Penerapan Algoritma Greedy dalam Permainan Kartu "Gaple"

Ahmad Rizal Alifio/13517076
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517076@std.stei.itb.ac.id

***Abstrak*—Makalah dokumen ini berisi penerapan algoritma klasik *Greedy* dalam permainan gaple, sebuah permainan yang cukup umum ditemukan di masyarakat. Permainan ini sangat bergantung pada strategi pemilihan kartu untuk mencapai poin sekecil kecilnya.**

Kata Kunci—Gaple, Greedy, Poin, Pemilihan Kartu, Strategi Pemilihan

I. Pendahuluan

A. Latar Belakang

Permainan gapleh, gaple, atau umum juga disebut permainan domino merupakan permainan yang umum dijumpai pada masyarakat. permainan ini menggunakan kartu domino yaitu sebuah kartu berukuran kecil dengan 2 buah segi pada sisi depan kartu. Terdapat beberapa variasi permainan ini namun penulis memilih permainan yang paling sering dijumpai dimainkan oleh masyarakat. Permainan ini menarik perhatian penulis sebab dalam permainannya hanya bergantung pada kecocokan segi untuk kartu yang telah dikeluarkan dan yang dimiliki pemain. Selain itu juga *goal* untuk memenangkan permainan memiliki tujuan yang jelas: jumlah angka yang tersisa harus sekecil mungkin. Dikarenakan 2 hal ini maka penulis merasa strategi *Greedy* adalah strategi yang seharusnya mangkus untuk menyelesaikan permainan.

B. Tujuan

Tujuan penulis membuat makalah tentang topik ini adalah:

-Menambah pemahaman tentang algoritma Greedy

-Mengimplementasikan algoritma greedy

-Mengasah strategi permainan gaple

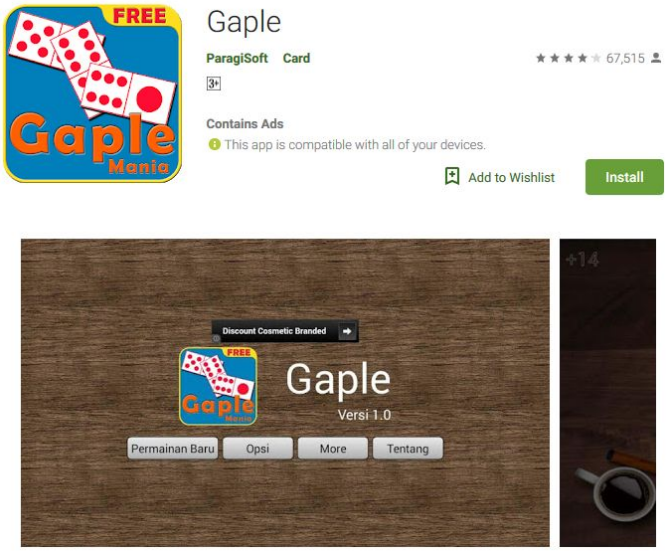
Gambar 1. Bidak Permainan Gaple, Sumber: http://kaskuszone.com/cara-menghitung-kartu-domino-dengan-tepat/kartu-domino/

II. Teori Dasar

A. Deskripsi Singkat Permainan Gaple

Gaple, Gapleh, atau Permainan Domino merupakan permainan dengan menggunakan kartu domino. Permainan ini bukanlah hal baru, sejarah permainan ini dapat ditelusuri asal muasalnya dari zaman Dinasti Song di Tiongkok sekitar abad ke-13. Permainan domino sendiri telah berevolusi menjadi beberapa variasi namun variasi paling terkenal adalah *Point Game* dimana pemain yang memiliki koleksi kartu dengan poin paling sedikit memenangkan permainan.

