Penggunaan Algoritma Brute Force dalam Memadukan (*Fusion*) Persona pada Persona 3 FES

Fitra Rahmamuliani 13513095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513095@std.stei.itb.ac.id

***Abstrak*— *Role-Playing Game (RPG)* adalah sebuah kategori permainan di mana pemain mengasumsikan dirinya sebagai pemeran utama dalam permainan dan berkolaborasi untuk membangun sebuah cerita bersama. Kategori RPG ini merupakan salah satu kategori permainan yang sering dibuat oleh para pengembang permainan di Jepang. RPG sering kali dimainkan oleh kalangan remaja hingga dewasa. Hal ini dikarenakan RPG memiliki sebuah drama berbahasa Inggris atau Jepang. Apabila kita tidak mengerti drama yang dimaksudkan, keberjalanan permainannya ke depan akan menjadi lebih sulit. Salah satu permainan berkategori RPG ini adalah Persona 3 FES. Pada Persona 3 FES, pemain dapat memadukan antara satu Persona dengan Persona lain menjadi Persona baru yang lebih baik. Oleh karena itu, diperlukan algoritma Brute Force untuk melihat satu-satu hasil dari perpaduan dua Persona. Makalah ini menjelaskan tentang penggunaan Brute Force itu sendiri dalam memadukan Persona pada Persona 3 FES.**

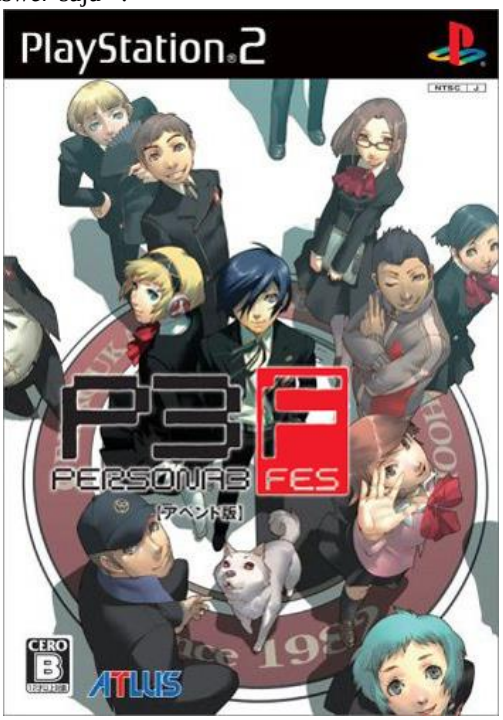
***Kata Kunci*—RPG, Role-Playing, *Game*, Persona, Persona 3 FES, *fusion*.**

I. PENDAHULUAN

Seiring berjalannya waktu, teknologi pun semakin berkembang. Oleh karena cepatnya perkembangan teknologi tersebut, semua alat di muka bumi ini mengarah ke digital, serba otomatis, dan serba cepat. Hingga teknologi berupa permainan pun terus dikembangkan dan semakin bervariasi hingga saat ini. Kategori RPG itu sendiri baru ada ketika sudah ditemukan teknologi untuk bermain secara virtual. Mungkin, masih banyak yang bingung apa maksud dari RPG. RPG atau *Role-Playing Game* adalah sebuah kategori permainan di mana pemain mengasumsikan dirinya sebagai pemeran utama dalam permainan dan berkolaborasi untuk membangun sebuah cerita bersama. Kategori RPG merupakan salah satu kategori permainan yang sering dibuat oleh para pengembang permainan di Jepang. RPG sering kali dimainkan oleh kalangan remaja hingga dewasa. Hal ini dikarenakan RPG memiliki sebuah drama dan alur cerita dalam bahasa Inggris atau Jepang. Apabila kita tidak mengerti drama yang dimaksudkan, keberjalanan permainannya ke depan akan menjadi lebih sulit.

Persona 3 FES adalah salah satu permainan berkategori RPG yang dikembangkan dan rilis ulang dari Persona 3 untuk *PlayStation* 2. Kata "FES" merupakan potongan dari kata "Festival". Oleh karena perkembangan itulah, Persona 3 FES dapat dikatakan lebih lengkap dibandingkan Persona 3. Persona itu sendiri merupakan sebuah seri dari permainan berkategori RPG yang dikembangkan dan dipublikasikan oleh Atlus. Seri ini adalah perubahan dari seri Megami Tensei yang berfokus pada memanggil dan mendatangkan roh jahat. Namun, seri Persona dipusatkan pada individu yang memiliki kemampuan untuk memanggil aspek dari jiwa mereka, yang dikenal sebagai Persona, menjadi ada.

Kembali ke Persona 3 FES, permainan ini dirilis di Jepang pada tanggal 19 April 2007. Permainan ini berisi sekitar 30 jam tambahan *gameplay* dibandingkan dengan Persona 3. Kebanyakan 30 jam tambahan ini berasal dari bentuk epilog yang bernama "The Answer" yang juga dikenal sebagai episode Aegis pada versi Jepang. Pada versi berbahasa Inggris, permainan ini berisi *The Journey* dan *The Answer*, sedangkan versi Jepang hanya memiliki *The Answer* saja[1].



Gambar 1 *Cover* depan Persona 3 FES

Sumber : http://ugrgaming.com/wp-content/uploads/2012/04/p3fes_basic_full.jpg (diakses pada 2 Mei 2015)

The Journey, juga dikenal sebagai "Episode Yourself" dalam versi Jepang, adalah versi yang disempurnakan dari Persona 3. Perubahan yang dilakukan adalah peningkatan jumlah Persona dalam permainan, video rahasia dari teman asrama protagonis telah ditambahkan, Koromaru dapat diambil di jalanan, beberapa *Social Link* telah dimodifikasi sedikit seperti Tanaka menjadi tersedia pada waktu yang berbeda, beberapa *quest* baru telah ditambahkan, kostum baru bisa dipakai dalam pertempuran, acara baru yang melibatkan Chidori Yoshino, Nagaki *Shine* telah dirombak total, dan *hard mode*.

