Penerapan Algoritma Greedy Dalam Permainan Clash Royale

Ahmad Fa'iq Rahman
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514081@std.stei.itb.ac.id

***Abstract*—Clash Royale merupakan salah satu permainan yang saat ini sering dimainkan oleh banyak kalangan darimulai pelajar, mahasiswa, sampai pekerja. Algoritma *greedy* merupakan salah satu algoritma yang dipelajari dalam kelas. Apakah ada hubungan antara kedua hal tersebut? Makalah ini akan menunjukkan bahwa algoritma greedy dapat dipakai dalam memenangkan permainan ini.**

Keywords—Algoritma Greedy; Clash Royale; Game;

I. PENDAHULUAN

Bermain adalah salah satu kegiatan yang dilakukan dalam mengisi waktu luang yang dimiliki oleh setiap orang. Disela kesibukan yang sangat padat, setiap orang pasti memiliki kegiatan yang dengan sengaja dilakukan untuk menghilangkan kejenuhan setelah mengerjakan kesibukan biasanya. Menonton, membaca, atau mendengar lagu merupakan contoh dari kegiatan yang dilakukan demi menghilangkan rasa bosan dalam melaksanakan rutinitas. Kegiatan lain yang tidak kalah sering dilakukan adalah bermain. Ada sangat banyak tipe permainan, darimulai permainan fisik, permainan papan, sampai permainan *mobile*. Pada makalah ini, akan dibahas salah satu permainan *mobile* yang sedang naik daun dan banyak dimainkan oleh orang-orang.

Permainan ini dinamakan *Clash Royale*, sebuah permainan yang dibuat oleh *mobile game developer* bernama Supercell. Permainan ini merupakan adaptasi dari permainan *Clash of Clans*, permainan lain yang juga dibuat oleh Supercell. Permainan yang memiliki tipe gabungan dari TCG (*Trading Card Game*), MOBA (*Mobile Online Battle Arena*), serta TD (*Tower Defense*) merupakan permainan yang sangat adiktif karena tampilannya yang sangat menarik, variasi kombinasi kartu yang sangat banyak, serta musuh yang beragam.

Gambar 1: *Clash Royale*
(https://i.ytimg.com/vi/x6aQ67jk58I/maxresdefault.jpg)

Salah satu algoritma yang dipelajari pada mata kuliah IF 2211 Strategi Algoritma merupakan Algoritma *greedy*. Algoritma yang memiliki arti yang sangat harafiah yaitu serakah, mengambil keputusan secara langsung tanpa memikirkan kedepannya. Makalah ini akan menunjukkan penerapan algoritma tersebut dalam permainan *Clash Royale* untuk memenangkannya. Memanfaatkan sifat alami algoritma *greedy* untuk menghancurkan menara musuh sebanyak-banyaknya.

II. DASAR TEORI

A. *Algoritma Greedy*

1. Prinsip Algoritma

Algoritma *greedy* adalah algoritma yang menggunakan prinsip *take what you can get now*. Algortima ini biasa digunakan untuk penyelesaian masalah optimasi. Masalah optimasi adalah masalah yang berbentuk mencari solusi optimum. Ada dua macam persoalan optimasi, maksimasi dan minimasi. Maksimasi berarti mencari jumlah maksimal yang bisa didapat dalam suatu persoalan sedangkan minimasi berarti mencari jumlah minimal yang bisa didapat dalam suatu persoalan.

Algoritma ini mengevaluasi pilihan langkah demi langkah. Pada setiap langkah, algoritma membentuk solusi dengan mengambil keputusan yang terbaik. Memecahkan masalah

langkah demi langkah dengan memilih optimum lokal dan berharap langkah sisanya mengarah ke optimum global.

Optimum global yang didapat dari algoritma ini tidak selalu merupakan solusi optimum terbaik karena algoritma ini beroperasi tidak secara menyeluruh terhadap seluruh alternatif solusi yang ada dan fungsi yang dipilih harus tepat. Oleh karena itu, algoritma ini sering digunakan untuk menghasilkan solusi hampiran dari suatu masalah.

