Penggunaan Algoritma *Branch and Bound* dalam Permasalahan Penjadwalan Disk pada Sistem Operasi

Muhammad Reza Irvanda 13512042[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13512042@std.stei.itb.ac.id

***Abstraksi* — Penjadwalan disk merupakan persoalan yang penting di dalam sebuah sistem operasi. Hal ini akan berpengaruh pada kecepatan akses dan penggunaan disk secara mangkus. Ada 6 algoritma yang umumnya digunakan dalam melakukan penjadwalan disk. Pada makalah ini, akan diperlihatkan penggunaan algoritma Branch and Bound pada permasalah penjadwal disk untuk melihat apakah algoritma ini bisa menjadi algoritma yang lebih baik daripada algoritma umum yang biasa digunakan dalam penjadwalan disk.**

***Istilah Kunci* —Branch and Bound, Penjadwalan Disk, Antrian, Jarak Akses.**

I. PENDAHULUAN

A. Penjadwalan Disk

Sistem operasi merupakan salah satu komponen penting yang ada dalam sebuah komputer. Sistem operasi membantu komputer dalam melakukan penggunaan perangkat keras secara lebih mangkus. Sistem operasi juga menjadi dasar dari semua perangkat lunak yang akan diimplementasikan pada sebuah komputer.

Melihat pentingnya peranan sebuah sistem operasi pada sebuah komputer, membuat banyak para informatikawan terus mencoba untuk melakukan peningkatan kualitas sistem operasi. Salah satu tugas dari sistem operasi adalah melakukan penjadwalan disk.

Penjadwalan disk adalah sebuah serangkaian metode yang diatur oleh sistem operasi dalam melakukan akses terhadap sebuah disk. Hal utama yang diatur oleh penjadwalan disk adalah urutan dalam melakukan akses terhadap disk-disk yang diminta dalam sebuah operasi. Penjadwalan disk yang baik akan meningkatkan kecepatan akses dan jumlah data yang bisa ditransfer[1].

B. Algoritma dalam Penjadwalan Disk

Silberschatz ,dalam bukunya[2], menjelaskan enam algoritma yang biasanya digunakan dalam melakukan penjadwalan disk. Keenam algoritma tersebut adalah algoritma FCFS, SSTF, SCAN, LOOK, C-SCAN, C-LOOK.

1. FCFS
   Algoritma *First Come First Serve* menggunakan pendekatan antrian. Dengan cara ini, urutan disk yang diakses akan sesuai dengan urutan masukknya permintaan dari disk tersebut. Cara ini merupakan cara paling alami dalam melakukan penjadwalan disk. Secara umum, cara ini tidak memberikan cara terbaik dalam mengakses disk.

2. SSTF
   Algoritma *Shortest-Seek-Time-First* memilih permintaan disk yang lokasinya paling dekat dengan disk yang sekarang sedang diproses. SSTF menggunakan prinsip *greedy* , yakni mengambil jarak terdekat dalam menentukan harapan dengan harapan jarak total yang akan ditempuh pada akhirnya menjadi jarak terpendek. Cara ini bisa saja menyebabkan *starvation* pada sebuah disk. Hal ini terjadi ketika saat mengakses sebuah disk, permintaan dari disk terbaru terus datang dekat dengan lokasi proses disk saat ini. Hal ini menyebabkan permintaan dari disk yang cukup jauh, lama untuk ditangani meskipun permintaan sudah ada jauh sebelum permintaan-permintaan baru yang lebih dekat.

3. SCAN
   Algoritma SCAN sering disebut sebagai algoritma *elevator*. Hal ini sesuai dengan prinsip algoritma ini, dimana permintaan yang akan diakses adalah permintaan terdekat yang berada dalam arah bekerjanya proses disk. Dengan kata lain, jika urutan akses disk sedang berada dalam keadaan terus menaik (lokasi lebih tinggi menjadi sasaran proses berikutnya), maka akses akan terus mengurut naik hingga mencapai ujung lokasi disk. Setelah itu, barulah akses dilakukan secara menurun. Hal ini berlaku juga jika dalam proses awalnya, akses cenderung dilakukan secara menurun.

