**ALGORITMA DAN PEMEROGRAMAN**

**PAPER ALGORITMA FUZZY DAN GREEDY**

Dosen Pengampu:

Slamet Risnanto, ST., M.Kom.



Oleh:

Isep Lutpi Nur (2113191079)

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS SANGGA BUANA YPKP BANDUNG**

**2020**

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Dalam matematika dan komputasi, algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, dalam hal ini berbeda dengan heuristik. Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan (logika Boolean dan perbandingan) sampai tugasnya selesai.

Desain dan analisis algoritma adalah suatu cabang khusus dalam ilmu komputer yang mempelajari karakteristik dan performa dari suatu algoritma dalam menyelesaikan masalah, terlepas dari implementasi algoritma tersebut. Dalam cabang disiplin ini algoritma dipelajari secara abstrak, terlepas dari sistem komputer atau bahasa pemrograman yang digunakan. Algoritma yang berbeda dapat diterapkan pada suatu masalah dengan kriteria yang sama.

Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

Ada banyak algoritma yang bisa di terapkan untuk menyelesaikan masalah, yang semuanya memiliki karakteristik, kelebihan dan kekurangannya masing-masing. Salah satunya adalah algoritma Greedy, yaitu yang umum di gunakan dalam pemecahan masalah optimasi.

* 1. **Rumusan masalah**

Berdasarkanlatar belakang yang telah ada, rumusan masalah dapat diuraikan sebagai berikut:

1. Apa yang dimaksud dengan algoritma Greedy?
2. Bagaimana karakteristik algoritma Greedy?
3. Apa saja kelebihan dan kekurangan dari algoritma Greedy?
4. Bagaimana skema umumdan contoh algoritma Gredy?
5. Apa yang dimaksud dengan Knapsack?
6. Bagaimana menyelesaikan permasalahan Knapsack?
7. Bagaimana cara mengimplementasikan algoritma Greedy ke dalam *job schedulling*?
   1. **Tujuan**

Berdasarkan rumusan masalah diatas, adapun tujuan kami adalah sebagai berikut :

1. Untuk mengetahui pengertian dari algoritma Greedy
2. Untuk mengetahui karakteristik apa saja yang terdapat pada algortima Greedy
3. Untuk mengetahui kelebihan dan kekurangan apa saja dari algoritma Greedy
4. Untuk mengetahui metode skema umum besera contohnya pada algoritma Greedy
5. Untuk mengetahui pengertian dari Knapsack
6. Untuk mengetahui cara penyelesaian masalah pada Knapsack
7. Untuk mengetahui cara mengimplementasikan algoritma Greedy ke dalam *job schedulling.*

**BAB II**

**PEMBAHASAN**

* 1. **Algoritma Greedy**

Dalam kehidupan sehari hari, banyak terdapat persoalan yang menuntut pencarian solusi optimum. Persoalan tersebut dinamakan persoalan optimasi(optimization Problems). Persoalan Optimasi adalah persoalan yang tidak hanya mencari sekedar solusi, tetapi mencari solusi terbaik. Solusi terbaik adalah solusi yang memiliki nilai minimum atau meksimum dari sekumpulan alternatif solusi yang mungkin Algoritma Greedy adalah salah satu algoritma yang dapat digunakan untuk mendapatkan solusi terbaik dan merupakan algoritma yang paling populer dalam hal ini. Secara Harfiah Greedy artinya rakus atau tamak, sifat yang berkonotasi negatif. Orang yang memiliki sifat ini akan mengambil sebanayak mungkin atau mengambil yang paling bagus atau yang paling mahal. Sesuai dengan arti tersebut, Prinsip Greedy adalah take what you can get now. Dalam kehidupan sehari hari Greedy dapat digunakan dalam masalah seperti :

1. Memilih beberapa jenis investasi
2. Mencari jalur tersingkat

Algoritma Greedy membentuk solusi langkah per langkah. Pada setiap langkah tersebut akan dipilih keputusan yang paling optimal. Keputusan tersebut tidak perlu memperhatikan keputusan selanjutnya yang akan diambil, dan keputusan tersebut tidak dapat diubah lagi pada langkah – langkah selanjutnya. Prinsip utama algoritma greedy adalah prinsip “*take what you can get now!*”. Maksud dari prinsip tersebut adalah sebagai berikut; pada setiap langkah dalam algoritma greedy kita ambil keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah–langkah selanjutnya, lalu kita namakan solusi tersebut dengan optimum lokal. Dengan cara pengambilan nilai optimum lokal pada setiap langkah diharapkan akan tercapainya optimum global, yaitu tercapainya solusi optimum yang melibatkan keseluruhan langkah dari awal sampai akhir.