2. Elemen Algortima

- Himpunan Kandidat, C
  Berisi kandidat pilihan yang bisa dipilih dari permasalahan yang ada.
- Himpunan Solusi, S
  Berisi pilihan yang dipilih pada setiap langkah (optimum lokal) untuk mendapat penyelesaian masalah (optimum global)
- Fungsi Seleksi
  Fungsi untuk memilih elemen dari himpunan kandidat untuk dimasukkan kedalam himpunan solusi.
- Fungsi Kelayakan.
  Fungsi untuk memeriksa kelayakan hasil seleksi untuk masuk ke himpunan solusi
- Fungsi Objektif
  Fungsi yang menunjukkan optimum global

3. Contoh Pemakaian Algoritma

A. Minimasi Waktu dalam Sistem (Penjadwalan)

Persoalan: Sebuah *server* (dapat berupa *processor*, pompa, kasir di bank, dll) mempunai *n* pelanggan (*customer*, *client*) yang harus dilayani. Waktu pelayanan untuk setiap pelanggan *i* adalah t_i. Minimumkan total waktu dalam sistem!

Tiga pelanggan dengan

$t_1 = 5, \quad t_2 = 10, \quad t_3 = 3,$

Enam urutan pelayanan yang mungkin:

==========

Urutan	*T*
1, 2, 3:	5 + (5 + 10) + (5 + 10 + 3) = 38
1, 3, 2:	5 + (5 + 3) + (5 + 3 + 10) = 31
2, 1, 3:	10 + (10 + 5) + (10 + 5 + 3) = 43
2, 3, 1:	10 + (10 + 3) + (10 + 3 + 5) = 41
3, 1, 2:	**3 + (3 + 5) + (3 + 5 + 10) = 29 ê (optimal)**
3, 2, 1:	3 + (3 + 10) + (3 + 10 + 5) = 34

Algoritma:

1. Urutkan pelanggan berdasarkan waktu pelayanan dalam urutan menaik
2. Pada setiap langkah, pilih pelanggan yang membutuhkan waktu pelayanan terkecil di antara pelanggan lain yang belum dilayani

Elemen:

$C = \{t_1, t_2, t_3\}$

$S = \{\}$

Seleksi = pelanggan dengan waktu pelayanan terkecil

Kelayakan = tidak ada

Objektif = waktu yang digunakan minimum

Kesimpulan:

Pada masalah minimasi waktu seperti ini, algoritma *greedy* selalu memenuhi fungsi objektif (mendapat solusi optimum).

B. Integer Knapsack

Persoalan: Terdapat beberapa barang, setiap barang memiliki berat dan nilai. Terdapat sebuah wadah dengan berat maksimal yang dapat dimuat. Tentukan barang mana saja yang dapat dimasukkan kedalam wadah sehingga mendapat nilai sebanyak-banyaknya dari barang-barang tersebut!

Algortima:

$C = \{barang_1, barang_2, \ldots, barang_n\}$

$S = \{\}$

1. *Greedy by profit*.

   Seleksi = barang yang mempunyai nilai terbesar

   Kelayakan = jumlah berat lebih kecil atau sama dengan kapasitas

   Objektif = mendapat nilai sebanyak-banyaknya

2. *Greedy by weight.*

   Seleksi = barang yang mempunyai berat teringan

   Kelayakan = jumlah berat lebih kecil atau sama dengan kapasitas

   Objektif = mendapat nilai sebanyak-banyaknya

3. *Greedy by density*.

   Seleksi = barang yang mempunyai p_i / w_i terbesar

   Kelayakan = jumlah berat lebih kecil atau sama dengan kapasitas

   Objektif = mendapat nilai sebanyak-banyaknya

Contoh:

$w_1 = 100$; $p_1 = 40$; $w_2 = 50$; $p_2 = 35$; $w_3 = 45$; $p_3 = 18$; $w_4 = 20$; $p_4 = 4$; $w_5 = 10$; $p_5 = 10$; $w_6 = 5$; $p_6 = 5$; Kapasitas = 100

Properti Objek				*Greedy By*			Solusi Optimal
i	w_i	p_i	p_i / w_i	*profit*	*weight*	*density*	
1	100	40	0.4	1	0	0	0
2	50	35	0.7	0	0	1	1
3	45	18	0.4	0	1	0	1
4	20	4	0.2	0	1	1	0
5	10	10	1.0	0	1	1	0
6	5	2	0.4	0	1	1	0
Total Bobot				100	80	85	95
Total Keuntungan				40	34	51	55

Ketiga strategi gagal memberikan solusi optimal

Kesimpulan:

Algoritma ini tidak selalu berhasil menemukan solusi optimal untuk masalah integer *knapsack*

B. *Clash Royale*

Permainan ini memiliki beberapa elemen penting yang perlu dijelaskan yaitu, kartu dan *field*:

1. Kartu

   Kartu pada permainan ini digunakan untuk menempatkan pasukan, mantra, ataupun bangunan kedalam *field*. Setiap kartu memiliki karakter tersendiri bergantung pada tipe kartu yang digunakan, contoh: semua kartu mantra bisa ditempatkan dimana saja sedangkan bangunan dan pasukan hanya dapat ditempatkan pada wilayah sendiri.