The Answer, juga dikenal sebagai "Episode Aegis" dalam versi Jepang, adalah sebuah epilog dari Persona 3, yang berlanjut dari akhir The Journey. Bagian ini hanya tersedia dalam satu level mode, yang seharusnya setara dengan *hard mode* Persona 3. Game sekitar 30 jam lamanya memungkinkan pemain untuk mengontrol Aigis. Cerita mengikuti SEES (nama tim) pada petualangan baru yang melibatkan penjara baru, Abyss of Time, yang telah mengunci SEES dalam lingkaran waktu yang tak terbatas dalam asrama. Pada bagian ini juga memperkenalkan karakter baru yang dapat dimainkan bernama Metis.

Seperti pada permainan dengan kategori RPG pada umumnya, permainan terbagi menjadi dua mode yaitu mode menjelajah dan mode pertarungan. Perbedaan antara permainan RPG biasa dengan Persona adalah pada permainan RPG biasanya, pada mode penjelajah pemain dapat berkeliling ke kota-kota yang ada. Sedangkan person berkeliling pada asrama, sekolah, di dalam kota, *shrine*, *mall*, *port station*, *shopping district*, tartarus (sarang musuh), bukan berpindah antara satu kota ke kota lainnya.

Pada mode menjelajah, pemain akan diberikan HP dan SP. Sedangkan Persona memiliki *strength, Magic, agility, endurance*, dan *luck*. Saat berada di sekolah, terkadang pemain diminta *input*, misalkan guru bertanya saat di dalam kelas, dan ketika menjawab soal ujian.

Pada mode pertarungan, *turn base* berdasarkan *agility* yang dimiliki oleh Persona. Jika *main character* kita memiliki banyak Persona, *turn* berdasarkan Persona yang sedang kita gunakan. Sedangkan teman-temannya hanya memiliki satu Persona. Musuh juga mempunyai *strength, magic, agility, endurance, luck, HP,* dan *SP*. Akan tetapi, yang diperlihatkan kepada pemain hanyalah HP. Sisanya tidak dilihatkan kepada pemain, namun dipergunakan untuk perhitungan *damage*.

Serangan sendiri memiliki beberapa tipe, yaitu *physical*, *magic, almighty,*dan *instant death*. Pada *physical*, pemain dapat memilih untuk *slash, pierce,* atau *strike* ketika menyerang musuh. Serangan ini sangat ditentukan oleh *strength* dan *endurance* musuh. Pada *magic*, pemain dapat memilih untuk menggunakan elemen api, es, listrik, atau angin. Serangan ini sangat ditentukan oleh *magic* dan *endurance*. Pada *almighty,* lawan tidak bisa melakukan *null, reflect, absorb,* ataupun *strength*, kecuali jika lawan menggunakan kemampuan khusus untuk membuat *almighty* menjadi *null*. Pada *instant death,* pemain dapat memilih untuk menggunakan *ligth* atau *dark.* Jika serangan ini berhasil, lawan akan langsung mati. Akan tetapi jika serangan ini gagal, maka tidak akan mempengaruhi apapun dikarenakan terdapat *luck* penyerang dan *luck* yang diserang. Selain dari *luck,* setiap kemampuan memiliki kemungkinan *damage* tersendiri berapa persennya.

Dalam tampilan layar dalam mode pertarungan, kita dapat melihat banyak sekali lambang-lambang yang ada. Dari kiri putaran melawan arah jarum jam pada gambar, terdapat lambang *skill, attack, wait, run, change Persona, tatic,* dan *item.* Dari namanya sudah jelas bahwa *skill* adalah kumpulan kemampuan yang dimiliki oleh Persona tersebut, *attack* merupakan serangan yang kita lakukan untuk musuh. *Wait* memiliki arti bahwa pemain akan menunggu atau tidak melakukan hal apapun (bisa dikatakan juga sebagai *end turn*).Sedangkan *run* memiliki arti bahwa kita akan lari dari arena pertarungan ini. Akan tetapi, tidak selamanya *run* akan berhasil lari. Jika lawan dari pemain adalah bos atau lawan yang jauh lebih hebat dari Persona yang dimiliki pemain, akan terdapat kemungkinan untuk tidak berhasil lari. Seperti namanya, *change Persona* memiliki arti mengganti Persona yang sedang di-*equip*.

Dalam pilihan *tactic,* pemain akan menentukan temannya akan melakukan apa, seperti *attack fallen* (terdapat Persona yang sedang *down* di *attack* kembali agar sembuh dari *down*), *conserve SP* (melakukan apa saja yang tidak menggunakan SP), *heal support* (akan melakukan *heal* sampai HP seluruh *party* penuh. Jika penuh, teman pemain akan bebas melakukan apapun), *support* (melakukan *attack up* yang akan berpengaruh ke *damage*, *defense up*, dan *evasion up* yang mungkin bisa digunakan jika menggunakan *luck*), *knock down* (menggunakan *physical* secara terus menerus dengan kemungkinan *critical* yang lebih tinggi sedikit. Hal ini dikarenakan jika *critical* Persona akan *down*, tapi tidak dapat membuat bos *down*), *wait* (*party*-nya akan menunggu. Biasanya digunakan ketika terdapat satu orang yang sedang *analyze*), *attack same* (apa yang di-*attack* oleh pemain, temannya akan melakukan *attack* dengan sasaran yang sama).

