Penerapan Algoritma *Brute force* dan *Greedy* pada Penjadwalan Disk

Abraham Krisnanda Santoso – 13510033
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510033@std.stei.itb.ac.id

***Abstract*—Komputer sudah menjadi salah satu kebutuhan primer di masa ini bagi beberapa kalangan. Semakin hari, kebutuhan pengguna semakin meningkat. Komputasi yang dilakukan akan semakin banyak dan kompleks. Sistem operasi harus dapat mengontrol penggunaan perangkat keras dan lunak secara efisien. Salah satu perangkat keras nya adalah *disk drives*. *Disk drives* dituntut untuk memiliki akses waktu yang cepat dan bandwidth yang besar untuk memenuhi permintaan I/O. Disinilah diperlukan algoritma penjadwalan disk agar dapat terjadi peningkatan kecepatan akses dan bandwidth. Ada banyak algoritma yang digunakan dalam penjadwalan disk. Beberapa algoritma penjadwalan disk yang sudah umum menerapkan algoritma *brute force* dan algoritma *greedy*. Penerapan kedua algoritma tersebut akan dibahas pada makalah ini.**

Kata kunci*—Penjadwalan disk, *brute force, greedy

I. PENDAHULUAN

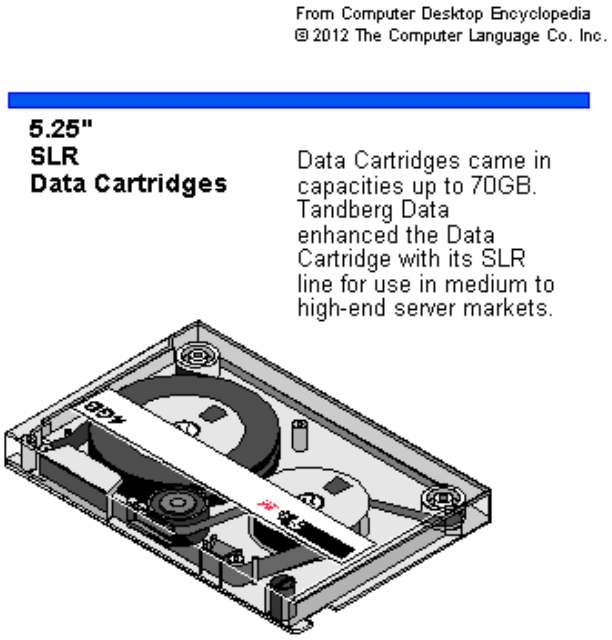
1.1 Pengertian Penjadwalan Disk

Komputer sudah menjadi salah satu kebutuhan primer di masa ini bagi beberapa kalangan. Semakin hari, kebutuhan pengguna semakin meningkat. Komputasi yang dilakukan akan semakin banyak dan kompleks. Data yang disimpan maupun diolah akan terus berkembang. Sistem operasi harus dapat mengontrol penggunaan perangkat keras dan lunak secara efisien untuk memenuhi setiap kebutuhan pengguna.

Pembahasan mengenai sistem operasi dikerucutkan ke dalam penyimpanan data. Penyimpanan data di dalam sistem komputer diatur oleh *file-system*, yang merupakan komponen dari sistem operasi. *File-system* terbagi dari tiga bagian :

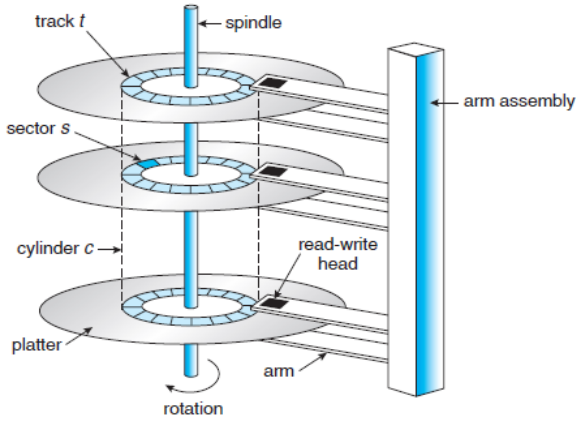
1. Antar muka pengguna dan programmer terhadap *filesystem*.
2. Struktur data internal dan algoritma yang digunakan sistem operasi dalam mengimplementasikan *filesystem*.
3. Struktur *mass-storage*.

Secara fisik data disimpan di dalam *mass-storage*. Pada beberapa waktu yang lalu, ketika komputer baru pertama kali ditemukan, media *mass-storage* yang digunakan adalah *magnetic tape*. *Magnetic tape* dapat menyimpan data dalam jumlah besar dan permanen. Namun permasalahannya adalah waktu yang dibutuhkan untuk mengakses data sangatlah lambat.



