Implementasi Algoritma Greedy dalam Permainan *Dicey Elementalis*

Rahmat Rafid Akbar - 13520090
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail) : 13520090@std.stei.itb.ac.id

***Abstract*—Bermain game adalah salah satu hobi bagi seseorang. Terlepas dari perbedaan umur, game dapat dimainkan mulai dari anak-anak, remaja, hingga orang dewasa. Dengan bermain game, seseorang mendapatkan rasa senang ketika menyelesaikan suatu *quest* (misi) ataupun berhasil dalam mengalahkan lawan yang cukup kuat. Game dapat dimainkan di dalam maupun di luar jaringan. Ada game *online* yang membutuhkan jaringan untuk pengoperasiannya dan ada juga game *offline* yang kerap kali tidak membutuhkan jaringan untuk mulai bermain. Salah satu genre game yang cukup menantang dan banyak dimainkan oleh orang-orang saat ini adalah game bergenre strategi. Pemain ditantang untuk menemukan kombinasi stategi yang baik untuk menyelesaikan permainan. Pada makalah kali ini, akan dibahas pemanfaatan algorima Greedy pada game strategi *Dicey Elementalist*.**

Keywords— Game; Strategi; Gambling; Dice; Dicey Elementalist; Algoritma; Greedy;

I. Latar Belakang

Perkembangan teknologi telah membawa kehidupan manusia satu tingkat lebih baik dari sebelumnya. Teknologi yang dikembangkan menghasilkan perangkat-perangkat yang memudahkan manusia dalam suatu urusuan dan pekerjaan. Salah satu teknologi yang paling berguna saat ini adalah perangkat elektronik. Perangkat elektronik memungkinkan kita menembus batas dunia nyata dengan menggunakan jaringan internet. Manusia dapat berinteraksi satu sama lain tanpa memandang jarak dan waktu. Menyesuaikan dengan perkembangan teknologi yang makin mumpuni, banyak orang yang juga mengembangkan berbagai macam fitur-fitur yang ada di dunia nyata agar dapat diakses melalui perangkat elektronik seperti Handphone dan Komputer. Tak terlepas pula pada perkembangan aplikasi permainan di ponsel dan komputer.

Banyak sekali game yang menarik perhatian dewasa ini, hal ini dikarenakan mudahnya akses dari perangkat dan hanya membutuhkan jaringan. Game-game yang dirilis oleh pembuatnya dapat di*download* dari situsnya langsung atau melalui aplikasi pengelola app seperti playstore, appstore dan lainnya. Game yang disediakan juga memiliki genre masing-masing yang tentunya berbeda pada *gameplay* atau jalan permainannya. Game dengan genre strategi membutuhkan pertimbangan agar dapat mencapai tujuan dengan sebaik mungkin. Pemain diminta untuk menentukan pilihan yang tepat ketika membuat keputusan. Kesalahan langkah atau pengambilan keputusan dapat menyebabkan kekalahan dan *game over* yang tentu saja tidak dikehendaki dalam bermain game.

Penyusunan strategi dapat dilakukan secara penerkaan asal-asalan semata maupun secara terprogram dengan menggunakan algoritma yang telah dipelajari pada mata kuliah IF2211 Stategi Algoritma. Pada game kali ini, Dicey Elementalist yang merupakan game strategi, berpusat pada optimalisasi kerusakan (*damage*) pada setiap permainan baik itu *damage* yang diterima ataupun *damage* yang diberikan kepada lawan. Salah satu algoritma yang dapat digunakan untuk optimalisasi adalah algoritma Greedy. Algoritma yang mencari keuntungan terbesar pada suatu tahap (lokal) dengan harapan keuntungan tersebut merupakan keuntungan terbesar untuk keseluruhan masalah (global).

II. Teori Dasar

A. Algoritma Greedy

Algoritma Greedy adalah salah satu algoritma yang dipakai dalam persoalan optimisasi, karena banyak persoalan yang jika ditelusuri secara detail akan memakan banyak waktu dan usaha. Sehingga, algoritma ini digunakan dengan harapan bisa mendapat aproksimasi nilai yang sedekat mungkin dari solusi optimalnya.