Pada awalnya permainan domino menggunakan *piece* keras seperti gambar 1, namun seiring perkembangan permainan ini makin digemari dan lama kelamaan muncullah versi kartu yang lebih terjangkau untuk dimiliki. Saat ini penulis telah menemukan permainan video untuk gaple ini yang dapat diunduh dari *PlayStore*. Anda dapat mencarinya dengan menggunakan kata kunci "Gaple" atau "Domino" di laman *PlayStore*, untuk menemukan permainan ini yang ditawarkan gratis dengan berbagai tampilan, bahasa, serta variasi mode permainan.



Gambar 2. Permainan Gaple berbahasa Indonesia di *PlayStore,* Sumber: *screenshot* dari hasil pencarian pada https://play.google.com/store?hl=en

1 set kartu domino berisi 28 kartu yang terdiri dari kombinasi pasangan 2 angka 1-6 (1-1, 1-2, … 6-6). Penghitungan poin untuk tiap kartu gaple adalah dengan menambahkan seluruh jumlah titik yang ada di kedua segi kartu, termasuk kartu balak (kartu dengan angka kedua segi sama) tanpa titik. Kartu tersebut bernilai 0 poin. Tentu peraturan poin ini dapat bervariasi sesuai lingkungan tempat memainkan, beberapa variasi yang telah penulis temui diantaranya poin kartu balak bernilai 2 kali poin jumlah titik, misalnya kartu balak 4 bukan bernilai 8 melainkan 16. Selain itu terdapat juga versi peraturan dimana jika pemain hanya memiliki kartu balak 0 pada akhir permainan, maka kartu balak tersebut memiliki poin 25, sementara jika pemain memiliki kartu lain selain kartu balak 0, maka kartu balak 0 tersebut akan memiliki nilai 0.

Aturan permainan gaple adalah sebagai berikut:

1. Permainan gaple maksimal hanya dapat dimainkan oleh 4 orang.
2. Setiap pemain masing masing mendapatkan 7 kartu dengan nilai yang acak.
3. Orang pertama yang kartunya habis akan menjadi pemenang.
4. Untuk putaran berikutnya, pemain yang memenangkan permainan sebelumnya berhak untuk membuang kartu duluan.
5. Bila seorang pemain mendapatkan lebih dari 5 kartu balak (kedua sisi kartu tersebut memiliki nilai yang sama) maka permainan diulang. Sebab pemain tersebut sudah dipastikan tidak akan menang yang mana hal ini dianggap tidak adil sehingga permainan harus diulang.
6. Jika kamu tidak ada kartu yang dapat dibuang maka kamu dapat melakukan pass.
7. Bila permainan tidak dapat dilanjutkan lagi (semua kartu pemain tidak ada yang dapat dibuang lagi) maka akan disebut gapleh. Untuk menentukan pemenang bila terjadi gapleh kita harus melihat nilai pada kartu yang tersisa. Pemain dengan nilai kartu terkecil dibandingkan dengan pemain lainnya akan menjadi pemenang.

Penulis tidak menemukan sumber yang menjelaskan mengapa di Indonesia, permainan ini didapat ditemui dimainkan di berbagai tempat oleh berbagai kalangan. Permainan ini cukup populer di Indonesia hingga kartunya pun dapat dijumpai dijual di warung-warung di pinggir jalan.

B. Algoritma *Greedy*

Algoritma *Greedy* adalah algoritma pemecahan masalah yang prinsipnya sederhana. *Greedy* sendiri secara bahasa berarti tamak atau rakus. Prinsip algoritma ini adalah mengambil keputusan yang memberi hasil paling sesuai terhadap sebuah fungsi yang dituju. Semboyan algoritma ini adalah : "*Take what you can get now!*".

Algoritma *Greedy* menggunakan prinsip optimalitas, artinya untuk setiap tahapan pengambilan keputusan untuk sebuah penyelesaian masalah algoritma ini akan mempertimbangkan semua kemungkinan yang saat itu tersedia lalu mengambil keputusan yang paling optimal, pilihan ini disebut optimal lokal. Cepat atau lambat algoritma *Greedy* akan mencapai akhirnya dan pada saat mencapai akhir masalah diharapkan rentetan pemilihan keputusan optimal lokal tadi akan terkumpul menjadi sebuah penyelesaian masalah secara utuh, atau disebut juga dengan sebutan optimal global.

