Penerapan Algoritma *Greedy* pada *Game Valkyrie Crusade*

Rangga Yustian M. - 13511017[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511017@std.stei.itb.ac.id

***Abstrak*—Algoritma *greedy* menjadi salah satu algoritma yang populer dalam pengembangan berbagai aplikasi, tidak terkecuali *game*. Algoritma *greedy* menjadi pilihan pengembang aplikasi yang menghendaki kompleksitas algoritma yang pendek dan tidak mengharapkan optimasi yang optimum absolut atau mendekati optimasi optimum. Makalah ini membahas penerapan algoritma *greedy* pada *game Valkyrie Crusade* dengan tujuan menganalisis bagian-bagian dari *game* ini yang dapat diterapkan algoritma *greedy* dan membantu pemain *game* ini dalam mencapai hasil terbaik.**

***Kata Kunci*—*Valkyrie Crusade*, algoritma *greedy*, *Knapsack problem*, optimasi.**

I. Pendahuluan

A. Latar Belakang

Dewasa ini, sudah banyak penerapan berbagai macam algoritma di berbagai macam aplikasi, salah satunya aplikasi *game*. Sebuah aplikasi *game* memiliki elemen yang tidak dapat dipisahkan, yaitu interaksi antara aplikasi itu sendiri dengan penggunanya. Oleh karena itu, perwujudannya adalah dengan mengimplementasikan berbagai macam algoritma yang memadai. Penulis memilih algoritma *greedy* karena implementasinya yang mudah dan handal dalam menentukan solusi dari berbagai macam persoalan yang tidak terlalu membutuhkan solusi optimum. Kemudian, penulis memilih aplikasi *game Valkyrie Crusade* karena *game* tersebut adalah minat penulis.

II. Algoritma Greedy

Algoritma *greedy* adalah algoritma pencarian solusi dari persoalan optimasi yang melihat pilihan terbaik pada saat itu juga [2]. Dengan kata lain, algoritma ini mencari solusi optimum lokal dan diharapkan menuju ke solusi optimum global yang mengarah ke solusi teroptimal. Oleh karena itu, keluaran dari algoritma ini sering kali bukan merupakan solusi optimal. Beberapa persoalan optimasi yang dapat diselesaikan dengan algoritma ini adalah *knapsack problem*, *minimum-spanning tree*, pencarian jalur terpendek, dan persoalan penukaran koin.

Algoritma *greedy* terdiri dari lima elemen, yaitu himpunan kandidat (*C*), himpunan solusi (*S*), fungsi seleksi, fungsi kelayakan, dan fungsi obyektif [3]. Himpunan kandidat adalah kumpulan entitas yang akan dipilih untuk memenuhi himpunan solusi. Himpunan solusi adalah sub himpunan dari himpunan kandidat yang sesuai dengan kriteria yang ditentukan. Fungsi seleksi adalah fungsi yang mengambil elemen optimal dari himpunan kandidat. Fungsi kelayakan adalah fungsi yang memeriksa apakah setiap elemen yang dipilih oleh fungsi seleksi memenuhi kriteria yang ditentukan. Fungsi obyektif adalah fungsi yang mengoptimisasi himpunan solusi.

Cara kerja algoritma *greedy* adalah sebagai berikut. Algoritma *greedy* mencari *S* yang merupakan himpunan bagian dari *C* dengan syarat *S* harus sesuai dengan kriteria yang ditentukan. Pencarian dihentikan apabila tidak ada *S* yang memenuhi kriteria yang ditentukan dan sebaliknya. Setelah ada *S* yang memenuhi kriteria, *S* dioptimisasi oleh fungsi obyektif. Berikut ini adalah *pseudocode* dari algoritma *greedy*.

```
  function greedy(input C: himpunan_kandidat) ->
himpunan_kandidat
  { Mengembalikan solusi dari persoalan optimasi
dengan
     algoritma greedy
       Masukan: himpunan kandidat C
         Keluaran: himpunan solusi yang bertipe
himpunan_kandidat
  }
  Deklarasi
       x : kandidat
       S : himpunan_kandidat

  Algoritma:
       S <- {} { inisialisasi S dengan kosong }
          while (not SOLUSI(S)) and (C != {}) do
                x <- SELEKSI(C) { pilih sebuah
kandidat dari C }
                C <- C - {x} {elemen himpunan
kandidat berkurang satu}
             if LAYAK(S U {x}) then
                   S <- S U {X}
             endif
          endwhile
          {SOLUSI(S) or C = {} }

          if SOLUSI(S) then
             return S
          else
             write('tidak ada solusi')
          endif
```

