Implementasi Algoritma *Greedy* pada Permainan Ludo

Sylvia Juliana, 13515070
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl, Ganesha 10 Bandung 40132, Indonesia
13515070@std.stei.itb.ac.id

***Abstrak*—Algoritma *Greedy* merupakan sebuah metode yang dirancang untuk memberikan solusi optimum dari sebuah persoalan optimasi. Namun, pada kenyataannya algoritma *Greedy* sering kali tidak menghasilkan solusi yang optimum. Permainan Ludo merupakan sebuah *board game* yang memerlukan strategi dalam penyelesaiannya. Strategi tersebut digunakan untuk memenangkan permainan dengan jumlah giliran paling sedikit.**

***Kata kunci*—greedy; solusi; optimum; Ludo**

I. PENDAHULUAN

Banyak persoalan yang menuntut hasil yang optimum. Algoritma *Greedy* merupakan salah satu metode yang dirancang untuk memberikan solusi optimum dari sebuah persoalan optimasi. Algoritma tersebut menyelesaikan persoalan langkah per langkah yang di setiap langkahnya dipilih optimum lokal. Gabungan seluruh solusi optimum lokal tersebut diharapkan menghasilkan solusi optimum global pada akhir langkah. Namun, pada kenyataannya sering kali algoritma *Greedy* tidak menghasilkan solusi yang optimum.

Permainan Ludo merupakan sebuah *board game* hasil penyederhanaan permainan Pachisi yang berasal dari India. Permainan ini memerlukan strategi dalam penyelesaiannya. Strategi tersebut digunakan agar didapat hasil yang optimum, yaitu memenangkan permainan dengan jumlah giliran paling sedikit. Pada makalah ini akan dibahas implementasi strategi atau algoritma *Greedy* pada permainan Ludo untuk menghasilkan solusi yang optimum tersebut.

II. ALGORITMA *GREEDY*

A. Pengertian

Algoritma *Greedy* merupakan sebuah metode yang dirancang untuk memberikan solusi optimum dari sebuah persoalan optimasi (*optimization problem*) [1]. Solusi optimum (terbaik) adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang mungkin. Terdapat dua macam persoalan optimasi, yaitu maksimasi (*maximization*) dan minimasi (*minimization*). Selain itu, terdapat dua elemen persoalan optimasi, yaitu kendala (*constraint*) dan fungsi obyektif atau disebut juga fungsi optimasi. Solusi yang memenuhi semua kendala disebut solusi layak (*feasible solution*). Solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum. [2]

Algoritma *Greedy* menyelesaikan persoalan dengan membentuk solusi langkah per langkah (*step by step*). Sesuai dengan kata *greedy* yang memiliki arti serakah dalam Bahasa Indonesia, pada setiap langkahnya algoritma *Greedy* akan mengambil keputusan yang dianggap paling baik yang dapat diperoleh pada saat itu tanpa memperhitungkan konsekuensi nantinya. [3]. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Prinsip algoritma *Greedy* adalah "*take what you can get now!*". [2]

Dengan pendekatan pengambilan solusi terbaik pada setiap langkah atau disebut dengan membuat pilihan optimum lokal (*local optimum*), diharapkan pada akhir langkah hasil gabungan seluruh optimum lokal yang didapat menghasilkan solusi optimum global (*global optimum*). [2]

B. Skema Umum

Algoritma *Greedy* disusun oleh elemen-elemen berikut [2] :

- Himpunan kandidat (*C*)

  Himpunan kandidat merupakan himpunan dengan elemen pembentuk solusi.

- Himpunan solusi (*S*)

  Himpunan solusi merupakan himpunan yang berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

- Fungsi seleksi (*selection function*)

  Fungsi seleksi merupakan fungsi untuk memilih kandidat yang paling memungkinkan mencapai solusi optimal. Namun kandidat yang sudah dipilih pada suatu langkah, tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

- Fungsi kelayakan (*feasible*)

  Fungsi kelayakan merupakan fungsi untuk memeriksa kelayakan dari solusi yang diberikan oleh suatu kandidat yang telah dipilih atau dengan kata lain, kandidat yang telah dipilih setelah ditambahkan pada himpunan solusi tidak akan melanggar kendala yang ada. Kandidat yang layak, akan dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak akan dibuang dan tidak pernah dipertimbangkan lagi.

