Penerapan Algoritma Greedy dan *Heuristic analysis* Dalam Permainan Kartu "41"

Timothy - 13517087
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
timmysutanto@gmail.com

***Abstract*—Permainan kartu merupakan permainan yang cukup populer di berbagai kalangan. Kita mengenal banyak sekali jenis permainan kartu. Salah satu jenis permainan kartu yang paling populer adalah permainan "41". Peraturan permainan ini mirip dengan peraturan permainan kuartet. Tujuan dari permainan ini adalah untuk mengumpulkan poin perhitungan kartu setinggi mungkin. Dalam makalah ini, akan dibahas mengenai teori algoritma *Greedy* yang dapat dipakai sebagai strategi dalam permainan ini. Perhitungan algoritma *greedy* ini akan dibantu dengan pendekatan metode *Heuristic Analysis* agar hasil yang dicapai lebih optimal. Pembahasan dalam makalah ini meliputi algoritma penentuan kartu yang dipilih pada setiap tahap permainan sesuai dengan objektif dari permainan ini serta pola dan metode heuristik yang akan dipakai untuk menyelesaikan masalah.**

Kata Kunci—permainan; algoritma; heuristik; maksimal

I. PENDAHULUAN

Hiburan masih merupakan salah satu kebutuhan manusia. Banyak aktivitas yang dapat kita lakukan untuk memenuhi kebutuhan ini. Hiburan dapat didapatkan melalui berbagai media. Masyarakat Indonesia seringkali mendapatkan hiburan melalui saluran televisi, radio, musik, maupun fasilitas-fasilitas yang terdapat di internet. Selain cara-cara di atas, cara orang untuk mendapatkan hiburan adalah melalui permainan.

Dalam era teknologi ini, perkembangan teknologi digital berkembang sangat pesat. Perkembangan ini memiliki dampak yang sangat besar dalam segala aspek kehidupan kita. Dalam dunia hiburan pun, kita dapat melihat berkembangnya permainan-permainan yang bersifat digital. Akan tetapi permainan klasik seperti permainan kartu masih cukup populer.

Kartu dapat dimainkan dalam berbagai macam peraturan. Peraturan permainan kartu yang banyak dikenal diantaranya adalah Capsa, Poker, 7 *spades,* "41" dan masih banyak lagi. Makalah ini akan membahas tentang permainan kartu "41". Dalam permainan ini terdapat banyak strategi yang biasa digunakan untuk memenangkan permainan.

II. DASAR TEORI

A. *Permainan kartu "41"*

Gambar 1.1 Permainan Kartu "41": http://4.bp.blogspot.com/-3rSLsI80P-E/VXQladxxBBI/AAAAAAAAAqQ/k-R3MiNGtnU/s1600/kartu-41-0.jpg

Dalam permainan ini, tujuan pemain adalah mengumpulkan kartu dengan lambang yang sama dan dengan nilai kartu yang setinggi mungkin agar mendapatkan poin perhitungan yang maksimal. Setiap kartu memiliki nilai yang berbeda-beda. Kartu angka, yaitu kartu yang tertulis angka 2 sampai 10 memiliki nilai sesuai dengan angkanya masing-masing. Kartu As memiliki nilai 1. Kartu bergambar seperti *Jack, Queen,* dan *King,* memiliki poin 10.

Perhitungan poin total seorang pemain dilakukan dengan menjumlahkan semua poin kartu yang dipegang. Kartu-kartu yang memiliki lambang yang sama akan dijumlah, sedangkan kartu-kartu yang memiliki lambang yang berbeda akan dikurang.

Dalam setiap giliran bermain, tiap pemain dapat mengambil satu kartu dari tumpukan kartu buangan

lawan, atau dari tumpukan kartu yang tertutup. Setelah mengambil, pemain harus membuang salah satu kartu yang dipegang ke tempat kartu buangan pemain yang akan mendapat giliran berikutnya. Permainan berakhir ketika salah satu pemain memiliki nilai total kartu 41 atau ketika tidak ada kartu lagi yang dapat diambil. Pada akhir permainan, pemain yang memiliki total nilai yang tertinggi adalah pemenangnya.

