APLIKASI ALGORITMA GREEDY DALAM INTERVAL SCHEDULING

Rizky Delfianto – NIM 13507032

Program Studi Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha 10 Bandung Indonesia 40132
e-mail: if17032@students.if.itb.ac.id

ABSTRAK

Algoritma *Greedy* merupakan algoritma yang ideal untuk menyelesaikan permasalahan optimasi baik maksimasi atau minimasi. Selain itu, algoritma *Greedy* juga sangat cocok untuk membantu dalam permasalahan penjadwalan. Salah satu permasalahan penjadwalan ayng dapat diselesaikan dengan algoritma *Greedy* adalah permasalahan *Interval Scheduling*, yaitu sebuah permasalahan dalam perancangan algoritma dimana diberikan *resources* berupa sejumlah pekerjaan yang diketahui waktu mulai dan waktu selesainya dan tujuannya adalah bagaimana mendapatkan sebanyak mungkin pekerjaan yang dapat diselesaikan dalam selang waktu tertentu.

Pendekatan algoritma *Greedy* untuk permasalahan ini ada empat macam, *Earliest Start Time*, *Earliest Finish Time*, *Shortest Interval*, dan *Fewest Conflicts*. Namun tiga dari empat pendekatan tersebut tidak menghasilkan solusi yang optimal. Satu-satunya pendekatan yang menghasilkan solusi optimal adalah *Earliest Finish Time* (EFT). Pernyataan tersebut dapat dibuktikan dengan metode induksi matematika.

Kata kunci: Algoritma *Greedy*, *Interval Scheduling*, EFT, induksi matematika.

1. PENDAHULUAN

Dalam menyelesaikan permasalahan optimasi, algoritma *Greedy* adalah algoritma yang paling sering digunakan. Namun, seperti halnya algoritma *Greedy* pada umumnya, tidak semua algoritma memberikan hasil yang optimal. Untuk dapat mengetahui apakah sebuah algoritma memberikan hasil yang optimal, tidak cukup jika hanya membuktikan dengan contoh-contoh saja namun juga harus dengan menggunakan metode pembuktian yang ilmiah.

Salah satu metode pembuktian ilmiah adalah induksi matematika. Induksi matematika digunakan untuk melakukan pengecekan terhadap hasil proses yang terjadi secara berulang sesuai dengan pola tertentu. Induksi matematika biasanya digunakan untuk membuktikan pernyataan yang bersifat universal.

Induksi matematika memiliki beberapa tahapan, *Basis Step*, *Inductive Step*, dan *Conclusion*. Misalkan S(n) dalah sebuah fungsi *proporsitional*. Pada tahap basis harus ditunjukkan bahwa S(1) benar. Pada tahap induksi akan diasumsikan bahwa S(k) benar dan harus dibuktikan dengan menggunakan S(k) bahwa S(k+1) juga benar. Sedangkan pada tahap konklusi cukup dinyatakan bahwa S(n) benar.

Berikut adalah contoh pernyataan yang akan dibuktikan dengan induksi matematika. Buktikan bahwa : $n^3 + 2n$ adalah kelipatan 3 untuk setiap n bilangan bulat positif.

Basis : Untuk n = 1 akan diperoleh :
$1 = 1^3 + 2(1) \rightarrow 1 = 3$, kelipatan 3
Induksi : misalkan untuk n = k asumsikan $k^3 + 2k = 3x$
Untuk n = k + 1 berlaku
$(k + 1)^3 + 2(k + 1)$ adalah kelipatan 3
$(k^3 + 3k^2 + 3k+1) + 2k + 2$
$(k^3 + 2k) + (3k^2 + 3k + 3)$
$(k^3 + 2k) + 3(k^2 + k + 1)$
↓ Induksi
$3x + 3(k^2 + k + 1)$
$3(x + k^2 + k + 1)$
Konklusi : $n^3 + 2n$ adalah kelipatan 3
Untuk setiap bilangan bulat positif n

2. METODE

2.1 Algoritma *Greedy*

Algoritma *Greedy* adalah algoritma yang paling populer dalam pemecahan persoalan optimasi. Algoritma ini membentuk solusi untuk tiap langkah dan dalam tiap langkah tersebut harus dibuat keputusan yang terbaik karena dalam tiap langkah terdapat banyak pilihan yang harus dieksplorasi dan dicari pilihan mana yang paling optimal dengan harapan jika kita memilih optimum lokal pada tiap langkah, dengan harapan nantinya akan dihasilkan optimum global. Ada dua macam permasalahan optimasi yang dapat diselesaikan menggunakan algoritma *Greedy*, yaitu Maksimasi dan Minimasi.