4. LOOK
   Algorima LOOK mirip dengan algoritma SCAN. Perbedaan kedua algoritma ini terletak di akhir pengaksesan. Jika pada algoritma SCAN, akses harus dihabiskan hingga mencapai ujung lokasi disk, pada algoritma LOOK, akses cukup dilakukan hingga permintaan terluar. Hal ini menjadikan algoritma ini menjadi relatif lebih baik daripada algoritma SCAN karena tidak adanya jarak sia-sia akibat keharusan untuk mengakses hingga ujung lokasi disk.

5. C-SCAN
   Algoritma C-SCAN (*Circular* SCAN) merupakan pengembangan dari algoritma SCAN. Pada SCAN, jika proses sudah mencapai lokasi disk yang berada di ujung, proses akan berbalik. Hal ini tidak terjadi pada C-SCAN. Pada algoritma ini, arah tetap dipertahankan dengan membuat sebuah mekanisme *warp*. Jika proses sudah mencapai ujung lokasi disk, maka proses akan berpindah ke ujung lainnya (jarak tidak dihitung dalam mekanisme ini) dan melanjutkan proses sesuai dengan arah awal pemrosesan permintaan disk. Hal ini menghemat jarak yang terbuang sia-sia karena harus berbalik arah pada algoritma SCAN.

6. C-LOOK
   Algoritma C-LOOK (*Circular* LOOK) merupakan pengembangan algoritma LOOK. Perubahan yang terjadi mirip dengan perubahan dari SCAN menjadi C-SCAN.

Keenam algoritma di atas lazim digunakan dalam melakukan penjadwalan disk pada sistem operasi. Dengan melihat kemangkusan penjadwalan berdasarkan jarak yang ditempuh pada saat melakukan urutan-urutan pemrosesan, algoritma yang sudah dijelaskan dirasa masih kurang mangkus karena tidak berlaku secara umum. Masing-masing algoritma bisa mangkus di suatu kasus, namun menjadi tidak mangkus jika diterapkan pada kasus lain. Di dalam makalah ini, pemakalah mencoba untuk menggunakan algoritma *Branch and Bound* guna memenuhi kemangkusan di setiap kasus uji coba.

C. Algoritma Branch and Bound

Algoritma *Branch and Bound (BnB)* adalah algoritma yang dikembangkan berdasarkan algoritma pencarian melebar (*Breadth First Search* atau BFS). Algoritma BnB memiliki ruang solusi yang diorganisasikan dalam bentuk pohon ruang status. Dalam bukunya[3], Rinaldi (2009 : 151) menyatakan :

Algoritma BnB mempercepat pencarian solusi dengan memilih simpul hidup yang menyatakan nilai ongkos(cost). Setiap simpul hidup diasosiasikan dengan sebuah ongkos yang menyatakan nilai batas (bound). Untuk setiap simpul X, nilai batas ini dapat berupa[HOR78]:

(a) Jumlah simpul dalam upapohon X yang perlu dibangkitkan sebelum simpul solusi ditemukan atau

(b) Panjang lintasan dari simpul X ke simpul solusi terdekat (di dalam upapohon X ybs)

Dalam melakukan pilihan terhadap simpul hidup mana yang akan dikunjungi, simpul hidup diurutkan berdasarkan nilai ongkosnya terurut membesar. Nilai ongkos dengan nilai yang lebih kecil akan menjadi prioritas dalam melakukan pencarian solusi. Setiap ada simpul hidup yang dihidupkan, simpul-simpul akan kembali diurutkan.

Algoritma BnB sering digunakan dalam persoalan-pesoalan Permainan 15-*Puzzle, Travelling Salesperson Problem* , dan persoalan N-Ratu.

II. Implementasi dan Percobaan

A. Definisi Kasus

Kasus yang akan coba diselesaikan, sesuai dengan penjelasan sebelumnya, adalah persoalan penjadwalan disk. Dalam percobaan ini, algoritma tidak langsung diimplementasikan dalam mesin komputer, melainkan mensimulasikan melalui berbagai contoh kasus berdasarkan letak permintaan proses disk.

