Penerapan Algoritma Greedy dan Algoritma BFS untuk AI pada Permainan Greedy Spiders

Rachmawaty 13509071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509071@std.stei.itb.ac.id

Abstrak — Makalah ini membahas tentang algoritma Greedy dan algoritma Breadth First Search atau lebih dikenal dengan BFS beserta penerapannya dalam sebuah mobile game yaitu Greedy Spiders. Algoritma Greedy sendiri merupakan salah satu algoritma yang populer dalam memecahkan persoalan optimasi, baik maksimasi ataupun minimasi. Algoritma BFS merupakan salah satu algoritma untuk traversal graf. Greedy Spiders merupakan turn-based puzzle game yang memiliki storyline yang cukup menarik. Pada permainan ini terdapat laba-laba yang ingin memakan serangga yang sudah terperangkap di jaring laba-laba tersebut. Bagi pemain, objektif dari permainan ini adalah menyelamatkan serangga yang terperangkap agar tidak dimakan oleh laba-laba. Pemain bisa menyelamatkan serangga dengan menggunakan tools yang diberikan. Namun, penerapan algoritma Greedy dan BFS yang akan dibahas dalam makalah ini adalah untuk menentukan bagaimana pergerakan laba-laba yang merupakan AI dalam permainan ini dengan objektif untuk mencapai serangga yang menjadi santapannya.

Kata Kunci — algoritma Greedy, algoritma BFS, Greedy Spiders, laba-laba, jaring laba-laba, serangga

I. PENDAHULUAN

Dewasa ini, *game* telah menjamur di beragam kalangan. Terdapat banyak jenis *game* yang ada, mulai dari *puzzle game*, RPG (*Role Playing Game*), *racing game*, dan lain sebagainya. *Game* yang semakin menjamur di masyarakat didukung dengan semakin terjangkaunya harga *device* yang dapat digunakan untuk bermain *game* yang dijual di pasaran. *Game developer* pun semakin banyak bermunculan karena pangsa pasar yang semakin luas. *Game* yang akan dibicarakan dalam makalah ini adalah salah satu *puzzle game* pada *mobile phone* atau perangkat lainnya yang beroperasi pada sistem operasi *mobile* yang sedang *booming*, yakni android. *Puzzle game* sendiri adalah jenis *game* yang banyak diminati karena biasanya memiliki *rules* atau aturan yang mudah dan biasanya menjadi permainan asah otak.

Game yang akan dibahas adalah Greedy Spiders. Greedy Spiders merupakan *game* yang termasuk pada jenis *puzzle game* dimana *game* ini menggunakan logika sederhana untuk menyelesaikannya. Pada permainan ini terdapat laba-laba yang ingin memakan serangga yang terperangkap pada jaring laba-laba tersebut. Objektif bagi pemain pada permainan ini adalah untuk menyelamatkan serangga yang terjebak pada jaring laba-laba sehingga tidak dapat dimakan oleh laba-laba. Laba-laba tersebut merupakan AI pada permainan ini yang akan menjadi lawan bagi pemain dalam menyelesaikan permainan ini. Laba-laba tersebut akan mencari jalan terbaik untuk mendapatkan serangga untuk dimakan sebelum serangga dapat dibebaskan oleh pemain. Apabila pemain dapat membebaskan semua serangga yang terperangkap, maka pemain telah menyelesaikan level pada permainan ini. Namun, apabila laba-laba bisa memakan salah satu dari serangga yang terperangkap, maka permainan berakhir dan pemain gagal melewati level tersebut.

Penulis sendiri tidak tahu pasti algoritma apa yang sebenarnya digunakan untuk *Artificial Intelligence* (AI) pada permainan Greedy Spiders ini. Namun, penulis akan mencoba membahas penggunaan algoritma Greedy dan algoritma BFS untuk menentukan pergerakan AI pada permainan ini sehingga laba-laba dapat memakan salah satu dari serangga yang terperangkap. Penerapan dari algoritma Greedy dan BFS pada permainan Greedy Spiders akan dibahas pada bab-bab selanjutnya.

