Aplikasi Algoritma *Greedy* untuk Pergerakan Musuh pada Permainan *Pac-Man*

Timotius Nugroho Chandra / 13508002
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18002@students.if.itb.ac.id

Abstrak—*Pacman* adalah suatu permainan 'sepanjang-masa' yang mungkin tak akan lekang oleh waktu. Permainan ini berawal lahir di tahun 1980 dan merupakan permainan yang sangat populer pada saat itu dan bahkan hingga saat ini.

Permainan *Pacman* termasuk salah satu jenis permainan yang beraliran labirin (*maze*). Konsep permainannya pun sangat sederhana : pemain harus menggerakkan karakter *Pacman* untuk memakan semua makanan yang ada dalam labirin tersebut semuanya sampai tak tersisa satu pun.

Sekilas permainan ini terlihat mudah, namun untuk membuatnya semakin menarik, ada karakter musuh yang dapat mengejar *Pacman*. Jika *Pacman* bersentuhan dengan karakter musuh ini, dia akan mati.

Makalah ini tidak menitikberatkan pada karakter *Pacman* dan bagaimana menyelesaikan permainan *Pacman*, namun pada karakter yang menjadi musuh *Pacman*. Objektif dari musuh *Pacman* adalah mengejar dan menabrakkan diri pada karakter *Pacman* yang ada dalam labirin tersebut sesegera mungkin sebelum *Pacman* memakan habis semua makanan dalam labirin tersebut.

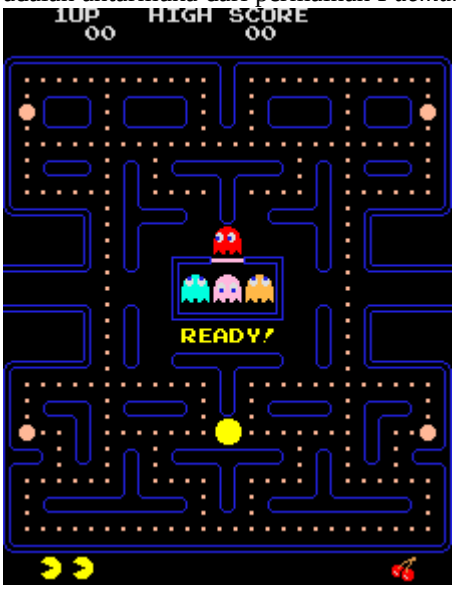
Kata kunci : *greedy*, *pacman*, *shortest path*, optimasi.

I PENDAHULUAN[1]

Permainan *Pacman* adalah sebuah permainan beraliran labirin (*maze*) dengan konsep yang sangat sederhana : diberikan suatu labirin dengan berbagai dinding rintangan di dalamnya, suatu titik yang merupakan posisi awal munculnya karakter *Pacman*, titik-titik yang harus dilewati, dan - yang paling menarik – beberapa musuh yang harus dihindari.

Objektif permainan *Pacman* adalah : pemain diharuskan menggerakkan karakter *Pacman* menyusuri labirin tersebut melewati semua titik-titik yang telah dispesifikasikan sebelumnya tanpa satu kali pun bertemu dengan musuh. Titik-titik dalam labirin yang harus dilewati oleh *Pacman* digambarkan sebagai makanan *Pacman*.

Berikut adalah antarmuka dari permainan *Pacman* :



Gambar 1 Antarmuka Permainan Pacman

Ketika permainan Pacman dimulai, selama beberapa detik musuh (digambarkan sebagai karakter yang berwarna biru, merah, merah mudah, coklat muda, dan memiliki mata) akan terkurung di kotak di bagian tengah permainan, setelah beberapa detik, karakter-karakter tersebut akan bebas dan mulai berjalan secara acak (random).

Untuk meningkatkan kemewahan dan kesulitan permainan Pacman, karakter musuh dapat diberi suatu kecerdasan khusus agar pada setiap tahap permainan, karakter musuh tersebut dapat menentukan rute yang paling pendek (minimum) agar semakin mendekati posisi karakter Pacman.

Persoalan mendekati karakter Pacman ini dapat diselesaikan dengan berbagai macam cara, salah satu cara yang cukup sangkil (walaupun tidak selalu menghasilkan hasil yang paling optimal) adalah dengan menggunakan **algoritma greedy**.

