Penerapan algoritma greedy pada berbagai macam tugas sistem operasi

Riady Sastra Kusuma / 13512024
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
riady.sk@student.itb.ac.id

Abstrak* – *Dalam makalah ini dibahas mengenai pemakaian algoritma greedy dalam mengatur penjadwalan CPU (CPU scheduling), penjadwalan disk (disk scheduling) dan alokasi memori (memory allocation) dalam sistem operasi untuk meningkatkan peformansi komputer. Makalah juga membahas pebandingan peformansinya jika tidak menggunakan algoritma greedy

***Index Terms*— Memory allocation, CPU scheduling, greedy, disk scheduling, operating system.**

I. PENDAHULUAN

Sistem operasi adalah software yang menghubungkan hardware dengan pengguna komputer. Dengan adannya sistem opearasi pengguna tidak perlu memikirkan hubungan aplikasi yang dijalankan dengan hardware komputer yang ada.

Sistem operasi mempunyai tugas yang banyak seperti mengatur penjadwalan CPU (*CPU scheduling*), mengatur pendjadwalan disk (*disk scheduling*) dan alokasi memori (*memory allocation*). Sistem operasi juga mengatur penggunaan hardware seperti prosesor, memori, dan perangkat input output lainnya. Karena tugasnya yang sangat banyak, penggunaan algoritma yang mangkus sangat dibutuhkan agar tugas-tugasnya dapat berjalan dengan baik.

Di dalam makalah ini akan dibahas pengaturan penjadwalan CPU, penjadwalan pembacaan disk dan alokasi memori dapat lebih efektif jika menggunakan algoitma greedy dibandingkan dengan brute force. Greedy juga tekadang lebih mangkus dari algoritma yang lainnya, karena greedy hanya membutuhkan kalkulasi yang sedikit daipada algoritma yang lebih kompleks.

Penjadwalan CPU dibutuhkan agar CPU efektif memproses proses yang ada agar proses dapat diselesaikan dengan waktu yang singkat. Dengan demikian, proses tidak menunggu lama untuk diproses.

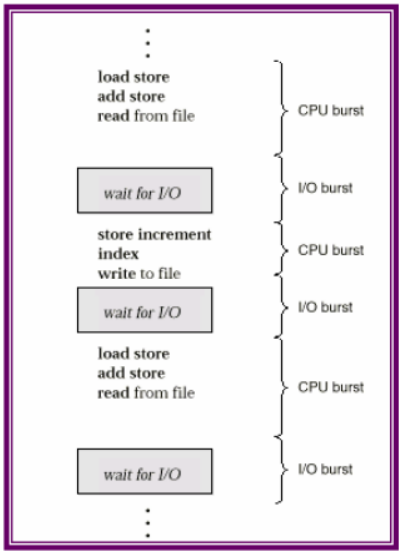
Alokasi memori dibutuhkan karena memori mempunyai kapasitas maksimal. Untuk menjalankan suatu aplikasi atau program, dibutuhkan ruang memori yang kosong untuk diproses. Maka dari itu algoritma yang mangkus dibutuhkan untuk menglokasikan memori secara efektif.

Penjadwalan pembacaan disk dibutuhkan agar pembacaan disk dapat terjadi secepatnya dan tidak melakukan perputaran disk yang lama.

II. TEORI DASAR

A. Penjadwalan CPU

Pada multiprogramming pasti akan ada proses yang berjalan bersamaan dalam suatu waktu. Sistem ini dipakai untuk memaksimalkan kerja CPU. Pada saat proses dijalankan terjadi siklus CPU dan menunggu I/O (input/ouput) yang disebut dengan CPU-I/O burst. Eksekusi proses dimulai dengan CPU burst lalu dilanjutna dengan I/O burst, diikuti CPU burst yang lain, lalu I/O burst dan seterusnya.