- Fungsi obyektif

  Fungsi obyektif merupakan fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Dengan kata lain, algoritma *Greedy* melakukan pencarian sebuah himpunan solusi (*S*) dari himpunan kandidat (*C*). Himpunan solusi yang dihasilkan harus memenuhi kriteria yang ditentukan, yaitu menyatakan suatu solusi dan dioptimasi oleh fungsi obyektif.

Pseudo-code algoritma *Greedy* secara umum adalah sebagai berikut [2] :

```
procedure Greedy (input C : himpunan_kandidat;
                  output S : himpunan_solusi)
{ Menentukan solusi optimum dari persoalan
  optimasi dengan algoritma Greedy.
  Masukan  : himpunan kandidat C
  Keluaran : himpunan solusi S  }

Deklarasi
  x : kandidat

Algoritma
  S ← {}      { inisialisasi S dengan kosong }
  while (belum SOLUSI(S)) and (C ≠ {}) do
    { pilih sebuah kandidat dari C }
    x ← SELEKSI(C)
    { elemen himpunan kandidat berkurang satu }
    C ← C - {x}
    if (LAYAK(S ∪{x}) then
       S ← S ∪ {x}
    endif
  endwhile
```

Pada *pseudo-code* di atas dapat dilihat bahwa pada akhir setiap pengulangan akan terbentuk solusi optimum lokal sedangkan pada akhir pengulangan akan terbentuk solusi optimum global. Namun pada kenyataanya, algoritma *Greedy* sering kali tidak menghasilkan solusi yang optimum melainkan hanya berupa solusi *sub-optimum* atau *pseudo-optimum*. Dihasilkannya solusi yang tidak optimum dapat disebabkan oleh [2] :

1. Algoritma *Greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (seperti pada metode *exhaustive search* pada algoritma *Brute Force*).
2. Pemilihan fungsi seleksi yang kurang tepat. Pada sebuah persoalan, terdapat kemungkinan adanya beberapa fungsi seleksi yang berbeda sehingga fungsi seleksi harus dipilih dengan tepat agar algoritma *Greedy* dapat bekerja dengan benar dan menghasilkan solusi yang optimum.

Oleh karena itu, beberapa persoalan yang diselesaikan dengan algoritma *Greedy* tidak berhasil memberikan solusi yang optimum. Jika persoalan yang akan dipecahkan tidak memerlukan hasil yang terbaik mutlak (benar-benar optimum), algoritma *Greedy* dapat menghasilkan solusi yang menghampiri optimum daripada menggunakan algoritma lain yang lebih rumit untuk menghasilkan solusi yang terbaik mutlak. Jika suatu persoalan yang diselesaikan menggunakan algoritma *Greedy* menghasilkan solusi optimum, keoptimalannya dapat dibuktikan secara matematis. [2]

C. Keuntungan dan Kerugian

Keuntungan penggunaan algoritma *Greedy* [3][4] :

- Sederhana

  Algoritma *Greedy* sering kali lebih mudah untuk dideskripsikan dan dibuat kodenya dibandingkan dengan algoritma lain.

- Efisien

  Algoritma *Greedy* sering kali terimplementasi dengan lebih efisien dibandingkan dengan algoritma lain. Selain itu, lebih mudah menganalisis waktu yang dibutuhkan sebuah algoritma *Greedy* untuk bekerja dibandingkan dengan algoritma lain.

Kerugian penggunaan algoritma *Greedy* [3][4] :

- Sulit dirancang

  Terkadang sulit untuk menemukan pendekatan algoritma *Greedy* yang benar pada sebuah persoalan. Tetapi, jika sudah ditemukan pendekatan yang benar, algoritma *Greedy* mudah untuk dirancang.

- Sulit dibuktikan

  Sulit untuk membuktikan benar atau tidaknya algoritma *Greedy* yang digunakan untuk memecahkan sebuah persoalan.

