Penerapan Algoritma Greedy pada Alokasi Memori

Muhammad Sulthan Adhipradhana, 13516035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516035@std.stei.itb.ac.id

***Abstrak*—Greedy adalah algoritma yang sering digunakan karena merupakan algoritma yang mendapatkan hasil mendekati hasil optimum. Greedy juga diterapkan pada sistem operasi sebagai algoritma untuk mencari alokasi memori. Alokasi memori merupakan mekanisme pemberian tempat bagi proses dari memori kosong. Algoritma alokasi memori terdiri dari first fit, best fit, dan word fit. Algoritma alokasi memori mempunyai beberapa pertimbangan, pertimbangan tersebut menyangkut dengan performansi, waktu eksekusi, dan juga fragmentasi.**

***Keywords*—Greedy, Alokasi Memori, First-fit, Best-fit, Worst-fit**

I. PENDAHULUAN

Algoritma greedy adalah algoritma yang sering digunakan dalam kehidupan sehari-hari. Algoritma greedy membahas mengenai persoalan optimasi. Algoritma greedy akan menghasilkan solusi optimum lokal yang diharapkan dapat mendekati hasil dari solusi optimum global.

Alokasi memori merupakan konsep yang ada pada sistem operasi. Alokasi memori digunakan untuk mengalokasikan memori untuk proses dengan memanfaatkan memori yang kosong. Dalam makalah ini, alokasi memori yang dibahas hanyalah alokasi memori kontigu, yang berarti memori yang dialokasikan harus mempunyai letak yang bersebelahan, tidak diperbolehkan untuk mengalokasikan memori yang mempunyai letak yang terpisah. Ada beberapa algoritma yang digunakan untuk mengalokasikan memori yaitu first-fit, best-fit, worst-fit. Masing-masing menggunakan algoritma greedy dengan fungsi objektif yang berbeda.

First-fit adalah prinsip alokasi memori dengan algoritma greedy menggunakan pendekatan lubang pertama yang cukup.

Best-fit adalah prinsip alokasi memori dengan algoritma greedy menggunakan pendekatan lubang cukup yang paling kecil.

Worst-fit adalah prinsip alokasi memori dengan algoritma greedy menggunakan pendekatan lubang cukup yang paling besar.

Dalam menentukan algoritma apa yang tepat, masing-masing mempunyai kelebihan dan kekurangan masing-masing. Hal yang harus diperhatikan dalam pemilihan adalah algoritma adalah waktu yang dibutuhkan untuk menjalankan, performansi dari alokasi memori dan juga fragmentasi memori yang dihasilkan.

Fragmentasi memori adalah kondisi dimana sebetulnya terdapat cukup memori untuk memenuhi kebutuhan, tetapi karena tidak kontigu, maka tidak dapat dialokasikan. Fragmentasi juga merupakan hal yang harus dipertimbangkan dalam pemilihan algoritma alokasi memori.

II. DASAR TEORI

A. Algoritma Greedy

Greedy dalam bahasa Indonesia berarti tamak atau rakus dan dalam konotasi negatif. Greedy mempunyai prinsip “take what you can get now” atau jika diterjemahkan berarti ambil apa yang dapat diperoleh sekarang. Dalam pemrograman, greedy selalu mengambil hasil terbaik yang diperoleh sekarang tanpa mempedulikan hasil yang diperoleh selanjutnya.

Algoritma greedy membuat solusi langkah per langkah. Pada setiap langkah, terdapat banyak pilihan yang harus dieksplor. Sehingga, untuk setiap langkah harus diambil keputusan yang terbaik. Langkah yang telah dibuat ini tidak boleh diubah pada langkah selanjutnya.

Dalam menyelesaikan masalah, pendekatan yang dipakai adalah membuat pilihan yang tampaknya seperti pilihan terbaik. Hal ini dilakukan dengan membuat pilihan optimum lokal (local optimum) dengan harapan pilihan ini dapat mendekati pilihan optimum global (global optimum).

Dari definisi yang telah disebutkan di atas, dapat disimpulkan bahwa setiap tahap algoritma greedy memuat :

1. Pengambilan pilihan terbaik pada tahap tersebut tanpa memperdulikan tahap selanjutnya.
2. Mempunyai harapan pemilihan optimum lokal dapat berakhir dengan optimum global.

