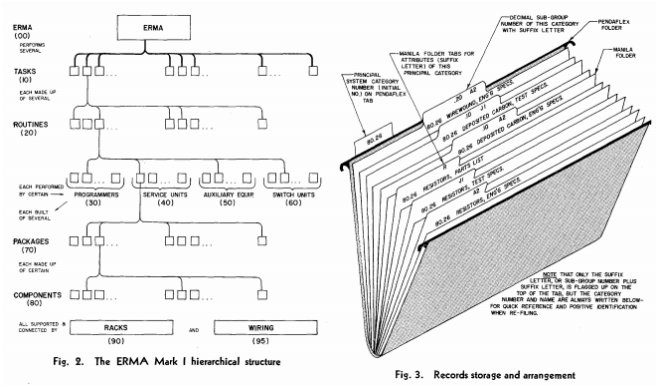
Tasya Lailinissa Diandraputri 13517141[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13517141@std.stei.itb.ac.id

***Abstrak*—Sistem operasi merupakan perangkat lunak yang sangat esensial pada setiap perangkat komputer dan laptop. Salah satu fungsi utama dari sistem operasi adalah melakukan pengelolaan dan pengendalian data masukan atau keluaran, termasuk pengalokasian memori pada penyimpanan sekunder atau memori utama dan penyimpanan sekunder. Setiap sistem operasi memiliki suatu sistem berkas. Salah satu sistem berkas yang umum digunakan suatu sistem operasi adalah sistem berkas hierarkis (*hierarchical file system*). Sistem berkas ini dapat direpresentasikan sebagai graf yang tidak mengandung sirkuit atau (*tree*). Saat menggunakan komputer atau laptop, pencarian suatu berkas atau direktori seringkali diperlukan. Pencarian tersebut termasuk pencarian pada graf sehingga dapat diselesaikan dengan algoritma *breadth first search* (pencarian melebar) dan *depth first search* (pencarian mendalam). Makalah ini membahas penggunaan algoritma *breadth-first search* dan *depth-first search* dalam pencarian suatu berkas atau direktori sistem operasi. Makalah ini disusun berdasarkan hasil studi pustaka. Dari hasil studi pustaka dan pengujian, algoritma *breadth-first search* dan algoritma *depth-first search* merupakan metode yang cocok dalam pencarian berkas dan direktori pada sistem berkas hierarkis. Kedua algoritma ini tidak selalu menghasilkan hasil yang sama terhadap input yang sama.**

Kata kunci—berkas; breadth-first search; depth-first search; direktori; pencarian

I. PENDAHULUAN

Sistem operasi yang digunakan pada perangkat komputer dan laptop memiliki sistem berkas yang digunakan untuk menyimpan berkas pengguna. Sistem berkas (*file system*) merupakan struktur logika untuk pengelolaan, penyimpanan, manipulasi, dan pengambilan data dari suatu memori. Terdapat banyak jenis sistem berkas. Pada umumnya, sistem operasi pada perangkat komputer dan laptop menggunakan sistem berkas hirarkis (*hierarchical file system*) yaitu sistem berkas yang dapat bercabang ketika terdapat suatu direktori lain. Dalam direktori tersebut, dapat disimpan berkas ataupun direktori. Hal ini mempermudah pengorganisasian berkas sehingga berkas tersimpan dengan teratur dan mudah diakses. Sistem berkas ini dapat direpresentasikan sebagai graf yang tidak memiliki sirkuit atau pohon (*tree*).



Gambar 1 Sistem berkas hierarkis
Sumber: [1]

Salah satu hal yang sering dibutuhkan oleh pengguna adalah mencari suatu berkas atau direktori tertentu yang terletak pada sistem berkas. Karena sistem berkas ini dapat direpresentasikan sebagai graf, algoritma pencarian dalam graf dapat digunakan untuk pencarian berkas ataupun direktori pada sistem berkas tersebut. Algoritma pencarian dalam graf yang dapat digunakan untuk permasalahan ini antara lain adalah algoritma *breadth-first search* dan *depth-first search*. Kedua algoritma ini merupakan algoritma yang umum digunakan dalam penelusuran graf.

Konsep dari kedua algoritma tersebut mirip namun berbeda. Perbedaan yang paling signifikan terhadap keduanya adalah urutan pencariannya. Algoritma *breadth-first search* memiliki arti pencarian melebar sehingga pemrosesan grafnya akan dengan cara melebar, sedangkan, algoritma *depth-first search* memiliki arti pencarian mendalam sehingga pemrosesan grafnya akan dengan mendalami suatu simpul hingga ke ujungnya.