Terdapat 2 kategori optimisasi yang biasanya dilakukan, yaitu :

1. Maksimalisasi suatu nilai (*value maximization*), dan
2. Minimalisasi suatu nilai (*value minimalization*).

Permasalahan yang dioptimisasi pun dapat beragam, mulai dari masalah biaya, penghasilan, waktu tunggu, kinerja, dan lainnya. Berikut adalah beberapa persoalan yang dapat diselesaikan menggunakan algoritma Greedy :

1. Persoalan penukaran uang (*coin exchange problem*)
2. Persoalan memilih aktivitas (*activity selection problem*)

3. Minimalisasi waktu kerja sistem CPU
4. Persoalan knapsack (*knapsack problem*)
5. Penjadwalan pekerjaan (*Job scheduling*)
6. Pohon merentang minimum (*minimum spanning tree*)
7. Lintasan terpendek pada graf (*shortest path on graph*)
8. Kode Huffman (*Huffman code*)
9. Pecahan mesir (*Egyptian fraction*)

Algoritma Greedy berjalan secara tahap demi tahap dan di setiap tahapnya akan dilakukan pememilihan keputusan yang paling optimal (pada tahap itu) tanpa memikirkan imbas kedepannya dan dengan harapan keputusan/ solusi tersebut akan mengarah dan menghasilkan hasil atau solusi akhir yang optimal pula untuk keseluruhan masalah.

Pemilihan solusi optimum lokal ini belum tentu mengarah kepada solusi optimum global karena bisa saja pemilihan keputusan pada suatu tahap malah memutus kemungkinan rute pencarian untuk mencapai solusi optimum global. Jadi algoritma ini tidak dapat memastikan bahwa solusi akhir yang dihasilkan adalah solusi paling optimal.

Namun, algoritma Greedy banyak dipakai karena langkah-langkahnya yang mudah dipahami dan mudah digunakan. Analisis efisiensi waktunya pun lebih mudah dibandingkan dengan algoritma lain, seperti algoritma *Dynamic Programming* yang terbilang cukup kompleks.

Dalam implementasi Algoritma greedy, terdapat beberapa elemen yang harus ditentukan agar memudahkan proses pencarian. Elemen-elemen tersebut adalah :

1. Himpunan kandidat (C)

   Himpunan kandidat berisi kandidat yang akan dipilih pada setiap tahap.

2. Himpunan solusi (S)

   Himpunan solusi berisi kandidat yang sudah dipilih. Himpunan ini merupakan subset dari himpunan kandidat, $S \in C$.

3. Fungsi solusi (*solution function*)

   Fungsi solusi digunakan untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.

4. Fungsi seleksi (*selection function*)

   Fungsi seleksi digunakan untuk memilih kandidat berdasarkan strategi greedy tertentu. Strategi ini bersifat *heuristic*.

5. Fungsi kelayakan (*feasibility function*)

   Fungsi kelayakan berfungsi untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi.

6. Fungsi objektif

   Fungsi objektif berisi tujuan dari algoritma tersebut, memaksimumkan atau meminimumkan. Fungsi ini juga mengecek apakah kandidat yang dipilih sudah mengarah kepada tujuan atau belum.

Berikut adalah gambaran implementasi algoritma Greedy secara umum :

Skema umum algoritma *greedy*:

