Penerapan Algoritma Greedy dalam Algoritma Disk Scheduling Shortest Seek Time First

Muhammad Fauzan Naufan / 13513062
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
mfauzannaufan@s.itb.ac.id

***Abstract*—Algoritma scheduling Shortest Seek Time First sejatinya merupakan implementasi lanjutan dari algoritma greedy. Algoritma SSTF merupakan salah satu algoritma disk scheduling yang mengutamakan seek time yang kecil. Dalam memilih request yang tepat, algoritma greedy digunakan.**

***Index Terms*—SSTF, greedy, optimal, algoritma.**

I. PENDAHULUAN

Algoritma adalah sebuah cara yang digunakan untuk menyelesaikan sebuah permasalahan. Dalam karya tulis ini, penulis membahas algoritma disk scheduling SSTF.

Algoritma scheduling diciptakan untuk penggunaan resource seefisien mungkin. Karena tanggung jawab dari sebuah sistem operasi adalah menggunakan hardware secara efisien. Waktu akses untuk disk terdiri dari dua komponen utama yaitu *seek time*, waktu yang dibutuhkan oleh head untuk bergerak ke sektor yang dituju dan *rotational latency*, waktu tambahan untuk memutar disk ke sektor tertentu.

Waktu akses dapat dioptimasi dengan mengatur urutan dari pelayanan request. Untuk mengatur urutan tersebut maka diciptakan algoritma scheduling.

Algoritma disk scheduling sendiri banyak macamnya yaitu FCFS (*first come first serve*), SSTF (*shortest seek time first*), SCAN, C-SCAN, LOOK, dan C-LOOK.

Algoritma scheduling SSTF menggunakan algoritma greedy dalam menentukan request yang akan diproses. Pada awalnya, head berada di posisi tertentu. Kemudian, request yang berjarak paling dekat dengan posisi head sekarang akan diproses terlebih dahulu. Maka, posisi head berubah. Selanjutnya, request yang berjarak paling dekat dengan posisi head sekarang akan diproses. Begitu seterusnya hingga semua request dipenuhi.

II. DASAR TEORI

Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi terbagi ke dalam dua jenis yaitu maksimasi dan minimasi. Jadi algoritma greedy akan mengarah ke maksimasi atau minimasi.

Greedy dalam bahasa Indonesia berarti rakus dan tamak. Greedy berprinsip "Take what you can get now". Algoritma greedy menyelesaikan masalah per langkah. Pada setiap langkah terdapat banyak pilihan yang harus dievaluasi. Lalu ditentukanlah solusi optimum lokal dengan harapan bahwa langkah sisanya mengarah ke solusi optimum global. Solusi optimum global merupakan solusi yang paling optimal dari sebuah persoalan.

Pada setiap langkah algoritma greedy, kita mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan sambil berharap bahwa dengan memilih solusi optimum lokal pada setiap langkah greedy, maka solusi optimum global akan tercapai.

Algoritma greedy mempunyai lima elemen. Elemen pertama adalah himpunan kandidat yaitu himpunan yang mungkin merupakan solusi dari persoalan. Elemen yang kedua adalah himpunan solusi yaitu himpunan yang merupakan solusi dari persoalan.

Elemen ketiga adalah fungsi seleksi yaitu fungsi yang melakukan maksimasi atau minimasi. Elemen keempat adalah fungsi layak yaitu fungsi yang memeriksa apakah sebuah himpunan kandidat melebihi batas-batas yang ditentukan. Elemen kelima adalah fungsi obyektif yaitu fungsi yang merupakan inti dari persoalan greedy.

Algoritma greedy melibatkan pencarian sebuah himpunan solusi yang merupakan himpunan bagian dari himpunan kandidat dalam hal ini himpunan solusi memenuhi beberapa kriteria yang ditentukan yaitu menyatakan suatu solusi dan himpunan solusi dioptimisasi dengan fungsi obyektif.

Misal pada masalah penukaran uang, himpunan kandidat adalah himpunan koin yang merepresentasikan nilai 100, 200, 500, dan 1000 paling sedikit mengandung

satu koin untuk setiap nilai. Himpunan solusinya adalah total nilai koin yang dipilih tepat sama jumlahnya dengan nilai uang yang ditukarkan.