Algoritma greedy mempunyai skema umum yang tersusun atas lima elemen yaitu :

1. Himpunan kandidat (C)

Himpunan yang mempunyai elemen-elemen pembentuk solusi. Dinamakan kandidat karena merupakan kandidat dari solusi. Untuk setiap langkah, dipilih satu elemen dari himpunan kandidat.

2. Himpunan solusi (S)

Himpunan yang berisi solusi dari persoalan. Merupakan himpunan bagian dari himpunan kandidat karena tiap elemennya merupakan bagian dari himpunan kandidat.

3. Fungsi seleksi

Fungsi yang digunakan untuk memilih salah satu elemen dari himpunan kandidat pada setiap langkah. Kandidat yang telah dipilih tidak dipertimbangkan di tahap selanjutnya. Fungsi seleksi selalu memilih nilai paling optimal dari himpunan kandidat.

4. Fungsi kelayakan

Funsgsi yang digunakan untuk menentukan apakah suatu elemen dari himpunan kandidat yang akan dipilih memenuhi batasan dan tidak melanggar kendala. Kandidat yang memenuhi fungsi ini dimasukan ke Himpunan solusi, selain itu tidak akan diperhitungkan lagi.

5. Fungsi obyektif

Tujuan dari pencarian solusi dengan algoritma greedy. Biasanya memaksimalkan atau meminimumkan nilai solusi.

Skema umum dari algoritma greedy dalam bentuk pseudocode adalah sebagai berikut,

```
function greedy(input C:himpunan_kandidat)->
himpunan_kandidat
{ Mengembalikan solusi dari persoalan
optimasi greedy
  Masukan : Himpunan kandidat C
  Keluaran : Himpunan solusi bertipe
himpunan_kandidat
}

Deklarasi
   x : kandidat
   S : Himpunan solusi

Algoritma
   S <- {} {Inisialisasi S dengan kosong}
   while (not SOLUSI(S)) and (C ≠ {}) do
      x <- SELEKSI(C){Pilih kandidat dari C}
      C <- C - {X} {Kurangi X dari C}
      if LAYAK(S U {X}) then
         S <- S U {X}
      endif
   endwhile
   {SOLUSI(S) or C = {}}

   if SOLUSI(S) then
      return S
   else
      write('tidak ada solusi')
   endif
```

Jika algoritma di atas dianalisis, ada kemungkinan algoritma greedy tidak selalu menghasilkan hasil yang optimum. Hal ini mungkin dapat disebabkan oleh dua kemungkinan. Kemungkinan yang pertama adalah algoritma greedy tidak mengecek seluruh kemungkinan jawaban, karena hanya mengambil yang terbaik untuk setiap tahap. Kemungkinan yang kedua adalah fungsi seleksi yang digunakan kurang merepresentasikan hasil optimum yang sebenarnya. Sehingga, diperlukan pemilihan fungsi seleksi yang bagus agar mendapatkan hasil optimum. Greedy memang tidak selalu menghasilkan yang optimum, naum greedy merupakan aproksimasi dari hasil optimum sebenarnya yang didapatkan dari algoritma kompleks.

B. Alokasi Memori

Memori adalah perangkat keras yang digunakan untuk menyimpan informasi secara temporer atau permanen. Pada kali ini, memori yang disebutkan adalah memori yang temporer. Memori digunakan untuk menyimpan proses-proses agar tiap proses dapat dilayani oleh CPU.

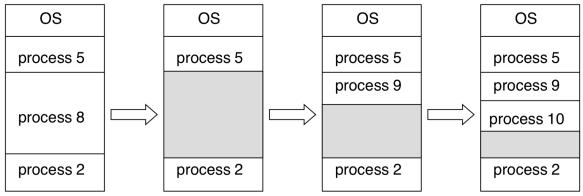
Proses adalah program yang sedang dieksekusi. Proses biasanya terdiri dari process stack, yang memuat data temporer seperti parameter fungsi, return address, dan variable lokal. Oleh karena itu, proses perlu dialokasikan di memori agar dapat dilayani. Untuk memasuki memori, proses menunggu pada *Input Queue* sebelum dialokasikan ke memori. *Input Queue* akan berisi proses-proses yang urutannya diatur sesuai dengan algoritma penjadwalan.