Batasan (*constrain*) dalam percobaan ini adalah penyederhanaan definisi kemangkusan dalam penjadwalan disk. Dalam kasus ini, kemangkusan hanya diukur berdasarkan jadwal total yang ditempuh dengan masing-masing algortima yang digunakan.

B. Persiapan Implementasi

1. Generator Bilangan

Dalam menentukan contoh kasus uji, digunakan sebuah generator kasus uji yang akan menghasilkan sebuah antrian bilangan (menyatakan posisi disk yang mengirimkan permintaan) sebanyak 20 buah secara acak dengan bilangan terbesar yang mungkin adalah 100. Adapun pseudo-code untuk generator ini adalah sebagai berikut.

```
procedure generator(output
Q:priority_queue , input
bilTerbesar:integer )

KAMUS
Max : integer
i : integer
tmp : integer

ALGORITMA
Max ← 20
i traversal [1...20]
      tmp ← random(bilTerbesar)
      Q.add(tmp)
end traversal
```

2. Perubah Ke Matriks

Jarak masing-masing lokasi disk terhadap lokasi disk lainnya dinyatakan dalam sebuah matriks 20x20 yang dihasilkan dari antrian yang sudah di-*generate* mnggunakan Generator Bilangan. Adapun pseudo-code untuk peubah ke matriks ini adalah sebagai berikut.

```
procedure keMatriks(output M:
Matriks[20][20] , input Q :
priority_queue )

KAMUS
Max : integer
i,j : integer
tmp : integer

ALGORITMA
Max ← 20
i traversal [1...20]
      j traversal [1...20]
      if(i!=j)
            tmp ← |Q[i]-Q[j]|
            M[i][j] ←tmp
      else
            M[i][j] ← -1
      end traversal
end traversal
```

3. Matriks Tereduksi

Dalam menentukan nilai ongkos dari sebuah simpul, kita menggunakan nilai ongkos matriks dari matriks tereduksi. Matriks tereduksi merupakan matriks yang didapatkan dengan mengurangi elemen-elemen matriks dengan nilai minimal dalam sebuah baris dan nilai minimal dalam sebuah kolom. Adapun pseudo-code penghitung nilai cost berdasarkan matriks tereduksi ini adalah sebagai berikut,

```
function
ReduksiMatriks(input/output M :
Matriks[20][20],)→ integer

KAMUS
i : integer
tmp,min : integer

ALGORITMA
tmp ←0
i traversal [1...20]
      min ← CariMinBaris(M,i)
      tmp ← tmp + min
      ReduksiBarisMatriks(M,i,min)
end traversal

i traversal [1...20]
      min ← CariMinKolom(M,i)
      tmp ← tmp + min
      ReduksiKolomMatriks(M,i,min)
end traversal

   → tmp
```

4. Implementasi Algoritma *Branch and Bound*

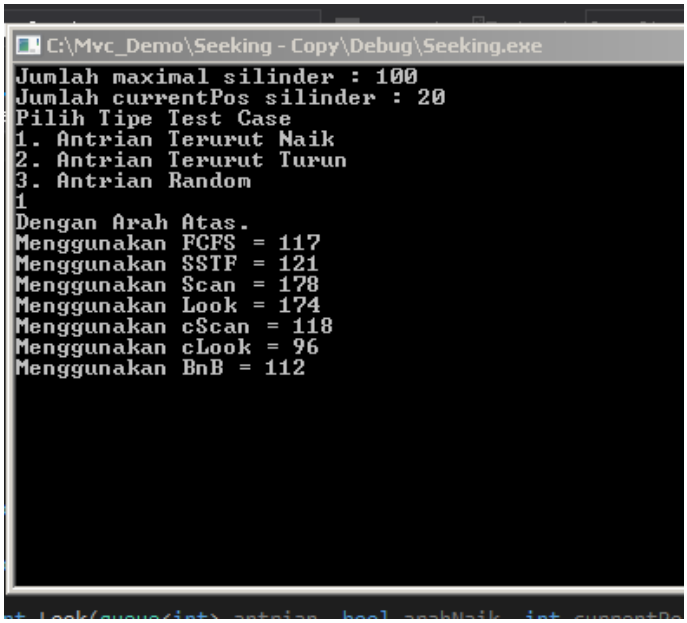
Algortima BnB yang akan digunakan diimplementasikan dalam program dengan dasar pseudo-code sebagai berikut.

