**ACTIVITY SELECTION PROBLEM**

**DENGAN ALGORITMA GREEDY**

**Aditya Gumilar – 1301184037**

Program Studi S1 Teknik Informatika

Fakultas Informatika

Telkom University

Wangisagara, Kabupaten Bandung

e-mail : [adityagum07@gmail.com](mailto:adityagum07@gmail.com)

**ABSTRAK**

**Algoritma Greedy, jika dalam Bahasa Indonesia Greedy yang berarti rakus. Jadi, algoritma greedy merupakan sebuah algoritma yang menggunakan konsep ambil yang terbaik “Take what you can get now”. Ambil yang terbaik disini maksudnya yaitu mengambil solusi optimal dari semua solusi yang telah dikumpulkan untuk memecahkan suatu permasalahan. Selain itu algoritma Greedy juga digunakan dalam masalah optimisasi waktu seseorang dalam melakukan pekerjaan, dimana terdapat waktu memulai (start) dan waktu berakhir (finish). Salah satu masalah yang bisa menggunakan algoritma Greedy yaitu Activity Selection Problem. Activity Selection ini merupakan sebuah masalah dalam pemilihan kegiatan yang dilakukan oleh satu orang yang ditandai dengan waktu mulai (Si) dan waktu selesai (Fi). Disini kita memilih jumlah kegiatan yang dilakukan oleh satu orang dengan asumsi bahwa satu orang tersebut hanya dapat beraktifitas dalam satu waktu.**

**Kata kunci : Activity Selection Problem, Pemilihan Aktifitas, Greedy, Interval Representation**

**PENDAHULUAN**

Activity Selection Problem adalah sebuah masalah dalam pemilihan kegiatan yang dilakukan oleh seseorang. Asumsikan seseorang tersebut memiliki beberapa aktifitas, dimana aktifitas tersbut kita anggap sebagai {A1,A2,A3,…,An} dimana didalam aktifitas tersebut kita memiliki waktu mulai (Si) dan waktu selesai (Fi). Dua aktifitas dapat

dikerjakan apabila waktu dari dua aktifitas tidak saling bentrok. Misalnya aktifitas a1 memiliki waktu dari jam 2 sampai jam 4, aktifitas selanjutnya memiliki waktu dari jam 4 sampai jam 6, maka aktifitas tersebut dapat di kerjakan. Jika aktifitas A1 memiliki waktu dari jam 1 sampai jam 3, dan aktifitas selanjutnya memiliki waktu dari jam 1 sampai jam 4, maka aktifitas tersbut tidak bisa dilakukan karena perolehan waktu dari kedua aktifitas tersebut bertentangan. Problem nya disini adalah kita harus mencari aktifitas atau kegiatan yang ada sebanyak mungkin, yaitu sama dengan himpunan aktifitas yang telah ditulis sebelumnya yakni {A1, A2, …, An} dengan catatan dari aktifitas tersebut memiliki waktu yang tidak bertentangan. Kita mempunyai kasus misalkan terdapat 10 aktifitas dengan waktu yang berbeda beda setiap aktifitasnya. Untuk lebih jelasnya, perhatikan table dibawah ini.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Si | 1 | 7 | 3 | 4 | 6 | 5 | 2 | 1 | 5 | 13 |
| Fi | 3 | 13 | 7 | 9 | 13 | 12 | 5 | 6 | 8 | 15 |

**Tabel 1. Tabel aktifitas**

**Keterangan :**

* A = Aktifitas
* Si = Start (Waktu mulai)
* Fi = Finish (Waktu selesai)

Masalah dari persoalan ini adalah kita menghitung berapakah jumlah maksimal dari akfititas diatas yang bisa diselesaikan.

1. **PENJELASAN PROGRAM**

Activity selection problem merupakan sebuah masalah dalam memilih urutan atau kegiatan dari beberapa aktifitas. Bahasa Pemrograman yang saya gunakan adalah C++. Kita tahu bahwa dalam activity selection problem terdapat sebuah aktifitas, waktu mulai dan waktu selesai. Kita menggunakan perbandingan waktu mulai dan waktu antara aktifitas satu dengan aktifitas lainnya agar menemukan jumlah maksimal aktifitias yang dapat dikerjakan. Kemudian hasil dari perbandingan tersebut akan di urutkan (sorting) dari aktifitas yang memiliki waktu paling cepat pengerjaannya.