Dalam pilihan *item*, terdapat *item* yang dapat *heal* HP, *heal* SP, *heal* HP SP, menyerang dengan menggunakan elemental, melakukan *attack up* atau *attack down* dan semua yang mirip dengan itu. Selain yang telah disebutkan sebelumnya, di dalam pilihan *item* juga terdapat penjelasan tentang sepatu, pakaian, dan *weapon* yang sedang digunakan.

Gambar 2 Tampilan ketika mode pertarungan

Sumber :
http://upload.wikimedia.org/wikipedia/en/e/ed/Persona3_battlescreen.jpg (diakses pada 4 Mei 2015)

Pada permainan Persona 3 FES ini, terdapat sebuah kemampuan untuk memadukan dua buah Persona yang disebut *fusion*. *Fusion* diartikan sebagai kemampuan yang mengombinasikan kekuatan dari dua atau lebih Persona yang spesifik untuk membuat efek berbeda yang biasanya berbentuk lebih kuat dari mantra yang dipanggil. Keuntungan yang pemain dapatkan jika melakukan *fusion* adalah mendapatkan jenis Persona baru. Selain itu, pemain mendapatkan Persona yang levelnya lebih tinggi jika level kedua Persona tidak terlalu jauh. Jika level kedua Persona terlalu jauh, akan dihasilkan Persona dengan level di bawah salah satu level yang pemain *fusion*. Keuntungan lainnya, pemain akan mendapatkan Persona dengan kemampuan yang dibawa dari Persona sebelumnya dalam *random chance*.

Fusion merupakan cara utama untuk mendapatkan Persona yang lebih baik, karena jika hanya dari *drop tartarus*, pemain hanya mendapatkan Persona dengan *skill basic* dan *base stat*-nya saja. Sedangkan jika pemain menggunakan *fusion*, terdapat kemampuan yang dapat dibawa dan *base stat*-nya ditambahkan dengan suatu perhitungan dari Persona bahan *fusion* tersebut.

Oleh karena itu, makalah ini akan membahas bagaimana penggunaan algoritma Brute Force dalam memadukan (*fusion)* Persona dalam Persona 3 FES. Selain itu, makalah ini juga akan membahas cara yang sedikit lebih optimal dari Brute Force yang biasa digunakan, namun masih berada dalam cakupan algoritma Brute Force.

II. DASAR TEORI

Algoritma Brute Force merupakan pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Biasanya didasarkan pada pernyataan pada persoalan (*problem statement*), dan definisi konsep yang dilibatkan. Algoritma Brute Force memecahkan persoalan dengan sangat sederhana, langsung, jelas (*obvious way*) [2].

Kerugian dari menggunakan algoritma Brute Force adalah tidak efisien dalam kasus yang umum. Ada pula yang mengartikan bahwa algoritma Brute Force merepresentasikan sebuah gaya pemrograman yang primitif [3]. Hal ini dikarenakan pemrograman sangat bergantung kepada kemampuan otak (komputer) untuk melakukan komputasi daripada menggunakan kemampuan berpikirnya (*programmer*) dalam menyelesaikan ataupun menyederhanakan persoalan. Sehingga sebaiknya pemrogram tidak bergantung penuh terhadap kemampuan komputer, tetapi mengandalkan pemikirannya sendiri. Jika pemrogram mengandalkan pemikirannya sendiri, solusi yang didapat mungkin lebih optimal baik segi waktu maupun hasil yang didapat. Akan tetapi, tidak semua masalah dapat diselesaikan dengan algoritma lain. Hampir semua masalah dapat diselesaikan dengan algoritma Brute Force.

Selain itu, algoritma Brute Force dianggap sebagai algoritma yang paling umum digunakan oleh kalangan pemrogram dalam menyelesaikan masalah yang ada. Hal ini dikarenakan algoritmanya berjalan secara terurut dan sistematis mencoba setiap cara yang mungkin dijadikan solusi dari masalah. Kemudian algoritma ini akan mengecek apakah setiap percobaan yang dijalankan sesuai dengan masalah yang diberikan. Sehingga dapat disimpulkan bahwa algoritma ini bukanlah algoritma yang cerdas ataupun efisien.

Jika ukuran *input* yang akan diproses dengan algoritma Brute Force semakin besar, waktu ataupun langkah yang dibutuhkan untuk menyelesaikan masalah akan meningkat. Sehingga algoritma Brute Force ini memakan waktu yang lama jika diberikan kasus ukuran *input* yang besar. Walaupun algoritma Brute Force ini umum digunakan, ternyata dengan umum pula algoritma Brute Force ini kurang disukai karena harus meng-enumerasi lebih banyak langkah untuk mendapatkan suatu solusi dibandingkan algoritma lain. Oleh karena kemudahan dan memungkinkan menyelesaikan hampir semua persoalan, algoritma Brute Force sering digunakan sebagai pembanding antara algoritma tertentu dengan dirinya sendiri. Hal tersebut bertujuan untuk mengetahui apakah algoritma lain yang dibandingkan tersebut efisien atau tidak.

Hal unik yang dapat ditemukan dari algoritma ini adalah kesederhanaannya yang memungkinkan pemrogram untuk membobol (*crack*) sistem operasi atau *website* atau *file* yang penting. Pembobol akan berusaha memasukkan kata sandi secara acak sesuai pola tertentu atau sesuai dengan informasi yang didapatkan.