* 1. **Kelebihan dan Kekurangan Algoritma Greedy**

2.2.1 Kelebihan Algoritma Greedy

Apabila kita melihat dari definisi dan pengertian algoritma greedy, maka bisa simpulkan bahwa kelebihan dari algoritma greedy adalah cepat dalam bertindak alias fast response. Apabila sebuah program membutuhkan penyelesaian masalah secara instant dan juga cepat, algoritma greedy adalah salah satu metode yang tepat. Algoritma greedy tidak membutuhkan waktu lama untuk memikirkan opsi – opsi lain yang bisa dilakukan, serta tidak perlu mempertimbangkan baik buruk serta konsekuensi dari apa yang diputuskan.

2.2.2 Kekurangan Algoritma Greedy

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada.
2. Pemilihan fungsi SELEKSI: Mungkin saja terdapat beberapa fungsi SELEKSI yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma bekerja dengan benar dan menghasilkan solusi yang benar-benar optimum. Karena itu, pada sebagian masalah algoritma.
3. Greedy tidak selalu berhasil memberikan solusi yang benar-benar optimum.
   1. **Skema Umum Algoritma Greedy**

Elemen – elemen yang digunakan dalam penerapan algoritma greedy antara lain :

1. Himpunan Kandidat, C.

Himpunan yang berisi elemen pembentuk solusi.

1. Himpunan Solusi, S.

Himpunan yang terpilih sebagai solusi persoalan.

1. Fungsi Seleksi, SELEKSI.

Fungsi yang memilih kandidat yang paling mungkin untuk mencapai solusi optimal.

1. Fungsi Kelayakan, LAYAK.

Fungsi yang memeriksa apakah suatu kandidat yang dipilih dapat memberikan solusi yang layak. Maksudnya yaitu apakah kandidat tersebut bersama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada.

1. Fungsi SOLUSI

Fungsi yang mereturn boolean. True jika himpunan solusi yang sudah tebentuk merupakan solusi yang lengkap; False jika himpunan solusi belum lengkap.

1. Fungsi Objektif

Fungsi yang mengoptimalkan solusi.

Skema umum dari algoritma greedy dapat kita tuliskan sebagai berikut :

1. Inisialisasi S dengan kosong.
2. Pilih sebuah kandidat dari C (dengan SELEKSI).
3. Kurangi C dengan kandidat yang telah terpilih di atas.
4. Periksa apakah kandidat yang dipilih tersebut bersama – sama dengan himpunan solusi membentuk solusi yang layak (dengan LAYAK). Jika ya, masukkan kandidat ke himpunan solusi; jika tidak buang kandidat tersebut dan tidak perlu ditelaah lagi.
5. Periksa apakah himpunan solusi yang sudah terbentuk telah memberikan solusi yang lengkap (dengan SOLUSI). Jika ya, berhenti; jika tidak, ulangi dari langkah 2.
   1. **Study kasus Algoritma Greedy**

Soal :

Masalah Penukaran Uang

Diberikan uang senilai A. Tukar A dengan koin-koin uang yang ada. Berapa jumlah minimum koin yang diperlukan untuk penukaran tersebut?

Jawab :

Tersedia banyak koin 1, 5, 10, 25

Uang senilai A = 32 dapat ditukar dengan banyak cara berikut:

32 = 1 + 1 + … + 1 (32 koin)

32 = 5 + 5 + 5 + 5 + 10 + 1 + 1 (7 koin)

32 = 10 + 10 + 10 + 1 + 1 (5 koin)

… dst

Minimum: 32 = 25 + 5 + 1 + 1 (4 koin)

Penjelasan :

1. Himpunan kandidat: himpunan koin yang merepresentasikan nilai 1, 5, 10, 25, paling sedikit mengandung satu koin untuk setiap nilai.
2. Himpunan solusi: total nilai koin yang dipilih tepat sama jumlahnya dengan nilai uang yang ditukarkan.
3. Fungsi seleksi: pilihlah koin yang bernilai tertinggi dari himpunan kandidat yang tersisa.
4. Fungsi layak: memeriksa apakah nilai total dari himpunan koin yang dipilih tidak melebihi jumlah uang yang harus dibayar.
5. Fungsi obyektif: jumlah koin yang digunakan minimum.

Kesimpulan

Strategi greedy:

Pada setiap langkah, pilihlah koin dengan nilai terbesar dari himpunan koin yang tersisa.