II. GREEDY SPIDERS, ALGORITMA GREEDY, DAN ALGORITMA BREADTH FIRST SEARCH (BFS)

A. Greedy Spiders

Gambar 1 Menu Utama Game Greedy Spiders

Greedy Spiders adalah sebuah *mobile game* untuk sistem operasi iOS dan android. Pengembang dari permainan ini adalah Blyts yang merupakan *software house*. Greedy

Spiders memiliki 128 level yang dibungkus dalam 4 buah level utama, yaitu Time to Eat, Wild Hills, Dry Escape, dan Scary Crypts.

Gambar 2 Empat Level Utama pada Permainan Greedy Spiders

Game ini merupakan *turn-based puzzle game* yang memiliki *storyline* yang cukup menarik. Pada permainan ini terdapat laba-laba yang ingin memakan serangga yang terperangkap pada jaring laba-laba tersebut. Objektif bagi pemain pada permainan ini adalah untuk menyelamatkan serangga yang terperangkap agar tidak dimakan oleh laba-laba. Pemain dapat menyelamatkan serangga dengan menggunakan tools yang diberikan, yaitu gunting untuk memotong jaring laba-laba, api untuk membakar jaring, *fake bugs* yang digunakan untuk menipu laba-laba sehingga laba-laba bergerak ke arahnya bukan ke arah serangga yang asli, atau menggunakan kekuatan supranatural. Terdapat tujuh serangga yang harus diselamatkan. Namun, pada tiap levelnya tidak seluruhnya muncul sekaligus.

Gambar 3 Jenis Serangga pada Permainan Greedy Spiders

Gambar 4 Contoh Interface Level 1-1pada Permainan Greedy Spiders

B. Algortima Greedy

Algoritma Greedy merupakan metode yang paling populer dalam memecahkan persoalan optimasi. Ada dua macam persoalan optimasi, yaitu maksimasi dan minimasi.

Prinsip yang digunakan dalam algoritma ini adalah "*take what you can get now!*" yang berarti ambil yang bisa didapat sekarang. Algoritma Greedy adalah algoritma yang memecahkan masakah langkah per langkah. Pada setiap langkah diambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan seperti prinsipnya dengan harapan bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Elemen-elemen algoritma Greedy :

1. Himpunan kandidat, C.
   Himpunan ini berisi elemen-elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

2. Himpunan solusi, S.
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi adalah himpunan bagian dari himpunan kandidat.

3. Fungsi seleksi (*selection function*)
   Fungsi ini dinyatakan dengan predikat seleksi. Merupakan dungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (*feasible*)
   Fungsi ini dinyatakan dengan predikat layak. Fungsi kelayakan ini merupakan fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang

sudah terbentuk tidak melanggar kendala (*constraints*) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. Fungsi obyektif
   Fungsi objektif ini merupakan sebuah fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Skema umum algoritma Greedy adalah sebagai berikut.

```
function greedy(input C: himpunan_kandidat)
→ himpunan_kandidat
{Mengembalikan solusi dari persoalan
optimasi dengan algoritma greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi yang bertipe
himpunan_kandidat}

Deklarasi
  x : kandidat
  S : himpunan_kandidat

Algoritma:
S ← {} {inisialisasi S dengan kosong}
while (not SOLUSI(S)) and (C ≠ {} ) do
  x ← SELEKSI(C){pilih sebuah kandidat
dari C}
  C ← C - {x} {elemen himpunan kandidat
berkurang satu}
  if LAYAK(S ∪ {x}) then
    S ← S ∪ {x}
  endif
endwhile
{SOLUSI(S) or C = {}}

if SOLUSI(S) then
  return S
else
  write('tidak ada solusi')
endif
```

Solusi optimum global yang diperoleh dari algoritma Greedy ini belum tentu merupakan solusi optimum (terbaik), tetapi *sub-optimum* atau *pseudo-optimum*. Hal ini dikarenakan algoritma Greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada dan terdapat beberapa fungsi seleksi yang berbeda sehingga kita harus memilih fungsi yang tepat jika menginginkan algoritma yang menghasilkan solusi optimal.