Alokasi memori yang akan dibicarakan adalah alokasi memori secara kontigu. Pada alokasi memori kontigu, setiap proses disimpan pada suatu bagian yang berlokasi bersebelahan dengan proses lainnya. Salah satu metode yang cukup sering digunakan dalam alokasi memori adalah membagi memori ke partisi-partisi yang mempunyai ukuran yang berbeda-beda (variable-partition).

Pada skema variable-partition, sistem operasi menyimpan tabel yang menyimpan bagian memori mana yang kosong dan bagian memori mana yang terisi. Pada awalnya, seluruh memori tersedia untuk proses dan dipartisi sebagai satu blok memori ukuran besar yang disebut sebagai lubang.

Proses yang sebelumnya perada pada *Input Queue* dialokasikan oleh sistem operasi ke dalam memori agar dapat dilayani oleh CPU. Proses yang telah selesai dijalankan dilepaskan dari memori. Sistem operasi akan mengalokasikan memori hingga tidak ada lagi ruang yang cukup untuk dialokasikan. Biasanya, sistem operasi menunggu hingga suatu proses selesai atau melanjutkan *Input Queue* agar mendapatkan proses dengan kebutuhan memori yang lebih sedikit yang dapat dialokasikan.

Dari penjelasan yang telah disebutkan, dapat diambil kesimpulan bahwa blok memori terdiri atas kumpulan lubang yang besarnya bervariasi dan tersebar dalam memori. Ketika proses datang, sistem akan mencari lubang yang besarnya cukup untuk proses tersebut.



Gambar 2.1 Lubang pada Memori
Sumber : Operating System Concepts

Untuk mengalokasikan proses ke dalam lubang-lubang tersebut dibutuhkan algoritma yang dapat menentukan lubang apa yang tepat untuk proses tersebut. Algoritma yang digunakan adalah algoritma First fit, Best fit, dan Worst fit.

Pembahasan dari tiap algoritma adalah sebagai berikut :

1. First Fit

First fit adalah algoritma untuk alokasi memori yang mengalokasikan memori proses pada lubang pertama yang cukup. Pencarian biasanya dimulai dari lubang pertama. Pencarian akan berhenti ketika menemukan lubang pertama yang cukup.

2. Best Fit

Best fit adalah algoritma untuk alokasi memori yang mengalokasikan memori proses pada lubang cukup yang mempunyai ukuran terkecil. Best fit menginginkan agar alokasi menghasilkan lubang sisa yang paling kecil.

3. Worst Fit

Worst fit adalah algoritma untuk alokasi memori yang mengalokasikan memori proses pada lubang cukup yang mempunyai ukuran terbesar. Strategi ini akan menghasilkan lubang sisa yang besar. Diharapkan lubang sisa yang besar ini dapat digunakan untuk proses yang lain.

Ketiga algoritma untuk alokasi memori ini menggunakan greedy tetapi dengan menggunakan fungsi seleksi yang berbeda. Tiap algoritma mempunyai keuntungan dan kerugian masing-masing. Dalam performansi, banyaknya ditemukan algoritma best fit dan first fit berfungsi dengan baik.

C. Fragmentasi

Saat proses dialokasikan dan dilepaskan dari memori, memori yang kosong atau lubang akan tersebar di memori. Fragmentasi adalah kondisi dimana sebetulnya terdapat cukup memori untuk memebuhi kebutuhan permintaan dari proses. Namun, karena memori kosong ini terbagi menjadi lubang-lubang kecil yang tersebar dalam memori, proses tidak dapat dialokasikan secara kontigu.

Ketika menggunakan algoritma alokasi memori first fit dan best fit terkena masalah fragmentasi. First fit yang mangalokasikan memori dengan lubang pertama yang cukup mempunyai kemungkinan lubang yang dialokasikan menghasilkan lubang kecil yang tidak dapat dialokasikan oleh proses selanjutnya. Selain itu, penggunaan best fit juga bermasalah karena terdapat kemungkinan alokasi memori akan menghasilkan lubang kecil yang tidak dapat dialokasikan oleh proses lain. Dibandingkan dengan algoritma worst fit, alokasi memori akan menghasilkan lubang yang besar, hal ini memungkinkan lubang sisa alokasi dapat dialokasikan oleh proses lain.