Misal: A = 32, koin yang tersedia: 1, 5, 10, dan 25

Langkah 1: pilih 1 buah koin 25 (Total = 25)

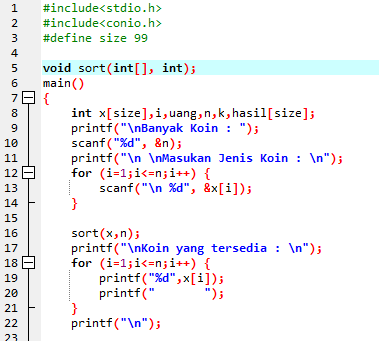
Langkah 2: pilih 1 buah koin 5 (Total = 25 + 5 = 30)

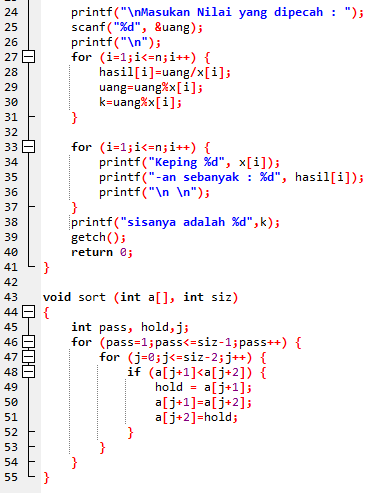
Langkah 3: pilih 2 buah koin 1 (Total = 25+5+1+1= 32)

Solusi: Jumlah koin minimum = 4 (solusi optimal!)

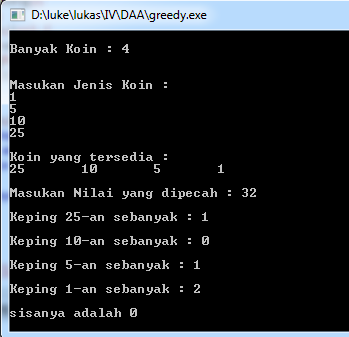
* 1. **Contoh Program Algoritma Greedy**

Input Program





Output Program



* 1. **Knapsack**

Knapsack adalah tas atau karung yang digunakan untuk memasukkan sesuatu. Tapi tidak semua barang bisa ditampung kedalam karung tersebut. Karung tersebut hanyadapat menyimpan beberapa objek dengan total ukurannya (weight) lebih kecil atau samadengan ukuran kapasitas karung.Knapsack merupakan salah satu masalah yang sering munculdalam kasus optimasi dan beberapa strategi algoritma yang bisa menghasilkan solusi optimal, diantaranya adalah Brute Force, algotitma genetika, Algoritma greedy. Pada makalah ini masalah knapsack akan diselesaikan dengan Greedy Algorithm yaitu solusi yang mencari nilai optimum. Algoritma ini memecahkan permasalahan langkah per langkah, yaitu:

1. Mengambil pilihan terbaik yang bisa diperoleh saat itu juga tanpa memperhatikan konsekuensi kedepan (prinsip “take what you can get now!”).
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Dalam algoritma Greedy ada beberapa strategi yang digunakan untuk memilih objek yang akan dimasukkan kedalam knapsack:

1. Greedy by profit. Pada setiap langkah, knapsack diisi dengan objek yang mempunyai keuntungan terbesar.Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yang paling menguntungkan terlebih dahulu.
2. Greedy by weight. Pada setiap langkah, knapsack diisi dengan objek yang mempunyai berat paling ringan.Strategi ini mencoba memaksimumkan keuntungan dengan memasukkan sebanyak mungkin objek ke dalam knapsack.
3. Greedy by density. Pada setiap langkah, knapsack diisi dengan objek yang mempunyai densitas, pi /witerbesar. Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yangmempunyai keuntungan per unit berat terbesar
   1. **Masalah Knapsack**

Masalah knapsack adalah suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang di mana setiap barang tersebut mempunyai berat dan profit masing masing, sehingga dari pemilihan barang tersebut didapatkan profit yang maksimum. Dalam dunia nyata permasalahan Knapsack ini sering sekali digunakan terutama pada bidang (jasa) pengangkutan barang (seperti pengangkutan peti kemas dalam sebuah kapal). Dalam usaha tersebut, diinginkan suatu keuntungan yang maksimal untuk mengangkut barang yang adadengan tidak melebihi batas kapasitas yang ada. Berdasarkan persoalan tersebut, diharapkanada suatu solusi yang secara otomatis dalam mengatasi persoalan itu. Problem Knapsack adalah permasalahan optimasi, dimana kita harus mencari solusi terbaik dari banyak kemungkinan yang dihasilkan.