Algoritma Brute Force ini menghasilkan algoritma-algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan, dan sebagainya. Selain itu, algoritma Brute Force juga menghasilkan algoritma baku atau standar untuk tugas-tugas komputasi penjumlahan atau perkalian dari sebuah deret bilangan atau menentukan minimum/maksimum di dalam sebuah data. Jika sudah bingung dan ragu dengan algoritma yang dapat memecahkan suatu masalah, algoritma Brute Force adalah jalan keluarnya. Namun, algoritma Brute Force ini tidak se-kontruktif atau se-kreatif teknik pemecahan masalah lainnya.

Teknik heuristik adalah suatu teknik untuk mempercepat pencarian solusi, yaitu dengan cara mengeliminasi beberapa kemungkinan solusi tanpa harus mengeksplorasinya secara penuh. Teknik neuritik ini dapat juga membantu kita memutuskan solusi mana yang harus dievaluasi pertama kali.

III. IMPLEMENTASI ALGORITMA BRUTE FORCE DALAM *FUSION* PERSONA

Dari semua 169 Persona dalam Persona 3 FES, yang diurutkan menjadi 22 *arcana* digunakan di Tarot. Hal yang paling esensial adalah kita dapat mengombinasikan lebih dari 6 Persona dan mendapatkan sebuah Persona dari sana, kecuali *fusion*-nya tidak valid. Faktor utama untuk validitas *fusion* adalah bahwa beberapa *arcana* tidak dapat digabungkan. Faktor lain adalah bahwa beberapa *arcana* hanya dapat menyatu hanya dengan menggunakan tiga atau lebih Persona.

Ide dari penggunaan algoritma Brute Force ini adalah mencoba mengombinasikan semua Persona di dalam repertoar pemain dan mengulangi seluruhnya sampai tidak terdapat Persona baru yang dapat dibentuk. Di waktu yang sama, untuk setiap *fusion*, Persona yang sudah digunakan disimpan bersama dengan hasilnya untuk dapat dilihat pada akhirnya bagaimana sampai ke Persona yang dibutuhkan.

Memeriksa apakah dan bagaimana Persona dapat dibentuk dari dua buah atau lebih Persona saat ini dalam kepemilikan pengguna kemudian memeriksa jika Persona termasuk dalam daftar Persona yang telah diciptakan melalui *fusion*. Jika ada, salah satu kebutuhan secara rekursif menyelesaikan semua bahan Persona yang nantinya akan digunakan untuk target Persona sampai semua bahan yang sudah ada dalam kepemilikan pemain. Jika tidak ada, Persona tidak bisa dibuat. Sebagai contoh, karena pemain tidak memiliki Persona yang diperlukan di tempat pertama atau mungkin rumus *fusion* akan menjadi gagal.

Berikut adalah *source code* (dalam bahasa C#) yang berkorespondensi untuk versi yang tidak optimal[4].

```
List&amp;lt;FusedPersonaList&amp;gt;
lists = new
List&amp;lt;FusedPersonaList&amp;gt;
();

// Initialize from the user's
currently owned Persona
lists.Add(UserDB.GetList());

while (true) {
    int lastCount =
lists.Last().Count;
    // Create the set of Persona
which can be created from the list
from
    // the last step.
    FusedPersonaList newList =
lists.Last().CreateNewList(Database,
UserDB);
    int currentCount =
newList.Count;

    // If the number of Persona has
not increased, stop looping.
    if (lastCount == currentCount) {
        break;
    } else {
        lists.Add(newList);
    }
}
```

Setelah *loop*, *item* terakhir dari *list* adalah lis dengan semua Persona yang dapat dipadukan. Implementasi *fusion* sendiri lempang (*straightforward*) ketika mengikuti aturan untuk *fusion* dari permainan Persona 3 FES itu sendiri.

3.1 Normal Spread Fusions

Dalam kasus *fusions* biasa, ketika kita secara sederhana mencoba untuk memadukan setiap Persona di dalam lis dengan setiap Persona lain di dalam lis, kita akan mencoba sebanyak n^2 *fusions*. Berikut adalah penggunaan algoritma Brute Force dalam memadukan setiap Persona dalam lis dan kemudian terdapat pemanggilan fungsi NormalSpreadFusion yang berisi hasil dari penggabungan p1 dan p2 sesuai dengan data yang dimiliki oleh program.

```
for (int i = 0; i < persona.Count;
i++)
{
  for (int j = 0; j < persona.Count;
j++)
  {
      Persona p1 = persona[i];
      Persona p2 = persona[j];
      Persona result =
NormalSpreadFusion(p1,p2);
  }
}
```

Bagaimana pun juga, terdapat banyak *fusions* yang tidak berguna di sana. Pertama kita tidak ingin mencoba memadukan Persona yang sama dengan dirinya sendiri, karena hal ini tidak valid. Hal ini berarti bahwa kita tidak membolehkan semua urutan dari 2 elemen (p_1, p_2), tapi kita membatasi untuk semua urutan dimana $p_1 \neq p_2$. Hasil ini menyebabkan sejumlah percobaan *fusion* yang lebih sedikit, yaitu $n^2 - n$. Bagaimanapun, kita masih mencoba semua perintah, berarti jika kita mencoba (p_1, p_2), kita akan mencoba (p_2, p_1) juga, walaupun hasil ini adalah sama. Bagaimanapun, apa yang kita ingin uji hanya *subsets* dari ukuran 2, yang tentu saja memiliki kemungkinan sebesar $\binom{n}{2}$. Berikut adalah teknik heuristik yang digunakan pada kode berikut[4].

```
for (int i = 0; i < persona.Count;
i++)
{
   for (int j = i + 1; j <
persona.Count; j++)
   {
      Persona p1 = persona[i];
      Persona p2 = persona[j];
      Persona result =
NormalSpreadFusion(p1, p2);
   }
}
```

Terlihat pada kode bahwa nilai awal j dengan algoritma Brute Force diawali dengan 0. Sesuai dengan penjelasan sebelumnya, maka kita dapat mengatur nilai awal j dengan i+1 sehingga memberikan hasil yang lebih efektif dan efisien daripada algoritma Brute Force biasa.