Algoritma Greedy mempunyai elemen-elemen antara lain :

1. Himpunan Kandidat, C, yaitu himpunan yang menjadi asal dari solusi yang diperoleh.
2. Himpunan Solusi, S, yaitu himpunan yang berisi solusi permasalahan.
3. Fungsi Seleksi (*Selection Function*), yaitu fungsi yang digunakan untuk memilih kandidat terbaik untuk dimasukkan ke himpunan solusi.
4. Fungsi Kelayakan (*Feasible*), yaitu fungsi yang digunakan untuk menentukan apakah sebuah kandidat dapat membentuk sebuah solusi.
5. Fungsi Objektif, yaitu fungsi yang memilih solusi mana yang paling optimal.

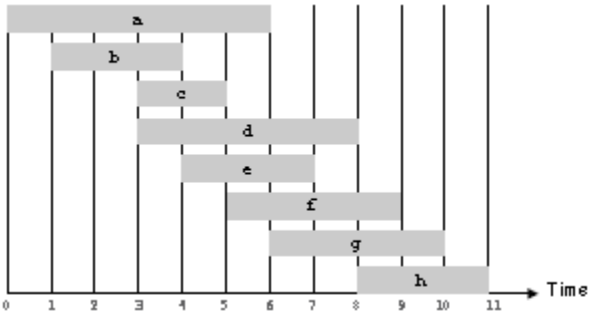
Dengan kata lain, algoritma *Greedy* akan mencari sebuah himpunan solusi, S, yang merupakan himpunan bagian dari himpunan kandidat, C. Dalam hal ini, himpunan S merupakan himpunan kandidat, C, yang dikenai fungsi seleksi dan kelayakan. Dan solusi akhir yang diambil adalah himpunan solusi, S, yang dikenai fungsi objektif.

Pilihan yang dibuat menggunakan algoritma *Greedy* ditentukan oleh pilihan-pilihan yang telah dibuat sampai saat ini namun tidak oleh pilihan-pilihan yang akan datang. Secara iteratif, pilihan *greedy* dilakukan sehingga membuat permasalahan yang ada menjadi permasalahan yang lebih kecil. Perbedaan mendasar antara algoritma *Greedy* dengan program dinamis adalah pada algoritma *Greedy* tidak pernah merubah pilihan yang telah dibuat sebelumnya.

Hasilnya, algoritma *Greedy* disebut sebagai metode *short-sighted* dan *non-recoverable*. Algoritma ini ideal untuk permasalahan yang mempunyai masalah optimasi dan sangat cocok untuk permasalahan yang sederhana. Perlu diketahui pula bahwa algoritma ini dapat digunakan sebagai algoritma seleksi untuk memprioritaskan pilihan dalam sebuah pencarian atau algoritma *branch and bound*. Ada beberapa variasi algoritma *Greedy*, yaitu *Pure Greedy*, *Orthogonal Greedy*, dan *Relaxed Greedy*.

2.2 *Interval Scheduling*

Interval Scheduling adalah sebuah kelas permasalahan pada ilmu komputer, khususnya pada perancangan algoritma. Pada permasalahan ini, diberikan sebuah himpunan tidak kosong $\{(s(i), f(i)), 1 \leq i \leq n\}$ yang berisi waktu mulai dan selesainya sejumlah *n* pekerjaan. Tujuan yang ingin diperoleh adalah sebuah solusi yang merupakan himpunan bagian terbesar (jumlah anggotanya terbanyak) yang berisi pekerjaan-pekerjaan yang *mutually compatible*, yaitu pekerjaan yang tidak meng-*overlap* pekerjaan yang lain. Permasalahan ini memodelkan keadaan dimana terdapat sebuah *resources*, himpunan pekerjaan, dan diperlukan penjadwalan yang mencakup sebanyak mungkin pekerjaan. Seperti dikatakan sebelumnya, algoritma *Greedy* dapat digunakan untuk menyelasaikan permasalahan penjadwalan, permasalahan ini juga dapat diselesaikan dengan algoritma *Greedy*.