* 1. **Knapsack 0/1**

Knapsack 0/1, yaitu suatu objek diambil seluruh bagiannya atau tidak sama sekali. Cara terbaik agar menguntungkan, bukan hanya dari hasilnya optimal tetapi juga banyaknya langkah yang dibutuhkan Diberikan n buah objek dan sebuah knapsack dengan kapasitas bobot W. Setiap objek memiliki properti bobot (weigth) wi dan keuntungan(profit) pi. Persoalan adalah memilih memilih objek-objek yang dimasukkan ke dalam knapsack sedemikian sehingga memaksimumkan keuntungan. Total bobot objek yang dimasukkan ke dalam knapsack tidak boleh melebihi kapasitas knapsack. Solusi persoalan dinyatakan sebagai vektor n-tupel :

X = {x1, x2, … , xn}

xi = 1 jika objek ke-i dimasukkan ke dalam knapsack,

xi = 0 jika objek ke-i tidak dimasukkan.

Persoalan 0/1 Knapsack dapat kita pandang sebagai mencari himpunan bagian (subset) dari keseluruhan objek yang muat ke dalam knapsack dan memberikan total keuntungan terbesar.

* 1. **Study Kasus Knapsack**

Data awal :

w1 = 6; p1 = 12 w2 = 5; p2 = 15

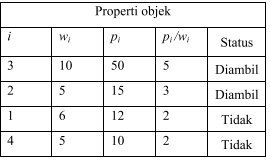
w3 = 10; p3 = 50 w4 = 5; p4 = 10

Kapasitas knapsack W = 16

Solusi dengan algoritma greedy :

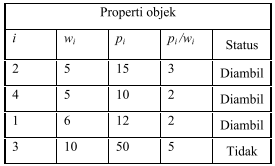
1. Greedy By Profit

Pada setiap langkah Knapsack diisi dengan obyek yang mempunyai keuntungan terbesar. Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yang paling menguntungkan terlebih dahulu. Pertama kali dilakukan adalah mengurutkan secara menurun obyek-obyek berdasarkan profitnya .Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau (sudah tidak ada obyek lagi yang bisa dimasukan).



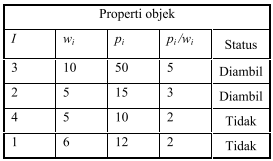
1. Greedy By Wight

Pada setiap langkah, knapsack diisi dengan objek yang mempunyai berat paling ringan. Strategi ini mencoba memaksimumkan keuntungan dengan memasukan sebanyak mungkin objek kedalam knapsack. Pertama kali yang dilakukan adalah mengurutkan secara menaik objek-objek berdasarkan weight-nya. Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau ( sudah tidak ada obyek lagi yang bisa dimasukan).

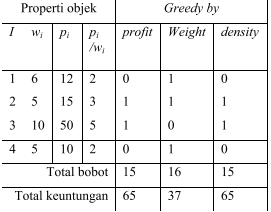


1. Greedy By Density

Pada setiap langkah, knapsack diisi dengan obyek yang mempunyai densitas terbesar (perbandingan nilai dan berat terbesar). Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yang mempunyai keuntungan per unit berat terbesar. Pertama kali yang dilakukan adalah mencari nilai profit per unit / density dari tiap-tiap objek. Kemudian obyek-obyek diurutkan berdasarkan densitasnya.



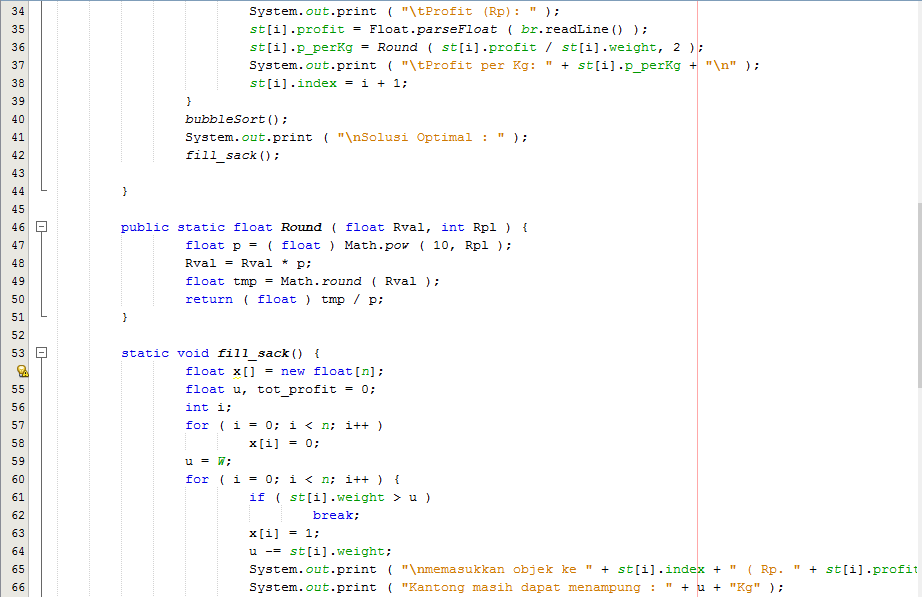
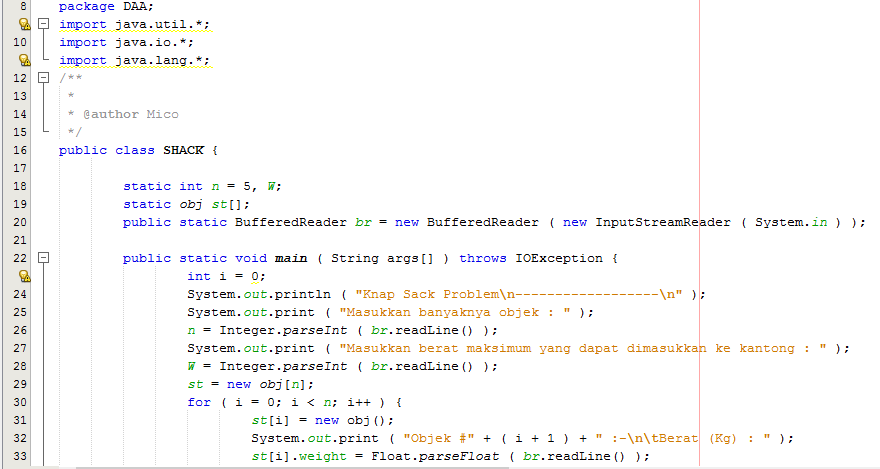
Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau (sudah tidak ada obyek lagi yang bisa dimasukan).