II DASAR TEORI[2]

Persoalan karakter musuh *Pacman* dalam menentukan arah mana yang harus dijalaninya untuk semakin mendekatkan dirinya kepada karakter *Pacman* dapat dikategorikan sebagai persoalan optimasi, dan persoalan optimasi cukup efektif dipecahkan dengan menggunakan algoritma *Greedy*.

Persoalan optimasi pada musuh *Pacman* ini termasuk persoalan minimasi, yaitu mencari rute terpendek saat ini dari posisi karakterk musuh ke posisi karakter *Pacman*.

Greedy diterjemahkan ke dalam bahasa Indonesia berarti : rakus, tamak, atau loba. Sesuai dengan namanya, prinsip dari algoritma *Greedy* adalah memilih keputusan yang terbaik pada setiap langkahnya (optimum lokal). Keputusa n – keputusan lokal yang diambil pada akhirnya menjadi solusi optimum yang mutlak (optimum global).

Elemen-elemen dari algoritma *greedy* adalah sebagai berikut :

- Himpunan Kandidat
- Himpunan Solusi : himpunan bagian dari himpunan kandidat yang merupakan solusi atas permasalahan.
- Fungsi Seleksi : fungsi yang memiliki unsur *greedy* di dalamnya, yang digunakan untuk menyeleksi himpunan kandidat.
- Fungsi Layak : untuk memeriksa apakah solusi yang dipilih merupakan solusi yang layak atau tidak.
- Fungsi Objektif : solusi yang dihasilkan optimum.

Namun demikian, kekurangan dari algoritma *greedy* adalah solusi akhir yang dibentuk oleh fungsi seleksi tidak selalu menghasilkan solusi global yang paling optimum. Hal ini dikarenakan algoritma *greedy* tidak memeriksa semua kemungkinan dan hanya mengambil yang terbaik relatif terhadap fungsi seleksi yang didefinisikan.

Meskipun demikian, algoritma *greedy* sangat cocok diterapkan untuk mendapatkan solusi yang mendekati optimum. Untuk dapat menghasilkan solusi yang semakin mendekati optimum, hal yang sangat menentukan adalah pemilihan fungsi seleksi, karena fungsi tersebut yang menentukan langkah mana yang diambil di tiap tahap.

Strategi atau pendekatan algoritma *greedy* pada suatu persoalan bukan hanya satu, namun bisa menjadi sangat banyak, dan masing – masing strategi bisa menghasilkan solusi yang berbeda-beda.

Karena algoritma *greedy* tidak menjamin optimalitas solusi, maka pemilihan fungsi seleksi pada algoritma ini menjadi sangat penting.

III IMPLEMENTASI

A. Elemen *Greedy*[2]

Elemen-elemen algoritma *greedy* pada permasalahan musuh *Pacman* adalah sebagai berikut :

- Himpunan Kandidat : himpunan titik-titik (*node*) yang merupakan posisi yang dapat dilalui oleh musuh *Pacman*
- Himpunan Solusi : himpunan titik-titik yang dipilih adalah rute yang berakhir pada posisi karakter *Pacman*.
- Fungsi Seleksi : titik (*node*) yang dipilih semakin mendekati posisi karakter *Pacman*.
- Fungsi Layak : titik yang dipilih dapat dilalui (bukan tembok atau karakter musuh lain)
- Fungsi Objektif : rute yang dipilih adalah rute yang paling optimum (dalam hal ini, paling pendek)

Fungsi seleksi pada persoalan ini dapat dijabarkan sebagai berikut :

- Jika karakter *Pacman* ada di sebelah kanan karakter musuh saat ini, maka musuh pindah ke kanan, jika tidak pindah ke kiri
- Jika karakter *Pacman* ada di sebelah atas karakter musuh saat ini, maka musuh pindah ke atas, jika tidak pindah ke bawah

Sebelum karakter musuh dipindahkan, terlebih dahulu dicek apakah langkah pemindahan tersebut sah (layak), dalam artian tidak ada dinding / tembok atau karakter musuh lain yang menghalangi.

B Kode Semu

Pada kode semua di bawah, digunakan dua tipe variabel bernama “musuh” dan “pacman” yang masing-masing merepresentasikan karakter musuh dan karakter *pacman*. Masing-masing tipe tersebut memiliki atribut X dan Y yang menunjukkan posisi absis dan ordinat tipe tersebut pada labirin permainan *pacman*.