Perbandingan antara algoritma yang telah disebutkan sebelumnya tidak hanya seputar waktu eksekusi tiap algoritma, tetapi juga seputar masalah fragmentasi. Mungkin, secara waktu eksekusi, algoritma first fit adalah algoritma yang terbaik, karena akan berhenti ketika ditemukan lubang pertama yang cukup. Secara performansi, algoritma best fit mungkin dapat mengalokasikan memori secara mangkus karena tidak menyisakan lubang, walaupun terdapat kemungkinan menyisakan lubang kecil yang akan berpotensi menyebabkan fragmentasi. Algoritma worst fit juga harus dipertimbangkan karena walaupun secara performansi ataupun segi waktu tidak sebagus algoritma lainnya, secara lubang yang dihasilkan mengecilkan kemungkinan adanya fragmentasi.

III. PEMBAHASAN

Pembahasan ini akan mengambil contoh terdapat 8 lubang yang tersebar pada suatu memori. Lubang tersebut mempunyai panjang masing-masing 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, dan 15 MB. Pada *Input Queue,* terdapat 3 proses yang membutuhkan memori masing-masing 12 MB, 10 MB, dan 9 MB.

Ketiga proses pada *Input Queue* tersebut akan dialokasikan ke dalam memori dengan menggunakan algoritma First fit, Best fit, dan Worst fit. Selanjutnya, hasil dari ketiga algoritma tersebut akan dibandingkan untuk mendapatkan kesimpulan hasil paling optimal dari kasus di atas.

A. First Fit

Dari penjelasan yang telah disebutkan di atas dan dengan menganalisis algoritma ini untuk mendapatkan komponen-komponen greedy, didapatkan hasil analisis sebagai berikut :

1. Himpunan Kandidat (C)

Himpunan kandidat dari permasalahan di atas menyatakan himpunan lubang kosong. Himpunan kandidat akan berupa larik yang memuat elemen bernilai 10, 4, 20, 18, 7, 9, 12, dan 15.

2. Himpunan Solusi (S)

Himpunan solusi dari permasalahan di atas menyatakan himpunan lubang kosong yang dipilih untuk dialokasikan sesuai urutan proses yang datang.

3. Fungsi Seleksi

Fungsi seleksi akan memilih lubang pertama yang cukup dengan kebutuhan memori proses pada tahap pencarian.

4. Fungsi Kelayakan

Fungsi kelayakan disini adalah mengecek apakah lubang yang akan dialokasikan mencukupi kebutuhan memori proses atau tidak.

5. Fungsi Obyektif

Fungsi obyektif disini merepresentasikan pemilihan lubang berdasarkan lubang pertama yang kosong dan mencukupi kebutuhan. Diharapkan dapat meminimisasi waktu eksekusi dari pencarian alokasi memori.

Setelah menganalisis komponen di atas, dapat diperoleh skema dari algoritma first fit sebagai berikut.

```
function first_fit(C : array of integer, queue
: array of integer) -> array of integer
{ Mencari lubang alokasi memori dengan metode
first fit
  Masukan : himpunan kandidat dan input queue
  Keluaran : lubang yang terpilih secara urut,
mengembalikan -1 jika tidak ada
}

DEKLARASI

  S : array of integer #himpunan solusi
  cp : integer #proses sekarang
  found : boolean #cek lubang ketemu
```

```
ALGORITMA

  S <- {}#inisialisai s dengan array kosong

  for cp in queue
    i <- 0
    found <- false

    while (not found) and (i < C.length) do
        if (C[i] >= cp) then
          found <- true
        else
          i = i + 1
        endif
    endwhile

    if (found) then
        S <- S U {C[i]}
        C <- C - {C[i]}
    else
        S <- S U {-1}
    endif

  endfor

  return S
```