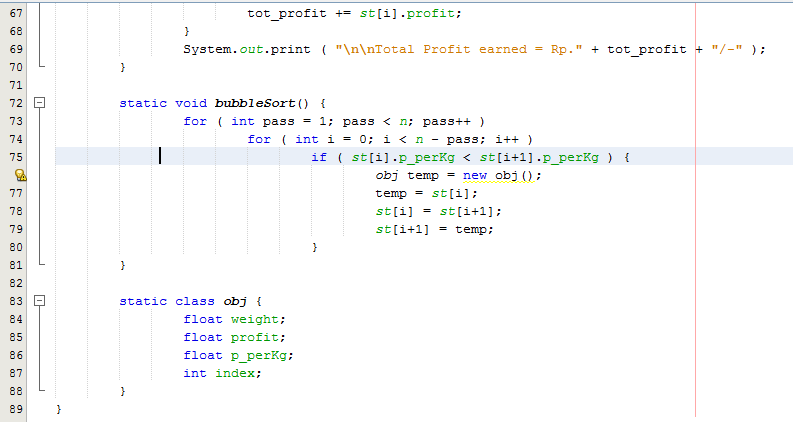


JADI Penggunaan 3 strategi diatas tidak menjamin akan memberikan solusi optimal karna total keuntungan ada 2 yang sama yaitu profit dan density.

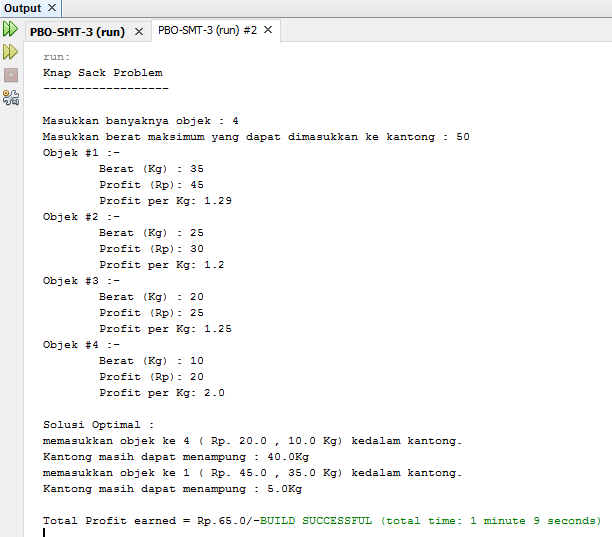
* 1. **Contoh Program Knapsack**

Input program





Output



* 1. ***Job Scedulling***

*Job Scedulling* adalah kegiatan pengalokasian sumber-sumber atau mesin-mesin yang ada untuk menjalankan sekumpulan tugas dalam jangka waktu tertentu(Baker,1974). *Job Scedulling* juga di definisikan sebagai rencana pengaturan urutan kerja serta pengalokasian sumber, baik waktu maupun fasilitas untuk setiap operasi yang harus diselesaikan (Vollman,1998). Dari beberapa definisi yang telah disebutkan maka dapat ditarik satu definisi “Penjadwalan adalah suatu kegiatan perancangan berupa pengalokasian sumber daya baik mesin maupun tenaga kerja untuk menjalankan sekumpulan tugas sesuai prosesnya dalam jangka waktu tertentu”.