D. Aplikasi

Beberapa contoh persoalan yang dapat diaplikasikan menggunakan algoritma *Greedy*, yaitu [1][2] :

- masalah penukaran uang,
- minimisasi waktu di dalam sistem (penjadwalan),
- *integer knapsack*,
- *fractional knapsack*,
- penjadwalan *job* dengan tenggat waktu (*job scheduling with deadlines*),
- pohon merentang minimum (*minimum spanning tree*) dengan menggunakan algoritma Prim dan Kruskal,
- pencarian jalur terpendek (*shortest path*) termasuk penggunaan algoritma Djikstra,
- pemampatan data dengan algoritma Huffman,
- pecahan Mesir (*Egyptian Fraction*),
- *connecting wires*,
- *travelling salesman problem*.

Berikut ini akan dibahas salah satu pengaplikasian algoritma *Greedy*, yaitu pada persoalan penukaran uang. Persoalannya adalah jika diberikan uang senilai *A* dan uang tersebut akan ditukar menggunakan koin-koin uang dengan nominal-nominal tertentu, maka akan dicari jumlah minimum koin yang diperlukan untuk penukaran uang tersebut. Contohnya [2] :

- dimiliki uang senilai 32
- tersedia koin-koin dengan nominal 1, 5, 10, dan 25
- kemungkinan kombinasi koin-koin hasil penukaran beserta jumlahnya, yaitu :

  32 = 1 + 1 + … + 1 (32 koin)

  32 = 5 + 5 + 5 + 5 + 10 + 1 + 1 (7 koin)

  32 = 10 + 10 + 10 + 1 + 1 (5 koin)

  dst.
- strategi *Greedy* untuk persoalan ini adalah pada setiap langkah dilakukan pemilihan koin dengan nominal paling besar dari himpunan koin yang tersisa dengan syarat tidak melebihi nilai uang yang ditukarkan
- elemen-elemen algoritma *Greedy* :

  1. himpunan kandidat (*S*)

     terdiri atas himpunan koin dengan nominal 1, 5, 10, dan 25 yang memiliki paling sedikit satu koin untuk setiap nilai,

     $$\{d_1, d_2, d_3, d_4\} = \{1, 5, 10, 25\}$$

  2. himpunan solusi (*C*)

     terdiri atas total nilai koin yang dipilih tepat sama jumlahnya dengan nilai uang yang ditukarkan (dalam persoalan ini, nilai uang yang ditukarkan adalah 32),

     $$X = \{x_1, x_2, x_3, x_4\},$$

     $x_i = 1$ jika d_i dipilih sedangkan $x_i = 0$ jika d_i tidak dipilih,

  3. fungsi seleksi

     adalah fungsi yang akan memilih koin yang memiliki nilai paling tinggi dari himpunan kandidat yang tersisa,

  4. fungsi kelayakan

     adalah fungsi yang memeriksa nilai total dari himpunan koin yang dipilih agar tidak melebihi nilai uang yang akan ditukarkan,

  5. fungsi obyektif

     adalah fungsi untuk menghasilkan solusi dengan jumlah koin yang digunakan seminimum mungkin,

     $$F = \sum x_i$$

     dengan kendala

     $$\sum d_i x_i = A$$

- agar pemilihan koin optimal, maka perlu dilakukan pengurutan himpunan koin dengan urutan yang menurun
- penyelesaian persoalan :

  1. memilih 1 buah koin dengan nominal 25 sehingga total = 25,
  2. memilih 1 buah koin dengan nominal 5 sehingga total = 25 + 5 = 30,
  3. memilih 2 buah koin dengan nominal 1 sehingga total = 25 + 5 + 1 + 1 = 32,

  solusi persoalan dengan penerapan algoritma *Greedy*, yaitu diperoleh jumlah koin sama dengan 4 dan merupakan solusi optimum.

Pada persoalan penukaran uang, algoritma *Greedy* tidak selalu menghasilkan solusi optimum karena bergantung pada nominal koin uang yang digunakan. Contohnya, jika dimiliki uang senilai 7 dan tersedia koin-koin dengan nominal 5, 4, 3, dan 1, maka solusi yang diperoleh dengan menggunakan algoritma *Greedy* adalah

7 = 5 + 1 + 1 (3 koin)

sedangkan solusi optimumnya adalah

7 = 4 + 3 (2 koin)