Gambar 2.1 Siklus CPU I/O burst [1]

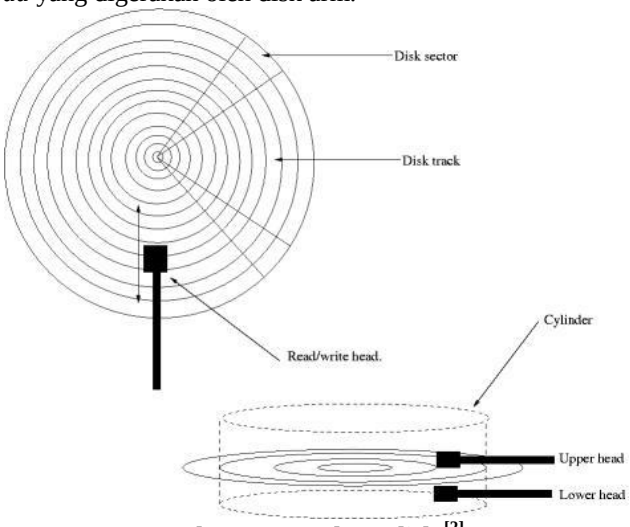
Pada saat proses diekskusi terdapat CPU burst yang pendek dan yang panjang. Biasanya pada proses yang sering menggunakan I/O, lebih banyak terdapat CPU burst yang pendek, seperti aplikasi Microsoft word, editor foto dan sebagainya. Sedangkan proses yang jarang menggunakan I/O mempunyai CPU burst yang sangat lama, seperti antivirus. Terkadang CPU akan mengurus proses yang memiliki CPU burst yang lama dibandingkan

CPU burst yang sebentar. Maka dari itu penjadwalan CPU sangat dibutuhkan agar semua proses dapat dijalankan.

Pada saat CPU menganggur, maka sistem operasi harus menyleksi proses-proses yang ada untuk dieksekusi. Pada proses menyeleksi tersebut, dibutuhkan algoritma yang mangkus agar semua proses dapat dieksekusi dengan waktu yang optimal. Pada saat proses dieksekusi, proses tersebut bisa ditunda untuk mengeksekusi proses yang lain. Hal ini biasa disebut penjadwalan *preemptive*. Penjadwalan preemptive biasa dilakukan agar tidak ada proses yang dieksekusi terus menerus dan banyak proses yang menunggu lama sampai proses tersebut selesai dieksekusi.

B. Penjadwalan disk

Disk adalah tempat penyimpanan data pada komputer. Disk mempunyai kapasitas yang besar dan harus mengakses data terus menerus. Disk menyimpan data pada sector-sektor yang ada pada silindernya. Data dibaca oleh *head* yang digerakan oleh disk arm.



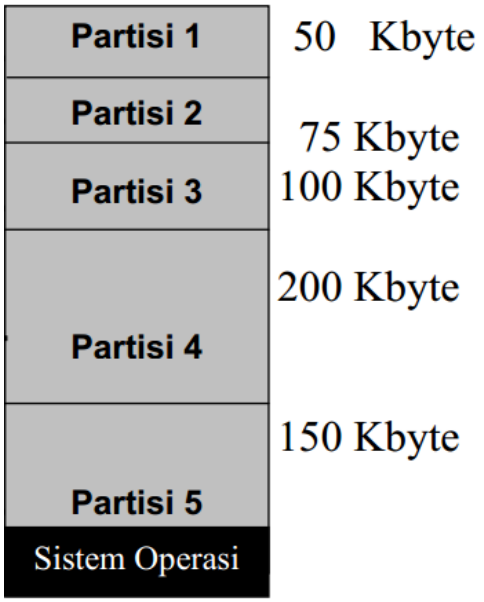
Gambar 2.2 Struktur disk [2]

Salah satu tugas sistem operasi adalah menggunakan *hardware* secara efisien. Efisiensi pada *disk drives* adalah waktu akses yang cepat pada disk. waktu akses pada disk memiliki 2 komponen utama, yaitu waktu pencarian dan waktu rotasi disk. Waktu pencarian adalah waktu yang dibutuhkan *disk arm* untuk menggerakan *head* ke bagian silinder disk yang mengandung sector yang dicari. Waktu rotasi disk adalah waktu tambahan yang dibutuhkan untuk menunggu rotasi sehingga sector yang diinginkan dapat dibaca oleh *head*.