Gambar 1 : Salah satu contoh *magnetic tape*

Sedangkan media *mass-storage* yang sering dipakai sekarang adalah *magnetic disks*. *Magnetic disks* terdiri dari berbagai plat disk seperti CD. Kedua sisi dari plat tersebut diselimuti oleh material magnetik. Data yang disimpan diletakkan pada material magnetik tersebut. Pembacaan dan penulisan pada *magnetic disks* menggunakan *read-write head*. *Head* terletak pada *disk arm* yang dapat menggerakkan *head*. Setiap permukaan pada plat disk dibagi menjadi beberapa *track*, yang dibagi lagi menjadi beberapa sektor. Kumpulan *track* akan membentuk sebuah bentuk silinder. Ketika disk sedang digunakan, kumpulan plat disk akan diputar. Untuk membaca data pada lokasi tertentu, *disk arm* akan digerakkan menuju silinder yang tepat. *Seek time* adalah Waktu yang dibutuhkan *disk arm* untuk mencapai silinder yang diinginkan.



Gambar 2 : Mekanisme pembacaan disk

Kebutuhannya adalah disk harus dapat diakses dengan cepat dan memiliki *bandwidth* yang besar. Waktu akses memiliki dua komponen, yaitu *seek time* dan *rotational latency*, yaitu waktu tambahan untuk *head* dalam mengakses sektor. Dalam pembahasan ini, waktu *rotational latency* diabaikan. *Bandwidth* adalah banyak nya byte yang dapat ditransfer berbanding dengan selisih waktu antara permintaan transfer data pertama kali dan permintaan transfer data terakhir. Waktu akses dan *bandwidth* dapat ditingkatkan dengan mengatur urutan I/O yang akan diproses terlebih dahulu.

II. Dasar Teori

2.1 Algoritma *Brute Force*

Algoritma *brute force* merupakan algoritma yang sangat sederhana. Penyelesaian masalah berdasarkan pernyataan masalah dan definisi konsep yang dilibatkan. Algoritma *brute force* secara umum tidak cerdas atau tidak mangkus, karena dibutuhkan langkah yang besar dalam menyelesaikan masalah. Meskipun tidak mangkus, algoritma *brute force* dapat diterapkan dalam semua permasalahan. Dalam masalah yang sederhana / kecil, algoritma *brute force* sangat cocok untuk digunakan.

2.2 Algoritma *Greedy*

Algoritma *greedy* merupakan algoritma yang populer dalam memecahkan persoalan optimasi. Algoritma *greedy* membentuk solusi langkah per langkah. Setiap langkah solusi menghasilkan banyak pilihan yang harus dieksplorasi, sehingga harus diambil keputusan terbaik. Keputusan terbaik yang diambil tidak dapat diubah lagi pada langkah sebelumnya. Algoritma *greedy* seakan-akan memberikan perolehan terbaik. Karena dengan membuat pilihan optimum lokal diharapkan akan mengarah ke solusi optimum global.

III. Penerapan Algoritma Brute Force

Salah satu algoritma penjadwalan yang paling sederhana adalah FCFS (First Come First Serve). Algoritma ini dapat dikatakan algoritma yang 'adil'. Proses yang memberikan permintaan akses I/O akan didahulukan yang memberikan permintaan pertama. Tetapi algoritma ini bukanlah algoritma yang paling efisien.

Algoritma FCFS pada penjadwalan disk merupakan penerapan dari algoritma brute force. Algoritma FCFS bisa direpresentasikan sebagai persoalan antrian. Setiap permintaan I/O akan dimasukkan ke dalam antrian. Setiap elemen yang ditemukan akan langsung diproses dengan menggerakkan *disk arm* ke elemen tersebut.

Pseudo-code algoritma :

```
procedure FCFS (input a : queue)
Deklarasi
i : integer
Algoritma
while (a tidak kosong) do
  pindahkan disk head ke posisi HEAD(a)
  hapus HEAD(a) dari antrian a
end while
{antrian sudah kosong, semua permintaan telah dipenuhi}
```

Sebagai contoh, antrian permintaan akses I/O blok silinder adalah sebagai berikut :