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← {}      { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
      C ← C – {x}        { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
          S ← S ∪ {x}     { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then    { solusi sudah lengkap }
    return S
  else
    write('tidak ada solusi')
  endif
```

- Pada akhir setiap lelaran (iterasi), solusi yang terbentuk adalah optimum lokal.
- Pada akhir kalang **while-do** diperoleh optimum global (jika ada).

Gambar 2.1. Gambaran Umum Implementasi Algoritma Greedy

B. *Game : Dicey Elementalist*

Game *Dicey Elementalist* adalah salah satu game bergenre strategi dengan campuran genre *gambling* didalamnya. Game ini bertujuan untuk mengalahkan monster yang ada di dalam sebuah dungeon 4 lantai (*floor*). Di setiap lantainya akan terdapat 10 buah ruangan (*room*).

Pada setiap ruangan, akan ditemukan seekor monster dan pemain harus mengalahkan monster tersebut untuk mencapai ruangan selanjutnya. Monster yang ditemui pada sebuah ruangan dapat berpangkat normal, elite, maupun boss. Namun, monster berpangkat boss hanya dan pasti muncul pada ruangan terakhir (ruangan ke-10) yang merupakan penjaga lantai sebelum turun ke lantai selanjutnya.

➢ *Another Entity in The Rooms*

Terkadang, di sebuah ruangan pemain tidak menemukan monster dan dapat menemukan entitas lain yang dapat mempengaruhi keberjalanan permainan. Entitas lain yang dapat ditemui selain monster pada sebuah ruangan dapat berupa dari salah satu entitas berikut :

1. *Sprite Spring (Blessing)*

   Akan terdapat sebuah batu kristal yang akan memberikan 2 pilihan berkat. Berkat ini dapat berupa uang, penambahan Hp, restorasi Hp atau efek buff. Pemain dapat memilih 1 dari 2 pilihan yang disediakan. Apabila tidak memilih salah satunya, pemain dapat mengambil uang tebusan sebagai gantinya.

2. *Book of Choice (Destiny)*

   Akan terdapat sebuah buku yang akan memberikan 2 pilihan takdir, setiap takdir akan memberikan efek yang berbeda terkadang sabuah takdir memberikan efek buff sekaligus debuff atau hanya salah satunya. Pemain dapat

memilih 1 dari 2 pilihan yang disediakan. Apabila tidak memilih salah satunya, pemain dapat mengambil uang tebusan sebagai gantinya.

3. *Chest Box (Gift)*

Akan terdapat sebuah peti harta yang akan memberikan 2 pilihan artefak atau 2 pilihan kartu spell. Pemain dapat memilih 1 dari 2 pilihan yang disediakan. Apabila tidak memilih salah satunya, pemain dapat mengambil uang tebusan sebagai gantinya.

4. *Goblin Merchant (Trade)*

Akan terdapat seorang pedagang dan pemain dapat menjual artefak serta membeli efek untuk pertarungan.

5. *Crow Seller (Shop)*

Akan terdapat seorang penjual dan pemain dapat membeli kartu maupun artefak yang ditawarkan oleh penjual.

6. *Demon Blacksmith (Upgrade)*

Akan terdapat seorang pandai besi dan pemain dapat meningkatkan efek dari kartu spell yang dimiliki. Pemain juga dapat menjual kartu spell dari dek.

7. *Poker Demon (Gambler)*

Pemain akan ditawarkan untuk bermain poker dadu, menang maupun kalah, pemain akan mendapatkan sebuah efek. Pemain dapat menolak untuk ikut bermain poker.

8. *Freedom Demon (Gambler)*

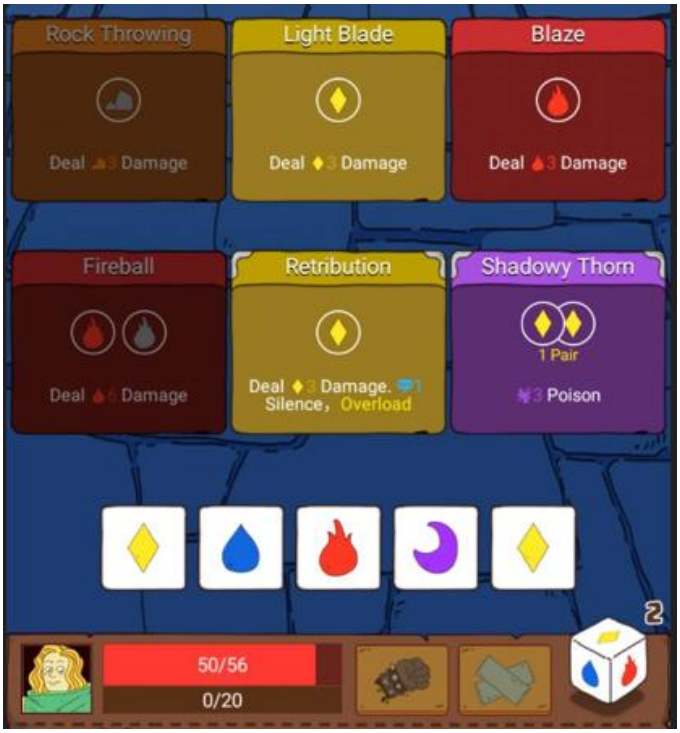