Kode semu algoritma *greedy* untuk menentukan arah pergerakan karakter musuh *Pacman* adalah sebagai berikut :

Fungsi seleksi :

```
procedure gerakMusuh(m:musuh,p:pacman)
{
  if(p.X() >= m.X and isOK(m.X+1, m.Y))
  then
       pindahKanan(m)
```

```
  else
  if(p.Y()>= m.Y and (isOK(m.X, m.Y+1))
  then
       pindahAtas(m)
  else
  if(isOK(m.X, m.Y - 1) then
       pindahBawah(m)
  else
  if(isOK(m.X-1, Y)) then
       pindahKiri(m)
}
```

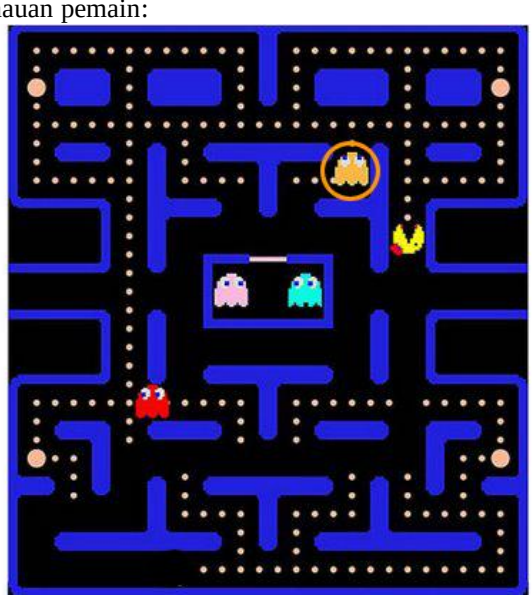
Fungsi layak : untuk menentukan apakah di posisi x dan y terdapat dinding atau karakter musuh lain yang menghalangi

```
function isOK(x, y:integer)-> boolean
{
  if(noDinding(x,y) && noMusuh(x,y))
       ➔ true
  else
       ➔ false
}
```

Dengan algoritma di atas, karakter musuh *pacman* digerakkan hingga mencapai suatu titik dalam labirin yang merupakan percabangan. Jadi fungsi gerakkanMusuh di atas akan dipanggil berulang-ulang setiap karakter musuh sampai di suatu percabangan.

IV Contoh Kasus

Berikut adalah salah satu contoh keadaan dalam permainan *pacman*, pada contoh kasus ini diasumsikan bahwa karakter *Pacman* tidak bergerak (diam saja di tempat), untuk menentukan apakah rute yang dipilih hasil algoritma *greedy* merupakan yang paling optimum atau tidak, walaupun dalam permainan *Pacman* yang asli karakter *Pacman* dapat bergerak-gerak sesuai dengan kemauan pemain:

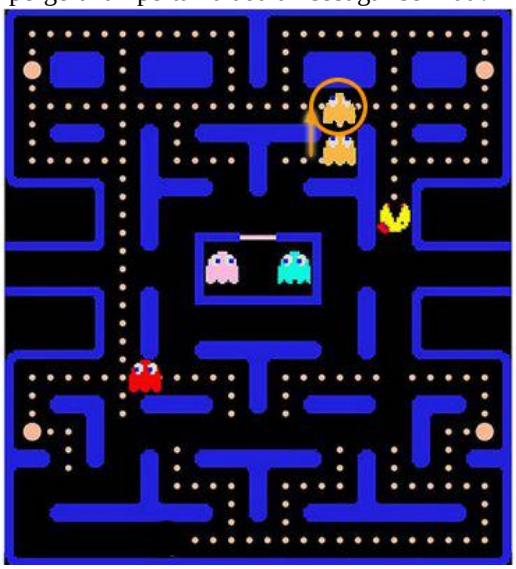


Gambar 2 Contoh Keadaan pada Permainan Pacman

Misal fungsi seleksi `gerakkanMusuh` diterapkan pada musuh *Pacman* yang berwarna oranye (pada gambar terlihat sebagai karakter yang dilingkari dengan lingkaran berwarna oranye).