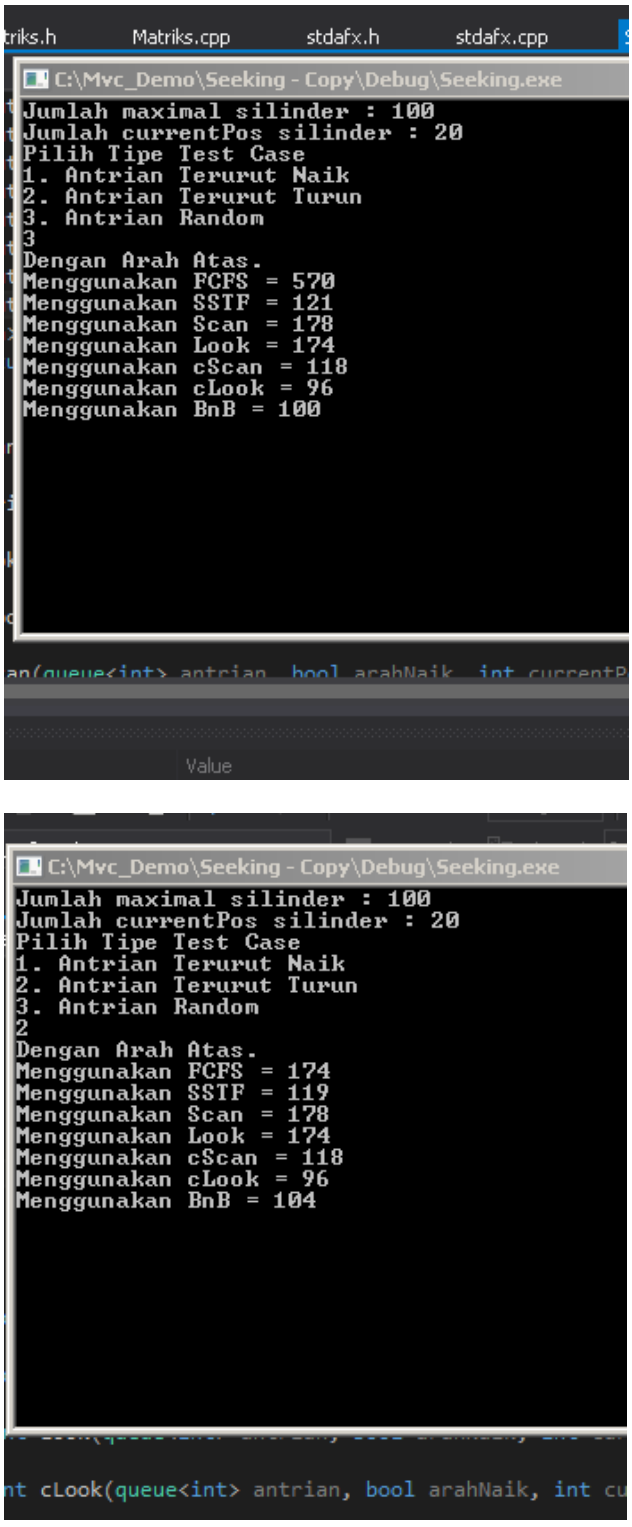
```
function MainAlgoritma(input M:
Matriks[20][20])→ integer

KAMUS
i,j : integer
jarak : integer
hasil : vector<int>
q1 : priority_queue //untuk menyimpan
masing-masing cost dan matriks dari simpul
hidup
ALGORITMA
M.ReducedMatriks();
q1.push(M);
while(!finish){
        M = q1.top();
        q1.pop();
        for(int i=0;i<M.getRow();i++){
              if(M.p[i][0]==INF){
        //sudah dikunjungi
              }
              else{
              Matriks nM(m);
              NSimpul++;
        nM.MakeAnak(nM.simpulTerakhir(),i);
              q1.push(nM);
              finish=nM.isFinished();
              }
        }
hasil ← q1.top.getUrutanTitik()
HitungJarak(M,hasil)
```