Gambar 1. *Pseudocode* algoritma *greedy*
(Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma",Institut Teknologi Bandung,2009.)

Dengan sifat algoritma *greedy* tersebut, algoritma ini belum tentu dapat menghasilkan solusi optimum. Alasannya, algoritma *greedy* tidak mencari semua alternatif solusi yang ada ketika proses pencarian himpunan solusi. Selain itu, banyak fungsi seleksi yang berbeda sehingga dapat menghasilkan solusi yang berbeda pula.

Dengan demikian, algoritma *greedy* memadai untuk keperluan-keperluan yang tidak membutuhkan solusi optimum yang presisi. Algoritma *greedy* juga membutuhkan waktu yang relatif lebih singkat dari pada algoritma yang dapat menghasilkan solusi optimum.

III. Valkyrie Crusade

Valkyrie Crusade adalah *collectible card game* berbasis Android dan iOS yang dibuat oleh Nubee Tokyo, anak perusahaan dari Nubee Pte Ltd, *game studio* berbasis Singapura. *Game* ini dirilis di Appstore Japan pada 18 Februari 2013 dan 19 Maret 2013 pada Google Play secara internasional[1].

Gambar 2. Logo *Valkyrie Crusade*
(valkyriecrusade.wikia.com)

Dasar permainan dari *game* ini adalah sebagai berikut. Pemain membangun kota yang ia miliki untuk mendapatkan tiga sumber daya utama, yaitu *gold*, *ether*, dan *iron*. Ketiga sumber daya utama ini digunakan untuk pengembangan kota selanjutnya dan pembangunan unit tempur yang berupa karakter-karakter perempuan yang direpresentasikan dengan kartu. Nantinya, karakter-karakter yang diperoleh digunakan dalam berbagai pertempuran, yaitu pertempuran melawan musuh yang datang pada map *campaign* dan *archwitch*, yaitu musuh utama yang menjadi daya tarik dari *game* ini. Contoh pertarungan dengan *archwitch* terlihat pada gambar berikut. Setelah pemain memenangkan pertarungan melawan *archwitch*, pemain diberikan penghargaan berupa karakter lain atau *item* yang dapat menunjang kebutuhan dalam permainan, seperti sepatu yang dapat digunakan untuk memulihkan *vitality* yang dikonsumsi. *Vitality* adalah stamina pemain dalam menjelajah berbagai peta yang disediakan *game* ini.

Gambar 3. Pertarungan melawan *archwitch* alias musuh utama
(dok. pribadi)

Setiap karakter mempunyai *rarity* yang berbeda. *Rarity* yang disediakan oleh *game* ini berupa *normal* (N), *rare* (R), *super rare* (SR), dan *ultra rare* (UR). Setiap *Rarity* ini memiliki versi *high.* Semakin tinggi tingkat *rarity*-nya, semakin sulit untuk diperoleh dan semakin kuat karakternya. Setiap karakter mempunyai delapan elemen utama, yaitu level, *rarity*, *skill*, elemen, status *attack*, status *defense*, status *health point* (dalam game, *soldier count*), dan *card cost*. *Card cost* adalah 'harga' karakter yang dapat ditampung dalam pasukan ofensif maupun defensif. Kemudian, g*ame* ini menyediakan empat elemen yang berbeda, yaitu *passion*, *cool*, *dark*, dan *light*. Keempat elemen ini saling memperkuat dan melemahkan satu sama lain. Kombinasi yang ada dari keempat elemen ini adalah *passion – cool* dan *light – dark*.