Pada kasus ini, penjabaran skema algoritma di atas adalah sebagai berikut,

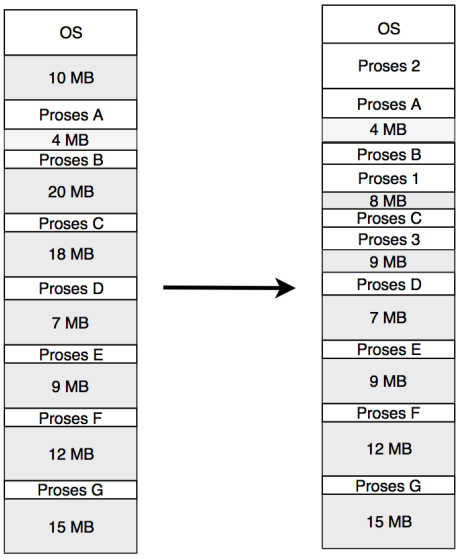
Pertama, untuk inisialisasi awal. Fungsi akan diinput dengan himpunan kandidat dan *input queue*. Setelah itu, himpunan solusi akan diinisialisasi dengan himpunan kosong.

Pada iterasi pertama, cp akan diisi dengan proses pertama yang berukuran 12 MB. Setelah itu, akan dilakukan iterasi mencari lubang kosong hingga ditemukan lubang kosong yang cukup. Pada iterasi pertama akan terhenti pada lubang berukuran 20 MB. Setelah itu, lubang tersebut akan dihilangkan dari himpunan kandidat dan lubang tersebut dimasukan ke dalam himpunan solusi sehingga himpunan solusi sekarang akan bernilai {20}.

Pada iterasi kedua, cp akan diisi dengan proses kedua yang berukuran 10 MB. Setelah itu, akan dilakukan iterasi mencari lubang kosong hingga ditermukan lubang yang cukup, pencarian akan dilakukan dari awal himpunan kandidat. Iterasi akan terhenti pada lubang yang berukuran 10MB. Setelah itu, lubang tersebut akan dihilangkan dari himpunan kandidat dan lubang tersebut dimasukan ke dalam himpuan solusi sehingga himpunan solusi sekarang akan bernilai {20,10}.

Pada iterasi ketiga, cp akan diisi dengan proses ketiga yang berukuran 9 MB. Setelah itu, akan dilakukan iterasi mencari lubang kosong hingga ditemukan lubang yang cukup, pencarian akan dilakukan dari awal. Iterasi akan terhenti pada lubang yang mempunyai ukuran 18 MB. Setelah itu, lubang tersebut akan dihilangkan dari himpunan kandidat dan lubang tersebut dimasukan ke himpunan solusi sehingga himpunan solusi sekarang akan bernilai {20,10,18}.

Setelah iterasi pada *input queue* telah selesai, maka algoritma ini akan mengembalikan himpunan solusi yang pada kasus ini bernilai {20,10,18}. Setelah itu, himpunan solusi ini akan digunakan sistem operasi untuk mengalokasikan proses sesuai dengan urutan *input queue*. Alokasi memori akan menyisakan lubang sebesar {8, 0, 9}. Hasil akhir dari alokasi memori ini akan menghasilkan ilustrasi berikut.



Gambar 3.1 Alokasi Memori dengan First Fit
Sumber : Dokumen Pribadi

B. Best Fit

Dari penjelasan yang telah disebutkan di atas dan dengan menganalisis algoritma ini untuk mendapatkan komponen-komponen greedy, didapatkan hasil analisis sebagai berikut :

1. Himpunan Kandidat (C)

Himpunan kandidat dari permasalahan di atas menyatakan himpunan lubang kosong. Himpunan kandidat akan berupa larik yang memuat elemen bernilai 10, 4, 20, 18, 7, 9, 12, dan 15.

2. Himpunan Solusi (S)

Himpunan solusi dari permasalahan di atas menyatakan himpunan lubang kosong yang dipilih untuk dialokasikan sesuai urutan proses yang datang.

3. Fungsi Seleksi

Fungsi seleksi akan memilih lubang cukup dengan ukuran yang paling kecil.

4. Fungsi Kelayakan

Fungsi kelayakan disini adalah mengecek apakah lubang yang akan dialokasikan mencukupi kebutuhan memori proses atau tidak.

5. Fungsi Obyektif

Fungsi obyektif disini merepresentasikan pemilihan lubang berdasarkan lubang cukup yang mempunyai ukuran terkecil. Diharapkan dapat meminimisasi sisa lubang dari hasil pencarian alokasi memori.