Pemain akan ditawarkan untuk bermain lempar dadu, apabila ada kombinasi dadu yang cocok dengan salah satu dari 6 penawaran dealer maka pemain dapat membawa pulang hadiah tersebut. Pemain dapat membawa maksimal seluruh hadiah jika memenuhi.

- ***Gameplay***

Pertempuran dengan monster dilakukan dengan pelemparan dadu-dadu elemen dan pemakaian *spell card*. Setiap *spell card* memiliki *requierments element* yang harus dipenuhi agar dapat dipakai. Elemen yang terdapat pada permainan ini ada 6, yaitu *fire*, *water*, *earth*, *wind*, *light* dan *dark*.

Jika seandainya *requirment* dari *spell card* belum terpenuhi, pemain dapat melempar ulang dadu-dadu elemen. Pelemparan ulang (*re-roll*) dapat dilakukan 2x secara default (dapat bertambah karena efek). Jika semisal ada salah satu dadu elemen yang dirasa cocok, pemain dapat menahannya (*hold*) sehingga pada *re-roll* selanjutanya tidak akan berganti. Jumlah dadu yang dapat dimainkan secara default adalah 5 buah, namun dapat ditambah dengan efek dari artefak atau buff.

Pemain memiliki atribut *Hp-bar* dan *furry-bar*. Apabila *Hp-bar* mencapai 0 maka pemain akan mati. *Furry-bar* mengindikasikan penggunaan skill, pemain secara default memiliki 2 skill aktif dan memiliki *requirment* jumlah *furry* untuk memakai skill tersebut. Pemain dapat menghasilkan *armor*, hal ini ditandai dengan *Hp-bar* berwarna biru dan jumlah armor di sisi kiri *Hp-bar*.



Gambar 2.2. Tampilan Dadu Elemen, Hand Card, dan Status Pemain

Pada awal permainan, pemain hanya memiliki 3 buah *spell card* dan tidak memiliki satupun artefak. Kartu dan artefak bisa didapatkan setelah mengalahkan seekor monster, dari sebuah entitas lain, ataupun ketika naik level. Jumlah kartu maksimal yang dapat dibawa ke dalam pertarungan (kartu di *hand*) adalah 6, namun pemain dapat memiliki lebih dari 6 kartu yang sisanya disimpan di *deck*. Pemain juga hanya dapat memakai maksimal 10 buah artefak ke dalam pertarungan.

Terdapat 7 kartu berdasarkan tipe elemennya, kartu tersebut dapat berupa salah satu dari 6 elemen di atas atau merupakan kartu biasa (tanpa elemen). Perbedaan tipe elemen kartu nantinya akan berpengaruh pada kombinasi kerusakan dari serangan yang diterima oleh lawan. Bergantung pada buff atau debuff yang diberikan, serangan akan semakin kuat.

- ***Buff dan Debuff***

Terdapat beberapa efek buff (peningkatan) dan debuff (penurunan) yang dapat diberikan kepada lawan atau pun diterima oleh pemain, diantaranya adalah :

1. Dodge (*buff*)

Buff ini berguna untuk menghindari serangan selanjutnya dari lawan.

2. Fury (*buff*)

Buff ini berguna untuk memakai sebuah kartu dua kali.

3. Reduce by (*buff*)

Buff ini berguna untuk mengurangi serangan dari lawan selama 1x pertarungan (saat melawan 1 monster).

4. Pierce (*buff*)

Buff ini berguna untuk memberikan *true damage* pada lawan terlepas dari efek buff maupun armor yang dipakainya.

5. Freeze (*debuff*)

DeBuff ini menyebabkan 1 buah dice lawan tidak dapat di *re-roll*. Freeze dapat ditumpuk hingga lebih dari 5 buah. Setelah membekukan dice lawan pada giliran (*turn*) tersebut, banyak efek Freeze akan berkurang sebanyak pemakaian. Apabila terdapat 6 Freeze maka 5 buah dice lawan akan dibekukan dan 1 Freeze akan tersisa untuk giliran selanjutnya.

6. Burn (*debuff*)