C. Percobaan

Setelah mempersiapkan kode program yang mengimplementasikan masing-masing pseudo-code di atas, didapatkan hasil sebagai berikut.





Gambar 1. Hasil Percobaan

D. Hasil Percobaan dan Analisis

Berdasarkan percobaan yang telah dilakukan, didapatkan hasil seperti pada Gambar 2. Ternyata, penggunaan algoritma *Branch and Bound* tidak memberikan hasil yang signifikan berupa pengurangan jarak yang ditempuh oleh proses penjadwalan disk. Algoritma *Branch and Bound* selalu kalah mangkus dibandingkan dengan algoritma C-LOOK.

Dalam hemat pemakalah, hal ini bisa terjadi karena C-LOOK memiliki mekanisme *warp* yang memungkinkan terjadinya pemangkasan jarak antar dua buah lokasi disk yang mengirimkan permintaan.

Algoritma *Branch and Bound* ternyata tidak bisa menjadi algoritma yang mangkus di semua contoh kasus. Dari contoh kasus di atas, algoritma C-LOOK yang berhasil menjadi algoritma yang terus mangkus.

Percobaan yang direkomendasikan untuk menguji algoritma *Branch and Bound* dari hasil yang didapatkan pada kali ini adalah dengan menambahkan mekanisme warp pada algoritma *Branch and Bound*. Hal ini bisa menjadi penunjang untuk algoritma BnB untuk mewujudkan kemangkusan yang diinginkan.

Selain itu, relatif dibandingkan dengan algoritma yang lain, waktu proses dengan algoritma Branch and Bound relatif lebih lama sehingga algoritma ini justru memiliki nilai kurang tersendiri dalam penggunaan sebagai algoritma dalam penjadwalan disk.

III. Kesimpulan dan Saran

Pada percobaan ini dapat disimpulkan bahwa algoritma *Branch and Bound* tidak cukup mangkus untuk dijadikan algoritma dalam melakukan penjadwalan disk. Algoritma *Branch and Bound* masih kalah mangkus dari algoritma C-LOOK.

Adapun, dari percobaan ini, bisa dilihat keunggulan algoritma C-LOOK karena adanya pemangkasan jarak karena mekanisme warp. Hal ini mungkin bisa diimplementasikan pada algoritma *Branch and Bound* untuk melihat apakah dengan ditambahkannya mekanisme ini, algoritma *Branch and Bound* bisa menjadi algoritma yang tepat dalam melakukan *disk scheduling*.

IV. Istilah

1. BFS : Breadth First Search, algoritma dalam melakukan pencarian solusi. Pendekatan yang digunakan adalah dengan melakukan pencarian melebar (pencarian dengan tingkatan sama) terlebih dahulu sebelum melakukan pecarian yang mendalam.
2. *Warp:* istilah yang digunakan untuk menunjukkan adanya pemotongan jarak atau waktu. Biasanya dijelaskan dalam film-film *Science-Fiction*.

Referensi

[1] Silberschatz, Abraham.2013.*Operating System Concepts 9th Edition.*New Jersey : Jhon Wiley and Sons. Hal 472 – 473.
[2] __. Hal 472 – 473.
[3] Munir, Rinaldi.2009.*Diktat Kuliah IF2211 Strategi Algoritma*. Program Studi Teknik Informatika ITB.
[4] B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.

[5] E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.
[6] J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.
[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

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, 19 Mei 2014

Muhammad Reza Irvanda
13512042