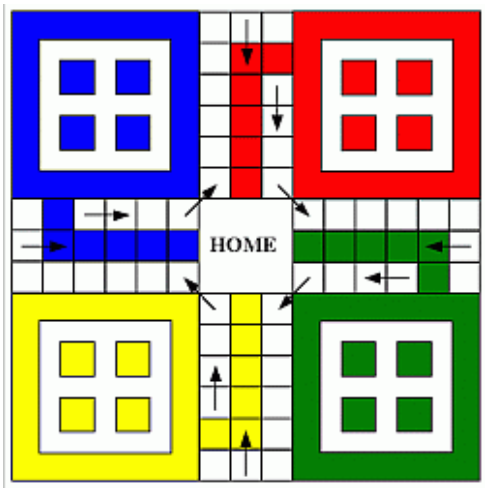
III. PERMAINAN LUDO

A. *Deskripsi Umum*

Ludo merupakan sebuah *board game* yang diadaptasi dari permainan Pachisi dari India. Ludo yang diciptakan pada akhir abad ke-19, masih menjadi salah satu permainan populer hingga saat ini. Walaupun pada saat bermain Ludo pemain dapat memilih banyak jalan, tetapi permainan ini juga membutuhkan keberuntungan dalam hal penentuan pemain yang menang dan kalah. Oleh karena itu, permainan Ludo merupakan salah satu permainan yang digemari juga oleh anak-anak. [5]

Gambar 1. Permainan Pachisi

(Sumber : https://en.wikipedia.org/wiki/Pachisi)



Gambar 2. Papan permainan Ludo

(Sumber : http://pachisi.vegard2.net/ludo.html)

Ludo dapat dimainkan oleh dua hingga empat pemain yang masing-masing pemainnya memiliki empat bidak. Pemain berlomba untuk menjalankan seluruh bidak yang dimilikinya melewati kotak-kotak bagian luar bentuk + pada papan permainan (pada gambar 2 ditandai dengan kotak-kotak berwarna putih). Jika sebuah bidak sudah mengelilingi seluruh kotak tersebut, bidak akan melewati kotak-kotak yang menuju ke tengah papan permainan untuk menyelesaikan perjalanannya. Pemain yang memenangkan permainan adalah pemain yang berhasil menjalankan seluruh bidak miliknya hingga ke tengah papan permainan. [5]

B. *Sejarah*

Pada abad ke-19, sebuah *board game* dari India bernama Pachisi menarik perhatian dunia bagian barat. Pachisi merupakan sebuah permainan yang terdiri atas 4 pemain. Dibutuhkan strategi dalam hal bermain Pachisi. Namun dilakukan beberapa penyederhanaan terhadap permainan Pachisi sehingga akhirnya dapat dimainkan oleh anak-anak. Hasil penyederhanaan Pachisi tersebut diberi nama Ludo.

Banyak hal yang diubah dari permainan Pachisi hingga akhirnya tercipta Ludo. Kulit kerang yang digunakan untuk menentukan pilihan jalannya pemain diganti menjadi dadu enam sisi. Tidak seperti Pachisi, kerja sama antar pemain tidak diperbolehkan pada permainan Ludo. Setiap pemain menjalankan masing-masing bidaknya hingga ke tujuan. Jalur perjalanan bidak juga dipermudah. [5]

Dengan adanya penyederhanaan Pachisi menjadi Ludo, Ludo menjadi lebih ideal untuk dimainkan anak-anak namun kurang cocok dimainkan orang dewasa [5].

C. *Peraturan*

Peraturan yang berlaku dalam permainan Ludo, yaitu [5] :

- Terdiri atas dua hingga empat pemain.
- Pada awal permainan, setiap pemain memiliki empat bidak pada daerahnya masing-masing. Daerah setiap pemain berada pada ujung-ujung papan permainan.
- Para pemain menentukan secara acak urutan pemain.
- Pemain memulai gilirannya dengan melempar dadu.
- Jika hasil pelemparan dadu yang didapat adalah enam, pemain dapat mengeluarkan bidak baru dari daerahnya untuk memulai perjalanan. Atau pemain juga dapat menjalankan bidak yang sudah ada di luar daerahnya sebanyak enam kotak.
- Hasil pelemparan dadu lain yang bernilai selain enam hanya dapat digunakan untuk menjalankan bidak yang sudah ada di luar daerah pemain dengan jumlah kotak sama dengan hasil pelemparan dadu tersebut. Jika pemain tidak memiliki bidak yang berada di luar daerahnya, permainan dilanjutkan ke pemain berikutnya.