1. **STRATEGI ALGORITMA**

Untuk menyelesaikan persoalan pada kasus activity selection problem, kita menggunakan Algortima Greedy. Karena Algoritma Greedy merupakan algoritma yang mencari solusi optimum, yaitu solusi yang bernilai minimum atau maksimum dari sekumpulan alternative solusi yang mungkin ada. Algoritma Greedy membentuk solusi step by step, “take what you can get now” maksudnya ambilah keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah berikutnya. Kita bisa simpulkan bahwa solusi tersebut dengan optimum lokal. Pada saat pengambilan nilai optimum lokal terjadi pada setiap langkah, diharapkan agar tercapainya optimum global, yaitu dengan tercapainya solusi optimum yang melibatkan keseluruhan langkah awal sampai akhir. Namun perlu diingat bahwa optimum global belum tentu merupakan solusi optimum (terbaik) karena terdapat beberapa fungsi seleksi (pemilihan) yang berbeda, sehingga kita harus memilih fungsi yang tepat jika ingin algoritma menghasilkan solusi optimal. Dalam menyelesaikan Activity selection problem menggunakan Greedy, terdapat 2 cara : Ada pengerjaan yang diurutkan terlebih dahulu berdasarkan nilai terkecil dari waktu selesai setiap aktifitias, dan ada yang tidak menggunakan urutan tersebut.

* 1. **Solusi dengan Algoritma Gredy**

Perhatikan urutan aktifitas yang terdapat pada tabel dibawah ini

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Si | 1 | 7 | 3 | 4 | 6 | 5 | 2 | 1 | 5 | 13 |
| Fi | 3 | 13 | 7 | 9 | 13 | 13 | 5 | 6 | 8 | 15 |

**Tabel 1. Tabel aktifitas**

Kita akan mencari urutan jumlah maksimal aktifitas dari tabel tersebut. Waktu selesai (Fi) pada tabel aktifitas tersebut tidaklah berurutan, maka kita harus urutkan terlebih dahulu waktu selesai(Fi) untuk setiap aktifitas.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Si | 1 | 2 | 1 | 3 | 5 | 4 | 6 | 5 | 7 | 13 |
| Fi | 3 | 5 | 6 | 7 | 8 | 9 | 13 | 13 | 13 | 15 |

**Tabel 2. Tabel Aktifitas setelah diurutkan berdasarkan (Fi)**

Setelah kita mengurutkan urutan aktifitas tersebut, kita akan mencari sekumpulan aktifitas yang memiliki waktu tidak bertentangan, sehingga menyebabkan aktifitas itu maksimal.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Si | 1 | 2 | 1 | 3 | 5 | 4 | 6 | 5 | 7 | 13 |
| Fi | 3 | 5 | 6 | 7 | 8 | 9 | 13 | 13 | 13 | 15 |

**Tabel 3. Pemilihan Aktifitas (1)**

1. Contoh : Aktifitas A3 memiliki Si=1 dan Fi=6 dimana Si=1 merupakan waktu mulai dan Fi=6 merupakan waktu selesai dari aktifitas A3. Kemudian kita cari aktifitas yang memiliki waktu lebih dari sama dengan Fi yang ada pada A3. Berarti kita memilih A7, karena A7 memiliki Si=6 dan Fi=13. Setelah itu kita cari lagi aktifitas yang memiliki waktu lebih dari sama dengan Fi pada A7. Kemudian kita pilih aktifitas A10 karena memiliki Si=13 dan Fi=15. Fi pada A10 merupakan waktu paling akhir didalam aktifitas.

Sehingga {A3, A7, A10} merupakan aktifitas yang dapat diselesaikan karena waktu yang ada pada ketiga aktifitas itu tidak saling bertentangan.