DeBuff ini menyebabkan 1 buah *spell card* lawan terkena efek terbakar (*burn*). Apabila lawan memakai *spell card* dengan efek *burn*, maka lawan akan menerima *4 normal damage*. Efek *burn* dapat ditumpuk hingga lebih dari 6 buah. Setelah membakar *spell card* lawan pada giliran (*turn*) tersebut, banyak efek *burn* akan berkurang sebanyak pemakaian. Apabila terdapat 7 *burn* maka 6 buah *spell card* lawan akan terbakar dan 1 *burn* akan tersisa untuk giliran selanjutnya.

7. Shocked (*debuff*)

DeBuff ini menyebabkan 1 buah artefak lawan tidak dapat digunakan. DeBuff ini akan hilang di akhir *turn*.

8. Weaken (*debuff*)

DeBuff ini menyebabkan lawan menjadi lemah, sehingga serangan yang dihasilkan akan dikurangi sebesar 40%. DeBuff ini dapat ditumpuk dan disetiap akhir *turn* akan dikurangi 1 buah.

9. Silence (*debuff*)

DeBuff ini menyebabkan 1 buah *spell card* lawan tidak dapat dipakai di *turn* selanjutnya. Efek *silence* dapat ditumpuk hingga lebih dari 6 buah. Setelah memblokir pemakaian *spell card* lawan pada giliran (*turn*) tersebut, banyak efek *silence* akan berkurang sebanyak pemakaian. Apabila terdapat 7 *silence* maka 6 buah *spell card* lawan akan terblokir dan 1 *silence* akan tersisa untuk giliran selanjutnya.

10. Poison (*debuff*)

Debuff ini akan memberikan *poison damage* sebanyak efek yang tertumpuk (*staked*) pada saat awal giliran lawan. Setelah giliran lawan berakhir, maka efeknya akan berkurang satu.

11. Locked (*debuff*)

DeBuff ini menyebabkan 1 buah dice lawan tidak dapat dipakai. Locked dapat ditumpuk hingga lebih dari 5 buah. Setelah mengunci dice lawan pada giliran (*turn*) tersebut, banyak efek Locked akan berkurang sebanyak pemakaian. Apabila terdapat 6 Locked maka 5 buah dice lawan akan dikunci (tidak bisa dipakai untuk memenuhi *requierment spell card*) dan 1 Locked akan tersisa untuk giliran selanjutnya.

12. Cursed (*debuff*)

DeBuff ini menyebabkan salah satu *spell card* yang ada di *hand* tidak menyebabkan apapun, namun kartu tetap terhitung dipakai saat *turn* tersebut. Efek *cursed* dapat ditumpuk. Setelah mengutuk 1 *spell card* lawan, efek ini akan berkurang satu.

13. Entangled (*debuff*)

DeBuff ini menyebabkan 1 buah *spell card* lawan tersegel sehingga memiliki 2x *requierment* untuk dapat dipakai. *Requirment* pertama untuk membuka segel kartu dan yang kedua untuk memakai kartu seperti biasa. Efek *entangled* dapat ditumpuk hingga lebih dari 6 buah. Setelah menyegel *spell card* lawan pada giliran (*turn*) tersebut, banyak efek *entangled* akan berkurang sebanyak pemakaian. Apabila terdapat 7 *entangled* maka 6 buah *spell card* lawan akan tersegel dan 1 *entangled* akan tersisa untuk giliran selanjutnya.

14. Vulnerable (debuff)

DeBuff ini menyebabkan serangan lawan menjadi lebih kuat, sehingga serangan yang diterima akan ditambah sebesar 40%. DeBuff ini dapat ditumpuk dan disetiap akhir *turn* akan dikurangi 1 buah.

- ***Artifacts***

Selain dari pemakaian buff dan efek kartu. Pemain dapat menggunakan artefak-artefak untuk meningkatkan kekuatan dan pertahanan. Artefak-artefak ini bisa didapat dari peti harta, setelah mengalahkan monster, ataupun dari peningkatan level.

- ***Levelling***

Pemain pada awal permainan berada pada level 1. Setiap mengalahkan monster, pemain akan menerima exp. Apabila monster yang dikalahkan adalah monster tipe normal, maka akan mendapatkan 1 exp. Apabila monsternya bertipe elite atau bos, exp yang didapatkan akan menyesuaikan dengan batasan exp untuk naik level.