Sistem operasi harus dapat menyeleksi permintaan yang memerlukan pelayanan disk dengan baik, agar waktu akses pada disk dapat terjadi secepat mungkin. Jika proses menyeleksinya tidak efisien, maka disk akan berputar dengan waktu yang lama dan banyak permintaan yang tidak dilayani pada waktunya.

C. Alokasi memori

Salah satu tugas sistem operasi adalah mengalokasikan memori program pada memori di ruang yang kosong. Di dalam memori terdapat partisi-partisi untuk dialokasikan suatu program.



Gambar 2.3 Contoh partisi-partisi pada memori

Pada masing-masing partisi dapat dimasukkan suatu program untuk dieksekusi. Program yang masuk ke dalam memori disebut proses. Proses yang dialokasikan hanya dapat masuk ke dalam 1 partisi saja dan tidak dapat melewati batas partisi yang lain.

Dibutuhkan suatu algoritma untuk mengisi partisi-partisi tersebut dengan program agar alokasi yang dilakukan tidak menyisakan memori yang banyak pada suatu partisi. Jika dalam suatu partisi tersisa memori yang tidak terpakai dan itu terjadi pada semua partisi, maka memori di anggap penuh karena suatu program sudah tidak cukup lagi masuk ke suatu partisi. Padahal jika semua sisa-sisa tersebut digabungkan maka program tersebut masih muat.

D. Algoritma greedy

Algortima greedy adalah algoritma untuk memecahkan persoalan optimasi. Algoritma ini cukup sederhana karena hanya mencari solusi optimum lokal yang dihapkan akan mencapai hasil optimum global. Tetapi tidak selamanya algoritma greedy menghasilkan solusi yang paling optimum.

Algoritma greedy tidak memperhitungkan langkah-langkah selanjutnya. Hal ini menyebabkan algoritma greedy mempunyai kelebihan dan kelemahannya tersendiri.

Kelebihan algoritma greedy adalah kalkulasinya yang tidak memakan waktu yang lama. Sedangkan kelemahannya adalah solusinya tidak selalu paling optimum.

Algoritma greedy terdiri dari 5 elemen yaitu :

a. Himpunan Kandidat, C
   Kumpulan dari semua kemungkinan pilihan yang dapat diambil
b. Himpunan Solusi, S
   Kumpulan dari pilihan-pilihan yang diambil yang akan menjadi solusi
c. Fungsi seleksi
   Fungsi untuk menyeleksi pilihan dari himpunan kandidat untuk dipilih dan dimasukkan kedalam himpunan solusi.
d. Fungsi kelayakan
   Fungsi untuk memeriksa apakah himpunan solusi tidak melewati batas dan sudah benar
e. Fungsi obyektif
   Fungsi yang menentukan himpunan solusi adalah solusi akhir

III. PEMBAHASAN

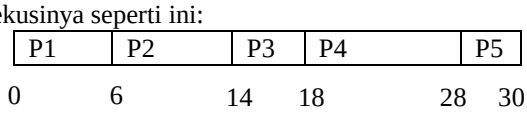
A. Algoritma greedy pada penjadwalan CPU

Algoritma greedy ini akan memilih proses yang memiliki waktu terkecil yang dimiliki proses untuk diselesaikan. Hal ini diharapkan agar proses-proses yang dapat dieksekusi secara cepat dan proses-proses lainnya tidak lama menunggunya. Setiap langkah yang diambil mengambil proses yang waktu tersisa untuk mennyelsaikannya paling sedikit. Kelebihan dari algoritma ini adalah waktu untuk rata-rata menunggunya lebih cepat.

Berikut adalah contohnya:

Process	Burst time (detik)
P1	6
P2	8
P3	4
P4	10
P5	2

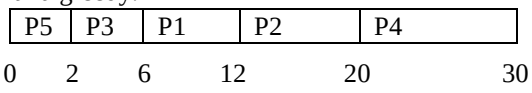
Jika tidak memakai algoritma greedy dan mengeksekusi proses yang datang terlebih dahulu maka gambaran eksekusinya seperti ini:



Waktu menunggu P1 adalah 0 detik, P2 menunggu selama 6 detik, P3 menunggu selama 14 detik, P4 menunggu selama 18 detik, dan P5 menunggu selama 28 detik. Jadi rata-rata waktu menunggunya adalah (0+6+14+18+28)/5 = 13,2 detik.