Gambar 4. Informasi karakter bernama *Omoikane*
(dok. pribadi)

Gambar tersebut menunjukkan informasi dari karakter bernama *Omoikane*. Karakter ini memiliki *rarity* (tingkat kelangkaan) *high super rare* (HSR). Selain itu, karakter ini memiliki level 60, *cost* 58, *attack* 9072, *defense* 8736 (*attack* dan *defense* tidak termasuk bonus yang ditulis dengan tulisan biru), *soldier count* 10920, dan *skill* penambah *attack* dari seluruh unit yang sedang dikendalikan pemain.

Setiap karakter dapat dikembangkan lebih jauh. Salah satu metode pengembangan karakter adalah menaikkan *level* karakter. Caranya dengan menggabungkan karakter yang akan dikorbankan kepada karakter yang akan dinaikkan levelnya. Ada cara lain juga, yaitu dibawa ke pertempuran. Setiap pertempuran akan menambah *experience point* ke semua karakter yang berpartisipasi dalam pertempuran.

Metode kedua dari pengembangan karakter adalah evolusi karakter. Untuk melakukan hal tersebut, pemain harus memiliki dua atau lebih karakter yang sama. Setelah itu, pemain dapat menggabungkan dua karakter yang sama sehingga karakter tersebut terlahir kembali. Jumlah dasar *attack*, *defense*, dan *soldier count* yang dimiliki karakter tersebut bertambah. Beberapa karakter tertentu dapat mengalami evolusi yang mengakibatkan karakter tersebut berubah sama sekali. Hal ini disebut *evolution accident*. Peluang terjadinya kejadian ini agak kecil, tetapi dapat diperbesar dengan item khusus yang dapat dibeli pada toko yang disediakan *game* ini.

Metode terakhir dari pengembangan karakter adalah penggabungan dua atau lebih karakter yang berbeda sehingga menghasilkan satu karakter yang berbeda. Di dalam *game* ini, metode ini bernama *amalgamation.* Tidak semua karakter dapat dikembangkan dengan metode ini.

Untuk mendapatkan karakter yang kuat, ada dua cara utama yang dapat dilakukan pemain. *Pertama*, bermain dengan tidak menghabiskan uang dari dunia nyata kepada *game* ini. Pemain dapat mendapatkan karakter-karakter kuat dengan mengalahkan *archwitch*. Walaupun demikian, peluang untuk mendapatkan karakter dengan *rarity* tertinggi dari *archwitch* (untuk sekarang ini, *super rare*) sangat kecil. Oleh karena itu, sangat dibutuhkan dedikasi dan kesabaran yang tinggi. *Kedua*, bermain dengan menggunakan uang dari dunia nyata kepada *game* ini. *Game* ini menyediakan mekanisme cepat untuk mendapatkan karakter-karakter yang dapat diperoleh pemain. Mekanisme tersebut bernama *summoning*.

Ada tiga tipe *summon* yang disediakan *game* ini, yaitu *free summon*, *premium summon*, dan *ultimate summon*. Untuk tipe pertama, pemain dapat melakukan *summon* dengan murah karena sumber daya yang dibutuhkan untuk *summon* tipe ini relatif mudah diperoleh. Untuk tipe kedua, biaya yang harus dikeluarkan untuk *summon* tipe ini relatif mahal karena sumber daya yang dibutuhkan sulit didapat. *Summon* tipe ini dapat menghasilkan karakter dengan *rarity* R dan SR. Berikut ini adalah hasil dari *premium summon* yang dilakukan oleh seorang pemain *game* ini.

Gambar 5. Hasil dari *premium summon* yang dilakukan seorang pemain *game* ini (dok. pribadi)

Gambar di atas menunjukkan karakter benama *Himiko* yang keluar dari *premium summon*. Peluang munculnya karakter dengan *rarity* SR sangat kecil. Oleh karena itu, pemain harus menyiapkan uang dalam jumlah besar untuk melakukan *premium summon* berkali-kali sampai muncul karakter SR.