III. IMPLEMENTASI DAN ANALISIS ALGORITMA GREEDY PADA PERMAINAN DICEY ELEMENTALIST

Sebelum menerapkan algoritma Greedy, kita harus menentukan elemen-elemen yang akan digunakan dalam pembuatan algoritma Greedy. Beberapa elemen-elemen Greedy yang dipakai pada permainan ini adalah :

1. Himpunan kandidat (C)

   Himpunan kandidat direpresentasikan oleh semua kemungkinan pemilihan kartu, artefak, urutan pemakaian kartu, pemberian efek *buff* dan *debuff*.

2. Himpunan solusi (S)

   Himpunan solusi direpresentasikan oleh pilihan yang telah dilakukan oleh pemain.

3. Fungsi solusi (*solution function*)

   Fungsi solusi direpresentakan dari perhitungan total *damage* yang dilakukan.

4. Fungsi seleksi (*selection function*)

   Fungsi seleksi yang digunakan dapat beragam karena fungsi ini tergolong *heuristic*. Bisa untuk memperbanyak damage yang dihasilkan atau untuk menigkatkan durability dengan menambah health dan armor.

5. Fungsi kelayakan (*feasibility function*)

   Fungsi kelayakan yang digunakan adalah apakah pemakaian sebuah kartu akan menyebabkan health berkurang banyak dan bisa menyebabkan kematian (kalah).

6. Fungsi objektif

   Fungsi objektif yang digunakan adalah minimalisasi *turn* agar damage yang diberikan musuh sedikit, dengan kata lain maksimalisasi pemberian damage sehingga pertempuran cepat selesai.

A. Analisis Algoritma Greedy

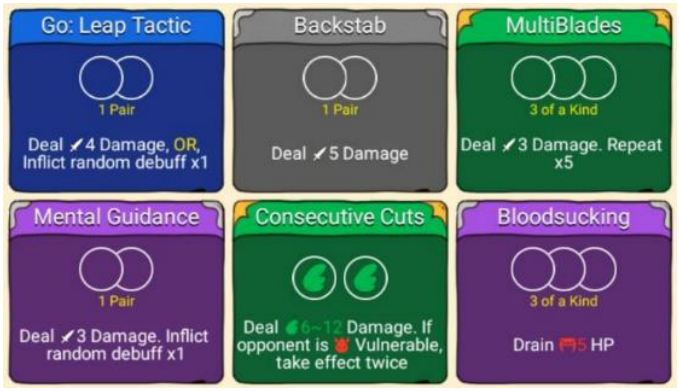
Berdasarkan fungsi objektif yang telah ditetapkan sebelumnya. Fungsi seleksi yang "dirasa" paling tepat digunakan adalah untuk menggunakan kartu dengan *damage* terbesar. Agar memberikan damage yang lebih besar, sebaiknya dicari terlebih dahulu kartu yang dapat memberikan efek buff peningkat serangan *Vunerable*. Namun, pemilihan kartu juga harus diperhatikan. Kartu yang dipilih haruslah memiliki *requierment* yang mudah didapat pada pelemparan dadu. Beberapa tipe requiement kartu jika diurutkan dari yang paling optipmal :

1. A pair no elemen
2. 1 kind no elemen
3. 1 kind with elemen
4. A pair with elemen
5. 2 pairs no elemen
6. 2 pairs with elemen
7. 3 kinds no elemen
8. 3 kinds with elemen
9. 4 kinds no elemen
10. 4 kinds with elemen

Kartu dengan *requirement* tanpa elemen lebih baik digunakan karena tidak terikat dengan sebuah elemen tertentu. Kartu dengan *requierment* sepasang dadu tanpa elemen berada pada posisi pertama karena untuk mendapatkannya cukup dengan meng-*hold* sebuah dadu dan melakukan *re-roll*. Selain itu probabilitas untuk mendapatkan dadu dengan elemen sepasang saat awal pertarungan cukup tinggi, karena ada 5 dadu dan hanya ada 6 elemen. Jika diasumsikan 4 dadu pertama semua elemennya berbeda, maka tinggal 2 elemen yang tidak termasuk. Pada dadu ke-5 terdapat 6 jenis kemungkinan dadu (ruang sampel) dengan 4 jenis dadu yang diinginkan (kejadian). Jadi probabilitasnnya adalah 4 dari 6 atau sebesar 66,67%.

Pemilihan susunan *hand card* sebaiknya dengan tipe *requierment* yang sama agar optimal, sehingga pemain hanya perlu mencari 1 kombinasi untuk memakai kumpulan kartu dengan tipe syarat pemakaian yang sama. Selain itu, juga sebaiknya diikuti dengan syarat tambahan 1 dadu. Contohnya terdapat kumpulan kartu dengan syarat pemakaian *A pair no element*. Pemain dapat mengambil kartu lain yang merupakan *3 kinds no element*, sehingga pemain cukup mencari kombinasi 1 dadu lagi yang cocok dengan kombinasi yang cocok dengan dadu A pair sebelumnya.