1. Pertimbangkan kembali ternyata kita masih bisa menemukan aktifitas lainnya selain aktifitas Selain itu kita juga bisa memulai aktifitas lain dengan syarat waktu yang dimiliki setiap aktifitas tidak bertentangan.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Si | 1 | 2 | 1 | 3 | 5 | 4 | 6 | 6 | 7 | 13 |
| Fi | 3 | 5 | 6 | 7 | 8 | 9 | 13 | 13 | 13 | 15 |

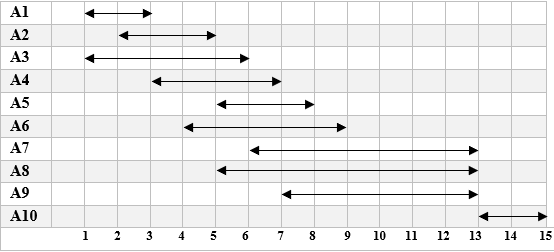
**Tabel 4. Pemilihan Aktifitas(2)**

Kita pilih aktifitas {A1, A4, A9, A10}, sebelumnya kita hanya memiliki 3 aktifitas saja sebagaimana telah terbukti pada point ke-1. Namun pada point ke 2, kita memiliki 4 aktifitas.

Kita dapat menyimpulkan bahwa solusi dalam menghitung jumlah maksimal dari aktifitas yang ada tidaklah unik, maksud tidak unik itu kita menemukan beberapa solusi untuk memecahkan permasalahan tersebut. Dari ketiga solusi diatas kita akan gabungkan beberapa solusi yang ada untuk menemukan solusi optimal secara global (menyeluruh).

* 1. **Solusi menggunakan Interval**

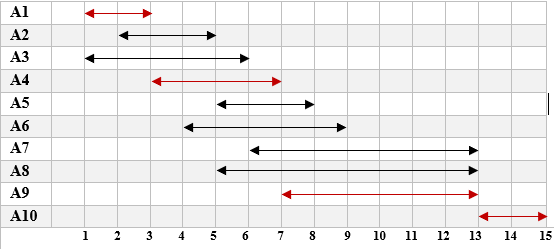
**Representation**

 Untuk kasus pemilihan aktifitas, kita bisa menggunakan Interval Representation. Untuk lebih jelasnya kita bisa lihat pada diagram dibawah ini.

**Gambar 1. Tabel Interval Representation**

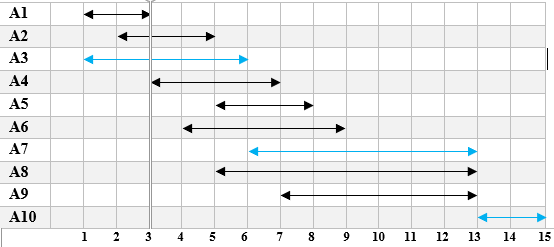
Untuk A1-A10 merupakan aktifitas, nilai dibawah merupakan waktu dan arah panah yang ada pada setiap baris aktifitas itu merupakan waktu yang terdapat pada masing-masing aktifitas. Dari sini kita bisa mencari aktifitas secara acak yang memungkinkan bahwa aktifitas ini tidak memiliki waktu yang bertentangan.

1. Kita memilih aktifitas A1, A4, A9 dan A10. Karena keempat aktfitas ini waktunya tidak saling bertentangan. Untuk lebih jelasnya kita liat saja arah panah kekiri dari setiap aktifitas. Pada A1 waktunya dimulai dari 1, A4 dari 3, A9 dari 7, dan A10 dari 13.



**Gambar 2. Memilih aktifitas pada Interval Representation (1)**

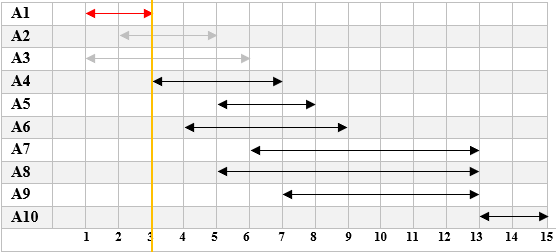
1. Bisa menggunakan contoh yang lain misalnya {A3, A7, A10}:



**Gambar 3. Memilih aktifitas pada Inteerval Representation (2)**

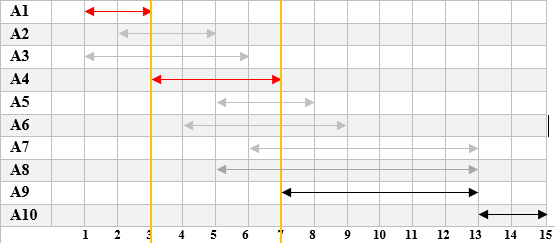
Kita memiliki beberapa solusi aktifitas, jadi untuk masalah ini daripada kita harus membuktikannya satu persatu, kita bisa menggunakan konsep Greedy. Caranya yaitu kita mencari aktifitas yang waktu selesainya lebih cepat.