Algoritma *Greedy* secara umum memiliki 5 elemen, yaitu:

1. Himpunan Kandidat

   Untuk setiap tahapan pemilihan solusi, terdapat beberapa kemungkinan pilihan yang dapat dipilih algoritma *Greedy*. Himpunan kandidat berisi seluruh kemungkinan tersebut yang nantinya akan dievaluasi oleh 3 fungsi selanjutnya. Sebagai contoh dalam persoalan pencarian lintasan dari simpul ke simpul pada graf, himpunan kandidat akan berisi kumpulan simpul tetangga dari simpul yang ditempati sekarang.

2. Fungsi Seleksi

   Untuk setiap langkah penyelesaian persoalan, fungsi ini menyeleksi himpunan kandidat untuk memilih keputusan yang paling optimal untuk mencapai strategi *Greedy* yang ditentukan. Kandidat yang

sudah dipilih tidak akan dipilih lagi dalam evaluasi himpunan kandidat yang selanjutnya.

3. Fungsi Kelayakan

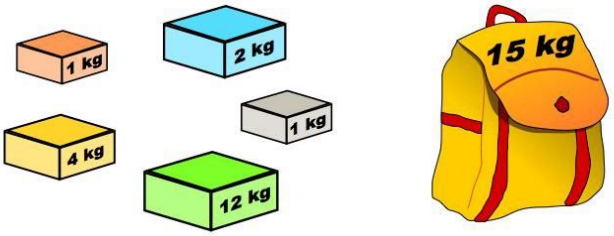
   Fungsi ini tidak selalu ada dalam sebuah algoritma *Greedy*, namun tidak berarti fungsi ini adalah fungsi sampingan. Fungsi kelayakan akan mengevaluasi pilihan dari fungsi seleksi dengan batasan yang ada. Jika pilihan melanggar batasan, kandidat dibuang dan evaluasi *Greedy* dimulai dari awal. Contohnya, dalam persoalan *Knapsack Problem*, jika fungsi seleksi mencari poin paling besar, maka fungsi kelayakan akan mengevaluasi apakah pilihan barang melebihi batasan berat yang diberikan.

4. Fungsi Objektif

   Fungsi objektif bergantung pada persoalan untuk mencari hasil akhir yang diinginkan. Untuk persoalan semacam *Knapsack Problem*, fungsi objektif akan memaksimalkan keuntungan, sementara untuk persoalan pohon merentang minimum graf, fungsi ini akan meminimalkan solusinya.

5. Himpunan Solusi

   Bagian akhir dari seluruh algoritma *Greedy,* himpunan solusi berisi kumpulan pilihan optimal lokal yang sekarang diharapkan menjadi optimal global.



Gambar 3. Ilustrasi Knapsack Problem, Sumber: https://medium.freecodecamp.org/how-to-solve-the-knapsack-problem-with-dynamic-programming-eb88c706d3cf

Singkatnya, algoritma *Greedy* melakukan pencarian kandidat, memeriksa validitas kandidat terhadap ketiga fungsi evaluator, lalu memasukan solusi lokal ke dalam himpunan solusi yang akan menjadi solusi global.

Dari perhitungan kompleksitas algoritma, algoritma *Greedy* lebih mangkus dan lebih cepat daripada algoritma *Brute Force* karena tidak mengevaluasi semua kemungkinan dengan segala variasinya, karena itu algoritma *Greedy* kerap digunakan sebagai algoritma alternatif jika lebih diperlukan kecepatan eksekusi pilihan daripada hasil akhir yang paling sempurna.