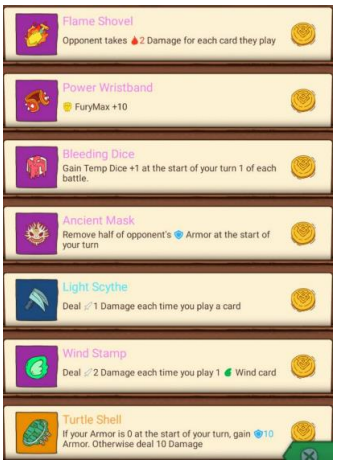
Berikut salah satu contoh susunan kartu pada tangan (*hand card*) yang menghasilkan damage lebih baik saat ini (pemilihan kartu telah dilakukan sebelumnya) :

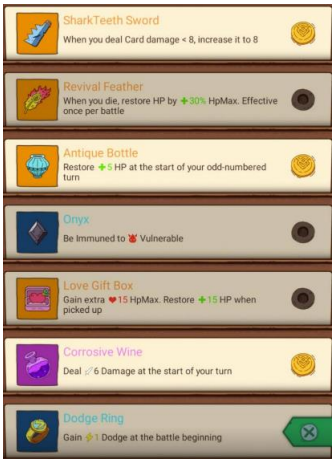


Gambar 3.1. Tampilan Hand Card dengan Requirments A Pair No Element

Pada susunan kartu diatas, terdapat 3 kartu dengan syarat pemakaian *A pair no element*. Selain itu, terdapat juga 1 kartu *a pair with* ***wind*** *element* dan 2 kartu *3 kinds no element*. Sehingga, untuk **optimal pada setiap langkah**, pemain cukup mencari kombinasi dadu dengan *3 kinds of* ***wind*** *element*. Sekurang-kurangnya pemain, dapat memakai 3 kartu dengan syarat *a pair no element* apabila setelah *re-roll* tetap tidak ditemukan kombinasi dadu yang optimal.

Pemilihan pemakaian artefak juga sama, memprioritaskan pemberian efek damage tambahan diatas efek lain seperti buff pelemah dan restorasi *health* maupun *damage*. Berikut salah satu strategi pemilihan artefak yang sesuai dengan algoritma Greedy yang diterapkan :



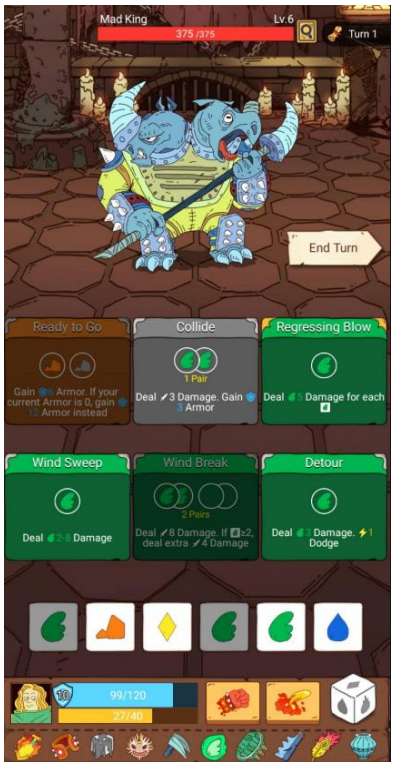


Gambar 3.2. Tampilan Artefak yang Dimiliki dan Artefak yang Sedang Dipakai Pemain

Terlihat bahwa 5 artefak yang tidak dipakai ke dalam pertempuran hanya memberikan efek peningkatan *endurance* dan *vitality*. Hal ini tentu berkebalikan dengan strategi Greedy kita yang membutuhkan total *damage* yang diberikan paling maksimal. Hampir keseluruhan artefak yang dibawa ke dalam pertempuran (pada contoh ini) memberikan tambahan kerusakan pada lawan, salah satunya adalah artefak *SharkTeeth Sword* yang memberikan efek 8 normal *damage* apabila serangan yang dihasilkan kartu lebih kecil dari 8.

- ***Greedy dalam pemakaian kartu***

Berikut adalah salah satu pertarunan yang akan dianalisis:



Gambar 3.3. Tampilan Pertarungan dengan Dino King ke-1

Terlihat bahwa kartu yang dimiliki hampir keseluruhan memiliki syarat pemakaian elemen bertipe *wind*. Selain itu, ada juga 1 kartu tipe *a pair no element*, 1 kartu tipe *a pair earth element* dan 1 kartu tipe *2 pair no element*. Pemain juga memiliki *Furry-bar* sebanyak 27 buah sehingga dapat menggunakan skil *vunerable* (skil 1) kepada lawan agar damage yang diberikan +40% dan juga memberikan 6 normal *damage*. Skil ini memakai 20 fury sehingga *Furry-bar* menjadi 17 dan tidak bisa memakai skil 2 (diperlukan 20 furry) yang berguna menggandakan 1 buah serangan. Namun, sesuai dengan strategi Greedy, untuk memaksimalkan kerusakan yang diberikan kepada lawan, pemain sebaiknya menambahkan kerusakan untuk setiap kartu yang dimainkan.

Pada pertarungan ini, pemain juga mendapatkan efek kerusakan tambahan pada lawan akibat artefak yang dipakai, yaitu : Ligth Scyte, Wind Stamp, Turtle Shell (semi attack), SharkTeeth Sword, dan Corrosive Wind.

Pertama-tama, pemain dapat memakai semua kartu yang memenuhi syarat pemakaian yaitu : Wind Sweep, Collide, Regressing Blow, dan Detour.

Total damage yang diberikan adalah:

- Basic Damage = 6 + [max(3,8) + max(2,8) + max(3*5,8) + max(3,8)]*140%

  Basic Damage = 6 + (8 + 8 + 15 + 8)*1.4 = 60.6
- Damage tambahan artefak = 4*1 + 3*2 + 10 + 6 + 10 = 36
- Total Damage = $\lceil \text{Basic} + \text{Artefak} \rceil$
- Total Damage = $\lceil 54{,}6 + 39 \rceil = 97$

Sehingga healt dari *Dino King* setelah *turn* selesai adalah $375 - 97 = 278$.