Sayangnya, kita masih mempunyai kompleksitas kuadrat dalam sejumlah uji *fusion*, tetapi kita telah

mengatur untuk mengurangi jumlah absolut perbandingan sedikit. Sebagai contoh, jika kita memiliki sekitar 150 Persona (yang dekat dengan kasus terburuk dari semua kecuali satu Persona berada di daftar), algoritma Brute Force akan menguji 22.500 *fusion*. Dengan berkurangnya jumlah *fusions*, maka kita hanya akan menguji sebanyak 11.175 *fusions*. Memang bukan perbandingan yang cukup besar, namun sangat memengaruhi kecepatan berjalannya program dan lebih efektif serta efisien dibandingkan dengan algoritma Brute Force.

3.2 Triangle Spread Fusions

Triangle Spread Fusions bekerja sedikit berbeda dengan *reguler spread fusions*. Jika kita ingin mendapatkan hasil dari *triangle fusion*, kita harus mengurutkan Persona berdasarkan level yang mereka miliki dan jumlah *Arcana* dalam kasus jika seri. Kemudian, sebuah *reguler fusion* dilakukan pertama (bawah) dua Persona, diikuti oleh *fusion* dua Persona (meskipun dengan aturan khusus) untuk mendapatkan Persona akhir.

Sekali lagi, algoritma Brute Force akan menempatkan semua urutan yang mungkin dengan tiga Persona untuk menguji dan secara internal mengurutkan mereka. Hal ini menyebabkan sejumlah kubik uji *fusion* n^3. Ketiga Persona harus berbeda, oleh karena itu kita dapat pengecualian *tupel* seperti (p_1, p_1, p_2). Hal ini menghasilkan jumlah percobaan sebanyak $n(n-1)(n-2)$. Sekali lagi daftar hanya berisi *subset* unik berukuran tiga, kita mendapatkan kemungkinan sebanyak $\binom{n}{3}$, yang kita bisa dapatkan dengan sebuah algoritma yang dideskripsikan sebelumnya.

Namun, kita dapat melakukan yang lebih baik, tidak dalam jumlah pengujian *fusion*, tapi kita dapat mengeliminasi pengurutan internal dan mencoba untuk meng-enumerasi Persona dalam urutan *lexicographical*. Urutan *lexicographical* adalah bentuk umum dari urutan alfabet kata yang berdasarkan pada pengurutan huruf depan.

Hal ini berarti untuk (p_1,p_2,p_3), $p_1 \leq p_2$ dan $p_2 \leq p_3$ ditahan (menggunakan urutan berdasarkan level dan *arcana* yang telah disebutkan sebelumnya). Seperti yang telah dijelaskan sebelumnya, kita tidak mengurangi jumlah pengujian *fusion*, tapi kita menyelamatkan diri dari kesulitan harus memesan tiga set Persona setiap kali kita jumpai kombinasi, karena kita yakin bahwa set tersebut sudah diurutkan. Jika kita asumsikan lagi 150 Persona, algoritma Brute Force akan memeriksa 3.375.000 tiga kali lipat untuk hasil *fusion*. Dengan optimasi, kita menguji 551.300 *fusions*, tanpa perlu mengurutkan hasil lagi secara internal, pada *cost* penyortiran *array* sebelum setiap putaran dilakukan.

Terdapat satu kemungkinan optimasi lagi. Katakan kita mulai dengan sekumpulan P_1 dengan ukuran sebesar k_1. Kita menguji semua kombinasi dari Persona di dalam himpunan, mengambil sekumpulan P_2 dengan ukuran sebesar $k_2 > k_1$. Jika kita lanjutkan bagian ini, kita akan mendapatkan perulangan yang banyak untuk menguji yang kita telah punya dalam himpunan yang kecil. Katakan perbedaan dari P_1 dan P_2 adalah P_x. Kita butuh menguji semua kombinasi dari Persona dalam P_x sama dengan semua kombinasi dari satu *member* P_x dan satu dari P_1 untuk mendapatkan sekumpulan P_3. Hal ini berarti kita harus melakukan pengujian sebanyak $\frac{k_x(k_x-1)}{2} + k_x k_1$. Katakan bahwa k_1 adalah 70, k_2 adalah 90, maka $k_x = 20$. Jika menggunakan algoritma Brute Force (algoritma *naive*) untuk menghitung P_3, kita butuh (hanya untuk *normal spread fusions*) mengecek sebanyak 4.005. Jika kita menggunakan pendekatan alternatif, kita hanya butuh mengecek sebanyak 1.590[4].

3.3 Caching Persona Levels

Hal pertama untuk diperhatikan adalah bahwa kita dapat memprediksi level sebuah Persona yang kita padukan akan mendapatkan level terlepas dari Persona saat ini dalam *inventor*, karena hanya dipengaruhi oleh *social link* yang berasosiasi dengan *arcana* Persona. Oleh karena itu, kita dapat melakukan *cache* semua kemungkinan level dari Persona, yang Aan berguna untuk *triangle fusions*.