Unsur-unsur vital didalam model-model penjadwalan adalah sumber-sumber dan tugas-tugas.Sumber-sumber biasanya dikenal dengan mesin-mesin sedangkan tugas-tugas dikenal dengan *job* atau pekerjaan.

Menurut L. Bethel dalam bukunya “*Industrial Organization and Management*” memberikan definisi penjadwalan atau scheduling sebagai berikut :

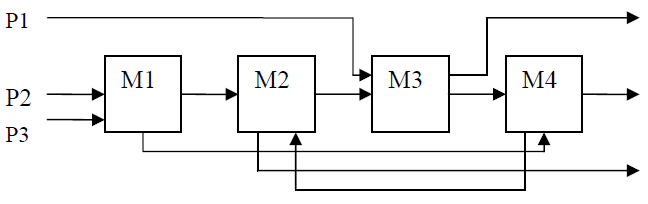
“Penjadwalan produksi merupakan proses penentuan pekerjaan yang akan dilakukan. Penjadwalan (*scheduling*) adalah suatu tahapan dari pengawasan produksi yang menetapkan pekerjaan dalam urut-urutan yang sesuai dengan prioritasnya dan kemudian dilengkapi pelaksanaan rencana tersebut pada waktu yang tepat dengan urutan yang benar, sehingga berhubungan dengan kapan suatu pekerjaan akan dilaksanakan pada suatu bagian produksi”.

* 1. **Tujuan penjadwalan**

Tujuan penjadwalan, adalah sebagai berikut:

1. Menurut Baker (1974), tujuan penjadawalan umumnya adalah sebagai berikut:
2. Meningkatkan produktifitas mesin, yaitu dengan mengurangi waktu mesin menganggur.
3. Mengurangi persediaan barang setengah jadi dengan jalan mengurangi jumlah rata-rata pekerjaan yang menunggu dalam antrian suatu mesin karena mesin tersebut sibuk.
4. Mengurangi keterlambatan suatu pekerjaan. Setiap pekerjaan mempunyai batas waktu (due date) penyelesaian, jika pekerjaan tersebut diselesaikan melewati batas waktu yang ditentukan maka pekerjaan tersebut dinyatakan terlambat. Dengan metoda penjadwalan maka keterlambatan ini dapat dikurangi, baik waktu maupun frekuensi.
5. Menurut Narasimhan (1985), penjadwalan yang baik seharusnya simpel, mudah dimengerti dan dapat dilaksanakan oleh pihak manajemen dan oleh siapapun yang menggunakannya. Aturan-aturan penjadwalan seharusnya cukup kuat tetapi mempunyai tujuan yang realistis sehingga cukup flexible untuk memecahkan masalah yang tidak terprediksi sebelumnya dan membolehkan satu perencanaan ulang.
6. Bedworth (1987) mengidentifikasikan beberapa tujuan dari aktivitas penjadwalan, adalah sebagai berikut:
7. Meningkatkan penggunaan sumber daya atau mengurangi waktu tunggunya, sehingga total waktu proses dapat berkurang dan produktivitas dapat meningkat.
8. Mengurangi persediaan barang setengah jadi atau mengurangi sejumlah pekerjaan menunggu dalam antrian ketika sumber daya yang ada masih mengerjakan tugas yang lain. Teori Baker mengatakan, jika aliran kerja suatu jadwal konstan, maka antrian yang mengurangi rata-rata waktu alir akan mengurangi rata-rata persediaan barang setengah jadi.
9. Mengurangi beberapa kelambatan pada pekerjaan yang mempunyai batas waktu penyelesaian sehingga akan meminimalisasi penalty cost (biaya kelambatan).
10. Membantu pengambilan keputusan mengenai perencanaan kapasitas pabrik dan jenis kapasitas yang dibutuhkan sehingga penambahan biaya yang mahal dapat dihindarkan.
    1. **Job Shop Scedulling**

*Job shop scheduling* secara umum lebih dikenal sebagai *shop floor control*, yang merupakan kegiatan penyusunan *input* (memasang yang diperlukan) menjadi *input*. Proses produksi dengan aliran *job shoop* berarti proses produksi dengan pola aliran atau rute proses pada tiap mesin yang spesifik untuk setiap pekerjaan dan mungkin berbeda untuk tiap *job*. Akibat aliran proses yang tidak searah ini, maka setiap *job* yang akan diproses pada satu mesin dapat merupakan *job* yang baru atau *job* dalam proses dan *job* yang keluar dari suatu mesin dapat merupakan *job* jadi atau *job* dalam proses. Dalam *job shop* tidak semua pkerjaan diasumsikan akan tepat satu kali menerima m operasi, dan untuk beberapa pekerjaan membutuhkan beberapa operasi dalam sebuah mesin.