II. DASAR TEORI

A. *Algoritma Breadth-First Search*

Algoritma *breadth-first search*/BFS (pencarian melebar) termasuk algoritma pencarian *uninformed*/*blind search* yaitu pencarian tanpa informasi tambahan. Pencarian solusi dengan algoritma *breadth-first search* dapat dilakukan dengan pendekatan graf statis ataupun graf dinamis.

Pencarian dimulai dari suatu simpul pada graf. Sesuai dengan namanya, pencarian dilakukan melebar dengan cara mengunjungi simpul yang bertetangga dengan simpul akar. Hal tersebut dilakukan pada simpul yang dikunjungi selanjutnya (simpul anak-anaknya) dan berlanjut ke suksesornya sampai simpul yang dicari ditemukan atau semua simpul telah dikunjungi.

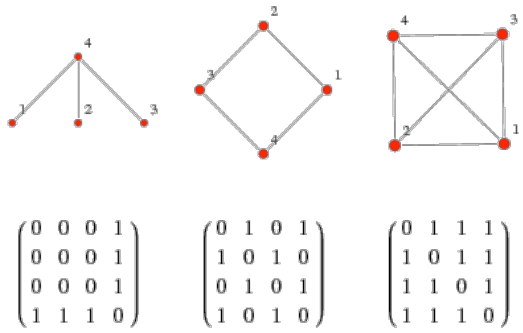
Struktur data yang digunakan algoritma *breadth-first search* adalah sebagai berikut.

1. Tabel *boolean* dengan elemen sebanyak *n* yaitu jumlah simpul. Tabel ini menyimpan informasi simpul yang telah dikunjungi maupun yang belum dikunjungi. Tabel ini diinisialisasi *false* karena sebelum pencarian, semua simpul belum dikunjungi. Jika simpul telah dikunjungi, elemen pada tabel diisi *true*. Deklarasi dan inisialisasi tabel *boolean* tersebut adalah sebagai berikut.

```
visited: array[1..n] of boolean
for i ← 1 to n do
      visited[i] ← false
```

2. Matriks ketetanggaan yang berukuran *n* × *n* dengan *n* sebagai jumlah simpul. Elemen matriks pada baris ke-*i* dan kolom ke-*j* bernilai 1 jika simpul *i* dan simpul j bertetangga, sedangkan 0 jika simpul *i* dan simpul j tidak bertetangga. Deklarasi matriks ketetanggaan tersebut adalah sebagai berikut.

```
adjacency: array[1..n] of array[1..n]
of integer
```



Gambar 2 Contoh matriks ketetanggaan dari graf
Sumber:
http://mathworld.wolfram.com/AdjacencyMatrix.html

3. Antrian yang menyimpan simpul yang akan dibangkitkan. Deklarasi antrian tersebut adalah sebagai berikut.

```
queue: array[1..n] of integer
```

Langkah-langkah yang digunakan algoritma *breadth-first search* adalah sebagai berikut.

1. Masukkan simpul akar ke dalam antrian. Jika simpul akar merupakan simpul solusi (*goal node*), solusi ditemukan dan pencarian dihentikan.
2. Jika antrian kosong, solusi tidak ditemukan dan pencarian dihentikan.
3. Kunjungi simpul anak-anak dari kepala (*head*) antrian. Jika simpul tersebut tidak mempunyai anak, kembali ke langkah kedua. Masukkan simpul anaknya pada antrian.
4. Jika suatu simpul anak tersebut merupakan simpul solusi, solusi ditemukan dan pencarian dihentikan. Jika tidak, kembali ke langkah kedua.