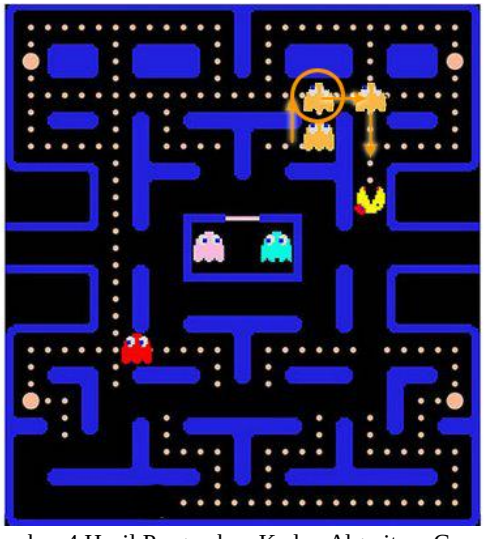
Posisi karakter musuh oranye berada di sebelah kiri karakter *Pacman* yang berwarna kuning (`m.X < p.X`), maka karakter musuh oranye seharusnya bergerak ke kanan, namun ternyata ada dinding yang menghalangi, maka dilakukan pengecekan lagi terhadap perbandingan posisi Y dan didapati posisi karakter musuh oranye berada di sebelah atas karakter *Pacman* (m.Y > p.Y) dan tidak ada dinding maupun karakter musuh lain yang menghalangi di atasnya, maka karakter musuh oranye dipindahkan ke atas.

Hasil pergerakan pertama adalah sebagai berikut :



Gambar 3 Hasil Pergerakan Pertama Algoritma Greedy

Setelah itu, diterapkan lagi algoritma *greedy* untuk kali kedua, posisi karakter oranye sekarang ada di sebelah kiri karakter *Pacman* (m.X < p.X) dan tidak ada yang menghalangi di sebelah kanannya, jadi karakter musuh bergerak ke kanan.

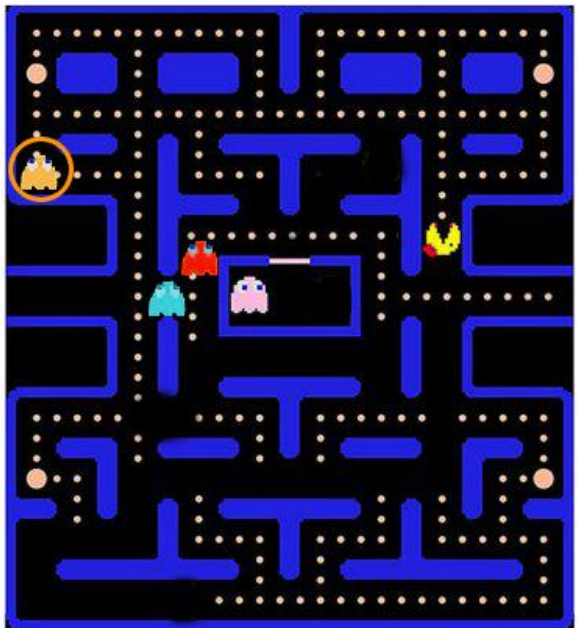


Gambar 4 Hasil Pergerakan Kedua Algoritma Greedy

Setelah bergerak ke kanan, algoritma *greedy* diterapkan lagi dan karakter musuh berada di atas *Pacman* (m.Y > p.Y), maka karakter musuh digerakkan ke bawah sampai bertemu dengan karakter *Pacman* : jarak yang ditempuh untuk menemukan *Pacman* adalah jarak yang paling pendek. Untuk kasus ini, algoritma *greedy* mangkus dan sangkil untuk menghasilkan optimal.

Namun sesuai dengan dasar teori, algoritma *greedy* tidak selalu dapat menghasilkan solusi yang optimal karena algoritma *greedy* tidak memeriksa semua kemungkinan.

Contoh kasus berikut adalah kasus lain dari permainan *pacman* yang ternyata tidak dapat diselesaikan secara optimum oleh algoritma *greedy* seperti contoh kasus pertama di atas, namun solusi yang dihasilkan tidak terlalu buruk.