C. Algoritma Breadth First Search (BFS)

Terdapat dua buah algoritma traversal untuk graf.. Traversal di dalam graf berarti mengunjungi simpul-simpul dengan cara yang sistematik. Dua buah algoritma tersebut adalah *Breadth First Search* (BFS) dan *Depth First Search* (DFS). Pada makalah ini, hanya akan dibahas mengenai BFS.

Untuk algoritma BFS, traversal dimulai dari simpul v. Algoritmanya adalah sebagai berikut.

1. Kunjungi simpul v.
2. Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu.
3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Jika graf berbentuk pohon berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum mengunjungi simpul-simpul pada aras *d*+1.

Pseudo-code algoritma BFS, yaitu diperlukan :

1. Matriks ketetanggan A = [a_{ij}] yang berukuran *n x n*,
   a_{ij} = 1, jika simpul *i* dan simpul *j* bertetangga,
   a_{ij} = 1, jika simpul *i* dan simpul *j* tidak bertetangga.
2. Antrian q untuk menyimpan simpul yang telah dikunjungi.
3. Tabel *boolean* yang bernama `dikunjungi`

   ```
   dikunjungi : array[1..n] of Boolean
   ```

   dikunjungi[i] = true jika simpul *i* sudah dikunjungi
   dikunjungi[i] = false jika simpul *i* belum dikunjungi

   Inisialisasi tabel:

   ```
   for i←l to n do
     dikunjungi[i]← false
   endfor
   ```

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian
BFS.
Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpul yang dikunjungi
dicetak ke layar }

Deklarasi
  w : integer
  q : antrian

  procedure BuatAntrian(input/output q :
antrian)
{membuat antrian kosong, kepala(q) diisi 0}

  procedure MasukAntrian(input/output q :
antrian, input v : integer)
{memasukkan v ke dalam antrian q pada
posisi belakang}

  procedure HapusAntrian(input/output q :
antrian, output v : integer)
{ menghapus v dari kepala antrian q }

  function AntrianKosong(input q : antrian)
→ boolean
{ true jika antrian q kosong, false jika
sebaliknya }

Algoritma:
  BuatAntrian(q) { buat antrian kosong }
  write(v) { cetak simpul awal yang
dikunjungi }
  dikunjungi[v]¬true { simpul v telah
dikunjungi, tandai dengan true}
  MasukAntrian(q,v) { masukkan simpul awal
```

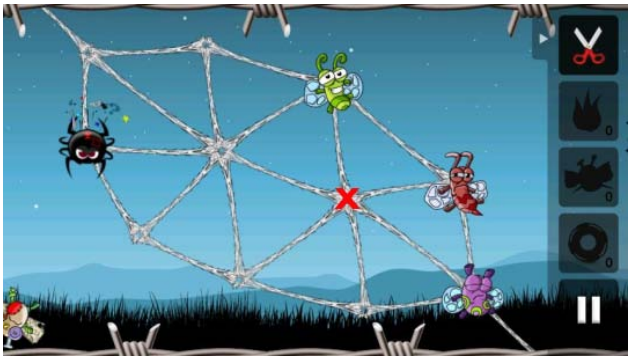
```
kunjungan ke dalam antrian }