- Pemain yang mendapat hasil pelemparan dadu senilai enam dan telah melakukan giliran, memiliki kesempatan untuk melempar dadu satu kali lagi dan melakukan giliran. Jika pada pelemparan kedua hasil yang didapat adalah enam, pemain mendapatkan satu kesempatan lagi. Namun, jika pada pelemparan ketiga hasil yang didapat adalah enam, permainan dilanjutkan pada pemain berikutnya.
- Sebuah bidak akan memulai perjalanannya dari kotak paling dekat daerah pemain yang berwarna sama dengan daerah tersebut. Lalu bidak akan menyusuri kotak pada papan permainan (pada gambar 2 ditandai dengan warna putih) hingga tersisa satu kotak sebelum kotak awal. Setelah itu, bidak akan menyusuri kotak berwarna sesuai daerah pemain yang menuju ke tengah papan permainan. Jika bidak sudah mencapai tengah papan permainan (pada gambar 2 ditandai dengan kotak bertuliskan "HOME"), maka perjalanan bidak selesai.
- Bidak dengan warna yang sama tidak diperbolehkan untuk menempati kotak yang sama.
- Jika sebuah bidak menempati kotak yang sudah berisi bidak berwarna lain, bidak yang sudah lebih awal menempati kotak tersebut harus kembali masuk ke daerah pemainnya.
- Pemain yang berhasil menyelesaikan seluruh perjalanan bidak untuk sampai ke tengah papan permainan akan memenangkan permainan. Namun, permainan dapat terus dilanjutkan untuk menentukan pemenang kedua maupun ketiga.

IV. Implementasi Algoritma Greedy pada Permainan Ludo

Dalam permainan Ludo, dapat diimplementasikan strategi atau algoritma *Greedy* untuk memenangkan permainan walaupun pada dasarnya permainan Ludo juga membutuhkan keberuntungan. Berikut ini akan dibahas implementasi algoritma *Greedy* pada permainan Ludo.

A. Elemen algoritma Greedy

1. Himpunan kandidat (*C*)

   Terdiri dari dua, yaitu mengeluarkan bidak dari daerah pemain dan menjalankan sebuah bidak yang sudah ada di luar daerah sesuai hasil pelemparan dadu.

2. Himpunan solusi (*S*)

   Terdiri dari langkah-langkah yang diambil hingga seluruh bidak pemain sampai pada tujuan atau menyelesaikan perjalanan (sampai pada papan permainan bagian tengah).

3. Fungsi seleksi

   Merupakan fungsi yang memilih langkah yang memiliki prioritas paling tinggi (prioritas permainan akan dijelaskan selanjutnya).

4. Fungsi kelayakan

   Merupakan fungsi yang memeriksa melanggar peraturan atau tidaknya sebuah kandidat jika dipilih oleh pemain.

5. Fungsi objektif

   Merupakan fungsi yang meminimumkan jumlah langkah pada permainan (jumlah giliran untuk memenangkan sebuah permainan).

B. Strategi Greedy

Berikut ini merupakan strategi *Greedy* yang saya kembangkan pada persoalan permainan Ludo dengan nomor urut terkecil menjadi prioritas paling tinggi untuk dilakukan pada saat bermain. Jika prioritas paling tinggi tidak dapat dilakukan, maka dilakukan pemeriksaan untuk melakukan prioritas dengan nomor urut berikutnya. Hal ini dilakukan terus hingga prioritas paling akhir.