Setelah melakukan *premium summon*, pemain akan mendapatkan suatu tiket khusus yang dapat digunakan untuk mengadakan *ultimate summon*. Tiket ini disebut *maiden ticket*. *Ultimate summon* dapat memunculkan karakter yang tidak dapat ditemui pada dua tipe *summon* sebelumnya. Jika karakter yang muncul pada *summon* tipe ini berupa karakter R, peluang untuk mendapatkan karakter SR meningkat sebesar 11%. Setelah pemain mendapatkan karakter SR, peluang untuk mendapatkan karakter SR selanjutnya dikembalikan ke peluang semula.

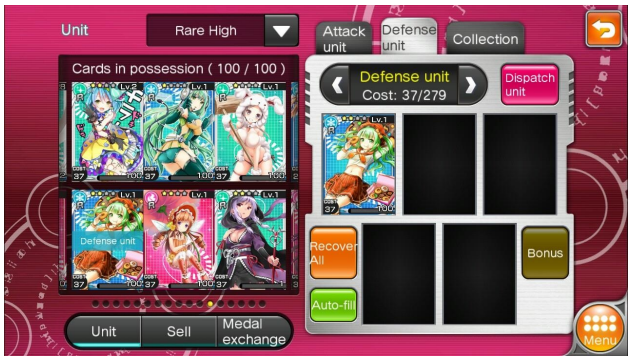
Gambar 6. Hasil dari *ultimate summon* yang dilakukan seorang pemain *game* ini (dok. pribadi)

Gambar di atas menunjukkan karakter bernama

Arianrhod dengan kelangkaan bertipe *super rare* yang muncul pada *ultimate summon*. Seperti *premium summon*, peluang dasar munculnya karakter dengan kelangkaan tipe tersebut sangat kecil.

IV. Aplikasi

Salah satu penerapan algoritma *greedy* pada game ini adalah pembangunan pasukan. Pembangunan pasukan adalah salah satu faktor penting dalam *game* ini. Untuk mendapatkan hasil yang memuaskan dalam pertarungan antar karakter di dalam *game* ini, pemain perlu memilih karakter yang dimiliki dengan cermat. Setiap pasukan dapat diisi maksimal lima karakter. Selain itu, setiap pasukan memiliki *unit cost*, yaitu batas jumlah *card cost* setiap karakter yang dapat ditampung pada pasukan tersebut. Jika pemain mencoba untuk memasukkan karakter yang dapat membuat jumlah *card cost* melebihi *unit cost*, karakter tersebut tertolak sehingga tidak dapat dimasukkan. Antarmuka pembangunan pasukan yang disediakan *game* ini adalah sebagai berikut.



Gambar 7. Antarmuka pembangunan pasukan (dok. pribadi)

Penulis menemukan empat optimasi penting yang dapat dilakukan, yaitu pemilihan karakter berdasarkan *attack* tertinggi, *defense* tertinggi, rasio *attack* dan/atau *defense* dengan *card cost* terendah.

Secara umum, *pseudocode* dari pemilihan karakter yang dioptimisasi dengan algoritma *greedy* adalah sebagai berikut.

```
function chooseUnit(input C:
card_group, unit_cost: integer,
GreedyBy: string, EnemyAvgElement:
string) -> card_group
{ Mengembalikan unit berdasarkan
attack terkuat
  Masukan: himpunan kartu C
  Keluaran: himpunan solusi yang
bertipe card_group
}
Deklarasi
  x : kandidat
  S : card_group

Algoritma
  S <- {}
  while (not SOLUSI(S)) and (C != {})
do
    x <- SELEKSI(C, GreedyBy,
EnemyAvgElement)
    C <- C - {x}
    if LAYAK(S U {x}) then
        S <- S U {X}
    endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then
    return S
  else
    write('tidak ada solusi')
  endif
```

Gambar 8. *Pseudocode* pemilihan karakter dengan algoritma *greedy*

Mula-mula, inisiasi himpunan solusi S dengan himpunan kosong. Kemudian, lakukan seleksi terhadap himpunan kandidat C dengan kriteria yang ditetapkan dengan variabel 'GreedyBy'. Variabel tersebut menyatakan jenis optimasi yang akan dilakukan. Fungsi seleksi yang digunakan memilih nilai terbesar dari optimasi yang dipilih. Contohnya, jika pengguna menginginkan optimasi berdasarkan *attack* terbesar, fungsi ini mencari karakter/kartu yang memiliki *attack* terbesar dan memasukannya ke dalam himpunan solusi apa bila kandidat tersebut layak.