Algoritma greedynya kira-kira seperti ini. Semua proses dibandingkan waktu pengerjaannya (burst time). Diambil proses yang paling sedikit. Lalu proses itu dieksekusi sampai selesai. Lalu diambil lagi burst timenya yang paling sedikit, lalu proses tersebut dieksekusi. Hal ini dilakukan sampai semua proses selesai dieksekusi.

Berikut adalah contoh eksekusi jika memakai algoritma greedy:



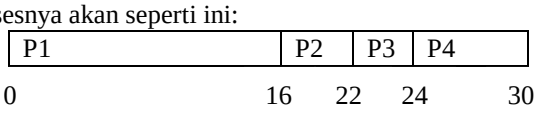
Waktu menunggu P1 adalah 6 detik, P2 menunggu selama 12 detik, P3 menunggu selama 2 detik, P4 menunggu selama 20 detik, dan P5 menunggu selama 0 detik. Jadi rata-rata menunggunya adalah (6+12+2+20+0)/5 = 10 detik.

Dapat dilihat bahwa rata-rata waktu proses menunggu lebih cepat dibandingkan tidak memakai algoritma greedy.

Contoh kasus lain jika diberikan waktu kedatangan prosesnya:

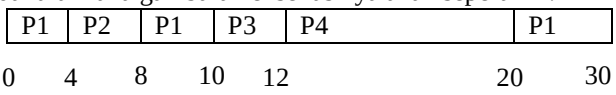
Proses	Waktu kedatangan (detik)	Burst time (detik)
P1	0	16
P2	4	4
P3	10	2
P4	12	8

Jika tidak memakai algoritma greedy dan mengeksekusi proses yang datang terlebih dahulu maka prosesnya akan seperti ini:



Waktu menunggu P1 adalah 0 detik, P2 menunggu selama 16 – 4 = 12 detik, P3 menunggu selama 22 – 10 = 12 detik, dan P4 menunggu selama 24 – 12 = 12 detik. Jadi rata-rata waktu menunggunya adalah (0+12+12+12)/4 = 9 detik.

Sedangkan jika mengambil waktu yang kecil terlebih dahulu maka gambaran eksekusinya akan seperti ini:



Yang terjadi adalah P1 dieksekusi selama 4 detik. Lalu P2 datang. Dibandingkanlah sisa waktu P1 dan P2. P1 mempunya sisa waktu pengerjaan selama 16 – 4 = 12. Sedangkan P2 mempunyai sisa waktu pengerjaan selama 4 detik. Maka dari itu P1 dihentikan dulu lalu P2 mulai dikerjakan. Setelah P2 selesai, hanya ada P1 yang menunggu. Maka dari itu P1 dieksekusi lagi. Pada detik ke 10 P3 datang. Sisa waktu pengerjaan P1 sebanyak 12 – 2 = 10 detik. Sedangkan P3 mempunyai sisa waktu pengerjaan selama 2 detik. Maka P3 diselesaikan terlerbih dulu. Dan pada detik 12, P3 selesai dikerjakan dan P4 datang. Sisa waktu pengerjaan P4 lebih sedikit dibandingkan dengan P1 yaitu 8 dan 10. Maka P4 dikerjakan terlebih dahulu dan sisanya P1 dikerjakan.

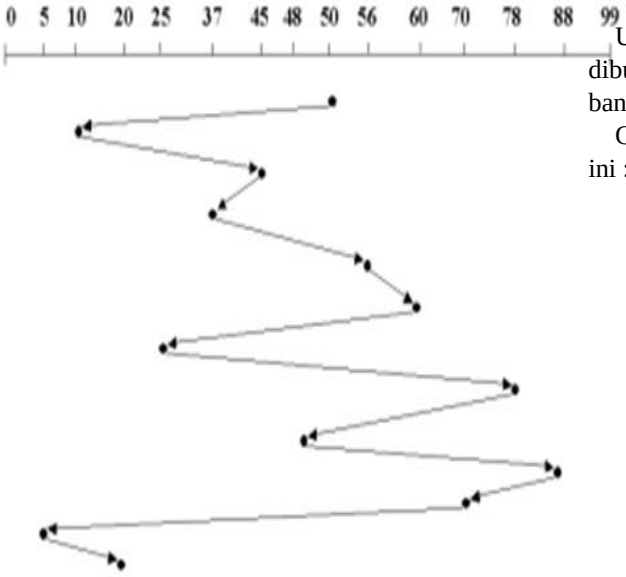
Waktu menunggu P1 adalah 20 – 4 – 2 = 14 detik, P2 menunggu selama 4 – 4 = 0 detik, P3 menunggu selama 10 – 10 = 0 detik, dan P4 menunggu selama 12 – 12 = 0 detik. Jadi rata-rata waktu menunggunya adalah (14+0+0+0)/4 = 3,5 detik.