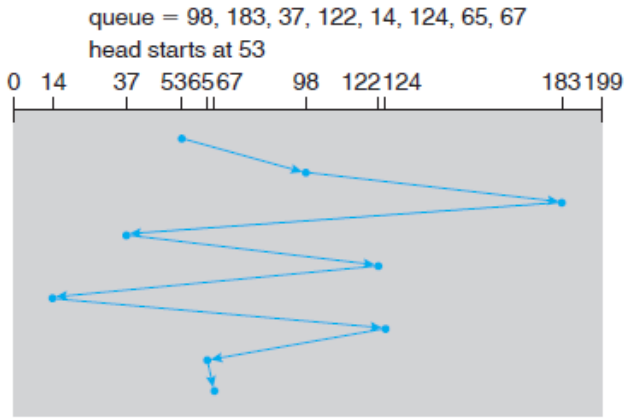
98,183,37,122,14,124,65,67

Misalkan *disk head* terletak pada silinder 53, maka *disk head* akan bergerak dengan urutan silinder seperti ini :

53 → 98 → 183 → 37 → 122 → 14 → 124 → 65 → 67

Langkah	Node			Cost
1	53	-->	98	45
2	98	-->	183	85
3	183	-->	37	146
4	37	-->	122	85
5	122	-->	14	108
6	14	-->	124	110
7	124	-->	65	59
8	65	-->	67	2
			total	640

seek time : 640 silinder.



Gambar 3 : Penjadwalan FCFS

IV. PENERAPAN ALGORITMA GREEDY

4.1. Penjadwalan SSTF

Algoritma penjadwalan disk yang lain adalah SSTF (shortest-seek-time-first). Algoritma SSTF memilih permintaan yang memiliki selisih *seek time* paling sedikit dari posisi head. Penjadwalan SSTF menerapkan algoritma greedy. Persoalan penjadwalan SSTF dapat dibentuk sebagai persoalan mencari jarak terpendek dari yang melalui setiap node hanya sekali. Strategi *greedy* yang digunakan untuk memilih node yang akan dikunjungi : Pada setiap langkah, pilih node yang belum pernah dikunjungi yang mempunyai jarak terdekat.

Misalkan terdapat sejumlah *n* node yang merepresentasikan antrian permintaan I/O. Lalu dihitung cost yang diperlukan untuk berpindah antar node.

Sebagai contoh, antrian permintaan akses I/O blok silinder adalah sebagai berikut :

98,183,37,122,14,124,65,67

Misalkan *disk head* terletak pada silinder 53, cost antar node dalam antrian akan terlihat seperti ini :

	53	65	183	98	122	14	124	37	67
53	-	12	130	45	69	39	71	16	14
65	12	-	118	33	57	51	59	28	2
183	130	118	-	85	61	169	59	146	116
98	45	33	146	-	24	84	26	61	31
122	69	57	61	24	-	108	2	85	55
14	39	51	169	84	108	-	110	23	53
124	71	59	59	26	2	110	-	87	57
37	16	28	146	61	85	23	87	-	30
67	14	2	116	31	55	53	57	30	-

Gambar 4 : Cost antar node / permintaan

Dengan menggunakan strategi *greedy*, maka langkah-langkahnya :

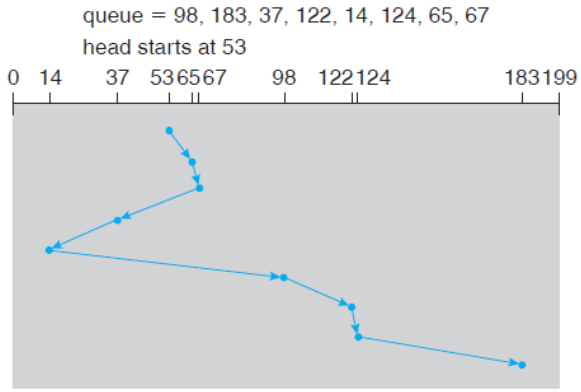
Langkah	Node	Cost
1	53 → 65	12
2	65 → 67	12 + 2 = 14
3	67 → 98	12 + 2 + 31 = 45
4	98 → 122	12 + 2 + 31 + 24 = 69
5	122 → 124	12 + 2 + 31 + 24 + 108 = 177
6	124 → 183	12 + 2 + 31 + 24 + 108 + 59 = 236

Jadi permintaan I/O akan dijalankan dengan urutan :

53 → 65 → 67 → 98 → 122 → 124 → 183

dengan cost 236 silinder.