Untuk pengujian algoritma *greedy* pada sampel yang dilampirkan pada makalah ini, penulis menetapkan *unit cost* sebesar 278. Format dari data yang diperoleh adalah <nama karakter> <*card cost*> <*rarity*> <*attack count*> <*defense count*> <*soldier count*> <elemen>

A. Pemilihan Pasukan berdasarkan Jumlah Attack *Tertinggi*

Berdasarkan sampel yang disertakan dalam lampiran pada makalah ini, penulis mendapatkan hasil sebagai berikut.

```
Hiderigami 62 HSR 9976 9328 10616 Passion
Astaroth 62 HSR 9856 9352 11104 Dark
Scylla 47 HSR 9567 9689 9205 Passion
Auxo 50 HSR 9353 9353 9733 Light
Hervor 52 HSR 9296 8960 9197 Cool
```

Gambar 9. Hasil pengujian optimasi pasukan berdasarkan jumlah *attack* tertinggi

Dari gambar tersebut, diketahui jumlah *card cost* yang diperoleh sebesar 273 yang berarti lebih kecil dari pada *unit cost* yang ditetapkan, yaitu sebear 278. Selain itu, algoritma *greedy* yang dirancang berhasil membuktikan bahwa karakter bernama *Hiderigami* memiliki jumlah *attack* tertinggi dibandingkan dengan karakter lain.

B. Pemilihan Pasukan berdasarkan Jumlah Defense *Tertinggi*

Berdasarkan sampel yang digunakan penulis, algoritma *greedy* yang didesain menghasilkan susunan karakter sebagai berikut.

```
Scylla 47 HSR 9567 9689 9205 Passion
Yuki Onna 52 HSR 9280 9424 10140 Dark
Auxo 50 HSR 9353 9353 9733 Light
Astaroth 62 HSR 9856 9352 11104 Dark
Hiderigami 62 HSR 9976 9328 10616 Passion
```

Gambar 10. Hasil pengujian optimasi pasukan berdasarkan jumlah *defense* tertinggi

Dari gambar tersebut, diketahui jumlah *card cost* yang diperoleh juga sebesar 273. Jumlah ini tetap lebih kecil dari pada *unit cost* yang ditetapkan, yaitu sebesar 278. Selain itu, algoritma *greedy* menemukan karakter bernama *Scylla* sebagai karakter yang memiliki jumlah *defense* tertinggi di antara karakter lain yang berada pada sampel yang digunakan.

C. Pemilihan Pasukan berdasarkan Rasio Attack *dengan* Card Cost *Terendah*

Hasil yang didapat penulis adalah sebagai berikut.

```
Muryan 20 HN 1540 2145 1925 Passion
Dryad 22 HN 1920 2280 2339 Light
Unicorn 22 HN 1925 2145 1870 Cool
Bishop 17 N 1500 1950 1600 Light
Military Band 20 HN 1760 1815 1870 Cool
```

Gambar 11. Hasil pengujian optimasi pasukan berdasarkan rasio *attack* dengan *card cost* terendah

Optimasi yang dilakukan menghasilkan karakter-karakter dengan tipe kelangkaan N dan HN karena rasio antara *attack* dan *card cost* rata-rata rendah.

D. Pemilihan Pasukan berdasarkan Rasio Defense *dengan* Card Cost *Terendah*

Hasil yang didapat penulis adalah sebagai berikut.

```
Dark Elf 26 HN 2480 2240 2060 Dark
Doll Master 22 HN 2145 1980 1705 Dark
Military Band 20 HN 1760 1815 1870 Cool
Fencer 19 HN 2145 1760 1485 Cool
Suiko 20 HN 2200 1870 1650 Passion
```

Gambar 12. Hasil pengujian optimasi pasukan berdasarkan rasio *defense* dengan *card cost* yang terendah

Mirip dengan metode optimasi rasio *attack* dengan *card cost* terendah, karakter yang didapat hanya berasal dari tipe kelangkaan HN karena rasio *defense* dengan *card cost* -nya rata-rata kecil.