Aliran Job Shop dimana P (Job) dan M (Machine)

Beberapa kriteria penting dalam masalah penjadwalan *job shop* adalah :

1. Kedatangan *job*

Masalah *job* yang ada selama ini adalah masalah statik yang biasa disebut *snapshot* dari sebuah sistem pada titik dalam waktu dan diproses untuk menyelesaikan masalah berdasarkan nilai yang berlaku walaupun banyak solusi algoritma tetapi perlu dipertimbangkan bahwa sebuah masalah dapat menjadi statik, pada kenyataannya masalah tersebut menjadi dinamik dengan sendirinya.

1. Jumlah dan mesin yang ada.

*Job shop* harus memiliki tampilan yang unik dengan mengimplementasikan solusi yang diperoleh dari penjadwalan dengan algoritma. Sebagai contoh, asumsi umur dari mesin yang memberikan tipe yang identik. Hal ini tidak sealu menjadi masalah, mesin yang utama dilihat dari berbagai jenis variasi seperti kondisi dari mesin atau kemampuan dati operator.

1. Jumlah Tenaga kerja di lantai produksi.

Jumlah tenaga kerja dan jenis mesin dari lantai produksi ditentukan dari kapasitas yang tersedia. Perencanaan kapasitas adalah aspek yang penting dari perencanaan produksi.

1. Aliran utama produksi.

Solusi diperoleh dari penjadwalan algoritma diperlukan *job* yang lengkap pada jumlah *order* yang tetap. Bagaimanapun pengurutan *job* berdasarkan hasil mesin dalam contoh aliram material dalam sistem.

1. Evaluasi dari alternatif solusi.

Pilihan yang obyektif ditentukan dari kenyamanan dan keefektifan dari aturan pengrutan. Secara umum lebih dari satu objek yang penting srhingga mustahil untukmenentukan aturan yang optimal.

*Job shop scheduling problem* merupakan salah satu masalah penjadwalan yang memiliki kendala urutan pemrosesan tugas, dan setiap tugas harus melalui setiap mesin tepat satu kali. Terdapat dua jenis metode yang biasa digunakan untuk menyelesaikan masalah *job shop scheduling problem*. Metode eksak, seperti pemrograman linier dan pemrograman nonlinier, dapat digunakan untuk ukuran *job shop scheduling problem* yang kecil. Sedangkan untuk ukuran masalah yang besar, digunakan suatu pendekatan secara aproksimasi, seperti *local search*, *simulated annealing*, *genetic algorithm*, *tabu search*, dan *ant colony optimization*.

Di dalam *Job shop scheduling problem*, diberikan suatu masalah dengan *n* pekerjaan dan *m* mesin. Setiap pekerjaan harus diproses tepat satu kali pada setiap mesin sesuai dengan urutannya masing-masing. Karena tugas adalah kombinasi dari pekerjaan dan mesin, maka dua tugas dari suatu pekerjaan tidak boleh dikerjakan pada waktu yang bersamaan, dan setiap mesin hanya dapat memproses paling banyak satu pekerjaan pada satu waktu. Tujuan dari *job shop scheduling problem* adalah mencari suatu jadwal yang meminimumkan waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan.

* 1. **Metode Dalam *Job Scedulling***

Dalam penjadwalan, banyak skema yang berbeda digunakan untuk menentukan pekerjaan spesifik mana yang akan dijalankan. Beberapa parameter yang dapat dipertimbangkan adalah sebagai berikut :

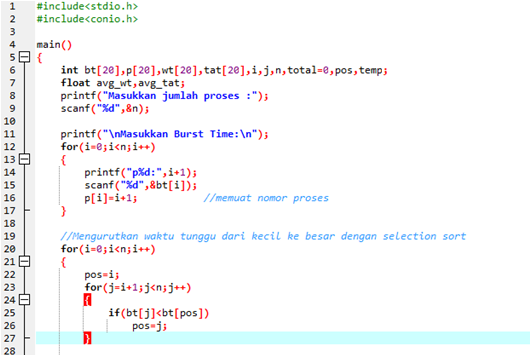
1. SPT (*Shortest Process Time*) : memilih operasi dengan waktu operasi yang terpendek.
2. LPT (*Longest Process Time*) : memilih operasi dengan waktu operasi terpanjang.
3. RANDOM (*Random Assignment*) : memilih operasi secara acak (random).
4. FCFS (*Frist Come, Frist Served*) : *job* yang yang pertama datang dikerjakan terlebih dahulu.
5. LCFS (*Last Come, Frist Served*) : *job* yang terakhir datang yang dikerjakan terlebih dahulu.
6. MWKR (*Most Work Remaining*) : memilih operasi dengan pekerjaan yang mempunyai pekerjaan sisa yang paling banyak yang dikerjakan terlebih dahulu.
7. EDD (*Earliest Due Date*) : memilih pekerjaan dengan due date yang yang paling awal.
   1. **Study Kasus**