Pseudocode algoritma *breadth-first search* adalah sebagai berikut.

```
procedure BFS (input v: integer, input w: integer)
{ Pencarian simpul graf dengan algoritma breadth-first
search dari simpul v ke simpul w }
KAMUS
    i, x: integer
    q: queue
    found: boolean
    visited: array[1..n] of boolean
    procedure MakeEmptyQueue(input/output q: queue)
    { Membuat antrian kosong, head(q) diisi 0 }
    procedure Add(input/output q: queue, input v: integer)
    { Memasukkan v ke dalam antrian q menjadikan v elemen
    terakhir q }
    procedure Del(input/output q: queue)

    { Menghapus kepala antrian q }
    function IsEmptyQueue(input q: queue) → boolean
    { Mengembalikan true jika antrian q kosong dan false
    jika sebaliknya }
ALGORITMA
    found ← false
    for i ← 1 to n do
        visited[i] ← false
    endfor
    MakeEmptyQueue(q) { Membuat antrian kosong }
    write(v) { Cetak simpul akar }
    visited[v] ← true { Tandai simpul akar menjadi telah
    dikunjungi }
    Add(q, v) { Memasukkan simpul akar ke dalam antrian }
    { Kunjungi simpul graf selama solusi belum ditemukan
    dan antrian belum kosong }
    while not found and not IsEmptyQueue(q) do
        Del(q) { Menghapus simpul yang telah dikunjungi
        dari antrian }
        for tiap simpul x yang bertetangga dengan v do
            if not visited[x] then
                write(x)
                Add(q, x)
                visited[x] ← true
            endif
        endfor
        if x = w then
            found = true
        endif
    endwhile { found or IsEmptyQueue(q) }
```

Kompleksitas waktu dari algoritma *breadth-first search* adalah eksponensial yaitu $1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$ dengan b merupakan *branching factor* atau maksimum percabangan yang mungkin dari suatu simpul dan d merupakan depth atau kedalaman dari solusi. Kompleksitas ruangnya adalah $O(b^d)$.

B. Algoritma Depth-First Search

Seperti algoritma *breadth-first search*, algoritma *depth-first search*/DFS (pencarian mendalam) merupakan pencarian tanpa informasi tambahan atau *uninformed*/*blind search*. Pencarian solusi dengan algoritma *depth-first search* dapat dilakukan dengan pendekatan graf statis ataupun graf dinamis. Pencarian dilakukan secara rekursif.

Pencarian dimulai dari suatu simpul pada graf. Pencarian dilakukan mendalam dengan cara mengunjungi simpul yang bertetangga dengan simpul akar. Lalu, simpul yang dikunjungi tersebut diproses dengan *depth-first search*. Ketika semua simpul yang bertetangga dengannya telah dikunjungi, terjadi pencarian runut-balik (*backtrack*). Proses tersebut dilakukan sampai solusi ditemukan atau semua simpul telah dikunjungi.

Struktur data yang digunakan algoritma *depth-first search* adalah sebagai berikut.

1. Tabel *boolean* dengan elemen sebanyak *n* yaitu jumlah simpul. Tabel ini menyimpan informasi simpul yang telah dikunjungi maupun yang belum dikunjungi. Tabel ini diinisialisasi *false* karena sebelum pencarian, semua simpul belum dikunjungi. Jika simpul telah dikunjungi, elemen pada tabel diisi *true*. Deklarasi dan inisialisasi tabel *boolean* tersebut adalah sebagai berikut.

```
visited: array[1..n] of boolean
for i ← 1 to n do
      visited[i] ← false
```

2. Matriks ketetanggaan yang berukuran $n \times n$ dengan n sebagai jumlah simpul. Elemen matriks pada baris ke-i dan kolom ke-j bernilai 1 jika simpul i dan simpul j bertetangga, sedangkan 0 jika simpul i dan simpul j tidak bertetangga. Deklarasi matriks ketetanggaan tersebut adalah sebagai berikut.

```
adjacency: array[1..n] of array[1..n]
of integer
```

Langkah-langkah yang digunakan algoritma *depth-first search* adalah sebagai berikut.

1. Masukkan simpul akar ke dalam antrian. Jika simpul akar merupakan simpul solusi (*goal node*), solusi ditemukan dan pencarian dihentikan.
2. Simpul diproses secara rekursif. Jika solusi ditemukan, pencarian dihentikan.
3. Jika semua simpul anak telah dikunjungi, terjadi pencarian runut-balik (*backtrack*) ke simpul terakhir yang mempunyai simpul anak. Kembali ke langkah kedua.

Pseudocode algoritma *depth-first search* adalah sebagai berikut.