{ kunjungi semua simpul graf selama antrian
belum kosong }
  while not AntrianKosong(q) do
    HapusAntrian(q,v) { simpul v telah
dikunjungi, hapus dari antrian }
    for w¬l to n do
      if A[v,w] = 1 then { v dan w
bertetangga }
        if not dikunjungi[w] then
          write(w) {cetak simpul yang
dikunjungi}
          MasukAntrian(q,w)
          dikunjungi[w]←true
        endif
      endif
    endfor
  endwhile
{ AntrianKosong(q) }
```

III. DESKRIPSI MASALAH

Seperti telah dijelaskan sebelumnya, pada permainan Greedy Spiders terdapat AI yaitu laba-laba yang akan memakan serangga yang terperangkap di jaring laba-laba tersebut. Namun, langkah yang dilakukan oleh laba-laba untuk mencapai titik di mana serangga berada tidak mudah karena ada gangguan dari pemain yang berusaha memutuskan jaring laba-laba ke arah serangga tersebut. Dikarenakan permainan ini merupakan *turn-based game* atau permainan berdasarkan giliran untuk bermain, laba-laba yang menjadi AI memperoleh giliran kedua yang berarti laba-laba akan bergerak setelah pemain melakukan langkahnya.

Laba-laba hanya bisa bergerak satu langkah dengan jarak satu titik pada jaring. Jaring laba-laba bisa diibaratkan sebagai graf yang memiliki bobot yang sama. Oleh, karena itu, laba-laba harus mencari jarak terpendek dari titik ia berada ke titik di mana serangga berada dan bergerak ke titik dimana kemungkinan ia bisa memakan serangga lebih besar. Titik di mana laba-laba memiliki kemungkinan memakan serangga paling besar yaitu apabila laba-laba berada pada titik ia memiliki 2 atau lebih serangga yang dapat dimakan dengan sekali langkah. Mengapa? Karena apabila satu jaring yang memiliki akses ke satu serangga diputus oleh pemain dan giliran berikutnya adalah AI, maka laba-laba masih memiliki satu atau lebih mangsa sehingga laba-laba dapat memakan salah satu serangga tersebut dan pemain gagal menyelesaikan level permainan tersebut.



Gambar 5 Titik X: Titik Kemungkinan Terbesar

Pada gambar di atas, titik x merupakan titik dimana laba-laba memiliki kemungkinan terbesar untuk memakan serangga. Apabila laba-laba berada di titik tersebut dengan sekali langkah lagi, laba-laba dapat memilih tiga buah serangga untuk dimakan yang menyebabkan permainan berakhir dan pemain gagal menyelesaikan level permainan tersebut. Oleh karena itu, untuk mencari titik x tersebut, digunakan Greedy By Serangga Terbanyak.

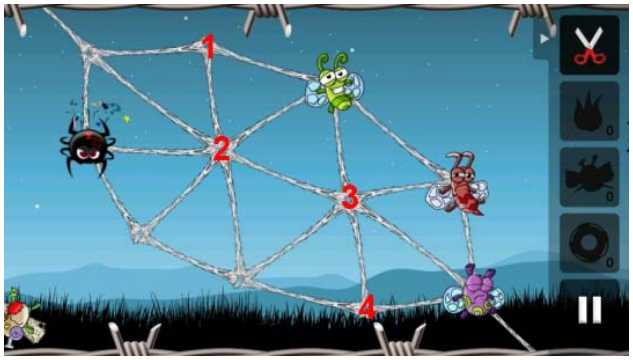