Setelah menganalisis komponen di atas, dapat diperoleh skema dari algoritma best fit sebagai berikut.

```
function best_fit(C : array of integer,queue :
array of integer) -> array of integer
{ Mencari lubang alokasi memori dengan metode
best fit
  Masukan : himpunan kandidat dan input queue
  Keluaran : lubang yang terpilih secara urut,
```

```
  mengembalikan -1 jika tidak ada
}

DEKLARASI

  S : array of integer #himpunan solusi
  cp : integer #proses sekarang
  ch : integer #lubang sekarang
  min : integer #lubang cukup terkecil

ALGORITMA

  S <- {}#inisialisai s dengan array kosong

  for cp in queue
     min <- infinity #min besar
     for ch in S
        if (ch >= cp) and (ch < min)then
           min <- ch
        endif
     endfor

     if (min ≠ infinity) then
        S <- S U {min}
        C <- C - {min}
     else
        S <- S U {-1}
  endfor

  return S
```

Pada kasus ini, penjabaran skema di atas adalah sebagai berikut,

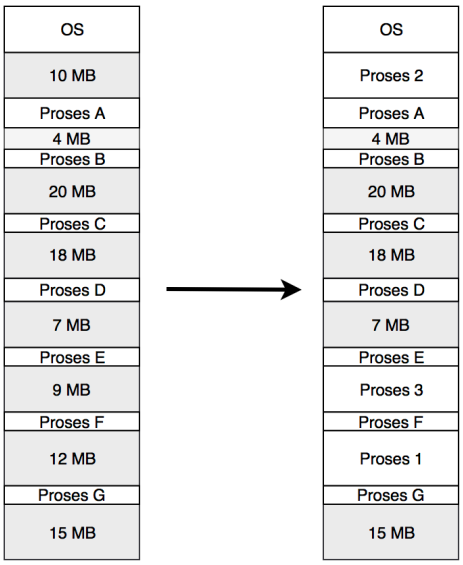
Pertama, sebagai inisialisasi awal, fungsi akan diiput dengan himpunan kandidat dan *input queue*. Setelah itu himpunan solusi akan diisi dengan himpunan kosong.

Pada iterasi pertama, cp akan diisi dengan proses pertama, yaitu proses dengan ukuran 12 MB. Min akan diisi dengan nilai besar. Setelah itu, akan dilakukan iterasi dari lubang pertama hingga lubang terakhir pada himpunan kandidat. Pada tiap lubang, dilakukan pengecekan untuk mengetes apakah lubang memadai kebutuhan proses dan apakah lubang mempunyai ukurang lebih kecil dari min. Hasil dari iterasi pertama mendapatkan lubang berukuran 12 MB. Lubang ini dimasukan ke himpunan solusi sehingga himpunan solusi bernilai {12} dan dihapus dari himpunan kandidat.

Pada iterasi kedua, cp akan diisi dengan proses kedua, yaitu proses dengan ukuran 10 MB. Min akan diisi dengan nilai besar. Setelah itu, akan dilakukan iterasi dari lubang pertama hingga lubang terakhir pada himpunan kandidat. Pada tiap lubang, dilakukan pengecekan untuk mengetes apakah lubang memadai kebutuhan proses dan apakah lubang mempunyai ukurang lebih kecil dari min. Hasil dari iterasi kedua mendapatkan lubang berukuran 10 MB. Lubang ini dimasukan ke himpunan solusi sehingga himpunan solusi bernilai {12, 10} dan dihapus dari himpunan kandidat.

Pada iterasi ketiga, cp akan diisi dengan proses ketiga, yaitu proses dengan ukuran 9 MB. Min akan diisi dengan nilai besar. Setelah itu, akan dilakukan iterasi dari lubang pertama hingga lubang terakhir pada himpunan kandidat. Pada tiap lubang, dilakukan pengecekan untuk mengetes apakah lubang memadai kebutuhan proses dan apakah lubang mempunyai ukurang lebih kecil dari min. Hasil dari iterasi ketiga mendapatkan lubang berukuran 9 MB. Lubang ini dimasukan ke himpunan solusi sehingga himpunan solusi bernilai {12, 10, 9} dan dihapus dari himpunan kandidat.