C. Kelebihan dan Kekurangan Algoritma *Greedy*

Sebagai sebuah algoritma, tentu ada hal yang ditawarkan oleh algoritma *Greedy* serta ada hal yang menjadi hambatan bagi algoritma ini untuk memberi hasil terbaik. Pertimbangan yang harus dipikirkan oleh seorang *Programmer* dalam menentukan pilihannya menggunakan algoritma *Greedy* adalah sebagai berikut:

- Kelebihan:
  - Sederhana, kompleksitas lebih rendah daripada *Brute Force*
  - Mangkus, eksekusi cepat karena evaluasi tidak banyak
  - Jika hasil yang diharapkan tidak memerlukan hasil terbaik, *Greedy* dapat menjadi pilihan.

- Kekurangan:
  - Bukan pilihan bagus jika membutuhkan hasil paling optimal
  - Perencanaan algoritma harus matang, sebuah persoalan dapat diatasi dengan berbagai strategi, karena itu memilih strategi paling tepat membutuhkan pertimbangan yang tidak sedikit
  - Tidak dapat digunakan untuk persoalan yang membutuhkan evaluasi semua kemungkinan secara global.

III. Penerapan Algoritma Greedy dalam Permainan Gaple

Kartu gaple dapat direpresentasikan dengan sebuah tipe data bentukan misalnya Card<int,int> dengan masing masing Integer menunjukkan satu dari dua nilai titik yang ada pada sebuah kartu gaple. Selanjutnya, pada permainan gaple, selalu terdapat 2 ekor segi yang menjadi tujuan untuk membuang kartu yang masing masing adalah ujung segi dari kartu pertama yang dibuang oleh pemain yang menang di permainan sebelumnya. Segi yang telah dibuang dapat direpresentasikan dengan tipe Integer dengan nama variabel ThrownEnd1 dan ThrownEnd 2. Algoritma greedy dalam persoalan ini akan dibuat untuk mengevaluasi kartu dengan poin tertinggi yang dapat dibuang ke 2 ekor tersebut. Pada makalah ini, strategi *Greedy* yang dipilih adalah untuk mengeluarkan kartu dengan poin terbesar.

Elemen-elemen permainan gaple yang diimplementasi dalam algoritma *Greedy* adalah sebagai berikut:

1. Himpunan Kandidat
   Himpunan solusi pada kasus ini berisi adalah seluruh kartu yang berada pada tangan pemain secara acak. untuk permainan dengan 4 pemain, akan terdapat 7 kartu gaple.

2. Fungsi Seleksi
   Fungsi untuk menghitung poin paling tinggi yang ada di tangan pemain.
3. Fungsi Kelayakan
   Fungsi ini akan menentukan apakah kartu yang dipilih dari fungsi seleksi sesuai dengan ujung segi kartu yang telah dibuang oleh pemain pada giliran sebelumnya.
4. Fungsi Objektif
   Memaksimalkan poin dari kartu yang dipilih. Hal ini diimplementasikan pada fungsi seleksi.
5. Himpunan Solusi
   Himpunan solusi ini hanya berisi 1 buah kartu yang akan dikeluarkan pada giliran pemain sesuai dengan hasil evaluasi fungsi di atas.