E. Pemilihan Pasukan berdasarkan Rasio Jumlah Attack *dan* Defense *dengan* Card Cost *Terendah*

Hasil yang didapat penulis adalah sebagai berikut.

```
Super Soldier 22 HR 6783 6783 7380 Light
Brunhild 34 HR 7619 7497 8346 Light
Assistant Loid 34 HR 7463 7205 7872 Cool
Head Clerk 22 HN 5040 4284 4396 Light
Tannenbaum Lancer 34 HR 7337 6956 7746
Light
```

Gambar 13. Hasil pengujian optimasi pasukan berdasarkan rasio jumlah *attack* dan *defense* dengan *card cost* yang terendah

Pada kali ini, hasil dari optimasi yang dilakukan dengan metode ini tidak melibatkan karakter dengan tipe kelangkaan SR. Hal ini disebabkan rasio tipe ini pada karakter dengan tipe kelangkaan SR rata-rata berjumlah besar.

V. Analisis Penerapan

Pembangunan pasukan pada *game* ini memiliki prinsip yang sama dengan *knapsack 0/1 problem*, yaitu mencari keuntungan maksimum yang disertai dengan suatu batasan. Pada *game* ini, *unit cost* setara dengan jumlah bobot maksimal yang dapat ditanggung oleh *knapsack*, *card cost* setara dengan beban per barang yang akan ditampung *knapsack*, dan jumlah *attack* maupun *defense* per karakter setara dengan keuntungan setiap barang. Namun, masih ada kemungkinan hasil yang dicapai belum sepenuhnya optimal, baik dari sisi komputasi maupun dari sisi *gameplay*.

Jika dilihat dari sisi komputasi, algoritma *greedy* hanya memilih kandidat yang merupakan optimum lokal sehingga tidak mencari kandidat lain yang sebenarnya memiliki peluang untuk mencapai solusi akhir yang optimum.

Begitu juga apabila dilihat dari sisi *gameplay*. Penulis menemukan bahwa para pemain *game* ini membangun pasukan mereka berdasarkan kebutuhan khusus. Pemilihan karakter tidak semata-mata melihat jumlah *attack* maupun *defense* saja, melainkan ada beberapa

faktor lain. Salah satunya, peluang sang karakter dalam mengaktifkan *skill*-nya dan juga jenis *skill* yang dimiliki oleh karakter tersebut.

VI. Kesimpulan

Algoritma *greedy* dapat digunakan dalam pembangunan pasukan. Walaupun demikian, hasil yang akan diterima masih belum memenuhi kebutuhan pemain. Algoritma *greedy* yang digunakan tidak memperhitungkan tingkah laku musuh dan *skill* musuh saat pertarungan.

VII. Sanwacana

Penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Masayu Leyla Khodra selaku dosen kelas penulis. Tidak lupa juga kepada Nubee yang telah mencipatakan *game* ini.

Daftar Pustaka

[1] http://nubee.com/2013/03/valkyrie-crusade-android-release, 02/12/2013, 17:23.
[2] Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma",Institut Teknologi Bandung, 2009.