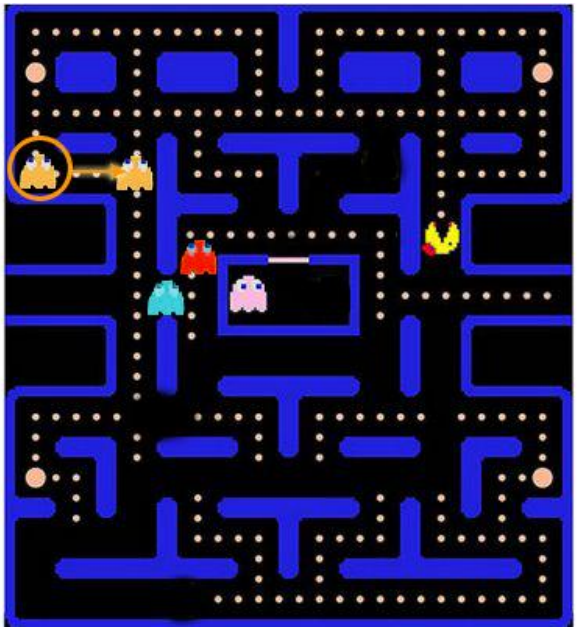


Gambar 5 Contoh Kasus Kedua

Pada contoh kasus yang kedua, tetap diasumsikan bahwa karakter *Pacman* tidak bergerak, selain itu, karakter musuh juga tidak ikut bergerak kemana-mana.

Misal fungsi seleksi diterapkan pada karakter musuh warna oranye sekali lagi (karakter musuh yang dilingkari lingkaran berwarna oranye). Karena musuh oranye ada di sebelah kiri posisi *Pacman* (m.X < p.X), maka musuh oranye digerakkan ke kanan.

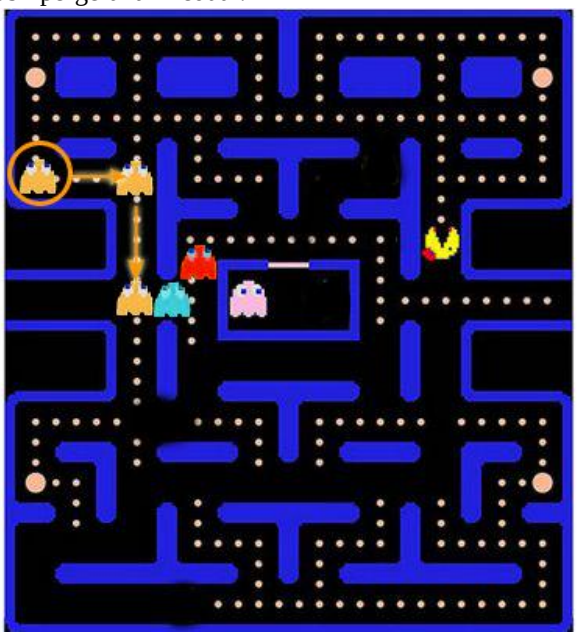
Hasil pergerakan pertama :



Gambar 6 Hasil Pergerakan Pertama

Setelah digerakkan ke kanan, posisi karakter musuh masih tetap di sebelah kiri *Pacman*, namun kali ini, musuh tidak bisa bergerak ke kanan lagi karena terhalang dinding, setelah dicek, ternyata karakter musuh berada di sebelah atas *Pacman* (m.Y > p.Y), maka, sesuai dengan algoritma *greedy* yang telah ditetapkan, karakter musuh digerakkan ke bawah.

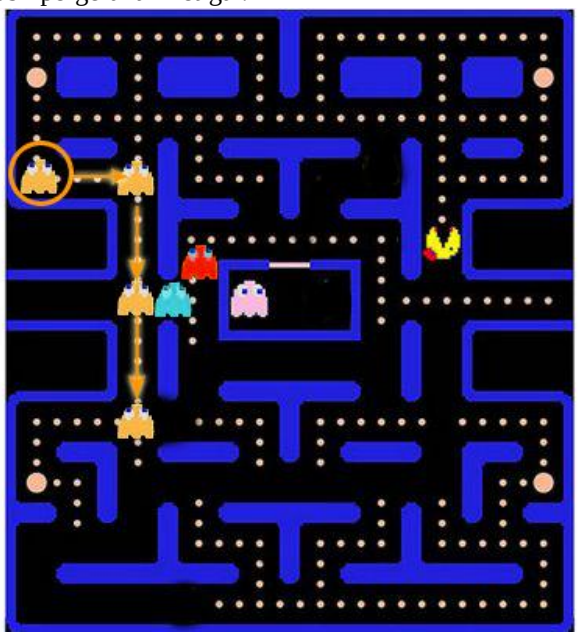
Hasil pergerakan kedua :