Tentu ini adalah solusi optimum lokal pada giliran ini (turn 1). Bila seandainya artefak yang digunakan tidak memprioritaskan *damage* tambahan maka tentu *damage* yang dihasilkan akan kurang dari 97. Begitu pula dengan pemakaian skil *venurable* dan pemilihan dadu yang di-*hold*. Jika semisal dadu yang di-*hold* adalah *earth dice* tentu *damage* yang dihasilkan pada saat itu akan berkisar antara 52–70 saja.

IV. KESIMPULAN

Penyelesaian game strategi dapat dilakukan secara gamblang (tebak-tebakan) atau secara terstruktur. Salah satu algoritma yang dapat digunakan adalah algoritma Greedy. Algoritma ini pada dasarnya telah diimplementasikan secara heuristik dalam kehidupan sehari-hari seperti kata slogan "take what you can get now!". Pengambilan keputusan yang terbaik saat itu juga tanpa melihat apakah keputusan itu baik secara keseluruhan.

Pada permainan *Dicey Elementalist* ini kita menggunakan pendekatan Greedy dengan mengambil total damage terbesar sebagai keputusan pada setiap tahap (*turn*) dibandingkan dengan mengambil total peningkatan *endurance* dan *vitality*. Hal ini agar pertarungan yang dilakukan dapat lebih singkat dan *damage* yang diterima pun dapat diminimalisasi.

Alternatif ke-2 yang mungkin adalah mengambil total peningkatan *endurance* dan *vitality* terbesar pada setiap turn. Namun, hal ini dirasa kurang cocok dengan tujuan awal atau fungsi objektif yang telah ditetapkan pada awal analisis.

Berdasarkan hasil analisis dan eksperimen, dapat dipastikan bahwa alternatif algoritma Greedy pertama merupakan salah satu algoritma terbaik walaupun belum dapat dipastikan apakah ada algortima lain yang lebih baik.

V. Ucapan Terima Kasih

Penulis ingin mengucapkan terima kasih sebesar-besarnya kepada beberapa pihak yang telah berperan besar dalam penyelesaian makalah ini :

1. Tuhan Yang Maha Esa karena berkat dan rahmat-Nya, makalah ini dapat terselesaikan dengan baik tanpa kurang satu bagian pun.
2. Dr. Ir. Rinaldi, M.T. sebagai dosen kelas K3 karena telah membimbing penulis selama pembelajaran Strategi Algoritma. Makalah IF2211 Strategi Algoritma – Sem. 2 Tahun 2021/2022
3. Seluruh tim dosen strategi algoritma yang telah berperan juga dalam kegiatan perkuliahan secara daring maupun luring.
4. Teman-teman seangkatan karena telah memberikan dukungan selama perkuliahan sehingga penulis dapat menulis makalah ini dengan baik.

Video Link at G-Drive

Video analisis terhadap implementasi algoritma Greedy yang dibahas pada makalah ini dapat dilihat pada link berikut :

https://drive.google.com/drive/folders/1n2iz8Ynr8deCkzL5Ah7eRl-6vFe_9eIH?usp=sharing

VI. References

[1] Dicey Elementalist Tricks, Hints, Q&A, Codes - HintsTricks.com (Diakses pada 23 Mei 2022)

[2] Status Effects | Dicey Dungeons Wiki | Fandom (Diakses pada 23 Mei 2022)

[3] Dicey Elementalist: Walkthrough Guide (iOS & Android) | AppsMeNow! (Diakses pada 23 Mei 2022)

[4] https ://www.elprocus.com/cryptography-and-its-concepts/ (Diakses pada 12 Desember 2021)

[5] Rinaldi Munir, Diktat Kuliah IF2120 : Algoritma Greedy Bag 1, Bandung : Program Studi Teknik Informatika Sekolah teknik Elektro dan informatika Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf (Diakses pada 23 Mei 2022)

VII. 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, 13 Mei 2022

Rahmat Rafid Akbar−13520090