Setelah mendapatkan titik x, maka AI harus mencari tahu mana jalur terpendek yang bisa ia lalui untuk mencapai titik tersebut. Untuk itulah digunakan algoritma BFS. Mengapa dipilih algoritma BFS (*Breadth First Search*) dibanding DFS (*Depth First Search*)? Keduanya memang sama-sama algoritma yang digunakan untuk traversal graf untuk menemukan suatu titik. Pada kasus ini, jaring laba-laba dianggap sebagai graf yang memiliki bobot yang sama. Titik yang dicari, yakni titik x, sudah pasti ditemukan. Sekarang tinggal mencari jarak terpendeknya. Baik algoritma BFS maupun DFS, jarak dari laba-laba ke titik x sama-sama direpresentasikan dengan kedalaman pohon yang dibuat. Apabila menggunakan algoritma DFS, pada penelusuran awal, titik x pasti langsung ditemukan tetapi belum tentu mendapatkan kedalaman yang paling minimum. Sedangkan, dengan menggunakan algoritma BFS, memang titik x tidak ditemukan dengan sekali penelusuran, namun apabila ditemukan pertama kali, maka itulah kedalaman yang paling minimum atau jarak terdekat yang bisa diperoleh.

IV. IMPLEMENTASI ALGORITMA GREEDY DAN BREADTH FIRST SEARCH (BFS)

Penerapan algoritma Greedy dan BFS pada AI permainan Greedy Spiders yaitu laba-laba dimaksudkan untuk memperoleh hasil yang maksimal. Maksimal dalam kasus yang akan dibahas di sini berarti memperoleh jalur tersingkat pada pergerakan laba-laba guna mencapai serangga yang akan menyusahkan pemain dalam menyelesaikan permainan Greedy Spiders. Pembahasan implementasi algoritma Greedy dan BFS akan diberikan dengan studi kasus pada suatu level di permainan Greedy Spiders. Level yang digunakan untuk studi kasus adalah level 2-4.

Gambar 6 Level 2-4 pada Permainan Greedy Spiders

Algoritma Greedy yang digunakan adalah Greedy by serangga terbanyak. Dengan menggunakan algoritma Greedy ini, penyelesaiannya adalah pertama-tama melihat seluruh titik pada jaring laba-laba dimana serangga berada. Kemudian akan dilihat titik mana yang berjarak satu langkah ke tiap serangga.



Gambar 7 Empat Titik yang Berjarak Satu Langkah dari Serangga

Titik yang berjarak satu langkah untuk memakan Greenie adalah titik 1,2, dan 3. Titik yang berjarak satu langkah untuk memakan Toby adalah titik 3 saja. Titik yang berjarak satu langkah untuk memakan Wormy adalah titik 3 dan 4.

Titik	Jumlah Serangga yang Bisa Dimakan
1	1
2	1
3	3
4	1

Tabel 1 Titik Terdekat dan Jumlah Serangga

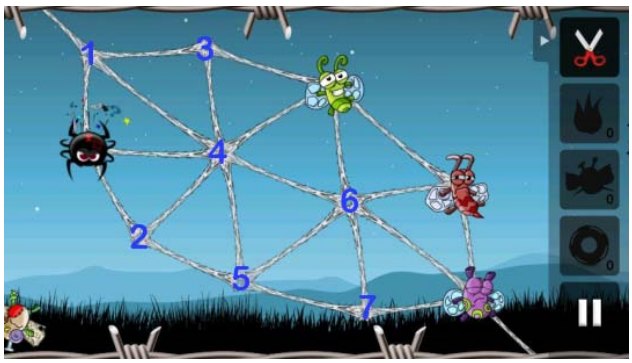
Dari tabel tersebut dapat diambil kesimpulan bahwa kemungkinan terbesar untuk laba-laba memakan serangga adalah apabila laba-laba berada di titik 3 dengan junlah serangga yang bisa ia makan berjumlah tiga, yaitu Greenie, Toby, dan Wormy.

Implementasi algoritma Greedy tersebut adalah sebagai berikut.

- Himpunan Kandidat
  Seluruh titik pada jaring laba-laba dengan jarak satu terhadap titik yang terdapat serangga
- Himpunan Solusi
  Titik dengan jarak satu langkah yang memiliki jumlah serangga terbanyak yang bisa dimakan
- Fungsi Seleksi
  Titik yang memiliki jumlah serangga terbanyak yang bisa dimakan dalam sekali langkah lagi
- Fungsi Layak
  Titik memiliki minimal satu serangga yang dapat dilahap dalam sekali langkah lagi

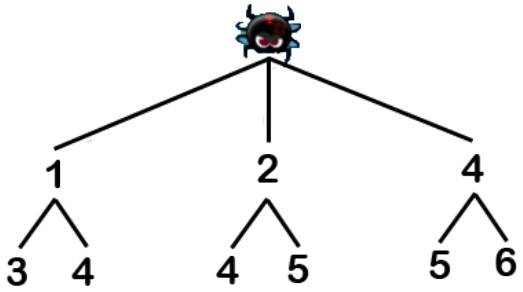