   Setiap kartu memiliki *level* yang menunjukkan kekuatan isi dari kartu tersebut. Semakin tinggi level kartu, semakin kuat kartu tersebut dan jika bertemu dengan kartu sama namun dengan *level* yang lebih rendah, kemungkinan kartu tersebut menang lebih tinggi dibandingkan jika dalam keadaan se-*level*.

   Kartu bisa didapatkan dengan membeli di toko pada permainan atau dari peti. Peti bisa didapat dari membeli di toko atau hasil memenangkan pertandingan melawan pemain lain. Kartu yang sama digabung menjadi satu untuk menaikkan level kartu tersebut, minimal kartu agar bisa naik level berbeda pada setiap levelnya.

   Setiap kartu memiliki biaya berupa *Elixir* untuk memanggil pasukan, mantra, atau bangunan yang ada pada kartu tersebut. *Elixir* tersebut akan dikurangi dari jumlah *Elixir* yang kita miliki saat pertandingan berlangsung. *Elixir* selalu bertambah 1 setiap detik dan pada satu menit terakhir, kecepatan akan bertambah 2 kali lipat.

   Pemain hanya dapat memiliki 10 maksimum *Elixir* pada pertandingan, jika sudah sampai angka 10, *Elixir* tidak akan bertambah lagi.

   Setiap kartu memiliki tingkat kejarangan masing-masing, terdapat tiga jenis tingkatan kejarangan yaitu, *common*, *epic*, *rare*, dan *legendary*. Tingkat kejarangan ini menunjukkan tingkat kesulitan untuk mendapatkan kartu tersebut dan tingkat kesulitan untuk menaikkan level dari kartu tersebut. Untuk menaikkan level, kita harus membayar dengan mata uang permainan ini, setiap level memiliki harga yang berbeda.

   Dalam pertandingan, kartu yang telah dipakai akan menghilang dan selanjutnya akan muncul kembali jika kartu lainnya sudah pernah keluar. Hal ini membuat kemungkinan dua buah kartu dengan isi yang sama untuk keluar dengan perbedaan waktu yg sedikit menjadi lebih kecil.

   Contoh:

   a. *Troop Card*

Gambar 2: Golem.
(http://vignette3.wikia.nocookie.net/clashroyale/images/d/d4/GolemCard.png/revision/latest/scale-to-width-down/218?cb=20160124213213)

Kartu yang akan mengeluarkan sebuah batu raksasa yang selalu mengincar bangunan. Jika hancur, akan membuat dua buah batu berukuran lebih kecil yang juga akan mengincar bangunan namun dengan kekuatan yang juga lebih kecil. Hanya dapat dipanggil pada wilayah sendiri

   b. *Spell Card*

Gambar 3: Rage.
(http://vignette1.wikia.nocookie.net/clashroyale/images/1/1c/RageCard.png/revision/latest/scale-to-width-down/218?cb=20160124213511)

Kartu yang membuat semua pasukan yang kita miliki dan berada pada daerah mantra untuk bergerak dan menyerang lebih cepat. Sangat berguna ketika sudah ada banyak pasukan yang dibuat dan berada dekat dengan menara musuh. Dapat dipanggil dimanasaja.

c. *Building Card*

Gambar 4: Inferno Tower.
(http://vignette2.wikia.nocookie.net/clashroyale/images/3/34/InfernoTowerCard.png/revision/latest/scale-to-width-down/218?cb=20160124213213)
Sebuah bangunan yang hanya dapat menyerang satu unit pasukan musuh dalam satu waktu. Walaupun begitu, serangan dari bangunan ini bertambah setiap waktunya sehingga sangat efektif untuk melawan unit pasukan dengan nyawa yang sangat banyak.

Masih banyak jenis dan tingkat dari kartu yang ada pada permainan ini. Setiap kartu memiliki penjelasan detil tentang isi dari kartu tersebut.

2. *Field*

Terdapat beberapa jenis *field* yang ada pada permainan ini. Perbedaan tersebut menunjukkan tingkat permainan dari para pemain di *field* tersebut. Namun, selain perbedaan warna dan lingkungan, tidak derdapat perbedaan yang mendasar dari *field-field* ini.

Dalam sebuah *field* terdapat dua buah wilayah, wilayah sendiri dan wilayah musuh. Wilayah ini merupakan daerah yang dibolehkan dalam penempatan kartu yang akan dipakai. Seperti yang telah disebutkan diatas, beberapa kartu hanya bisa digunakan pada wilayah sendiri. Kedua wilayah dipisahkan oleh sebuah sungai.