Dapat dilihat bahwa pebedaan yang signifikan dari algoritma greedy untuk mengurangi waktu menunggu rata-rata dari semua proses. Algoritma greedy selalu menghasilkan hasil yang optimum untuk permasalahan meminimalkan waktu menunggu. Tapi kelemahan dari algoritma greedy ini untuk mengatur penjadwalan CPU adalah ada proses yang lama sekali menunggunya dan bahkan tidak di proses-proses jika terus berdatangan proses yang mempunyai burst time yang lebih kecil. Hal ini dapat menyebabkan suatu aplikasi tidak dapat bekerja.

B. Algoritma greedy pada penjadwalan disk

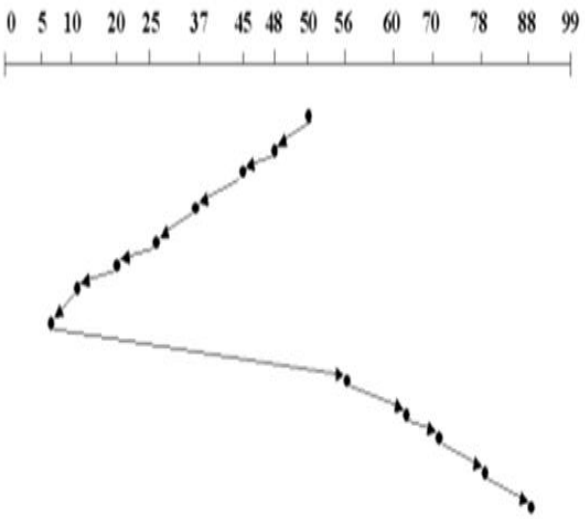
Algoritma greedy pada penjadwalan disk adalah mencari jarak permintaan (request) terdekat dari poisisi head untuk dibaca. Algoritmanya kira-kira seperti ini. Untuk setiap posisi request pada antrian, dicari jarak yang paling dekat dengan posisi head sekarang. Lalu head menuju ke posisi itu. Lalu dicari lagi pada setiap posisi request pada antian yang terdekat dengan posisi head sekarang. Hal ini dilakukan sampai semua request di antrian habis.

Contoh posisi request pada suatu antrian adalah : 10, 45, 37, 56, 60, 25, 78, 48, 88, 70, 5, 20. Dengan posisi head sekarang adalah 50 dan posisi paling maksimal adalah 99. Maka tanpa algoritma greedy, masing-masing request akan diproses dari depan. Gambaran pergerakan headnya akan seperti ini :



Jarak yang ditempuh oleh head adalah 362. Sedangkan jika memakai algoritma greedy dalam kasus tersebut jaraknya akan menjadi lebih sedikit.

Awalnya head ada pada posisi 50. Dicari jarak yang terdekat dengan posisi 50. Didapatkan posisi 48 adalah jarak tedekat yang ada pada antrian. Maka head sekarang menuju ke posisi 48. Lalu di cari lagi posisi yang terdekat dari posisi head sekarang yaitu 48. Didapatkan request pada posisi 45. Maka sekarang head menuju ke posisi 45. Hal ini terjadi sampai antriannya kosong. Gambaran pergerakan disk akan seperti ini :



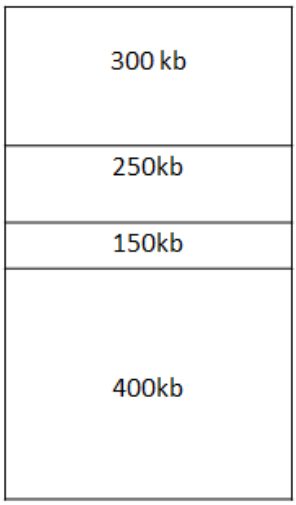
Jarak yang ditempuh oleh head adalah. 128. Dapat dilihat bahwa perbandingannya sangat signifikan dibandingkan memproses request tanpa algoitma greedy yaitu 362.