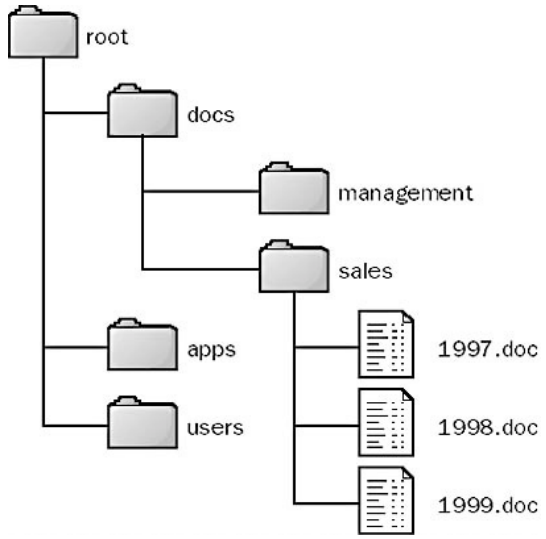
```
procedure DFS (input v: integer, input w: integer)
{ Pencarian simpul graf dengan algoritma depth-first
search }
KAMUS
    x: integer
    visited: array[1..n] of boolean { Tabel boolean
    penanda simpul yang telah dikunjungi }
    adjacency: array[1..n] of array[1..n] of integer {
    Matriks ketetanggaan }
ALGORITMA
    for i ← 1 to n do
        visited[i] ← false
    endfor
    write(v)
    visited[v] ← true
    x ← 1
    while x ≤ n and x ≠ w do
        if adjacency[v, x] = 1 then { simpul v dan simpul
        x bertetangga }
            if not visited[x] then
                DFS(x, w) { Rekurens }
            endif
        endif
        x ← x + 1
    endwhile { x > n or x = w }
```

Sama seperti algoritma *breadth-first search*, kompleksitas waktu dari algoritma *depth-first search* adalah eksponensial. Kompleksitas waktunya adalah $O(b^m)$ dengan b merupakan *branching factor* atau maksimum percabangan yang mungkin dari suatu simpul dan m merupakan maksimum kedalaman. Kompleksitas ruangnya adalah $O(bm)$.

III. Analisis Masalah dan Implementasi Program

Sistem berkas hierarkis dapat direpresentasikan dengan graf. Representasinya membentuk graf yang tidak memiliki sirkuit sehingga dinamakan pohon (*tree*). Komponen-komponen pada representasi graf sistem berkas hierarkis adalah sebagai berikut.

1. Akar pada pohon merupakan representasi dari direktori akar atau *directory root*. Direktori ini menyimpan semua berkas dan direktori yang berada dalam sistem berkas. Direktori ini juga disebut alur mutlak (*absolute path*) dari sebuah berkas atau direktori dalam sebuah sistem berkas. Pada sistem operasi Windows dan MS-DOS, memiliki lebih dari satu direktori akar (*multiple root directory*), sedangkan sistem operasi macOS hanya memiliki satu direktori akar.
2. Cabang pada pohon merupakan representasi dari direktori lain selain direktori akar. Direktori ini pada umumnya berbentuk *folder* yang dapat dimasukkan berkas ataupun direktori lain.
3. Daun pada pohon merupakan representasi dari berkas. Berkas yang disimpan dapat memiliki ekstensi yang berbeda-beda. Dalam suatu direktori, dapat ditemukan banyak berkas. Jumlah berkas maksimum ditentukan oleh memori yang terdapat pada perangkat tersebut.



Gambar 3 Representasi graf pada sistem berkas hierarkis
Sumber: [3]

A. Pencarian Berkas dan Direktori pada Sistem Berkas Hierarkis Menggunakan Algoritma Breadth-first Search

Langkah-langkah algoritma *breadth-first search* pada sistem berkas hierarkis adalah sebagai berikut.

1. Masukkan direktori akar ke dalam antrian. Jika direktori akar merupakan direktori yang dicari, solusi ditemukan dan pencarian dihentikan.
2. Jika antrian kosong, solusi tidak ditemukan dan pencarian dihentikan.
3. Kunjungi simpul (direktori atau berkas) dari kepala (*head*) antrian. Jika simpul tersebut tidak mempunyai anak, kembali ke langkah kedua. Masukkan simpul anaknya pada antrian.
4. Jika suatu simpul anak tersebut merupakan simpul solusi, solusi ditemukan dan pencarian dihentikan. Jika tidak, kembali ke langkah kedua.