Pada masing-masing wilayah terdapat tiga buah menara, dua menara penjaga dan satu menara inti. menara penjaga akan selalu menyerang musuh yang berada pada jangkauan serangnya sedangkan menara inti hanya akan menyerang jika pernah terserang atau ada menara penjaga yang hancur.

Menara penjaga yang hancur berharga satu mahkota, sedangkan menara inti berharga dua mahkota. Namun, pemain hanya dapat mengumpulkan paling banyak tiga mahkota dalam satu pertandingan. Nyawa dari menara inti lebih banyak dari nyawa yang dimiliki oleh menara penjaga.

Semakin tinggi level pemain, semakin tinggi nyawa yang dimiliki oleh ketiga menara tersebut.

Gambar 5: BattleField.
(http://img.2pcdn.com/2016/clashroyale//2016/04/07/20160407195801881.jpg)

3. Cara Bermain

Setiap pemain memiliki *deck* serta menara masing-masing. *Deck* masing-masing pemain akan diacak sehingga muncul empat dari delapan kartu yang telah dipilih.

Diberikan tiga menit dalam satu pertandingan dengan satu menit terakhir kecepatan penambahan *Elixir* dan permainan secara umum berubah menjadi dua kali lipat. Dalam tiga menit ini, pemain berlomba mengumpulkan mahkota sebanyak-banyaknya dengan cara menghancurkan menara lawan. Pemain yang mengumpulkan mahkota terbanyak akan menjadi pemenang dan sebaliknya. Dalam pertandingan melawan pemain lain, jika setelah tiga menit kedua pemain mengumpulkan jumlah mahkota yang sama, waktu pertandingan akan ditambahkan selama satu menit. Jika setelah satu menit itu tetap

sama jumlah perolehan mahkota, pertandingan akan berakhir seri. Sedangkan jika pemain bertanding melawan teman (tidak mempengaruhi peringkat pemain) waktu tambahan yang diberikan adalah tiga menit sehingga kemungkinan untuk seri berkurang.

Pemain yang menang, mendapat tambahan poin untuk meningkatkan peringkatnya, begitu pula sebaliknya. Semakin tinggi peringkat seseorang semakin kuat lawan yang akan ditemukan.

Setiap kartu yang baru masuk kedalam papan permainan tidak dapat langsung ditempatkan ke *field* karena terdapat waktu untuk persiapan kartu tersebut.

Selain melawan pemain lain, pemain juga dapat berlatih melawan bot yang disediakan oleh pembuat permainan. Pertandingan melawan bot relatif sangat mudah jika dibandingkan dengan melawan pemain lain.

Dalam pertandingan, pemain bisa saling berkomunikasi satu dengan lainnya namun dengan cara yang terbatas, hanya dengan *emoticon* atau kata yang telah dipilihkan dan dibuat oleh pembuat permainan ini. Komunikasi dibatasi untuk mengurangi kemungkinan spam yang dilakukan oleh orang-orang yang tidak bertanggung jawab.

III. PENERAPAN ALGORITMA

Pada bab ini akan dijelaskan bagaimana penerapan algoritma *greedy* pada permainan *Clash Royale*. Dengan menerapkan algoritma ini, diharapkan pemain bisa memenangkan pertandingan baik melawan orang ataupun *bot* yang disediakan sistem permainan. Algoritma *greedy* dapat diterapkan pada dua tahap permainan ini yaitu, tahap pemilihan kartu serta pertandingan itu sendiri.

A. *Penerapan pada Pemilihan Kartu*

Setiap pemain memiliki kartu standar saat pertama kali memulai permainan ini. Kartu-kartu standar ini akan langsung masuk pada *deck* pertama dari pemain. Seiring dengan naiknya level permainan dari pemain, kartu yang didapat semakin banyak.

Dengan beragamnya jenis kartu yang dimiliki, kita dapat menggunakan algoritma *greedy* untuk mengambil kartu untuk dimasukkan kedalam *deck* untuk dibawa kedalam pertandingan.

Tujuan dari penggunaan algoritma ini adalah untuk mendapatkan kebutuhan *Elixir* tersedikit untuk permainan. Dengan memiliki sedikit kebutuhan *Elixir*, kartu akan lebih cepet berputar karena bisa dengan mudah dipanggil kedalam lapangan pertandingan.