3.4 Caching Fusions

Sebuah implementasi Brute Force dari skema *caching* untuk hasil *fusion* akan menggunakan *array* dua dimensi (tiga dimensi untuk *triangle fusions*) untuk menyimpan kemungkinan hasil dari setiap *fusions*. Namun, karena aturan *fusions* dari permainan Persona 3 FES ini sendiri, sebagian besar dari kombinasi Persona sebenarnya tidak valid. Untuk *normal spread fusions*, jumlah kemungkinan kombinasi adalah $n^2 = 169^2 = 28.561$ dengan hanya 9998 dari mereka menghasilkan sebuah Persona yang valid. Efek ini bahkan lebih diucapkan dalam kasus *triangle fusions*, di mana kemungkinan kombinasi berjumlah $n^3 = 169^3 = 4.826.809$. Jumlah *fusion* yang valid ketika memulai, dari Persona, yang dimiliki pemain saat ini bergantung pada level dari Persona. Jika kita menggunakan level dasar dari siap Persona, hanya ada 13.344 kombinasi yang menghasilkan sebuah Persona baru.

3.5 Goal-Directed Lookup

Selama mengisi *cache*, kita dapat menyimpan *fusion* sedemikian rupa bahwa kita bisa mendapatkan daftar semua *fusion* yang menghasilkan Persona yang diberikan. Oleh karena itu, sekarang kita tidak perlu memeriksa semua kombinasi seperti sebelumnya, kita dapat menghilangkan *fusion* yang akan menghasilkan sebuah Persona yang sudah kita miliki di set "Persona yang sudah dibuat". Selain itu, kita bisa *cache* daftar Persona yang muncul sebagai bahan untuk Persona tertentu, sehingga kita dapat memeriksa daftar ini melawan Persona yang tersedia untuk melihat apakah itu mungkin untuk membuat Persona dari yang sudah tersedia. Namun, dapat ditebak bahwa dalam kasus ini, mudah untuk hanya memeriksa semua dan mendapatkannya hampir dengan kecepatan yang sama.

3.6 Costs

Skema *caching* tentu saja menggunakan *memory cost*. Setiap *normal spread fusion*, kita butuh menyimpan dua bahan Persona beserta hasil Persona. Karena terdapat

169 Persona di dalam permainan Persona 3 FES, sebuah referensi ke Persona akan masuk ke dalam *byte*. Oleh karena itu, kita membutuhkan 3 *byte* untuk *cache* satu hasil *fusion*. Seperti yang telah disebutkan di atas, terdapat 10.580 *fusion* yang valid mungkin dalam himpunan semua Persona. Oleh karena itu, kita mempunyai 3 * 10.580 *bytes*, sekitar 31 KB. Pada *triangle spread fusions*, kita tidak dapat memberikan jumlah pasti tanpa menggunakan level yang sebenarnya dari Persona pemain. Asumsikan bahwa jumlah yang dihasilkan dari semua Persona dalam level dasar, 13.986, dekat dengan kasus terburuk. Hal ini berarti bahwa kita membutuhkan (dengan tambahan *byte* untuk Persona ke tiga) 4 * 13.986 *bytes* = 55 KB.

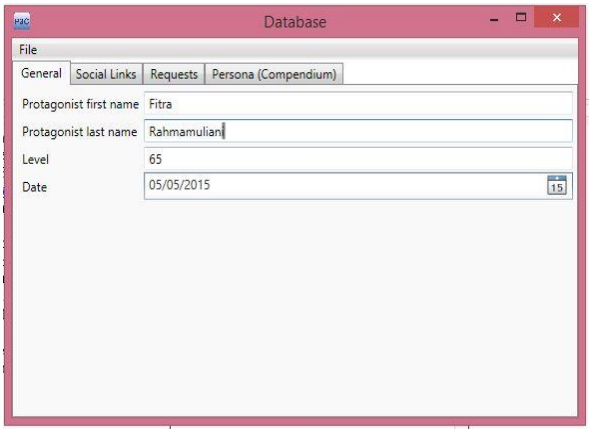
3.7 Caching Triangle Fusion Results

Satu hal penting mengenai hasil *caching triangle fusions* adalah sebagai berikut. Dalam *triangle fusions*, level sebenarnya yang tertinggi (bukan dasar) dari tiga Persona yang terlibat diperlukan untuk mengetahui dalam rangka memadukan mereka. Oleh karena itu, *cache* untuk *triangle fusions* perlu menggunakan level Persona dalam *compendium* pemain saat ini. Agar dapat mengorbankan pengurutan tiap kali himpunan yang sewenang-wenang dengan 3 Persona diuji, kita dapat menyimpan hasil dari tiap permutasi *fusion* dalam *cache*. Perona yang belum terdapat dalam *compendium* pemain, kita dapat prediksi level dasar mereka dalam peringkat *social link*. Dengan cara ini, Persona dapat dimasukkan dalam urutan yang sewenang-wenang, pada *cost* menggunakan memori lebih untuk informasi yang berlebihan pada permutasi.