Dengan menggunakan langkah-langkah algoritma *breadth-first search*, *pseudocode* implementasi program pencarian berkas dan direktori pada sistem berkas hirarkis adalah sebagai berikut.

```
procedure BFSDirSearch (input nama: string, )
{ Pencarian path nama file atau direktori dengan algoritma
breadth-first search dari direktori akar }
KAMUS
    i: integer
    root: string
    q, path: queue
    found: boolean
    visited, done: array[1..n] of boolean
    procedure MakeEmptyQueue(input/output q: queue)
    { Membuat antrian kosong, head(q) diisi "" }
    procedure Add(input/output q: queue, input v: string)
    { Memasukkan v ke dalam antrian q menjadikan v elemen
    terakhir q }
    procedure Del(input/output q: queue)
    { Menghapus kepala antrian q }
    function IsEmptyQueue(input q: queue) → boolean
    { Mengembalikan true jika antrian q kosong dan false
    jika sebaliknya }
ALGORITMA
    found ← false
    for i ← 1 to n do
        visited[i] = false
    endfor
    root ← "root"
```

Program implementasi pencarian berkas dan direktori dengan algoritma *breadth-first search* diuji dengan pengujian sebagai berikut.

```
root
├─desktop
│   └─tubes
│      └─Main.java
├──documents
│   └─Main.java
│
└─app
   ├─games
   │  ├─2048
   │  ├─Mahjong
   │  ├─Minesweeper
   │  └─Tetris
   ├─iMovie
   └─Safari
```

```
BFS Search Directory
Input: Main.java
Path:
root
documents
Main.java
```

Gambar 4 Hasil pengujian program pencarian berkas dan direktori dengan algoritma *breadth-first search*

B. Pencarian Berkas dan Direktori pada Sistem Berkas Hierarkis Menggunakan Algoritma Depth-first Search

Langkah-langkah algoritma *depth-first search* pada sistem berkas hierarkis adalah sebagai berikut.

1. Masukkan simpul akar ke dalam antrian. Jika simpul akar merupakan simpul solusi (*goal node*), solusi ditemukan dan pencarian dihentikan.
2. Simpul diproses secara rekursif. Jika solusi ditemukan, pencarian dihentikan.
3. Jika semua simpul anak telah dikunjungi, terjadi pencarian runut-balik (*backtrack*) ke simpul terakhir yang mempunyai simpul anak. Kembali ke langkah kedua.

Dengan menggunakan langkah-langkah algoritma *depth-first search*, *pseudocode* implementasi program pencarian berkas dan direktori pada sistem berkas hierarkis adalah sebagai berikut.

```
procedure DFSDirSearch (input nama: string)
{ Pencarian path nama file atau direktori dengan algoritma
depth-first search dari direktori akar }
KAMUS
    x: integer
    visited: array[1..n] of boolean
    adjacency: array[1..n] of array[1..n] of integer
ALGORITMA
    for i ← 1 to n do
        visited[i] ← false
    endfor
    write(v)
    visited[v] ← true
    x ← 1
    while x ≤ n and x ≠ w do
        if adjacency[v, x] = 1 then { simpul v dan simpul
        x bertetangga }
            if not visited[x] then
                DFS(x, w) { Rekurens }
            endif
        endif
        x ← x + 1
    endwhile { x > n or x = w }
```

Program implementasi pencarian berkas dan direktori dengan algoritma *depth-first search* diuji dengan pengujian yang sama dengan pengujian yang digunakan pada program implementasi dengan algoritma *breadth-first search*.

```
root
├─desktop
│   └─tubes
│       └─Main.java
├──documents
│   └─Main.java
│
└─app
  ├─games
  │ ├─2048
  │ ├─Mahjong
  │ ├─Minesweeper
  │ └─Tetris
  ├─iMovie
  └─Safari
```

```
DFS Search Directory
Input: Main.java
Path:
root
desktop
tubes
Main.java
```

Gambar 5 Hasil pengujian program pencarian berkas dan direktori dengan algoritma depth-first search

C. Analisis Hasil Pengujian

Pada algoritma *breadth-first search* (pencarian melebar), semua simpul pada direktori akar (*root directory*). Setelah itu, pencarian dilanjutkan dengan mengecek simpul anak dari setiap simpul pada direktori akar. Proses tersebut dilakukan sampai berkas atau direktori yang dicari ditemukan. Jika semua simpul telah dikunjungi namun tidak menemukan berkas atau direktori yang dicari, berkas atau direktori yang dicari tidak terdapat pada sistem berkas dan program menampilkan pesan tidak ditemukannya berkas yang dicari.