Algortima greedy pada penjadwalan disk sangat efektif. Tapi ada kelemahannya yaitu jika proses yang datang selalu dekat pada posisi head yang sekarang, maka request yang berada jauh dari head tidak akan diproses.

C. Algoritma greedy pada alokasi memori

Untuk mengalokasikan program pada partisi di memori dibutuhkan algoritma yang efektif agar tidak terjadi banyak sisa memori pada setiap partisi.

Contoh kasusnya adalah tersedia partisi-partisi seperti ini :

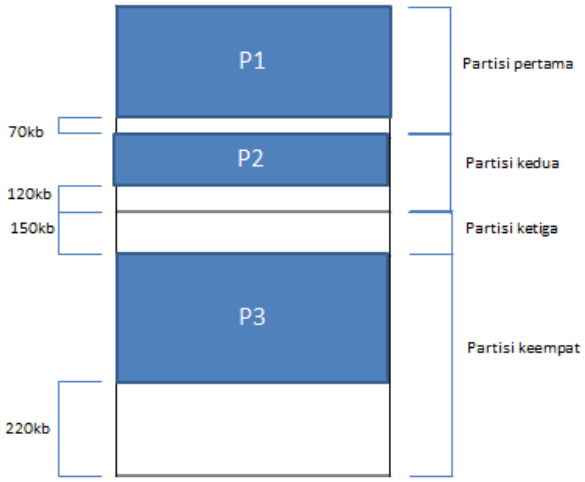


Lalu ada program-program seperti berikut:

Program	Besar memori (kb)
P1	230
P2	120
P3	280
P4	370

Program-program tesebut ingin dialokasikan pada partisi-partisi di atas. Tanpa algoritma greedy, program akan dialokasikan secara berurutan di tempat yang dapat menampung progam tersebut.

Proses yang terjadi adalah P1 dialokasikan langsung pada partisi pertama, sehingga partisi pertama sekarang mempunyai sisa 70kb. Lalu P2 ingin dialokasikan di partisi pertama, tetapi tidak cukup karena P2 mempunyai besar 120kb, sedangkan sisa partisi pertama hanya 70kb. Jadi P2 dialokasikan ke partisi selanjutnya yaitu partisi kedua yang mempunyai sisa 250kb. Setelah P2 dialokasikan sisanya menjadi 130kb. Lalu P3 ingin dialokasikan. P3 tidak cukup pada partisi pertama, kedua maupun ketiga. Jadi P3 dialokasikan pada partisi keempat. Sisa partisi keempat adalah 220kb. Setelah itu yang terakhir P4 ingin dialokasikan. Tetapi P4 tidak cukup dipartisi manapun. Sehingga P4 tidak dapat diproses. Berikut adalah gambarannya:

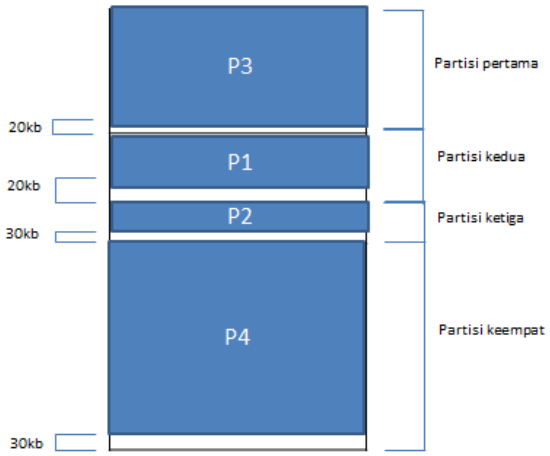


P4 tidak dapat dialokasikan di partisi-partisi di atas. Padahal jika dijumlahkan sisa-sisa memori yang ada, sebenarnya P4 dapat dialokasikan jika mengalokasikan program-progam sebelumnya itu tepat.