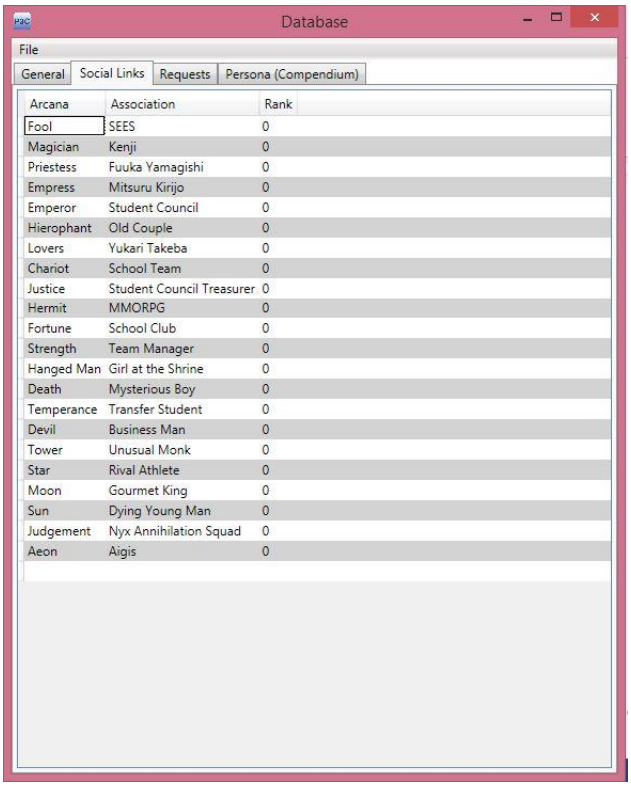
IV. PROGRAM YANG BERKAITAN

Setelah penulis mencari artikel yang berkaitan dengan makalah ini, penulis menemukan sebuah program yang sudah selesai dibuat dan menggunakan hasil optimasi dari Brute Force. Program ini dapat diakses pada *link* http://www.mehm.net/download/P3FESCalculator.zip. Program ini dinamakan Persona 3 FES Caculator. Dengan Persona 3 FES Calculator ini, kita diperbolehkan untuk masuk ke status saat ini dalam Persona 3 FES dengan diisi sendiri maupun *import* informasi dari permainan yang disimpan, dan kemudian kita gunakan untuk menghitung jika kita dapat memadukan Persona tertentu (dalam kemungkinan beberapa cara) dari Persona yang telah kita miliki.

Berikut adalah beberapa gambar ketika penulis menggunakan program tersebut.

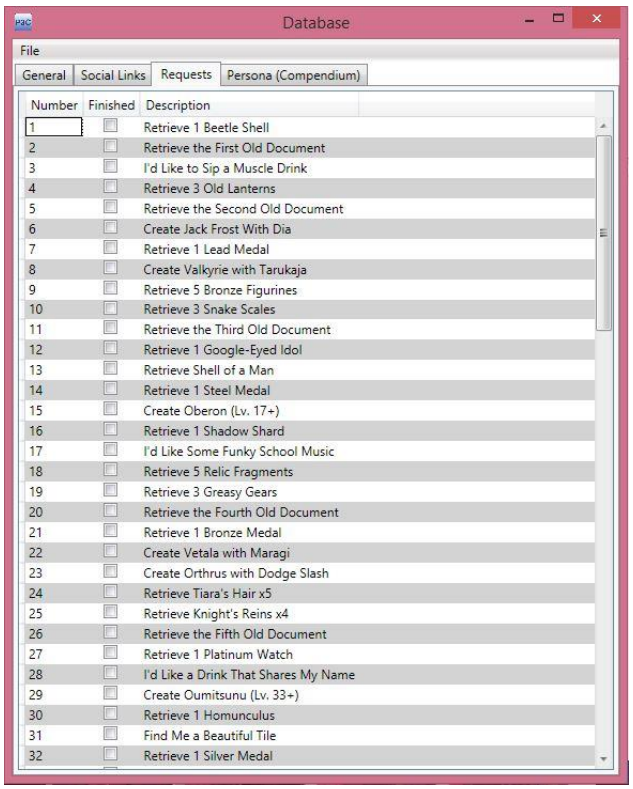


Gambar 3 Pengisian Basis Data Pemeran Utama (digunakan pada 5 Mei 2015)



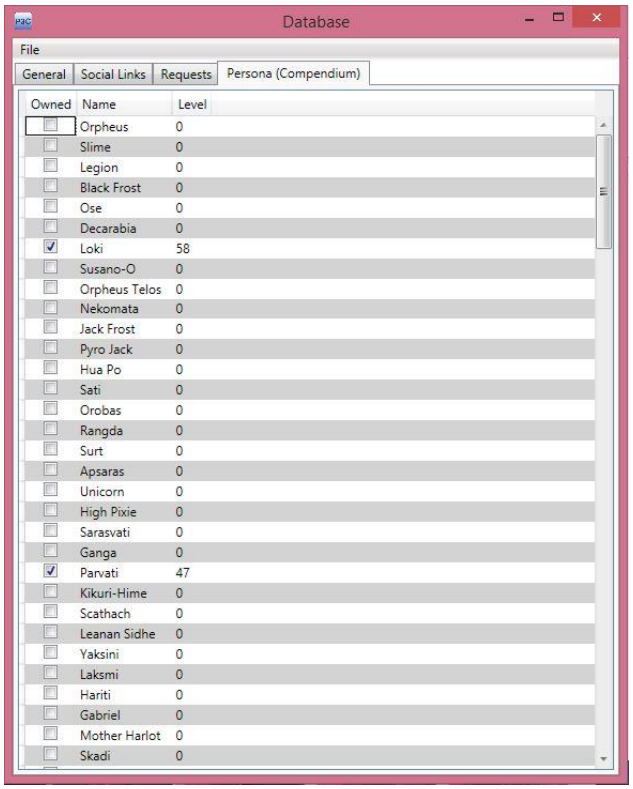
Arcana	Association	Rank
Fool	SEES	0
Magician	Kenji	0
Priestess	Fuuka Yamagishi	0
Empress	Mitsuru Kirijo	0
Emperor	Student Council	0
Hierophant	Old Couple	0
Lovers	Yukari Takeba	0
Chariot	School Team	0
Justice	Student Council Treasurer	0
Hermit	MMORPG	0
Fortune	School Club	0
Strength	Team Manager	0
Hanged Man	Girl at the Shrine	0
Death	Mysterious Boy	0
Temperance	Transfer Student	0
Devil	Business Man	0
Tower	Unusual Monk	0
Star	Rival Athlete	0
Moon	Gourmet King	0
Sun	Dying Young Man	0
Judgement	Nyx Annihilation Squad	0
Aeon	Aigis	0