Setelah iterasi pada *input queue* telah selesai, maka algoritma ini akan mengembalikan himpunan solusi yang pada kasus ini bernilai {12,10,9}. Setelah itu, himpunan solusi ini akan digunakan sistem operasi untuk mengalokasikan proses sesuai dengan urutan *input queue*. Alokasi memori akan menyisakan lubang sebesar {0, 0, 0}. Hasil akhir dari alokasi memori ini akan menghasilkan ilustrasi berikut.



Gambar 3.2 Alokasi Memori dengan Best Fit
Sumber : Dokumen Pribadi

C. Worst Fit

Dari penjelasan yang telah disebutkan di atas dan dengan menganalisis algoritma ini untuk mendapatkan komponen-komponen greedy, didapatkan hasil analisis sebagai berikut :

1. Himpunan Kandidat (C)

Himpunan kandidat dari permasalahan di atas menyatakan himpunan lubang kosong. Himpunan kandidat akan berupa larik yang memuat elemen bernilai 10, 4, 20, 18, 7, 9, 12, dan 15.

2. Himpunan Solusi (S)

Himpunan solusi dari permasalahan di atas menyatakan himpunan lubang kosong yang dipilih untuk dialokasikan sesuai urutan proses yang datang.

3. Fungsi Seleksi

Fungsi seleksi akan memilih lubang cukup dengan ukurang yang paling besar.

4. Fungsi Kelayakan

Fungsi kelayakan disini adalah mengecek apakah lubang yang akan dialokasikan mencukupi kebutuhan memori proses atau tidak.

5. Fungsi Obyektif

Fungsi obyektif disini merepresentasikan pemilihan lubang berdasarkan lubang cukup yang mempunyai ukuran terbesar. Diharapkan dapat meminimisasi adanya fragmentasi dengan menghasilkan sisa lubang yang besar.

Setelah menganalisis komponen di atas, dapat diperoleh skema dari algoritma worst fit sebagai berikut.

```
function worst_fit(C : array of integer, queue
: array of integer) -> array of integer
{ Mencari lubang alokasi memori dengan metode
worst fit
  Masukan : himpunan kandidat dan input queue
  Keluaran : lubang yang terpilih secara urut,
mengembalikan -1 jika tidak ada
}

DEKLARASI

  S : array of integer #himpunan solusi
  cp : integer #proses sekarang
  ch : integer #lubang sekarang
  max : integer #lubang cukup terbesar

ALGORITMA

  S <- {}#inisialisai s dengan array kosong

  for cp in queue
     max <- 0 #max terkecil
     for ch in S
        if (ch >= cp) and (ch > max)then
           max <- ch
        endif
     endfor

     if (max ≠ 0) then
        S <- S U {max}
        C <- C - {max}
     else
        S <- S U {-1}
  endfor

  return S
```

Pada kasus ini, penjabaran skema di atas adalah sebagai berikut,

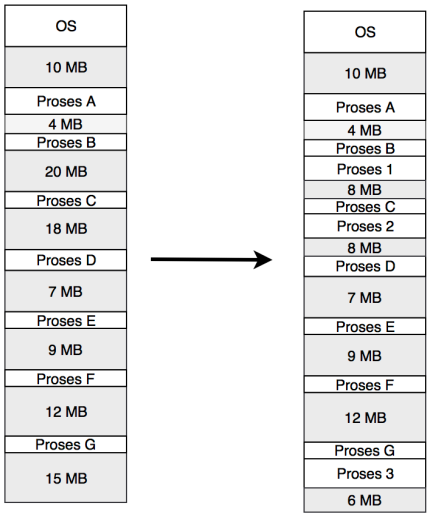
Pertama, untuk inisialisasi, fungsi akan diinput dengan himpunan kandidat dan *input queue*. Setelah itu, himpunan solusi akan diisi dengan himpunan kosong.

Pada iterasi pertama, cp akan diisi dengan proses pertama yang bernilai 12 MB. Max juga akan diisi dengan nilai yang kecil. Setelah itu, akan dilakukan iterasi untuk tiap lubang dan melakukan pengecekan apakah lubang tersebut memadai kebutuhan dari proses dan mempunyai ukuran lebih besar dari max. Hasil dari iterasi pertama akan menghasilkan lubang dengan ukuran 20 MB. Lubang ini akan dimasukan ke himpunan solusi menjadi {20} dan dihapus dari himpunan kandidat.