B. Teori Dasar Algoritma Greedy

1. Definisi Algoritma Greedy

Algoritma Greedy adalah salah satu bentuk metode yang dapat digunakan untuk menyelesaikan berbagai permasalahan. Algoritma ini merupakan cara yang paling populer dalam menyelesaikan permasalahan optimasi. Tujuan dari optimasi permasalahan dibagi menjadi dua, yaitu minimasi dan maksimasi.

Minimasi adalah permasalahan untuk mendapatkan hasil yang seminimal mungkin. Contoh permasalahan ini dapat kita temukan dalam masalah penukaran uang. Dalam kasus ini, kita menginginkan jumlah uang hasil penukaran yang sesedikit mungkin dan bernilai sama dengan nilai yang ditukar. Contoh lainnya dapat kita temukan dalam permasalahan *Travelling Salesman Problem* untuk mencari jalan dengan bobot yang sesedikit mungkin.

Maksimasi adalah permasalahan untuk mendapatkan hasil yang semaksimal mungkin. Salah satu permasalahan yang menggunakan prinsip ini adalah knapsack problem. Hasil yang diharapkan adalah kombinasi barang yang total beratnya memenuhi syarat maksimal beban dan menghasilkan keuntungan atau *value* yang semaksimal mungkin. Contoh penggunaan prinsip maksimasi lainnya adalah dalam persoalan penentuan jadwal

Penyelesaian dengan algoritma ini dilakukan dalam pembagian tahap-tahap. Dalam penggunaan metode *greedy,* prinsip yang digunakan adalah "*Take what you can get now"*, sehingga dalam setiap tahapnya algoritma akan mencari solusi optimal lokal yang dapat diambil. Solusi optimal lokal ini bergantung pada objektif algoritma yang sedang dijalankan. Jika objektif program tersebut adalah minimasi, maka program akan mengambil pilihan yang menghasilkan hasil yang sesedikit mungkin dan juga sebaliknya. Solusi optimal lokal ini kemudian diharapkan akan membawa perhitungan pada solusi global yang optimal.

Penggunaan algoritma ini mengandung beberapa kelebihan dan kekurangan. Algoritma *greedy* cenderung mengeluarkan hasil dengan waktu yang relatif singkat dibandingkan dengan algoritma lain seperti *brute force* dan lain-lain. Kompleksitas dari algoritma ini juga cenderung sederhana. Akan tetapi, hasil yang didapatkan melalui algoritma ini tidak dijamin optimal secara global dalam kasus-kasus tertentu. Misalkan dalam persoalan *Travelling Salesman Problem*, hasil algoritma *greedy* tidak dijamin optimal.

2. Elemen-Elemen Dalam Algoritma Greedy

Untuk menjelaskan elemen-elemen dalam algoritma ini, akan digunakan contoh permasalahan penukaran uang, dimana perhitungan diharapkan akan menghasilkan kombinasi penukaran uang dengan jumlah uang yang sesedikit mungkin dengan nilai total yang sama dengan jumlah uang yang ditukar.

- Himpunan kandidat

  Himpunan ini berisi dengan kemungkinan kemungkinan solusi yang akan diambil oleh algoritma *greedy*. Dalam permasalahan di atas, himpunan ini berisi himpunan koin yang dapat diambil dalam proses penukaran tersebut,

- Himpunan solusi

  Himpunan ini berisi solusi-solusi optimal lokal yang telah diambil sebelumnya. Pada akhir perhitungan, himpunan ini diharapkan akan menjadi solusi optimal global dari permasalahan yang ingin diselesaikan. Dalam kasus di atas, himpunan ini berisi koin-koin yang telah diambil pada langkah-langkah sebelumnya.

- Fungsi seleksi

  Elemen ini merupakan elemen yang menentukan optimal atau tidaknya hasil akhir dari algoritma *greedy*. Fungsi ini berisikan pertimbangan-pertimbangan algoritma dalam menentukan pilihan solusi optimal lokal dalam setiap tahap perhitungannya. Dalam permasalahan di atas fungsi inilah yang akan menentukan koin mana yang akan diambil sesuai dengan pertimbangan pertimbangan yang ada. Fungsi ini tidak bisa merubah solusi yang telah diambil sebelumnya atau dalam kata lain, keputusan pengambilan pilihan bersifat final dalam tiap tahapnya.