Elemen:
C = Semua kartu yang dimiliki oleh pemain
S = 8 kartu terpilih dengan rata-rata terkecil
Seleksi = ambil kartu dengan kebutuhan elixir tersedikit
Kelayakan = -
Objektif = kartu dengan rata-rata terkecil

Gambar 6: Hasil Optimasi Pemilihan Kartu

Hasil penerapan algoritma *greedy* dalam pemilihan kartu untuk mendapatkan rata-rata harga *Elixir* yang dibutuhkan yaitu 2.3.

B. *Penerapan pada Pertandingan*

Inti dari permainan ini adalah untuk menghancurkan menara lawan sebanyak-banyaknya. Banyak cara yang bisa diterapkan untuk melakukan hal tersebut, salah satunya adalah dengan menggunakan algoritma *greedy*.

Algoritma *greedy* yang akan diterapkan memilih kartu dengan harga terkecil yang bisa dipilih untuk selanjutnya ditempatkan pada *field* yang ada.

Dengan menggunakan cara pilih tersebut diharapkan kartu bisa berotasi lebih sering sehingga kartu yang dapat keluar lebih sering dan semakin besar kemungkinan hancurnya menara musuh.

Elemen:
C = kartu yang ada di layar
S = pilihan kartu
Seleksi = ambil kartu dengan harga termurah
Kelayakan = harga kartu lebih kecil atau sama dengan jumlah *Elixir*
Objektif = menara musuh hancur sebanyak-banyaknya

Algoritma:

1. Periksa apakah ada kartu yang memiliki harga yg murah dan memenuhi jumlah *Elixir*. Jika tidak ada, tunggu.
2. Jika ada, letakkan kartu pada *field*.
3. Ulangi tahap 1.

Dengan algoritma tersebut ternyata tidak selalu membuahkan kemenangan terhadap pemain yang menggunakannya.

IV. ANALISIS

A. *Penerapan pada Pemilihan Kartu*

Dalam pemilihan kartu, algoritma *greedy* diterapkan untuk memperkecil rata-rata *Elixir* yang dibutuhkan pada pertandingan. Hal ini juga dilakukan untuk rotasi kartu yang lebih cepat sehingga pemain dapat mengeluarkan berbagai macam kartu dengan cepat.
Hasil dari optimasi yang dilakukan algoritma *greedy* yaitu 2.3 dan hasil pemilihan normal yaitu 4.1. Perbedaan yang sangat besar dan signifikan ini menunjukkan keberhasilan tujuan pertama dari penerapan algoritma *greedy* yaitu memperkecil rata-rata *Elixir*. Secara implisit, tujuan kedua juga berhasil dilakukan sehingga dapat dikatakan bahwa penerapan algoritma ini sangatlah berhasil.

B. *Penerapan pada Pertandingan*

Dengan tujuan untuk memenangkan pertandingan, algoritma ini ternyata terlalu sederhana. Masih banyak aspek lain yang perlu dipertimbangkan untuk menyelesaikan permasalahan yang cukup rumit seperti ini.
Itulah yang membuat tidak semua pertandingan yang dimainkan menggunakan algoritma ini selalu membuahkan kemenangan. Tingkat kerumitan permasalahan ini tidaklah setara dengan algoritma yang digunakan. Namun, dengan algoritma ini, kita dapat melihat dasar yang sangat baik yang nantinya bisa dikembangkan lebih jauh sehingga dapat diterapkan pada setiap pertandingan yang dilakukan.

V. KESIMPULAN

Algoritma *greedy* merupakan salah satu cara mendekati solusi yang diinginkan. Untuk masalah pertama, algoritma *greedy* dapat dengan mudah menyelesaikan masalah tersebut, yaitu menggunakan kartu dengan rata-rata *Elixir* termurah. Namun untuk masalah kedua, algoritma yang sederhana ini belum bisa memastikan kemenangan dari seorang pemain. Hal ini dikarenakan lebih kompleksnya masalah yang ada. Sebenarnya, algoritma ini sudah membuat dasar yang baik dalam menyelesaikan permasalahan tersebut. Dengan sedikit modifikasi, algoritma ini dapat menjadi pilihan yang baik untuk memastikan kemenangan pemain dalam permainan ini.

REFERENCES

[1] https://clashroyale.com/, waktu akses : 8 Mei 2016, pukul 12:00
[2] http://clashroyale.wikia.com/wiki/Cards, waktu akses : 8 Mei 2016, pukul 12.13
[3] https://en.wikipedia.org/wiki/Knapsack_problem, waktu akses : 8 Mei 2016, pukul 13:00
[4] Munir, Rinaldi. 2014. Algoritma Greedy (2014).ppt

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, 8 Mei 2015

Ahmad Fa'iq Rahman / 13514081