Fungsi seleksinya adalah pilihlah koin yang bernilai tertinggi dari himpunan kandidat yang tersisa. Fungsi kelayakannya adalah memeriksa apakah nilai total dari himpunan koin yang dipilih tidak melebihi jumlah uang yang harus dibayar. Fungsi obyektifnya adalah jumlah koin yang digunakan minimum.

Solusi yang diberikan algoritma greedy belum tentu merupakan solusi terbaik tetapi pseudo-optimum. Alasannya adalah karena algoritma greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada. Selain itu terdapat beberapa fungsi seleksi yang berbeda, sehingga untuk mendapatkan solusi optimal maka fungsi seleksi yang benar harus dipilih.

Walaupun algoritma greedy belum tentu memberikan solusi optimal, tapi algoritma greedy sering berguna untuk menghasilkan solusi hampiran daripada menggunakan algoritma yang lebih rumit untuk menghasilkan solusi yang eksak.

Masalah-masalah yang dapat diselesaikan dengan algoritma greedy misalnya activity selection problem dan integer knapsack.

Dalam masalah integer knapsack, kita mengenal tiga jenis penggunaan algoritma greedy yaitu

1. Greedy by profit
   Pada setiap langkah objek yang mempunyai keuntungan besar akan dipilih. Berdasar pada prinsip bahwa memaksimumkan keuntungan didapat dengan memilih objek yang menguntungkan terlebih dahulu

2. Greedy by weight
   Pada setiap langkah objek yang mempunyai berat teringan akan dipilih. Berdasar pada prinsip bahwa memaksimumkan keuntungan didapat dengan memasukkan objek sebanyak-banyaknya ke dalam knapsack.

3. Greedy by density
   Pada setiap langkah, objek yang mempunyai profit/berat terbesar akan dipilih. Berdasar pada prinsip bahwa memaksimumkan keuntungan didapat dengan memilih objek yang mempunyai keuntungan per unit berat terbesar.

Contoh :

$w1 = 6$; $p1 = 12$; $w2 = 5$; $p2 = 15$;
$w3 = 10$; $p3 = 50$; $w4 = 5$; $p4 = 10$
Kapasitas *knapsack* $K = 16$

Properti objek				*Greedy by*		
i	w_i	p_i	p_i/w_i	*profit*	*weight*	*density*
1	6	12	2	0	1	0
2	5	15	3	1	1	1
3	10	50	5	1	0	1
4	5	10	2	0	1	0
Total bobot				15	16	15
Total keuntungan				65	37	65

- Solusi optimal: $X = (0, 1, 1, 0)$
- *Greedy by profit* dan *greedy by density* memberikan solusi optimal!

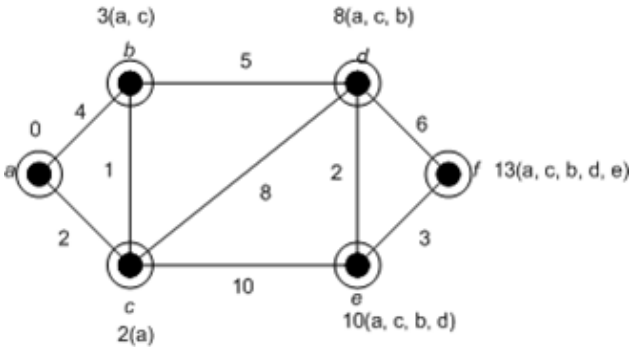
Gambar 1. Contoh penerapan greedy by profit, greedy by weight, dan greedy by density.

Selain itu persoalan lain yang menggunakan algoritma greedy adalah penjadwalan job. Persoalan utamanya adalah memilih job untuk mendapatkan keuntungan maksimal.