Gambar 4 Basis Data *Social Links* dari pemain



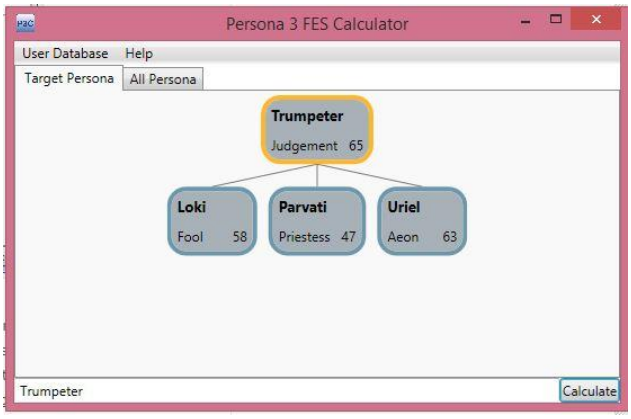
Gambar 6 Basis Data Permintaan

Pada Basis Data Permintaan, terdapat kolom *finished* dan *Description*. Di sini, pengguna program atau pemain dapat mencentang apakah deskripsi tersebut telah selesai dilakukan atau tidak. Jika pengguna program telah memiliki data penyimpanan permainan, program secara otomatis akan mengatur permintaan mana yang sudah selesai dan permintaan mana yang belum selesai.



Gambar 7 Basis Data Persona (Compendium)

Sama seperti basis data permintaan, terdapat kolom *owned*, *name*, dan level. Dari kolom tersebut dapat dilihat persona mana yang sudah kita miliki dan berada pada level berapa. Di sini, pengguna program atau pemain dapat mencentang apakah persona tersebut telah dimiliki atau tidak. Jika pengguna program telah memiliki data penyimpanan permainan, program secara otomatis akan mengatur persona mana yang sudah dimiliki dan persona mana yang belum dimiliki.



Gambar 5 Contoh Penggunaan *Triangle Spread Fusions*

Pada gambar 5 di sudut kiri bawah, tertulis nama "Trumpeter". Nama tersebut adalah *input* dari pengguna program tentang persona apa yang dia cari dan apakah dia bisa mendapatkan persona tersebut dari persona yang dimilikinya. Ketika pengguna menekan tombol *calculate*, program akan otomatis men-*generate* bagan persona, di mana Persona yang kita cari garisnya diberi warna oranye.

Gambar 8 Semua Persona yang Bisa Didapat

Pada gambar 8, terlihat banyak sekali data Persona yang bisa dimiliki oleh pemain. Pada bagian kiri sebelum tanda "=" (sama dengan) adalah Persona baru yang bisa didapatkan, di dalam kurung terdapat angka yang merupakan level dari Persona baru yang mungkin bisa kita dapatkan. Sedangkan bagian sebelah kanan merupakan Persona-Persona yang salah satunya merupakan Persona yang kita miliki, jika digabungkan dengan Persona yang belum kita miliki, maka kita akan mendapatkan Persona baru di sebelah kirinya.

V. KESIMPULAN

Penggunaan algoritma Brute Force sangat berguna dan mampu menyelesaikan masalah *fusion* Persona pada Persona 3 FES. Akan tetapi, ternyata algoritma Brute Force tersebut tidak efektif dan efisien. Sehingga, kita dapat mengoptimalkan algoritma Brute Force tersebut sedikit lebih baik. Walaupun bukan mengubah algoritmanya menjadi algoritma dengan nama lain.

Mengoptimalkan algoritma jauh lebih efisien. Hal ini dikarenakan pada algoritma Brute Force terkadang terjadi kemacetan dalam kode entah itu dikarenakan memori yang digunakan ataupun kecepatan akses. Oleh karena optimalisasi algoritma, kecepatan program meningkat dengan drastis. Jika kita melakukan optimalisasi kode, pastikan bahwa pemrogram benar-benar mengoptimalkan kecepatan atau memori.

VI. UCAPAN TERIMA KASIH

Penulis bersyukur kepada Allah SWT oleh karena rahmat dan hidayahnya, penulis dapat menyelesaikan makalah berjudul "Penggunaan Algoritma Brute Force dalam Memadukan (*fusion*) Persona pada Persona 3 FES ini berhasil diselesaikan tepat waktu. Penulis juga berterima kasih kepada Dr. Ir. Rinaldi Munir dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc untuk arahan dan ajaran mereka pada mata kuliah Strategi Algoritma sehingga penulis mendapatkan pengetahuan yang sangat berguna dalam menyelesaikan makalah ini dan masih butuh belajar lagi. Selain itu, penulis juga tidak lupa berterima kasih kepada kedua orang tua, Krisna Reynaldi, Elietzer Christanto, Adhitya Rusman, Andra Wahyu Purnomo, keluarga, dan seluruh teman yang telah memberikan masukan, dukungan, semangat, dan doa mereka.

REFERENCES

[1] http://megamitensei.wikia.com/wiki/Persona_3_FES diakses pada 2 Mei 2015.
[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Algoritma%20Brute%20Force%20(2015).ppt diakses pada 4 Mei 2015
[3] http://dictionary.reference.com/browse/brute+force diakses pada 4 Mei 2015.
[4] http://mehm.net/blog/?p=203 diakses pada 2 Mei 2015.

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, 4 Mei 2015

Fitra Rahmamuliani 13513095