- Fungsi kelayakan

  Elemen ini merupakan penguji dari hasil yang didapatkan. Fungsi ini akan melakukan pengecekan syarat yang berlaku dari algoritma tersebut. Dalam kasus di atas, fungsi ini akan menguji apakah jumlah koin dalam himpunan solusi masih dibawah atau sama dengan jumlah uang yang ditukar di awal.

- Fungsi objektif

  Fungsi ini adalah tujuan dari optimalisasi algoritma *greedy*. Jika fungsi objektif dari algoritma ini adalah minimasi, maka fungsi seleksi akan mencari solusi yang menghasilkan hasil yang seminimal mungkin. Sebaliknya jika fungsi objektif dari algoritma ini adalah

maksimasi, maka fungsi seleksi akan mencari solusi yang menghasilkan hasil yang semaksimal mungkin.

3. Pseudocode Umum Algoritma Greedy

- Fungsi Seleksi

function Pemilihan(C : Himpunan_Kandidat) -> Kandidat
{Melakukan pemilihan berdasarkan objektif algoritma *greedy* yang dipakai. Fungsi ini kemudian akan mengembalikan pilihan yang terpilih}

- Fungsi Layak

function isLayak(S : Himpunan_Solusi) -> boolean
{Melakukan pengecekan solusi agar memenuhi *constraints* sesuai dengan permasalahan. Jika himpunan solusi memenuhi *constraints*, maka fungsi ini akan mengembalikan *true*. Sebaliknya jika himpunan ini melanggar *constraints* yang berlaku, maka fungsi akan mengembalikan nilai *false*}

- Fungsi Solusi

function isSolusi(S : Himpunan_Solusi) -> boolean
{Melakukan pengecekan pada himpunan solusi. Jika himpunan S merupakan solusi, maka fungsi ini akan mengembalikan nilai *true*, begitu juga sebaliknya}

- Algoritma *Greedy*

function greedy(C : Himpunan_Kandidat) -> Himpunan_Solusi
{Memeriksa himpunan kandidat yang diterima melalui parameter dalam tahap tahap pemilihan. Pemilihan tiap tahap ditentukan oleh fungsi solusi. Fungsi ini kemudian akan menghasilkan himpunan solusi yang diharapkan menghasilkan hasil yang optimal sesuai dengan fungsi objektifnya}

Deklarasi:

```
S : Himpunan_Solusi
K : Kandidat
```

Algoritma:

```
//Inisialisasi himpunan solusi dengan himpunan kosong
S <- {}

//Lakukan pemilihan sampai ditemukan solusi atau himpunan C merupakan himpunan kosong
while((not isSolusi(S)) or (C != {})) do
  K <- Pemilihan(C)
  C <- C - {K}
  if (isLayak(S + {K})) then
    S <- S + {C}
  if (isSolusi(S)) then
    return S

//Jika algoritma tidak menghasilkan solusi, maka fungsi akan mengembalikan himpunan kosong sebagai tanda tidak adanya solusi
if (not isSolusi(S)) then
  return {}
```

C. *Heuristic Analysis*

Heuristic Analysis adalah strategi yang dapat dipakai komputer untuk mempercepat penyelesaian masalah. Metode ini biasa dikenal dalam perangkat lunak *antivirus*. Kegunaan metode ini adalah untuk mempermudah perangkat lunak dalam mengenali ancaman-ancaman yang terkandung dalam sistem. Perangkat lunak ini mencoba untuk mengenali sifat dan pola-pola aktivitas yang terdapat pada bagian yang sedang diperiksa dan kemudian perangkat lunak ini akan mencocokan pola yang didapatkan tersebut ke dalam *database* yang dimiliki.

Selain *antivirus*, metode heuristik ini juga dapat digunakan dalam permasalahan lain. Contohnya adalah pencarian kata dalam kamus. Pada program sederhana, algoritma pencarian akan memulai pencocokan *string* dari awal atau akhir dari kamus. Dengan menggunakan heuristik, algoritma akan melakukan perhitungan tertentu untuk menentukan dari bagian mana pencarian akan dimulai.

Contoh kasusnya adalah pencarian kata bola pada kamus. Salah satu metode heuristik yang dapat dipakai adalah menghitung perkiraan posisi kata yang memiliki huruf awal b pada kamus. Perhitungan dapat dilakukan dengan

$$Halaman\ Awal = \frac{2}{26} X\ N$$

Karena huruf ‘b’ merupakan huruf dengan urutan kedua pada kamus dan terdapat 26 alfabet, maka perkiraan heuristik posisi kata adalah bagian ke-$\frac{2}{6}$ dari keseluruhan kamus.