Langkah	J	$F = \sum p_i$	Keterangan
0	{}	0	-
1	{1}	50	layak
2	{4,1}	50 + 30 = 80	layak
3	{4, 1, 3}	-	tidak layak
4	{4, 1, 2}	-	tidak layak

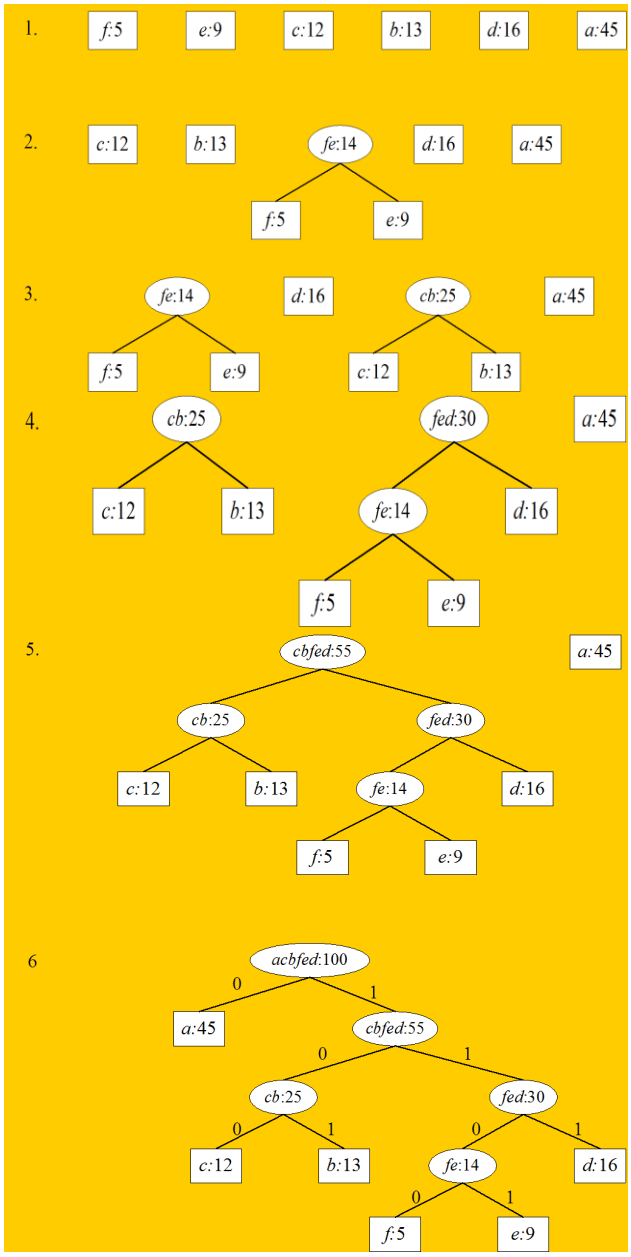
Gambar 2. Contoh penerapan algoritma greedy dalam persoalan penjadwalan job.

Persoalan yang dapat diselesaikan dengan menggunakan algoritma greedy adalah persoalan lintasan terpendek dengan algoritma Dijkstra.



Gambar 3. Contoh penerapan algoritma greedy dalam persoalan lintasan terpendek dengan algoritma Dijkstra

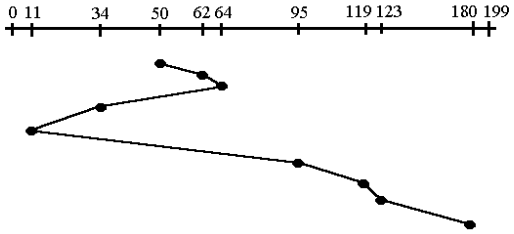
Persoalan lain yang dapat diselesaikan oleh algoritma greedy adalah persoalan pemampatan data dengan algoritma Huffmann. Prinsip kode Huffmann adalah karakter yang paling sering muncul di dalam data akan diberikan kode yang lebih pendek sehingga dibutuhkan algoritma greedy untuk menyelesaikannya.



Gambar 4. Ilustrasi dari penggunaan algoritma greedy pada pemampatan data dengan Kode Huffmaan

III. ALGORITMA SHORTEST SEEK TIME FIRST

Algoritma *shortest seek time first* (SSTF) didasarkan pada pemikiran bahwa semua request yang berada dekat dengan posisi head harus dilayani terlebih dahulu daripada request yang jauh. Sesuai namanya, algoritma SSTF memilih request yang memiliki *seek time* terkecil.