Gambar 7 Hasil Pergerakan Kedua

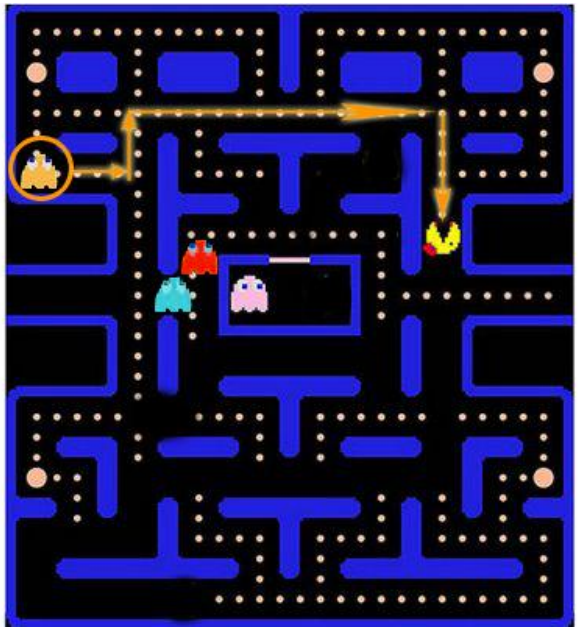
Setelah digerakkan ke bawah, posisi karakter musuh oranye ada di sebelah kiri dan di bawah *Pacman*. Algoritma *greedy* diterapkan sekali lagi dan karakter musuh seharusnya digerakkan ke kanan, namun karena ada karakter musuh lainnya di sana, maka karakter musuh oranye digerakkan ke bawah sekali lagi.

Hasil pergerakan ketiga :



Gambar 8 Hasil Pergerakan Ketiga

Pada hasil pergerakan ketiga diatas, karakter oranye bergerak ke bawah sekali lagi, dan dari sini terlihat bahwa rute yang ditempuh oleh karakter musuh oranye sudah tidak mungkin menjadi optimal lagi jika dibandingkan dengan solusi optimal (rute terpendek) yang mungkin dari posisi asal karakter oranye ke *Pacman* seperti pada gambar di bawah ini :



Gambar 9 Solusi Optimal Pergerakan Musuh Oranye

Solusi yang telah dicapai oleh algoritma *greedy* hingga pada gambar 8 di atas menunjukkan bahwa algoritma *greedy* yang dipilih ternyata tidak selalu menghasilkan solusi yang paling optimal. Namun demikian, karena objektif dari karakter musuh *Pacman* ini tidak harus selalu bergerak pada rute yang merupakan solusi paling optimum, maka algoritma *greedy* cukup baik untuk mendapatkan hampiran-hampiran yang mendekati solusi paling optimum tersebut.

V. KESIMPULAN

Strategi yang digunakan algoritma *greedy* cukup ampuh untuk diterapkan pada persoalan-persoalan optimasi. Hasil solusi yang didapatkan oleh algoritma *greedy* cenderung mendekati / menghampiri hasil solusi optimal, selain itu, yang menjadi kelebihan algoritma *greedy* sendir adalah implementasinya yang sangat sederhana dan langsung.

Namun demikian, jika yang dibutuhkan adalah solusi yang mengharuskan didapatkannya solusi yang paling optimum suatu algoritma *greedy* harus dapat dibuktikan secara matematis terlebih dulu bahwa algoritma tersebut dapat menghasilkan solusi yang paling optimum, jika tidak, maka tidak bisa dijamin bahwa algoritma *greedy* yang dibuat akan optimum untuk semua kasus.

Untuk kasus optimasi yang tidak mengharuskan didapatkannya solusi yang paling optimum, algoritma *greedy* sangat cocok untuk diterapkan, daripada menerapkan algoritma lain yang lebih kompleks dan rumit atau menerapkan algoritma *brute force* yang kompleksitasnya relatif lebih besar bila dibandingkan dengan algoritma *greedy*.

REFERENSI

[1] *Pac-Man*. http://en.wikipedia.org/wiki/Pacman. Waktu akses : 7 Desember 2010 pukul 23:00

[2] Munir. Rinaldi. “Diktat Kuliah IF2251 Strategi Algoritmik”. 2005. Bandung: Program Studi Teknik Informatika – Institut Teknologi 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, 9 Desember 2010

Timotius Nugroho Chandra / 13508002