* Pertama, kita pilih A1 karena waktu selesainya lebih cepat. Setelah itu kita lihat aktifitas lain yang waktunya bertentangan dengan A1, jika terdapat waktu yang bertentangan atau bentrok dengan A1. Untuk lebih jelasnya saya menggunakan pembatas sebagai tanda.

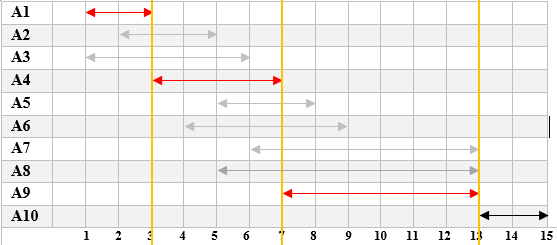
**Keterangan :** Aktiftias yang dipilih (merah) Aktifitas yang di abaikan (abu-abu), pembatas (jingga)

**Gambar 4. Memilih dan membatasi setiap aktifitas pada Interval Representation**

* Selanjutnya kita mencari aktifitas yang waktu selesainya lebih cepat, yaitu ada pada A4. Setelah itu abaikan aktifitas yang bentrok dengan A4.

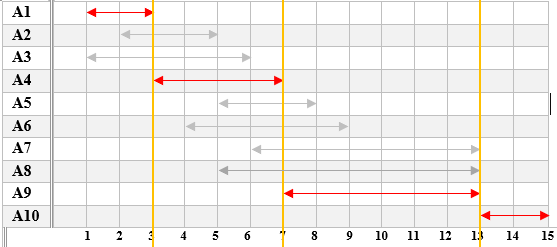


**Gambar 5. Memilih dan membatasi setiap aktifitas pada Interval Representation (2)**

* Kita lanjutkan untuk mencari aktifitas, pada Gambar 6, yang memiliki waktu penyelesaian lebih cepat adalah A9. Jadi kita memilih A9.

**Gambar 6. Memilih dan membatasi setiap aktifitas pada Interval Representation (3)**

* Selanjutnya kita mencari lagi aktifitas yang waktunya bertentangan. Namun pada gambar 6, kita tidak menemukan aktifitas yang bentrok setelah A9, dan hanya tersisa satu aktifitas yaitu A10, dan aktifitas A10 memiliki waktu yang tidak bertentangan.



**Gambar 7. Memilih dan membatasi setiap aktifitas pada Interval Representation (4)**

Pada gambar 7. Tidak ada aktifitias lagi, sehingga pada gambar 8 merupakan cara terakhir untuk mencari aktifitas yang tidak bertentangan.

Jadi, dengan menggunakan konsep greedy ini kita mendapatkan bahwa jumlah maksimal aktifitas yang kita dapatkan adalah sebanyak 4, yaitu {A1, A4, A9, A10}.

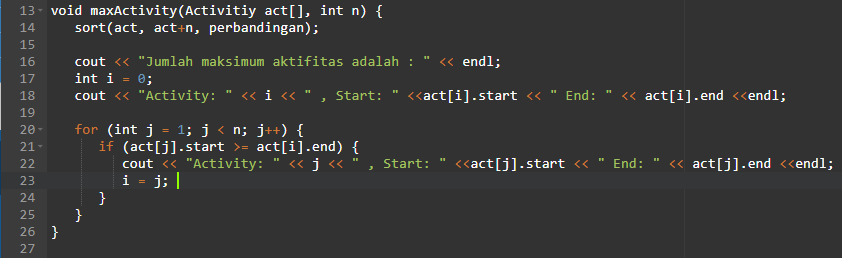
Operasi utama adalah melakukan pengecekan setiap aktifitas. Apabila waktu selesai (Fi) atau inputan setiap aktifitas berurutan, maka kompleksitas waktu nya adalah T(n) = 1 + 2 + 3 + … + (n-1) = *n2-n*/2. Karena urutan aktifitas dalam permasalahan yang kita bahas memiliki waktu penyelesaian (Fi) berurutan, maka dapat disimpulkan bahwa T(n) = O(n).