1. Jika terdapat bidak musuh yang berada di bagian belakang bidak pemain dalam jarak satu hingga enam kotak, maka bidak pemain dipindahkan sesuai hasil pelemparan dadu. Namun, jika terdapat dua bidak atau lebih yang memiliki kondisi tersebut, dipilih bidak dengan jarak yang lebih dekat dengan bidak musuh. Jika terdapat dua bidak atau lebih dengan kondisi tersebut dan memiliki jarak yang sama dengan bidak musuh, dipilih bidak yang lebih dekat jaraknya dengan tujuan akhir. Langkah ini dilakukan jika dalam jarak sesuai hasil pelemparan dadu di depan bidak pemain tersebut, tidak terdapat bidak musuh lain. Hal ini dilakukan untuk menghindar dari bidak musuh agar bidak pemain tidak kembali masuk ke daerah pemain.
2. Jika terdapat bidak pemain yang berada di bagian belakang bidak musuh dengan jarak sesuai dengan hasil pelemparan dadu, maka bidak pemain dipindahkan sesuai hasil pelemparan dadu. Namun, jika terdapat dua bidak atau lebih dengan kondisi tersebut, maka dipilih bidak yang lebih dekat jaraknya dengan tujuan akhir. Langkah ini dilakukan jika dalam jarak sesuai hasil pelemparan dadu di depan bidak pemain tersebut, tidak terdapat bidak musuh lain. Hal ini dilakukan untuk mengembalikan bidak musuh ke daerahnya.
3. Jika hasil pelemparan dadu yang didapat adalah enam, maka dilakukan pengeluaran bidak baru dari daerah pemain. Hal ini dilakukan jika tidak ada bidak musuh yang berada pada jarak satu hingga enam kotak di bagian belakang kotak awal pengeluaran bidak pemain. Jika sudah tidak terdapat bidak pada daerah pemain, dilakukan pemindahan bidak yang memiliki jarak

paling dekat dengan tujuan akhir. Pemindahan dilakukan jika dalam jarak sesuai hasil pelemparan dadu di depan bidak pemain tersebut, tidak terdapat bidak musuh lain. Jika terdapat bidak musuh lain, dilakukan pemindahan bidak lain pemain dengan jarak paling dekat dengan tujuan akhir.

4. Jika hasil pelemparan dadu yang didapat bernilai selain enam, maka dilakukan pemindahan bidak yang memiliki jarak paling dekat dengan tujuan akhir. Pemindahan dilakukan jika dalam jarak sesuai hasil pelemparan dadu di depan bidak pemain tersebut, tidak terdapat bidak musuh lain. Jika terdapat bidak musuh lain, dilakukan pemindahan bidak lain pemain dengan jarak paling dekat dengan tujuan akhir.

Langkah-langkah ini hanya dapat dilakukan jika sudah terdapat paling sedikit satu buah bidak pemain yang berada di luar daerah pemain (melakukan perjalanan menuju tujuan akhir).

V. KESIMPULAN

Board games Ludo dapat diselesaikan dengan menerapkan strategi atau algoritma *Greedy*. Pada strategi penyelesaiannya, dapat dibuat prioritas-prioritas pengambilan keputusan untuk setiap giliran. Penerapan algoritma *Greedy* pada permainan Ludo mungkin tidak menghasilkan solusi yang optimum. Hal ini dikarenakan permainan Ludo juga bergantung pada keberuntungan pemain.

UCAPAN TERIMA KASIH

Pertama-tama saya ingin mengucapkan syukur kepada Tuhan Yang Maha Esa karena oleh rahmatNya saya dapat menyelesaikan makalah ini. Saya juga ingin berterima kasih kepada Ibu Masayu Leylia Khodra, Ibu Nur Ulfa Maulidevi, dan Bapak Rinaldi Munir yang telah mengajarkan mata kuliah IF2211 Strategi Algoritma pada semester 2 tahun 2016/2017 ini. Selain itu, saya juga ingin mengucapkan terima kasih kepada orang tua dan rekan-rekan yang telah memberikan semangat dan dorongan kepada saya untuk terus belajar.

REFERENSI

[1] *Data Structures – Greedy Algorithms.* *https://www.tutorialspoint.com/data_structures_algorithms/greedy_algorithms.htm*; Dikunjungi 17 Mei 2017.

[2] Munir, Rinaldi. 2004. *Bahan Kuliah ke-3 IF2251 Strategi Algoritmik Algoritma Greedy*. Bandung: Institut Teknologi Bandung.

[3] *Basics of Greedy Algorithms.* *https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/*; Dikunjungi 17 Mei 2017.

[4] *Greedy Algorithms Part One*. *https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/13/Small13.pdf*; Dikunjungi 18 Mei 2017.

[5] *Ludo*. *http://www.cyningstan.com/game/956/ludo*; Dikunjungi 18 Mei 2017.

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, 19 Mei 2017

Sylvia Juliana, 13515070