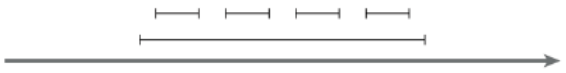
Gambar 1. Contoh permasalahan *Interval Scheduling*

Dalam penyelesaiannya menggunakan algoritma *Greedy*, masalah ini dikerjakan menggunakan asumsi bahwa himpunan pekerjaan tersebut terurut dalam sebuah urutan. Kemudian dilakukan penambahan kedalam solusi selama pekerjaan tersebut *compatible* dengan pekerjaan-pekerjaan yang ada pada solusi saat ini.

Pertanyaan yang timbul adalah dengan urutan apa kita menyelesaikan permasalahan ini. Ada empat kemungkinan jawaban :

1. *Earliest Start Time*. Pekerjaan diasumsikan terurut menaik berdasarkan waktu mulainya (s(i)).
2. *Earliest Finish Time*. Pekerjaan diasumsikan terurut menaik berdasarkan waktu selesainya (f(i)).
3. *Shortest Interval*. Pekerjaan diasumsikan terurut menaik berdasarkan durasi pekerjaan (f(i) - s(i)).
4. *Fewest Conflicts*. Untuk tiap pekerjaan *i*, hitung jumlah pekerjaan yang berkonflik dengannya. Pekerjaan terurut menaik berdasarkan jumlah pekerjaan yang berkonflik dengan pekerjaan tersebut.

Untuk menyelesaikan permasalahan ini, seperti dapat dilihat di atas, ada empat macam metode penyelesaian yang mungkin dipilih. Namun seperti kita ketahui bersama, tidak semua metode *Greedy* menghasilkan solusi yang optimal. Untuk itu perlu diketahui mana metode yang menghasilkan solusi optimal dan mana yang tidak. Untuk mengetahui mana solusi yang optimal dapat dilihat pada penjelasan dengan gambar di bawah ini.



Gambar 2. Contoh permasalahan yang tidak optimal bila diselesaikan dengan metode *Earliest Start Time*

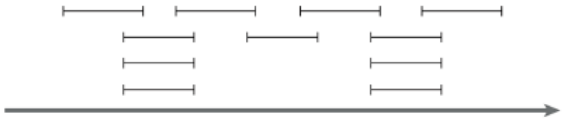
Dapat dilihat pada gambar, terdapat lima buah pekerjaan yang harus di atur penjadwalannya, sehingga diperoleh jumlah pekerjaan terbanyak yang dapat dikerjakan dalam selang waktu yang ada. Garis berpanah menunjukkan waktu yang ada.

Bila permasalahan seperti pada gambar diselesaikan dengan metode *Earliest Start Time*, hasil yang diperoleh

tidak akan optimal. Hal ini disebabkan karena algoritma akan memilih pekerjaan yang pada gambar memiliki waktu mulai terkecil namun memiliki durasi yang sangat panjang. Bila pekerjaan ini dipilih maka tidak akan ada lagi pekerjaan yang dapat ditambahkan ke dalam solusi yang ada yang *compatible* dengan solusi yang sudah ada. Sedangkan hasil optimal yang dapat diperoleh bukanlah satu pekerjaan melainkan empat pekerjaan.

Gambar 3. Contoh permasalahan yang tidak optimal bila diselesaikan dengan metode *Shortest Interval*

Bila permasalahan seperti pada gambar diselesaikan dengan metode *Shortesst Interval*, hasil yang diperoleh tidak akan optimal. Hal ini dikarenakan algoritma akan memilih pekerjaan yang pada gambar terletak di bawah yang memiliki durasi terkecil. Bila pekerjaan ini dipilih, maka tidak akan ada lagi pekerjaan yang dapat ditambahkan ke dalam solusi yang ada yang *compatible* dengan solusi yang sudah ada. Sedangkan hasil optimal yang dapat diperoleh bukanlah satu pekerjaan melainkan dua pekerjaan.