Berikut adalah potongan kode dalam bahasa C++ untuk menentukan pilihan kartu yang seharusnya dikeluarkan pemain.

```
#include <iostream>
#include <vector>

using namespace std;

#define Card pair<int,int>

int poin(Card c){
   return c.first + c.second;
}

bool isExistVec(Card test, vector<Card> evaluated){
    //Fungsi IsExist untuk mencari apakah kartu pernah dievaluasi atau tidak
   for(Card x : evaluated) {
       if(test == x){
           return true;
       }
   }
   return false;
}

Card maxPoin(vector<Card> PlayerCards, vector<Card> evaluated){
   //FUNGSI SELEKSI
   Card maxRes;
   maxRes.first = 0; maxRes.second = 0;
   for(Card x : PlayerCards){
           if (poin(maxRes) <= poin(x) && isExistVec(x, evaluated)){
           maxRes = x;
       }
   }

   return maxRes;
}

bool evalCard(int ThrownEnd1, int ThrownEnd2, Card x){
   //FUNGSI KELAYAKAN
   //ThrownEnd berisi nilai ujung segi dari kartu-kartu yang telah dibuang para pemain
       if(x.first == ThrownEnd1 || x.second == ThrownEnd1 || x.first == ThrownEnd2 || x.second == ThrownEnd2){
           return true;
       } else {
           return false;
       }
}

Card greedy(int ThrownEnd1, int ThrownEnd2, vector<Card> PlayerCards){
   //ThrownEnd berisi nilai ujung segi dari kartu-kartu yang telah dibuang para pemain
   vector<Card> evaluated;
   Card Result;

   do{
           Result = maxPoin(PlayerCards, evaluated);
       evaluated.push_back(Result);
       } while (!evalCard(ThrownEnd1, ThrownEnd2, Result) && !isEmptyVec(PlayerCards));

   if(isEmptyVec(PlayerCards)){
           //Kalau hal ini terjadi maka di eksekusi keputusan nanti pemain akan melakukan pass (tidak mengeluarkan kartu dari tangan pemain)
       Result = nil;
   }
   return Result;
}
```

Potongan kode di atas memperlihatkan bagaimana strategi greedy "Mengeluarkan kartu dengan poin terbesar" dapat menentukan kartu mana yang harus dipilih agar kartu yang dikeluarkan untuk berusaha mendapat poin sisa terkecil.

Berikut adalah penjelasan dari fungsi yang ada di potongan kode di atas:

- Fungsi `poin(Card c)`
  Fungsi ini mengembalikan nilai poin dari sebuah kartu gaple.
- Fungsi `isExistVec(vector<Card> s)`
  Fungsi ini adalah implementasi dari fungsi isExist untuk vector, berhubung vector tidak memiliki fungsi isExist bawaan.
- Fungsi `maxPoin(vector<Card>,vector<Card>)`
  fungsi ini adalah implementasi fungsi seleksi yang akan memilih kartu dengan poin terbesar yang ada di tangan pemain
- Fungsi `evalCard(int ThrownEnd1, int ThrownEnd2, Card x)`
  Fungsi kelayakan, fungsi ini akan memeriksa apakah kartu yang dipilih dapat dikeluarkan sesuai dengan ujung segi yang ada pada papan permainan.
- Fungsi `greedy(int ThrownEnd1, int ThrownEnd2, vector<Card> PlayerCards)`
  Fungsi ini adalah fungsi utama dimana semua fungsi kecil tadi dipanggil sehingga akhirnya akan menghasilkan pilihan kartu (atau pass) paling optimal untuk dikeluarkan pada giliran pemain kali ini.

IV. Analisis

Penulis melakukan percobaan 10 kali pada permainan gaple digital melawan komputer untuk menguji algoritma permainan yang dikembangkan pada tugas makalah ini. Hasil percobaan disajikan pada tabel berikut ini:

Kondisi	**Jumlah**
Menang	7
Tidak Menang	3
Total	10

Tabel 1. Hasil percobaan implementasi algoritma *Greedy*. Perhatikan bahwa kondisi kedua adalah tidak menang karena poin akhir belum tentu menunjukkan poin tersisa yang paling banyak

Pengujian ini tentunya tidak dapat menghasilkan data yang komprehensif mengingat jumlah pengujian yang sedikit. Namun penulis rasa ini dapat menggambarkan potensi kemenangan–atau setidaknya ketidakkalahan–yang dapat diraih dengan menggunakan cara bermain berdasarkan algoritma *Greedy*.