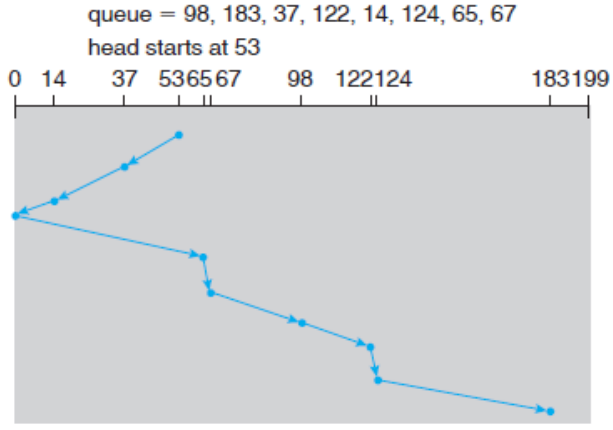
Terlihat adanya improvisasi performa menjadi sepertiga bagian apabila dibandingkan dengan algoritma *brute force* yang diterapkan pada penjadwalan FCFS.



Gambar 5 : Penjadwalan SSTF

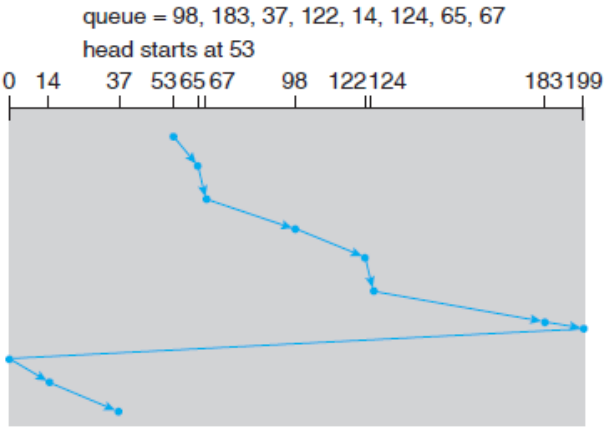
4.2 Penjadwalan CLOOK

Ada dua buah algoritma penjadwalan disk yang menggerakkan *disk head* ke seluruh lebar disk, yaitu algoritma SCAN dan C-SCAN. *Disk arm* akan digerakkan dari satu ujung disk ke ujung lainnya. Kedua algoritma ini didesain untuk membuat waktu tunggu antar proses memiliki panjang yang sama.



Gambar 6 : Penjadwalan SCAN

Pada penjadwalan SCAN, *disk head* bergerak ke silinder paling ujung dari disk. Setelah sampai paling ujung, *disk head* berbalik ke arah yang berlawanan menuju silinder di ujung lainnya sambil melayani permintaan yang dilewati saat perjalanan menuju ujung disk.



Gambar 7 : Penjadwalan CSCAN

Pada penjadwalan CSCAN, terdapat perbedaan dengan penjadwalan SCAN. Ketika *disk head* sudah sampai ujung silinder disk, *disk head* akan berbalik arah dan bergerak secepatnya menuju silinder di ujung disk yang satu lagi tanpa melayani permintaan yang dilalui dalam perjalanannya.

Pada prakteknya, kedua algoritma ini tidak diimplementasikan. Algoritma yang lebih dikenal adalah penjadwalan LOOK dan C-LOOK, Penjadwalan ini mengimplementasikan metode pergeseran ke ujung-ujung disk, namun ujung disk yang ditempuh hanya sebatas permintaan I/O yang ada. Sedangkan Penjadwalan SCAN dan CSCAN tanpa melihat ada atau tidaknya permintaan I/O, akan dilakukan pergeseran *disk head* ke paling ujung disk.

Penjadwalan C-LOOK juga menerapkan algoritma *greedy*. Berikut adalah elemen-elemen algoritma *greedy*:

1. Himpunan kandidat : permintaan – permintaan I/O yang terletak pada antrian.
2. Himpunan solusi : terbentuknya urutan pelayanan permintaan I/O.
3. Fungsi seleksi : pilihlah permintaan dari antrian permintaan yang bernilai > 0 dan bernilai minimum, apabila tidak ada yang > 0, pilih yang paling minimum.
4. Fungsi layak : permintaan yang dipilih tidak mengakibatkan *disk head* berganti arah lebih dari satu kali dalam melayani sebuah antrian permintaan.
5. Fungsi obyektif : cost dalam melayani seluruh permintaan I/O dalam antrian minimum dan lebih kecil dari dua kali panjang silinder disk.

Pada Penjadwalan LOOK, cost dari satu permintaan ke permintaan lainnya tidak dihitung secara absolut seperti penjadwalan SSTF. Jika cost bernilai negatif, berarti *next* permintaan berada di sebelah kiri *current* permintaan.