Kemungkinan terburuk adalah apabila masing-masing titik hanya bisa memakan satu serangga. Maka akan langsung dicari titik terdekat dari posisi laba-laba.

Setelah diperoleh titik tersebut, kemudian dilakukan pencarian jalur untuk menuju titik tersebut. Pencarian jarak ini dapat menggunakan algoritma *Breadth First Search* atau BFS di mana kedalaman pohon merepresentasikan jarak yang akan dilalui.



Gambar 8 Titik-Titik Penelusuran

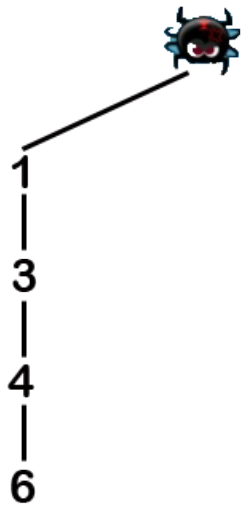
Pohon yang akan dibentuk dengan menggunakan algoritma BFS adalah sebagai berikut.



Gambar 9 Pohon BFS

Titik yang dicari adalah titik dengan nomor 6. Dapat dilihat pada gambar pohon yang terbentuk dengan menggunakan algoritma BFS di atas, titik 6 ditemukan pada kedalaman dua. Dari titik di mana laba-laba berada (L) hingga titik 6 dengan penelusuran BFS akan melalui titik-titik 1 → 2 → 4 → 3 → 4 → 4 → 5 → 6. Jalur terpendek untuk laba-laba lalui adalah dari titik L ke titik 4 kemudian ke titik 6 (L → 4 → 6).

Untuk pembanding, penulis akan memperlihatkan penelusuran menggunakan algoritma DFS (*Depth First Search*) untuk mencapai titik 6. Pohon yang akan dibentuk dengan DFS ini adalah sebagai berikut.



Gambar 9 Pohon DFS

Dengan menggunakan algoritma DFS akan diperoleh pohon yang bisa dilihat pada gambar di atas. Satu kali penelusuran akan mencapai titik yang dituju namun belum tentu memiliki kedalaman minimum. Kedalaman yan gdiperoleh dari pohon yang dibentuk oleh algoritma DFS di atas adalah 4 yang berarti dengan 4 kali langkah baru laba-laa akan mencapai titik tersebut. Pada algoritma DFS titik yang dilalui dan jalur yang diperoleh adalah sama, yaitu L → 1 → 3 → 4 → 6.

V. KESIMPULAN

Algoritma Greedy dan algoritma BFS dapat diterapkan pada AI permainan Greedy Spiders. Greedy yang digunakan adalah Greedy by serangga terbanyak, di mana akan dicari suatu titik yang apabila laba-laba tersebut berada di titik tersebut, laba-laba memiliki dua atau lebih pilihan serangga untuk dimakan dalam sekali langkah lagi. Dengan berada di titik yang seperti itu, pemain pasti gagal menyelesaikan satu level permainan Greedy Spiders. Kemudian, algoritma BFS akan dipanggil setelah menemukan titik yang harus dicapai. Algoritma BFS ini dapat memberikan jarak minimum yang harus ditempuh oleh laba-laba menuju titik tersebut.

VI. UCAPAN TERIMA KASIH

Makalah ini dibuat untuk memenuhi tugas mata kuliah IF3051 Strategi Algoritma. Penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir selaku dosen mata kuliah ini yang senantiasa mengajar dan membimbing kami.

Ucapan terima kasih juga diberikan kepada semua pihak yang telah membantu dalam pembuatan makalah ini sehingga makalah ini dapat penulis selesaikan.

DAFTAR REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Informatika Sekolah Teknik Elektro dan Informatika ITB, 2006.

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, Desember 2011

Rachmawaty
13509071