Gambar 4. Contoh permasalahan yang tidak optimal bila diselesaikan dengan metode *Fewest Conflicts*

Bila permasalahan seperti pada gambar diselesaikan dengan metode *Fewest Conflicts*, hasil yang diperoleh tidak akan optimal. Hal ini dikarenakan algoritma akan memilih pekerjaan yang pada gambar terletak di baris kedua yang ada ditengah karena memiliki jumlah pekerjaan lain yang berkonflik dengannya, yaitu dua, yang bila dibandingkan dengan pekerjaan lain yang minimal memiliki jumlah konflik empat menjadikan pekerjaan tersebut memiliki jumlah konflik yang paling sedikit. Bila pekerjaan ini dipilih, maka jumlah maksimal pekerjaan yang dapat dikerjakan adalah tiga. Sedangkan hasil optimal yang dapat diperoleh bukanlah tiga pekerjaan melainkan empat pekerjaan.

2.2 *Interval Scheduling* dengan *Earliest Finish Time* (EFT)

Dari pembahasan pada bagian sebelumnya, diketahui bahwa tiga dari empat metode *Greedy* yang dapat digunakan untuk menyelesaikan masalah *Interval Scheduling* tidak selalu menghasilkan solusi yang optimal. Hasil dari pembahasan tersebut menjadikan pendekatan *Greedy* dengan *Earliest Finish Time* merupakan metode yang paling optimal. Bila kita menggunakan metode *Earliest Finish Time*, pekerjaan harus diurutkan terlebih dahulu berdasarkan waktu selesainya secara menaik kemudian menambahkan pekerjaan ke dalam himpunan solusi jika pekerjaan tersebut *compatible* dengan solusi yang telah ada seperti bisa dilihat dari *pseudo-code* berikut.

```
Sort pekerjaan berdasarkan waktu
selesainya sehingga f(1) ≤ f(2) ≤ ...
≤ f(n).
A ← ϕ  //Solusi mula-mula kosong
for j = 1 to n
{
if (pekerjaan j compatible dengan A)
A ← A ∪ {j} //tambahkan pekerjaan j ke
              dalam A jika compatible
}
return A
```

Algoritma di atas akan mengembalikan himpunan A yang merupakan solusi optimal. Untuk membuktikan bahwa metode ini merupakan metode yang menghasilkan solusi optimal digunakan metode induksi. Misalkan O sebagai himpunan solusi optimal, akan sulit memperlihatkan bahwa A = O, namun kenyataannya kita cukup menunjukkan bahwa |A| = |O| (Daripada membuktikan bahwa semua pilihan pekerjaan yang ada di A merupakan semua pekerjaan yang ada pada O, cukup dibuktikan bahwa jumlah pekerjaan yang dipilih pada A dan O adalah sama. Dengan begini A optimal.).

Misalkan $i_1,i_2,\ldots,i_k$ merupakan himpunan pekerjaan pada A secara terurut membesar berdasarkan waktu selesainya dan $j_1,j_2,\ldots,j_m$ merupakan himpunan pekerjaan pada O yang terurut pula dengan $m \geq k$. Misalkan pula f(i) merupakan waktu selesai dari pekerjaan *i*.

Teorema
Untuk semua $r \leq k$, $f(i_r) \leq f(j_r)$.
Bukti
Pembuktian dengan Induksi matematika:
Basis : Teorema di atas benar untuk r = 1karena himpunan solusi masih kosong dan kita memilih secara *Greedy*.
Induksi : Asumsikan teorema di atas benar untuk r = n $f(i_n) \leq f(j_n)$ maka kita mendapatkan

$$f(i_n) \leq f(j_n) \leq s(j_{n+1}),$$

dengan $s(j_{n+1})$ adalah waktu mulai dari pekerjaan j_{n+1}. Dengan begini, kita dapatkan bahwa pekerjaan j_{n+1} dapat dipilih saat gilirannya tiba. Maka, dengan cara pembuktian yang sama, semua pekerjaan j_r akan dapat dikerjakan karena pekerjaan j_r dimulai setelah atau sama dengan saat pekerjaan j_{r-1} selesai dikerjakan.
Konklusi : $f(i_r) \leq f(j_r)$ untuk semua $r \leq k$.

Pembuktian belum selesai sampai disini karena kita masih harus membuktikan algoritma *Greedy* optimal untuk *Interval Scheduling.*

Teorema
Algoritma *Greedy* dengan EFT optimal untuk *Interval Scheduling*
Bukti
Bila solusi yang dihasilkan tidak optimal berarti himpunan A mempunyai sejumlah *k* pekerjaan dan O memunyai sejumlah *m* pekerjaan dengan m > k.
Dengan menggunakan lemma yang telah kita buktikan sebelumnya, setelah sejumlah *k* pekerjaan, j_{k+1} (pekerjaan di O ke-*k*+1) tetap dapat dipilih setelah i_k (pekerjaan di A ke-*k*) selesai dikerjakan. Hal ini membuktikan bahwa apapun pekerjaan yang ada di himpunan O akan bisa dimasukkan ke dalam himpunan A. Dan kardinalitas dari A akan tetap selalu sama dengan kardinalitas O.
Kesimpulan
A merupakan hasil optimal karena |A|=|O|.