III. STRATEGI GREEDY DAN HEURISTIK DALAM PERMAINAN KARTU “41”

Dalam permainan ini, pemain mengejar nilai total perhitungan kartu yang semaksimal mungkin. Maka dari itu, fungsi objektif yang dimiliki dalam penerapan algoritma ini dalam permainan kartu “41” adalah maksimasi. Agar perhitungan kartu maksimal, maka perlu diperhatikan dua strategi.

Strategi pertama adalah mengumpulkan kartu dengan lambang yang sama. Jika pada tahap awal perhitungan

pemain memegang kartu dengan jenis lambang yang lebih dari satu, maka pemain akan berusaha untuk membuang kartu yang dengan lambang yang jumlah nya paling sedikit. Pada kasus lambang-lambang yang berbeda tersebut memiliki jumlah yang sama seperti dalam kasus dua kartu berlambang hati dan dua kartu berlambang wajik, maka pemain akan berusaha untuk membuang kartu dengan nilai total yang lebih kecil.

Strategi kedua adalah mengumpulkan kartu yang bernilai tinggi. Strategi ini digunakan ketika pemain sedang memegang empat kartu yang memiliki lambang yang sama. Jika kartu yang diambil pada gilirannya memiliki nilai yang paling rendah diantara empat kartu sebelumnya, maka pemain akan membuang kartu yang baru terambil tersebut. Tetapi jika di antara empat kartu yang kita pegang memiliki nilai yang lebih rendah dibandingkan kartu baru yang diambil, maka pemain akan membuang kartu tersebut.

Saat pemain mengambil kartu pada awal gilirannya, maka pemain akan memilih untuk mengambil dari *discard pile* miliknya atau mengambil dari tumpukan kartu yang tertutup. Pada saat itu, pemain akan melakukan perhitungan penilaian nilai total kartu maksimal yang bisa didapatkan melalui kombinasi 4 kartu dari 5 kartu yang ada. Jika perhitungan menghasilkan nilai yang lebih tinggi dari perolehan nilai sebelum pengambilan, maka pemain akan membuang kartu yang memiliki nilai paling rendah pada *discard pile* pemain yang akan mendapat giliran jalan berikutnya.

Metode heuristik yang akan digunakan akan membantu pemain memenangkan permainan dengan menentukan kartu yang akan dibuang pada setiap langkahnya. Metode ini akan mencegah para lawan untuk mendapatkan kartu yang bernilai tinggi dan memiliki lambang yang sama dengan yang mereka sedang kumpulkan. Cara yang dapat dipakai adalah dengan memperhatikan pola pergerakan lawan. Pemain akan mencatat lambang-lambang kartu yang sedang dikoleksi oleh pemain lawan.

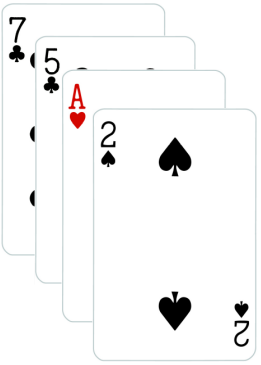
Pemain dapat mengetahui lambang yang sedang dikoleksi oleh pemain lawan ketika lawan memilih untuk mengambil kartu dari *discard pile* miliknya. Jika lambang pilihan lawan sudah diketahui, maka pemain akan menghindari membuang kartu dengan lambang tersebut pada giliran-giliran berikutnya. Tujuannya adalah agar pemain lawan tidak mengambil kartu yang pemain buang dan menambah nilai total perhitungan kartunya.

Pada kondisi awal permainan dimana pola pergerakan lawan masih belum dapat dikenali, fungsi seleksi kartu yang akan dibuang berjalan dengan pertimbangan yang berbeda. Fungsi seleksi akan memilih kartu yang memiliki nilai paling kecil untuk dibuang. Langkah ini akan mencegah pemain untuk membuang kartu bernilai tinggi yang dapat diambil oleh pemain lawan.