Tentunya *Greedy* bukanlah algoritma omnipoten yang dapat mengevaluasi semua aspek. Hal penting yang penulis rasa perlu disampaikan setelah memperhatikan permainan gaple dengan algoritma *Greedy* adalah ketidakmampuan greedy untuk menghitung probabilitas kematian sebuah ujung segi kartu. Contohnya adalah sebagai berikut:

Bayangkan sebuah permainan gaple yang sudah berjalan, saat ini giliran pemain untuk mengeluarkan kartu. Ujung segi yang muncul adalah 4 dan 1. Katakanlah algoritma *Greedy* memutuskan bahwa kartu 4-5 yang memiliki poin 9 adalah kartu dengan poin terbesar sehingga harus dibuang. Namun algoritma *Greedy* tidak menyimpan data tentang sudah berapa kali kartu bersegi 5 muncul, dalam kasus ini kartu bersegi 5 telah keluar sebanyak 5 kali, saat pemain mengeluarkan kartu 4-5, maka ujung segi tersebut akan mati karena tidak ada lagi pemain yang memiliki kartu bersegi 5. Hal ini meningkatkan kemungkinan permainan berakhir lebih cepat.

Hal ini juga dapat dieksploitasi oleh lawan dimana pemain lain mengeluarkan kartu dengan tujuan seperti di atas, lawan bisa saja mengeluarkan kartu dengan ujung segi yang tidak kita miliki, sehingga kita harus melakukan pass dan total poin pemain akan tetap tinggi.

V. Kesimpulan

Permainan gaple adalah permainan sederhana namun memiliki pertimbangan yang tidak sedikit. Secara umum, strategi greedy relatif menghasilkan solusi yang cukup baik untuk persoalan yang banyak berisi pertimbangan numerik.

Algoritma *Greedy* hanyalah satu alternatif dalam mengevaluasi pilihan solusi, dalam kasus ini khususnya pada permainan gaple. Namun masih ada hal yang belum dapat ditangani oleh algoritma *Greedy* karena keterbatasan kemampuan sederhananya. Selain itu keberuntungan pembagian kartu juga mempengaruhi kemungkinan kemenangan.

Akhir kata, penulis memberikan dukungan jika pembaca memiliki inspirasi baru untuk mengembangkan strategi permainan gaple dengan algoritma lain.

VI. Ucapan Terima Kasih

Puji Syukur penulis panjatkan kepada Allah SWT. karena berkat nikmat dan karunianya penulis dapat menyelesaikan makalah ini. Selanjutnya penulis ingin mengucapkan terima kasih kepada dosen mata kuliah Strategi Algoritma IF2211 Kelas 1 Ibu Nur Ulfa Maulidevi, Koordinator Mata Kuliah Strategi Algoritma IF2211 Bapak Rinaldi Munir, juga bersama jajaran dosen mata kuliah lainnya yang telah memberi kesempatan bagi penulis untuk mengeksplorasi dunia dan memanfaatkan ilmu yang telah diajarkan. Kemudian tidak lupa terima kasih pada orang tua penulis yang memberikan

bantuan, semangat, dan dorongan tanpa henti kepada penulis, rekan-rekan peserta mata kuliah Strategi Algoritma IF2211 yang telah berjuang bersama sama selama 1 semester berjerih payah dalam menggali ilmu di bangku perkuliahan, serta teman-teman terdekat penulis yang tanpa kehadiran dan semangat dari mereka penulis mungkin tidak akan termotivasi untuk menyelesaikan makalah ini.

VII. Referensi

[1] Lo, Andrew. "The Game of Leaves: An Inquiry into the Origin of Chinese Playing Cards," *Bulletin of the School of Oriental and African Studies*, University of London, Vol. 63, No. 3 (2000): 389-406.

[2] Rodney P. Carlisle (2 April 2009). *Encyclopedia of Play*. SAGE. p. 181. ISBN 978-1-4129-6670-2.

[3] Kelley, Jennifer A.; Lugo, Miguel (2003). *The Little Giant Book of Dominoes*. Sterling. ISBN 1-4027-0290-6.

[4] https://www.pagat.com/tile/wdom/math.html

[5] https://www.pagat.com/tile/wdom/history.html

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, 26 April 2019

Ahmad Rizal Alifio 13517076