Terlampir (dalam bentuk video), atau bisa di unduh pada link berikut ini :

1. <https://www.youtube.com/watch?v=jkJUrbsvlfc&t=234s>
2. <https://www.youtube.com/watch?v=KBGzqVWnYY4>
3. https://www.youtube.com/watch?v=MfaQjs3m\_tw&t=1s
   1. **Contoh Program**

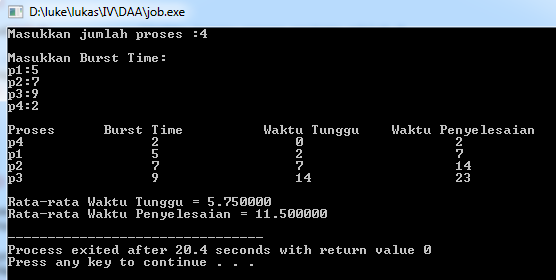
Pada contoh program berikut menggunakan aturan SPT (Shortest Process Time), yaitu memilih operasi dengan waktu operasi yang terpendek.

Input Program





Output Program



**BAB III**

**PENUTUP**

* 1. **Kesimpulan**

Algoritma Greedy adalah salah satu algoritma yang dapat digunakan untuk mendapatkan solusi terbaik dan merupakan algoritma yang paling populer dalam hal ini. Algoritma Greedy membentuk solusi langkah per langkah. Pada setiap langkah tersebut akan dipilih keputusan yang paling optimal. Prinsip Greedy adalah take what you can get now. Dalam kehidupan sehari hari Greedy dapat digunakan dalam masalah seperti :

* 1. Memilih beberapa jenis investasi
  2. Mencari jalur tersingkat

Kelebihan dari algoritma greedy adalah cepat dalam bertindak alias fast response. Apabila sebuah program membutuhkan penyelesaian masalah secara instant dan juga cepat, algoritma greedy adalah salah satu metode yang tepat. Masalah knapsack adalah suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang di mana setiap barang tersebut mempunyai berat dan profit masing masing, sehingga dari pemilihan barang tersebut didapatkan profit yang maksimum. Dalam algoritma greedy ada 3 strategi yang digunakan untuk memilih objek yang akan dimasukkan kedalam knapsack yaitu greedy by weight, greedy by profit, greedy density. Penjadwalan adalah suatu kegiatan perancangan berupa pengalokasian sumber daya baik mesin maupun tenaga kerja untuk menjalankan sekumpulan tugas sesuai prosesnya dalam jangka waktu tertentu

* 1. **Saran**

Menyadari bahwa penulis masih jauh dari kata sempurna, kedepannya penulis akan lebih fokus dan details dalam menjelaskan tentang makalah di atas dengan sumber - sumber yang lebih banyak yang tentunya dapat di pertanggung jawabkan. Untuk saran bisa berisi kritik atau saran terhadap penulisan juga bisa untuk menanggapi terhadap kesimpulan dari bahasan makalah yang telah di jelaskan.

**DAFTAR PUSTAKA**

aryo.lecture.ub.ac.id/files/2013/03/DAA-IV-Greedy-Algorithm.pptx

http://www.metode-algoritma.com/2013/02/contoh-program-algoritma-greedy.html

http://agusodow.blogspot.co.id/2011/12/algoritma-greedy.html

http://dosenit.com/ilmu-komputer/pengertian-algoritma-brute-force-dan-greedy

https://www.youtube.com/watch?v=z68CE9mbK98

http://ibrohim-almawardi.blogspot.co.id/2015/05/algoritma-greedy-dan-contoh-progamnya.html

https://www.techopedia.com/definition/7882/job-scheduling

http://www.thecrazyprogrammer.com/2014/08/c-program-for-shortest-job-first-sjf.html

http://elib.unikom.ac.id/files/disk1/64/jbptunikompp-gdl-s1-2006-jovanmaxyt-3157-bab-2.pdf\

http://id.shvoong.com/social-sciences/economics/2235594pengertian-penjadwalan-produksi/#ixzz1qPz1cIPy

https://docs.google.com/viewer?a=v&q=cache:AmOaWYdGrQcJ:elib.unikom.ac.id/download.php?

https://www.youtube.com/watch?v=jkJUrbsvlfc&t=234s

https://www.youtube.com/watch?v=KBGzqVWnYY4&t=4s

https://www.youtube.com/watch?v=MfaQjs3m\_tw

https://www.youtube.com/watch?v=h-358yaQWjw