Sebagai contoh, antrian permintaan akses I/O blok silinder adalah sebagai berikut :

98,183,37,122,14,124,65,67

Misalkan *disk head* terletak pada silinder 53, cost antar node dalam antrian akan terlihat seperti ini :

	14	37	53	65	67	98	122	124	183
14	-	23	39	51	53	84	108	110	169
37	-23	-	16	28	30	61	85	87	146
53	-39	-16	-	12	14	45	69	71	130
65	-51	-28	-12	-	2	33	57	59	118
67	-53	-30	-14	-2	-	31	55	57	116
98	-84	-61	-45	-33	-31	-	24	26	85
122	-108	-85	-69	-57	-55	-24	-	2	61
124	-110	-87	-71	-59	-57	-26	-2	-	59
183	-169	-146	-130	-118	-116	-85	-61	-59	-

Gambar 8 : Cost antar permintaan / node

Dengan algoritma *greedy* maka langkah – langkah yang diambil adalah sebagai berikut :

Langkah 1 : Dari node 53 ke node 65
(Total cost = 12), hal ini terjadi karena fungsi seleksi adalah memilih cost > 0 dan paling minimum (area hijau), apabila tidak ada (area hijau habis / terpilih semua), pilih yang paling minimum (area putih).

Langkah 2 : Dari node 65 ke node 67
(Total cost = 12 + 2 = 14)

Langkah 3 : Dari node 67 ke node 98
(Total cost = 12 + 2 + 31 = 45)

Langkah 4 : Dari node 98 ke node 122
(Total cost = 12 + 2 + 31 + 24 = 69)

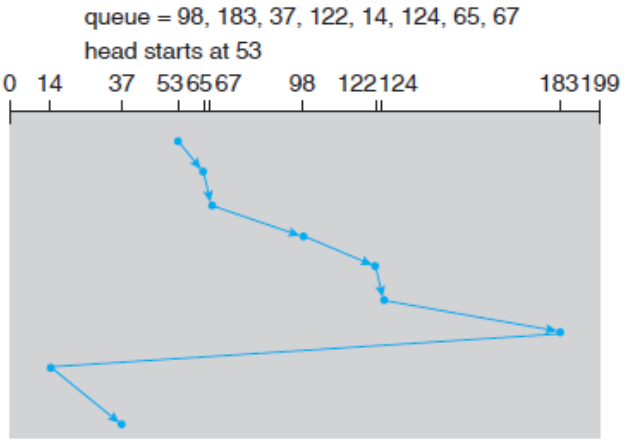
Langkah 5 : Dari node 122 ke node 124
(Total cost = 12 + 2 + 31 + 24 + 2 = 71)

Langkah 6 : Dari node 124 ke node 183
(Total cost = 12 + 2 + 31 + 24 + 2 + 59 = 130)

Langkah 7 : Dari node 183 ke node 14
(Total cost = 12 + 2 + 31 + 24 + 2 + 59 + |-169| = 299), hal ini terjadi karena node 183 tidak memiliki area hijau, sehingga dipilih nilai minimum dari area putih node 183. Penambahan cost dilakukan secara mutlak.

Langkah 8 : Dari node 14 ke node 37
(Total cost = 12 + 2 + 31 + 24 + 2 + 59 + |-169| + 23 = 322)

Total cost = 322 silinder.



Gambar 9 : Penjadwalan C-LOCK

IV. KESIMPULAN

Sampai saat ini secara umum hanya ada 5 algoritma penjadwalan seperti yang telah disebutkan di bagian atas semua. Ada cara lain untuk mendapatkan cost paling minimum untuk penjadwalan disk secara non-teknis (tanpa menghitung *rotational latency* dan memory yang dibutuhkan dalam memproses algoritma, hanya menghitung *seek-time*). Caranya adalah dengan membuat permasalahan penjadwalan disk menjadi permasalahan Travelling Salesperson Problem.

Tetapi sampai saat ini, penerapan algoritma *greedy* pada penjadwalan SSTF masih dianggap yang paling efisien untuk meningkatkan kecepatan akses dan bandwidth.

REFERENCES

[1] Munir, Rinaldi. 2009. Diktat Kuliah IF3051 Strategi Algoritma Program Studi Teknik Informatika, Sekolah Tinggi Elektro dan Informatika, Insitut Teknologi Bandung.

[2] Silberschatz, Abraham. *Operating System Concepts Essentials* . Jefferson city: USA, John Wiley & Sons, Inc., 2011, pp 457–567.

[3] Zhang,Lisa. "New Algorithms for the Disk Scheduling Problem," *IEEE.*, to be published.

[4] http://www.pcmag.com/encyclopedia_term/
Tanggal akses : 21 Desember 2012
Waktu akses : 07 : 03

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, 21 Desember 2012

Abraham Krisnanda Santoso
13510033