Kompleksitas waktu dari penerapan algoritma ini merupakan hasil dari proses pengurutan pekerjaan berdasarkan waktu selesainya dan untuk tiap pekerjaan dilakukan pengecekan terhadap waktu mulainya apakah terletak setalah atau sama dengan waktu selesai dari pekerjaan terakhir yang dipilih (apakah *compatible*). Hasilnya, O(n log n + n) = O(n log n).

3. KESIMPULAN

Algoritma *Greedy* adalah algoritma yang sangat ideal untuk masalah optimasi dan juga dapat digunakan untuk menyelesaikan permasalahan penjadwalan seperti *Interval Scheduling*. Dalam menyelesaikan persoalan *Interval Scheduling* dengan algoritma *Greedy*, himpunan pekerjaan sebelumnya diurutkan berdasarkan sebuah aturan dan kemudian dilakukan penambahan terhadap himpunan solusi apabila pekerjaan tersebut *compatible* dengan pekerjaan terakhir yang dipilih.

Pendekatan untuk menyelesaikan persoalan *Interval Scheduling* ada empat macam, yaitu *Earliest Start Time*, *Earliest Finish Time*, *Shortest Interval*, dan *Fewest Conflicts*. Seperti halnya sifat algoritma *Greedy* bahwa tidak semua persoalan yang diselesaikan dengan algoritma *Greedy* menghasilkan solusi optimal, hal yang sama terjadi juga pada persoalan ini. Pendekatan *Earliest Start Time*, *Shortest Interval*, dan *Fewest Conflicts* ketiganya tidak menghasilkan hasil optimal untuk beberapa kasus oleh karenanya tidak dapat digunakan untuk menyelesaikan masalah optimasi pada *Interval Scheduling.*

Sedangkan pendekatan *Earliest Finish Time*, merupakan satu-satunya pendekatan yang memberikan hasil optimal untuk permasalahan *Interval Scheduling*. Dengan menggunakan metode induksi matematika, dapat dibuktikan bahwa untuk tiap kasus *Interval Scheduling*, pendekatan *Earliest Finish Time* akan menghasilkan solusi yang optimal. Kompleksitas dari algoritma yang menerapkan pendekatan *Earliest Finish Time* ini adalah O(n log n).

REFERENSI

[1] English Wikipedia
http://en.wikipedia.org/wiki/Greedy_algorithm
Tanggal akses : 1 Januari 2010 pukul 23.56

[2] English Wikipedia
http://en.wikipedia.org/wiki/Interval_scheduling
Tanggal akses : 1 Januari 2010 pukul 23.57

[3] Lecture of Greedy Algorithm
http://courses.cs.vt.edu/~cs5114/spring2009/lectures/lecture04-greedy-scheduling.pdf
Tanggal akses : 1 Januari 2010 pukul 23.54

[4] Munir, Rinaldi, "Matematika Diskrit", Program Studi Informatika Institut Teknologi Bandung, 2007.

[5] Lecture of Greedy Algorithm
http://www.cse.psu.edu/~sofya/cse565/lecture-notes/CSE565-F07-Lec-06.pdf
Tanggal akses : 2 Januari 2010 pukul 00.00

[6] Lecture of Interval Scheduling
http://www.cs.umd.edu/class/fall2009/cmsc451/lectures/Lec04-interval.pdf
Tanggal akses : 2 Januari 2010 pukul 00.01

[7] Lecture of Greedy Algoithm in Interval Scheduling
http://lonati.dsi.unimi.it/algo/0910/lab/kowalski6.pdf
Tanggal akses : 1 Januari 2010 pukul 23.59

[8] Kuliah mengenai Induksi Matematika
http://swelandiah.staff.gunadarma.ac.id/Downloads/files/12920/Pertemua+ke+VI.doc
Tanggal akses : 2 Januari 2010 pukul 21.52