Gambar 5. Grafik perpindahan head pada algoritma SSTF dimana pergerakan head dimulai dari posisi 50 dan berakhir pada posisi 180.

Contoh dari algoritma SSTF adalah sebagai berikut. Sebuah disk memiliki space dari 0-99. Sedangkan request terjadi pada posisi 49, 91, 18, 61, 7, 62, 32, 33. Head saat ini ada di posisi 26. Request terdekat sebelum posisi 26 adalah posisi 18, sedangkan request terdekat setelah posisi 26 adalah 32. Maka head akan menuju ke posisi 32, karena 32-26=6 < 26-18=8.

Request terdekat sebelum posisi 32 adalah 18, sedangkan request terdekat sesudah posisi 32 adalah 33. Karena 33-32=1 < 32-18=14, maka head akan menuju ke posisi 33. Request terdekat sebelum posisi 33 adalah 18, sedangkan request terdekat setelah posisi 33 adalah 49. Maka head akan menuju ke posisi 18, karena 33-18=15 < 49-33=16.

Selanjutnya request terdekat sebelum posisi 18 adalah 7 sedangkan request terdekat setelah posisi 18 adalah 61. Karena 18-7=11 < 61-18=43, maka head akan menuju ke posisi 7. Lalu, karena tidak ada request sebelum posisi 7, maka head akan menuju ke request terdekat setelah posisi 7 yaitu posisi 49. Karena tidak ada request sebelum posisi 49, maka head akan menuju ke request terdekat setelah posisi 49 yaitu posisi 61.

Kemudian tidak ada request sebelum posisi 61. Maka, head akan menuju request terdekat setelah posisi 61 yaitu posisi 62. Karena hanya tersisa satu buah request lagi di posisi 91, maka head langsung menuju ke posisi 91.

Algoritma SSTF telah selesai dilakukan dengan rute pembacaan request adalah 26-32-33-18-7-49-61-62-91. Jarak total pembacaan head adalah 6 + 1 + 15 + 11 + 42 + 12 + 1 + 29 = 117.

Algoritma SSTF merupakan pengembangan dari algoritma FCFS (*first come first serve*) yang sangat boros dalam perpindahan head.

IV. PENERAPAN ALGORITMA GREEDY DALAM ALGORITMA SSTF

Algoritma greedy dalam algoritma SSTF dilakukan per langkah perubahan head. Dalam memilih request yang akan dipilih, algoritma greedy memilih request dengan jarak paling minimal dengan posisi head saat ini. Hal tersebut dilakukan hingga semua request terlayani.

Contoh dari penerapan algoritma greedy dalam algoritma SSTF adalah sebagai berikut. Sebuah disk memiliki space dari 0-99. Sedangkan request terjadi pada posisi 49, 91, 18, 61, 7, 62, 32, 33. Head saat ini ada di posisi 26.

Elemen-elemen dari algoritma greedy dari persoalan ini adalah sebagai berikut.

1. Himpunan kandidat : himpunan request yang merepresentasikan nilai 49, 91, 18, 61, 7, 62, 32, 33.
2. Himpunan solusi : himpunan yang semua requestnya telah dilayani
3. Fungsi seleksi : pilihlah request yang memiliki jarak terdekat dengan posisi head saat ini
4. Fungsi layak : memeriksa apakah request yang dilayani tidak melebihi jumlah request yang ada
5. Fungsi obyektif : seek time yang dilalui minimum

Langkah 1 (Posisi head : 26)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	32	6
2	33	7
3	18	8
4	7	19
5	49	23
6	61	35
7	62	36
8	91	65

Jarak paling kecil dengan head saat ini adalah 6 yang didapat apabila head berpindah ke posisi 32, maka request yang dipilih adalah request pada posisi 32. Selanjutnya head berpindah ke posisi 32