Pada iterasi kedua, cp akan diisi dengan proses kedua yang bernilai 10 MB. Max juga akan diisi dengan nilai yang kecil. Setelah itu, akan dilakukan iterasi untuk tiap lubang dan melakukan pengecekan apakah lubang tersebut memadai kebutuhan dari proses dan mempunyai ukuran lebih besar dari max. Hasil dari iterasi kedua akan menghasilkan lubang dengan ukuran 18 MB. Lubang ini akan dimasukan ke himpunan solusi menjadi {20, 18} dan dihapus dari himpunan kandidat.

Pada iterasi ketiga, cp akan diisi dengan proses ketiga yang bernilai 9 MB. Max juga akan diisi dengan nilai yang kecil. Setelah itu, akan dilakukan iterasi untuk tiap lubang dan melakukan pengecekan apakah lubang tersebut memadai kebutuhan dari proses dan mempunyai ukuran lebih besar dari max. Hasil dari iterasi ketiga akan menghasilkan lubang dengan ukuran 15 MB. Lubang ini akan dimasukan ke himpunan solusi menjadi {20, 18, 15} dan dihapus dari himpunan kandidat.

Setelah iterasi pada *input queue* telah selesai, maka algoritma ini akan mengembalikan himpunan solusi yang pada kasus ini bernilai {20,18,15}. Setelah itu, himpunan solusi ini akan digunakan sistem operasi untuk mengalokasikan proses sesuai dengan urutan *input queue*. Alokasi memori akan menyisakan lubang sebesar {8, 8, 6}. Hasil akhir dari alokasi memori ini akan menghasilkan ilustrasi berikut.



Gambar 3.3 Alokasi Memori dengan Worst Fit
Sumber : Dokumen Pribadi

IV. KESIMPULAN

Dari hasil pengujian yang telah dilakukan, didapatkan bahwa tiap algoritma mempunyai keuntungan dan kerugian masing-masing. Algoritma first fit berjalan dengan cepat dan menghasilkan lubang yang cukup besar. Algoritma best fit tidak menyisakan lubang sama sekali. Algoritma worst fit menghasilkan lubang dengan ukuran terbesar. Dari segi waktu, algoritma first fit merupakan yang terbaik. Dari segi performansi, algoritma best fit merupakan yang terbaik karena tidak menyisakan lubang. Dari segi fragmentasi, algoritma worst fit merupakan yang terbaik karena menyisakan lubang yang besar.

Pada pemilihan algoritma, tidak ada algoritma yang terbaik. Penggunaan algoritma pada kasus yang salah akan menghasilkan hasil terburuk. Parameter-parameter pada alokasi memori seperti waktu pencarian dan performansi perlu dipertimbangkan. Oleh karena itu, pemilihan algoritma yang

tepat di antara ketiga algoritma alokasi memori harus diperhatikan untuk mendapatkan hasil yang diharapkan.

IV. UCAPAN TERIMA KASIH

Pertama, saya ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa, atas rahmatnya saya dapat menyelesaikan makalah Strategi Algoritma. Tak lupa saya berterima kasih kepada orang tua saya, atas perjuangan dan dukungan mereka saya dapat terbantu mendapatkan pendidikan ini. Saya juga mengucapkan terima kasih kepada Bapak Rinaldi Munir, Ibu Nur Ulfa Maulidevi, dan Ibu Masayu Leylia Khodra selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma. Saya juga mengucapkan terima kasih kepada rekan-rekan saya yang membantu dan memberikan ide kepada saya dalam pembuatan makalah Strategi Algoritma ini.

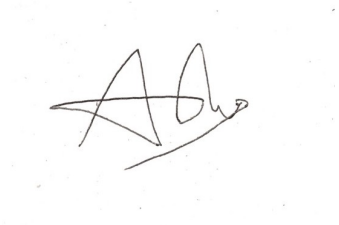
REFERENSI

[1] Munir, Rinaldi. "Diktat Kuliah IF2211 Strategi Algoritma", Informatika. Bandung : 2009

[2] Silberschatz, Abraham et al. "Operating System Concepts" 9th ed. Courier press. United States of America : 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, 12 Mei 2018
Muhammad Sulthan Adhipradhana, 13516035