Dengan menggunakan algoritma greedy, mengalokasikan program-progam akan lebih efektif. Prosesnya adalah saat ingin mengalokasikan memori, cari selisih memori kosong pada setiap memori dengan besarnya program yang ingin dialokasikan. Jadi fungsi seleksinya adalah selisih memori yang kosong pada partisi dengan program yang ingin dialokasikan haruslah minimal.

Maka proses yang terjadi pada kasus diatas adalah sebagai berikut. P1 ingin dialokasikan, selisih memori yang kosong pada partisi kedua dengan besarnya P1 adalah yang minimal yaitu 20kb. Sehingga P1 dialokasikan pada partisi kedua. Sekarang partisi kedua mempunyai sisa memori kosong sebanyak 20kb. Lalu P2 ingin dialokasikan, maka dicari selisih memori yang kosong dengan P2 yang minimal. Didapatkan partisi ketiga dengan selisih 30kb. Maka P2 dialokasikan pada partisi ketiga. Sekarang Partisi ketiga mempunyai sisa memori sebesar 30kb. Lalu P3 ingin dialokasikan. Di dapat partisi pertama adalah yang minimal dengan selisih 20kb. Maka P3 dialokaskan pada partisi pertama. Sekarang tersisa P4 yang ingin dialokasikan. Partisi yang masih tersedia dan cukup untuk P4 adalah partisi keempat. Maka P4 dialokasikan pada partisi keempat.

Berikut adalah hasil akhirnya :



Dengan algoritma greedy, maka semua program dapat dialokasikan dengan efisien. Dan sisa-sisa memorinya sangatlah sedikit dibandingkan tanpa algoritma greedy.

Dengan perbedaan yang jauh dapat dibilang algoritma greedy efektif dalam mengalokasikan program pada memori.

IV. KESIMPULAN

Algoritma greedy pada pengaturan penjadwalan CPU cukup efektif karena waktu menunggu proses-proses sudah minimal dan optimum. Tapi ada beberapa kasus yang tidak bisa ditangani oleh algoritma greedy yaitu kasus suatu proses yang membutuhkan waktu yang lama lama sekali dieksekusi karena banyak proses-proses yang membutuhkan waktu yang sebentar dan oleh algoritma greedy proses-proses yang sebentar itu diutamakan untuk dieksekusi. Jadi algoritma greedy hanya efektif untuk mengurangi waktu menunggu proses rata-rata.

Algoritma greedy pada pengaturan penjadwalan disk cukup efektif karena dapat mengurangi waktu akses dengan mengurangi jarak yang ditempuh oleh head. Hal ini terjadi karena dengan algoritma greedy, request yang diproses hanyalah yang paling dekat dengan posisi head. Dengan demikian ada kasus yang tidak dapat ditangani yaitu request yang terletak jauh dari head akan lama diprosesnya jika request-request terus berdatangan disekitar head. Jadi algoritma greedy hanya efektif untuk mengurangi waktu akses tapi tidak menjamin semua request akan dilayani.

Algoritma greedy pada pengaturan alokasi memori efektif digunakan, karena akan membeikan sisa memory

yang sedikit pada setiap memorinya, sehingga lebih banyak program yang dapat dialokasikan.

. DAFTAR PUSTAKA

[1] Silberschatz, "Operating system concepts", 9th ed. Hoboken : John Wiley & Sons Inc.

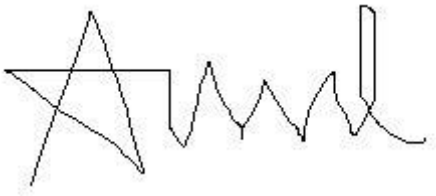
Munir,Rinaldi "Diktat Strategi Algoritma",Bandung : Penerbit ITB, 2009.

[2] http://heim.ifi.uio.no/~inf3151/disk_structure.jpg diakses pada tanggal 17 Mei 2014

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, 17 Mei 2014



Riady Sastra Kusuma / 13512024