Langkah 2 (Posisi head : 32)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	33	1
2	18	14
3	49	17
4	7	25
5	61	29
6	62	30
7	91	59

Jarak paling kecil dengan head saat ini adalah 1 yang didapat apabila head berpindah ke posisi 33, maka request yang dipilih adalah request pada posisi 33. Selanjutnya head berpindah ke posisi 33

Langkah 3 (Posisi head : 33)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	18	15
2	49	16
3	7	26
4	61	28
5	62	29
6	91	58

Jarak paling kecil dengan head saat ini adalah 15 yang didapat apabila head berpindah ke posisi 18, maka request yang dipilih adalah request pada posisi 18. Selanjutnya head berpindah ke posisi 18.

Langkah 4 (Posisi head : 18)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	7	12
2	49	31
3	61	43
4	62	44
5	91	73

Jarak paling kecil dengan head saat ini adalah 12 yang didapat apabila head berpindah ke posisi 7, maka request yang dipilih adalah request pada posisi 7. Selanjutnya head berpindah ke posisi 7.

Langkah 5 (Posisi head : 7)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	49	42
2	61	54
3	62	55
4	91	84

Jarak paling kecil dengan head saat ini adalah 42 yang didapat apabila head berpindah ke posisi 49, maka request yang dipilih adalah request pada posisi 49. Selanjutnya head berpindah ke posisi 49.

Langkah 6 (Posisi head : 49)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	61	12
2	62	13
3	91	42

Jarak paling kecil dengan head saat ini adalah 12 yang didapat apabila head berpindah ke posisi 61, maka request yang dipilih adalah request pada posisi 61. Selanjutnya head berpindah ke posisi 61.

Langkah 7 (Posisi head : 61)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	62	1
2	91	30

Jarak paling kecil dengan head saat ini adalah 1 yang didapat apabila head berpindah ke posisi 62, maka request yang dipilih adalah request pada posisi 62. Selanjutnya head berpindah ke posisi 62.

Langkah 8 (Posisi head : 62)

No	Posisi request	Jarak ($\|Head-x_i\|$)
1	91	29

Jarak paling kecil dengan head saat ini adalah 29 yang didapat apabila head berpindah ke posisi 91, maka request yang dipilih adalah request pada posisi 91. Selanjutnya head berpindah ke posisi 91.

Hasil penerapan algoritma greedy pada algoritma SSTF

Urutan	Posisi request	Jarak
1	32	6
2	33	1
3	18	15
4	7	12
5	49	42
6	61	12
7	62	1
8	91	29
Total		117

Maka rute perpindahan head yang dipilih melalui algoritma SSTF adalah 26 - 32 - 33 - 18 - 7 - 49 - 61 - 62 - 91 dengan total bobot 117.

V. KESIMPULAN

Algoritma SSTF dasarnya adalah algoritma greedy yang digunakan untuk menjadwalkan pelayanan request pada disk. Karena sifat dasar algoritma greedy adalah solusi greedy belum tentu solusi yang optimal, maka hasil algoritma SSTF pun tidak optimal.

VI. ACKNOWLEDGMENT

Penulis mengucap syukur kepada Allah subhanahu wa ta'ala karena berkat ridha-Nya karya tulis ini bisa dibuat.

Penulis juga hendak memberikan ucapan terima kasih kepada dosen kami, Pak Rinaldi Munir dan Bu Nur Ulfa Maulidevi atas pengajaran dan bimbingannya selama satu semester ini. Semoga ilmu yang telah diajarkan bernilai manfaat bagi kami para muridnya.

Penulis juga hendak mengucapkan terima kasih kepada kedua orang tua yang memberikan dukungan kepada penulis supaya karya tulis bisa dibuat dengan baik dan lancar.

Penulis hendak mengucapkan terima kasih kepada semua pihak yang telah mendukung dalam pembuatan karya tulis ini.

REFERENCES

[1] Rinaldi Munir. Diktat Kuliah Strategi Algoritma. 2009.
[2] Abraham Silberschatz. *Operating System Concepts.* Wiley, 2013.

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, 5 Mei 2015

Muhammad Fauzan Naufan 13513062