1. **FUNGSIONALITAS PROGRAM**

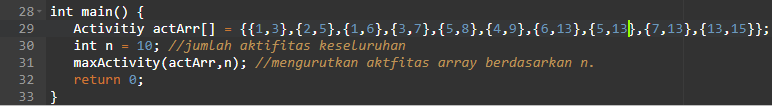


**Gambar 8. Function Perbandingan**

Fungsi perbandingan digunakan untuk membandingkan antar aktifitas dan perolehan waktu dari setiap aktifitas yang ada. A merupakan Actifitas pertama dan A2 merupakan aktiftias kedua.

**Gambar 9. Function maxActivity**

* Gambar 13 merupakan fungsi aktifitas maksimal yang memiliki parameter (Activity act[]) untuk mengurutkan hasil aktifitas dalam bentuk array dan (int n) sebagai variable untuk menghitung jumlah aktifitas. Kita menginisialisasi n = 0. Untuk lebih jelasnya akan ada di baris ke 30.
* Pada baris ke 14, sort digunakan untuk mengurutkan antar aktifitas dan kemudian kita lakukan perbandingannya. Pada saat melakukan perbandingan, usahakan aktifitas tersebut terurut.
* Pada baris 17 terdapat int I yang di assign dari 0. I disini sebagai aktifitas keberapa yang akan kita eksekusi terlebih dahulu. Di program yang menginisalisasi I dari 0, sehingga aktifitas akan dimulai dari 0. Aktifitas 0 disini adalah aktifitas ke 1. Karena urutan dalam array berbeda.
* Pada baris 18 untuk mencari aktifitas keberapa yang memiliki waktu pengerjaan lebih cepat.
* Pada baris 20-23 merupakan perulangan untuk untuk mencari aktifitas yang memiliki waktu pengerjaannya lebih cepat dibandingkan aktifitas lain. J disini sebagai penentu index untuk aktifitas selanjutnya. Jika aktifitas J memiliki waktu mulai yang lebih besar sama dengan waktu akhir dari I, maka J akan menggantikan I. Begitu seterusnya hingga aktifitas selesai.

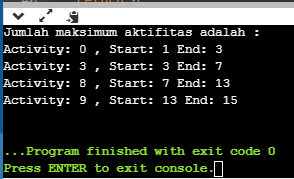


**Gambar 10. Main Menu**

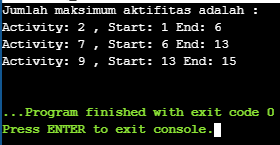
* Pada baris ke 29, merupakan urutan aktifitas beserta waktu mulai dan waktu selesainya.
* Pada baris 30, n sebagai jumlah aktifitas yang akan dieksekusi.
* Pada baris 31 function yang menggunakan maxActivtiy yang sebelumnya telah di jelas. Kemudian max activity ini akan mengurutkan data sama dengan yang ada didalam parameternya yaitu actArr merupakan nilai/value pada aktifitas yang ada, dan n merupakan jumlah aktifitasnya.

1. **SCREEN SHOOT OUTPUT**

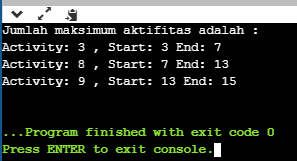
**PROGRAM**



**Gambar 11. Hasil Output (Jika I = 0)**



**Gambar 12. Hasil Output (Jika I = 2)**



**Gambar 13. Hasil Output (Jika I = 3)**

1. **REFERENSI**

*Activity Selection Problem | Greedy Algorithm*. (n.d.). Retrieved from CodesDope: https://www.codesdope.com/course/algorithms-activity-selection/

Chitranayal, A. R. (n.d.). *Activity Selection Problem | Greedy Algo-1*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/activity-selection-problem-greedy-algo-1/

Surf, F. (2015). *Algoritma Greedy*. Retrieved from slideplayer.info: https://slideplayer.info/slide/2789226/

Wahyono, P. (n.d.). Greedy Algorithm. *Greedy Algorithm*.