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, 20 Desember 2013

ttd

Rangga Yustian M.
13511017

LAMPIRAN

Berikut ini adalah data yang digunakan dalam pengujian algoritma *greedy*. Format dari data ini adalah <Nama karakter> <*card cost*> <*rarity*> <*attack count*> <*defense count*> <*soldier count*> <elemen>

```
Cu_Chulainn 58 HSR 9810 9236 11368 Passion
Scylla 47 HSR 9567 9689 9205 Passion
Hiderigami 62 HSR 9976 9328 10616 Passion
Omoikane 58 HSR 9072 8736 10920 Cool
Hervor 52 HSR 9296 8960 9197 Cool
Orihime 53 HSR 8624 8960 9449 Cool
Arianrhod 56 HSR 9072 8736 10360 Light
Nicola 56 HSR 9072 8624 10500 Light
Auxo 50 HSR 9353 9353 9733 Light
Lilim 43 HSR 8512 7616 10080 Dark
Astaroth 62 HSR 9856 9352 11104 Dark
Yuki_Onna 52 HSR 9280 9424 10140 Dark
Cybele 44 SR 6320 6480 6840 Passion
Fortuna 39 SR 6400 6400 7200 Passion
Princess 36 SR 5600 5760 5760 Passion
Librarian 34 SR 5440 5920 5040 Cool
Himiko 57 SR 8863 8751 9718 Cool
Cyborg 37 SR 6400 5920 5280 Cool
Konohanasakuya 37 SR 5760 6420 7200 Light
Kikurihime 43 SR 6160 6320 6000 Light
Athena 39 SR 6370 6370 6000 Light
Forneus 44 SR 6560 6160 6930 Dark
Hypnos 43 SR 6560 6320 6660 Dark
Nyx 48 SR 6400 6640 7110 Dark
Sword_Master 34 HR 6901 6664 7560 Passion
Millionaire 32 HR 6664 6552 7280 Passion
Ratatoskr 34 HR 7204 6956 7892 Passion
Assistant_Loid 34 HR 7463 7205 7872 Cool
Kushinadahime 36 HR 6525 7240 7840 Cool
The_Poor 38 HR 6579 6555 6880 Cool
Tannenbaum_Lancer 34 HR 7337 6956 7746 Light
Brunhild 34 HR 7619 7497 8346 Light
Super_Soldier 22 HR 6783 6783 7380 Light
Crusher 42 HR 7751 7269 8010 Dark
Skeleton 34 HR 7453 6590 8038 Dark
Shapeshifter 34 HR 6956 7204 8038 Dark
Roulette 28 R 4930 4760 5400 Passion
Melanippe 28 R 4590 4930 5600 Passion
Gunner 27 R 4760 4590 5400 Passion
Idler 15 R 2720 2720 2800 Cool
Kama 28 R 5100 4750 5300 Cool
Hacker 28 R 5190 5190 4800 Cool
Little_Fox 28 R 4930 5020 5200 Light
Kesalan_Patharan 28 R 4850 5020 5300 Light
New_Girl 29 R 5808 5704 6248 Light
Basilisk 28 R 4760 5270 5100 Dark
Death 27 R 4850 4590 5100 Dark
Ariel_(Strawhat) 28 R 4760 5020 5500 Dark
Suiko 20 HN 2200 1870 1650 Passion
Flower_Girl 20 HN 1870 2035 1815 Passion
Muryan 20 HN 1540 2145 1925 Passion
Fencer 19 HN 2145 1760 1485 Cool
Military_Band 20 HN 1760 1815 1870 Cool
Unicorn 22 HN 1925 2145 1870 Cool
Dryad 22 HN 1920 2280 2339 Light
Pixie 18 HN 1920 1920 1680 Light
Head_Clerk 22 HN 5040 4284 4396 Light
Dark_Elf 26 HN 2480 2240 2060 Dark
Doll_Master 22 HN 2145 1980 1705 Dark
Franken 20 HN 2035 1925 1705 Dark
Adventurer 17 N 1850 1900 1400 Passion
Bugbear 18 N 1950 1800 1700 Passion
Beastmaster 18 N 1950 1850 1700 Passion
Assassin 17 N 2000 1600 1450 Cool
Diver 17 N 1700 2100 1400 Cool
Lacrosse_Master 18 N 1850 1800 1650 Cool
Bishop 17 N 1500 1950 1600 Light
Cleric 15 N 2430 3510 2380 Light
Priest 15 N 1450 1800 1350 Light
Cyclops 18 N 3600 3150 2950 Dark
Homunculus 18 N 1800 1950 1500 Dark
Lamia 17 N 1850 1700 1500 Dark
```