Pada algoritma *depth-first search* (pencarian mendalam), satu simpul pada direktori akar (*root directory*) dan pencarian dilanjutkan mendalam pada direktori tersebut sampai simpul tidak memiliki anak lagi. Saat sampai pada simpul tanpa anak, program melakukan runut-balik (*backtrack*) sampai kepada simpul yang memiliki anak yang belum dikunjungi. Sama seperti pada algoritma *breadth-first search*, proses tersebut dilakukan sampai berkas atau direktori yang dicari ditemukan. Jika semua simpul telah dikunjungi namun tidak menemukan berkas atau direktori yang dicari, berkas atau direktori yang dicari tidak terdapat pada sistem berkas dan program menampilkan pesan tidak ditemukannya berkas yang dicari.

Hasil eksekusi dua program yang menggunakan dua algoritma yang berbeda yaitu *breadth-first search* dan *depth-first search* dengan pengujian yang sama menghasilkan hasil yang berbeda. Hal ini dikarenakan simpul (berkas atau direktori) pada sistem berkas unik hanya pada direktorinya sendiri. Hal ini memperbolehkan terdapatnya nama berkas atau direktori yang sama, tetapi pada direktori yang berbeda Dengan diperbolehkannya hal tersebut, pencarian menggunakan dua strategi algoritma berbeda dapat menghasilkan dua hasil yang berbeda. Sistem berkas menyimpan berkas atau direktori dengan *path*. Dapat dipastikan *path* setiap berkas atau direktori unik karena tidak ada berkas atau direktori pada direktori yang sama dan nama yang sama.

IV. KESIMPULAN

Kesimpulan yang dapat diambil dari makalah ini adalah sistem berkas pada perangkat komputer dan laptop yang umumnya digunakan adalah sistem berkas hierarkis (*hierarchical file system*). Sistem berkas tersebut dapat direpresentasikan sebagai pohon (*tree*) yang memungkinkan pencarian berkas dan direktori pada sistem berkas tersebut menggunakan algoritma pencarian dalam graf. Algoritma yang dapat digunakan adalah algoritma *breadth-first search* dan algoritma *depth-first search*. Dengan konsep yang berbeda, dua algoritma tersebut dapat memberikan hasil yang berbeda terhadap masukan yang sama. Hal tersebut ditunjukkan pada hasil pengujian dalam makalah ini.

Perbedaan hasil yang dihasilkan algoritma *breadth-first search* dan algoritma *depth-first search* terhadap masukan yang sama disebabkan oleh nama berkas atau direktori sama tetapi dalam direktori yang berbeda. Nama *path* setiap berkas atau direktori merupakan nama yang unik secara universal.

UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kepada Allah SWT karena atas izin, pertolongan, berkat, rahmat, dan hidayah-Nya, penulis dapat menyelesaikan tugas makalah ini dengan lancar. Pada kesempatan kali ini, dengan segala kerendahan hati, penulis ingin menyampaikan ucapan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., Ibu Drs. Nur Ulfa Maulidevi, dan Ibu Dr. Masayu Leylia Khodra, S.T., M.T. selaku dosen Mata Kuliah IF2211 Strategi Algoritma atas segala pembelajaran dan bimbingan yang telah diberikan dalam bentuk apapun. Penulis juga ingin menyampaikan terima kasih kepada kedua orang tua yang selalu memberikan dukungan dan memberikan doa kepada penulis. Ucapan terima kasih juga penulis haturkan kepada teman-teman yang telah memberikan bantuan dan dukungan dalam penyusunan makalah ini.

REFERENSI

[1] Barnard, III, G. A. dan Fein, L.. 1958. "Organization and retrieval of records generated in a large-scale engineering project". New York: Ampex Corporation, Redwood City, Calif.

[2] Munir, Rinaldi. 2009. *Diktat Kuliah IF2211 Strategi Algoritma*. Bandung: Institut Teknologi Bandung.

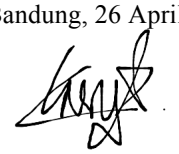
[3] Pinto, Miguel. 2010. "file system". http://www.thenetworkencyclopedia.com/entry/file-system/. Diakses pada tanggal 24 April 2019 pukul 21:22 WIB.

[4] Silberschatz, Avi. 2012. *Operating System Concepts Ninth Edition*. Connecticut: John Wiley & Sons, Inc.

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, 26 April 2019



Tasya Lailinissa Diandraputri 13517141