Kompleksitas algoritma *greedy* yang dipakai dalam permasalahan ini adalah O(M) dimana M adalah banyaknya giliran yang didapatkan pemain sepanjang permainan. Kompleksitas yang dimiliki bersifat linear karena jumlah pemeriksaan nilai maksimal yang dapat diperoleh sama dalam setiap tahapnya, yaitu memilih 4 kartu dari 5 kartu.

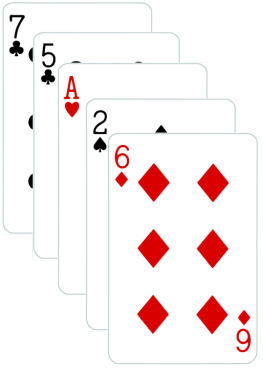
IV. Penerapan Heuristik algoritma Greedy dalam permainan kartu "41"

Ilustrasi pencarian solusi dengan algoritma *greedy* dalam permainan kartu "41" akan digambarkan sebagai berikut



Gambar 4.1 Posisi kartu awal

Gambar di atas adalah contoh keadaan kartu awal yang dapat dimiliki pemain. Perhitungan kartu maksimal dapat dilakukan dengan beberapa cara. Jika pemain mengumpulkan kartu berlambang hati, maka perhitungan poin pemain menjadi 1-2-5-7 = -13. Kemungkinan kedua adalah jika pemain mengumpulkan kartu berlambang sekop, maka perhitungan poin pemain menjadi 2-1-5-7 = -11. Pada kemungkinan terakhir, jika pemain mengumpulkan kartu berlambang keriting, maka perhitungan nilai kartu menjadi 7+5-2-1 = 9.



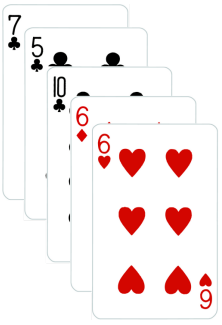
Gambar 4.2 Posisi kartu kedua

Gambar di atas menunjukkan contoh kondisi kartu pemain setelah mengambil dari tumpukan kartu yang tertutup. Melalui metode perhitungan di atas, nilai total maksimal yang dihasilkan adalah 9 yaitu dengan mengumpulkan kartu berlambang keriting.

Dalam kasus ini terdapat tiga kemungkinan kartu yang dapat dibuang, yaitu 6 wajik, 2 sekop, dan As hati. Pada tahap ini metode heuristik belum dapat mendeteksi kartu yang sedang dikoleksi lawan. Oleh karena itu algoritma akan memilih untuk membuang kartu As hati karena memiliki nilai yang paling rendah.

Setelah fungsi seleksi menentukan kartu yang dibuang, maka giliran pemain sudah selesai. Setelah ini program akan berusaha untuk memperhatikan pola pergerakan lawan dengan membaca kartu yang diambil lawan.

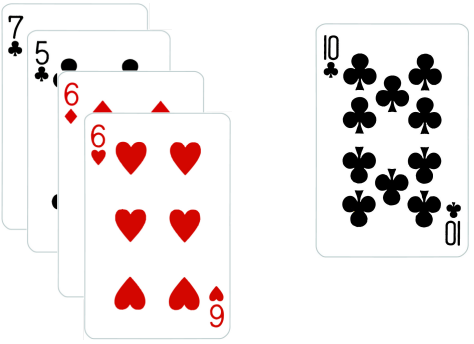
Jika lawan mengambil kartu dari tumpukan kartu yang tertutup, maka pertimbangan fungsi seleksi akan berjalan seperti sebelumnya. Akan tetapi, jika lawan mengambil kartu yang kita buang, maka algoritma akan mengingat lambang kartu tersebut dan menyimpannya dalam suatu variabel. Misalkan lawan mengambil kartu As hati yang sebelumnya dipilih fungsi seleksi untuk dibuang, maka program akan menyimpan "hati" dalam variabel tersebut. Dengan demikian, maka algoritma akan berusaha untuk menjaga kartu yang memiliki lambang hati jika ada kartu dengan lambang lain yang dapat dibuang pada langkah itu.



Gambar 4.3 Posisi kartu ketiga

Gambar di atas menunjukkan kondisi yang mungkin terjadi pada beberapa giliran berikutnya. Pada saat ini, program sudah memiliki fungsi heuristik yang berjalan dengan baik karena algoritma telah menyimpan lambang kartu yang dikoleksi oleh lawan. Pada posisi ini, perolehan kartu maksimal yang dapat diperoleh adalah 7+5+10-6 = 16.

Pemain memiliki dua pilihan kartu untuk dibuang, yaitu 6 hati dan 6 wajik. Disini kita lihat kedua kartu memiliki nilai kartu yang sama. Tanpa fungsi heuristik, kartu manapun yang dipilih untuk dibuang tidak akan memiliki perbedaan pada hasil yang akan didapatkan. Dengan fungsi heuristik, algoritma akan melihat lambang kartu yang sedang dikoleksi pemain lawan. Sehingga fungsi seleksi akan memilih untuk membuang kartu 6 wajik dengan pertimbangan bahwa pemain lawan dapat menambah nilai total kartunya jika mengambil kartu 6 hati yang dibuang.



Gambar 4.3 Posisi kartu keempat

Posisi kartu di atas menggambarkan suatu kondisi dimana kartu pada *discard pile* memiliki lambang yang sama dengan yang sedang dikoleksi oleh pemain. Jumlah nilai dari empat kartu pemain adalah 7+5-6-6=0. Dengan mengambil kartu pada *discard pile,* maka nilai perhitungan maksimal yang bisa didapatkan adalah 7+5+10-6=16. Karena nilai maksimal bertambah jika mengambil kartu dari *discard pile*, maka algoritma akan memilih untuk mengambil kartu dari *discard pile* dan mengabaikan tumpukan kartu yang tertutup pada giliran tersebut.

Pola ini akan terus berulang sampai fungsi solusi terpenuhi. Dalam permainan ini, fungsi solusi yang berlaku adalah jumlah perhitungan kartu pemain sama dengan 41 atau tidak ada kartu lagi pada tumpukan kartu tertutup

V. KESIMPULAN DAN SARAN

Kesimpulan yang bisa didapatkan setelah mengerjakan percobaan ini adalah bahwa algoritma *greedy* cukup baik untuk dipakai dalam permainan ini. Algoritma ini dapat membantu pemain untuk mendapatkan hasil yang lebih optimal dengan memilih kartu yang akan dipertahankan.

Metode *Greedy* juga membantu pemain untuk memilih untuk mengambil kartu dari tumpukan kartu yang tertutup atau dari *discard pile* yang dimiliki oleh pemain.

Metode *Heuristic Analysis* juga dapat membantu mengoptimalkan pilihan fungsi seleksi dalam menentukan kartu yang akan dibuang. Metode ini akan mencegah lawan mendapatkan keuntungan dari kartu yang dibuang melalui fungsi seleksi algoritma *greedy*.

Kelebihan algoritma ini adalah penentuan solusi yang lebih cepat jika dibandingkan dengan pembangkitan solusi melalui algoritma *brute-force.* Selain itu, algoritma ini bersifat cukup fleksibel. Setiap pemain dapat menerapkan strateginya masing-masing untuk memenangkan permainan ini.

Terlepas dari semua keuntungan yang bisa didapatkan, penerapan algoritma *greedy* dalam permasalahan permainan kartu "41" ini masih memiliki berbagai kekurangan. Tidak ada jaminan secara matematis bahwa hasil yang didapatkan

merupakan hasil yang optimal. Hal ini dikarenakan tidak selalu lambang sama terbanyak di awal giliran dapat dijadikan acuan mengingat banyaknya kemungkinan kartu-kartu yang akan didapatkan pada giliran-giliran berikutnya.

Oleh karena itu, penerapan algoritma *greedy* pada permainan kartu "41" ini perlu dikembangkan lebih lanjut. Bagian dari algoritma ini yang masih perlu diperbaiki antara lain menghitung probabilitas kartu yang akan diambil jika mengambil dari tumpukan kartu yang tertutup. Selain itu perlu dikembangkan juga fungsi objektif lain yang dapat digunakan sebagai acuan algoritma ini.

VI. REFERENSI

[1] http://acbl.mybigcommerce.com/52-playing-cards/ diakses pada tanggal 26 April 2019, pukul 02.07

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf diakses pada tanggal 26 April 2019, pukul 05.54

[3] https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/ diakses pada tanggal 26 April 2019, pukul 06.02

[4] Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung.

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, 29 April 2012

Timothy 13517087