Penggunaan Brute Force dalam Mencari Pergerakan pada Permainan Pokémon Shuffle

Kevin Iswara / 13515085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
keviniswara18@gmail.com

***Abstract*—Pokémon Shuffle adalah permainan puzzle yang menggunakan karakter pokémon sebagai inti dari permainan. Pemain diharuskan untuk mengalahkan musuh dengan menghabiskan *hit point* musuh dengan cara memberikan serangan untuk dapat mencoba menangkapnya. Serangan dihasikan dari mencocokkan tiga atau lebih Pokémon yang ada. Algoritma Brute Force dapat digunakan untuk mencari pergerakan mana yang menghasilkan combo maksimum. Semakin tinggi combo yang dihasilkan, damage juga akan semakin meningkat.**

Keywords—pokémon shuffle; puzzle, brute force; combo

I. PENDAHULUAN

Permainan puzzle merupakan salah satu permainan yang cukup populer untuk semua golongan masyarakat. Walaupun terkadang permainan jenis ini hanya seperti itu saja, beberapa golongan masyarakat sangat menyukainya dan merasa tertantang untuk menyelesaikan permainan puzzle tersebut karena permainan jenis ini memerlukan pemikiran untuk pergerakan puzzle tersebut dan juga strategi-strategi yang sesuai dengan kondisi yang ada sesuai dengan permainan puzzle yang dimainkan. Permainan ini juga memiliki sangat banyak kemungkinan untuk dijalankan. Pergerakan permainan jenis ini juga biasanya bersesuaian dengan pilihan / pergerakan yang dipilih pemain. Dengan kata lain, pilihan / pergerakan yang dipilih pemain ini berpengaruh pada tahap selanjutnya.

Salah satu permainan puzzle yang banyak disukai oleh anak-anak hingga remaja adalah Pokémon Shuffle. Permainan yang dirilis pada 18 Februari 2015 awalnya hanya dapat dimainkan di 3DS. Namun pada tanggal 25 Agustus 2015, versi untuk iOS dan Android yang disebut Pokémon Shuffle Mobile (Jepang: ポケとるスマホ版 PokéToru Smartphone Version) dirilis di Amerika Utara, Inggris dan Australia.[2]

II. DASAR TEORI

A. Algoritma Brute Force

Algoritma Brute Force adalah sebuah pendekatan yang sangat *straightforward* untuk memecahkan sebuah masalah. Biasanya algoritma ini didasarkan kepada *problem statement* dan definisi konsep yang dilibatkan. Algoritma ini memecahkan masalah dengan sangat sederhana, langsung dengan cara yang jelas.

Pada umumnya, algoritma *brute force* ini sangat tidak mangkus dan juga tidak cerdas. Hal tersebut dikarenakan algoritma ini membutuhkan langkah yang besar dalam penyelesaiannya, terutama apabila masalah yang harus dipecahkan adalah masalah yang masukkannya berukuran besar, sehingga harus dicoba satu per satu dan memakan banyak waktu. Algoritma ini paling tidak rumit untuk diimplementasikan, sehingga untuk permasalahan yang berukuran kecil, algoritma ini sering dipilih. Algoritma ini biasanya dijadikan basis untuk dibandingkan dengan algoritma lainnya (untuk mencari yang lebih mangkus). Walaupun algoritma ini tidak mangkus, hamper semua masalah yang ada dapat diselesaikan dengan algoritma *brute force* ini, bahkan ada malasah-masalah yang hanya dapat diselesaikan dengan algoritma *brute force*.[1]

Seperti algoritma pada umumnya, alogirtma ini memiliki kelebihan dan kekurangan sebagai berikut :

Kelebihan
1. Dapat digunakan untuk memecahkan hamper sebagian besar masalah.
2. Sangat sederhana sehingga mudah dimengerti.
3. Mudah untuk diimplementasi.

Kekurangan
1. Kurang mangkus.
2. Waktu eksekusi yang sangat lama sehingga tidak dapat diterima.

B. Pokémon Shuffle

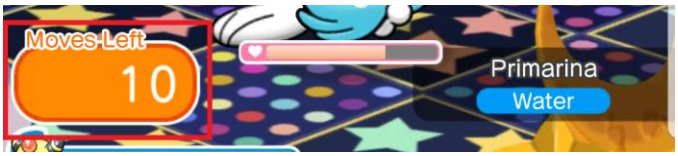
Pokémon Shuffle adalah permainan yang dapat didownload dan dimainkan secara *free-to-play* walaupun didalamnya tetap ada batasan-batasannya. Permainan ini memaksimalkan banyak *stage* permainan yang dapat dimainkan secara berurutan dengan cara memksimalkan jumlah *heart* yang ada pada game. Pada saat pertama kali mulai jumlah *heart* yang ada adalah lima, dan untuk setiap *stage* permainan akan menghabiskan satu buah *heart*. *Heart* ini akan bertambah sejumlah satu untuk setiap tiga puluh menit.

Permainan puzzle pada Pokemon Shuffle adalah dengan mencocokkan tiga atau lebih pokemon yang sama dalam enam kotak yang ada dalam Area Puzzle yang berukuran 6x6 yang telah disediakan untuk mengalahkan pokemon liar baik yang sudah pernah ditangkap maupun yang belum ditangkap. Cara untuk mencocokkan tiga atau lebih pokemon adalah dengan memindahkan dua posisi pokemon yang berbeda.[3]

a.

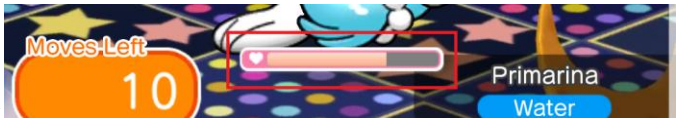
Fig. 1. Contoh Tampilan Pokemon Shuffle

Sesuai dengan semboyan utama dari game ini, "*Gotta Catch'em All"*, tujuan utama dari game ini adalah untuk menangkap semua Pokemon yang ada. Untuk mencoba menangkap pokemon, pemain harus mengalahkan pokemon liar tersebut. Untuk mengalahkan pokemon liar itu, pemain harus memberikan *damage* atau kerusakan yang sama atau lebih besar dari *Hit Out Pokemon's Hit Point* (HP) yang harus dilakukan dalam jumlah pergerakan yang terbatas ataupun waktu yang terbatas. Batasan ini diberikan dari permainan itu sendiri tergantung pada pokemon liar yang dilawan. Jika berhasil menangkap suatu pokemon liar, maka pemain dapat menggunakan pokemon liar tersebut untuk membantunya dalam pertarungan selanjutnya.



b.

Fig. 2. Pergerakan yang tersisa



c.

Fig. 3. HP Bar Pokemon Liar

Beberapa hal penting yang harus diperhatikan dalam permainan ini adalah sebagai berikut :

1. Penangkapan pokemon

   Penangkapan pokemon akan terjadi ketika pokemon liar yang belum pernah ditangkap oleh pemain dikalahkan. Penangkapan pokemon ini bergantung pada persentase keberhasilan menangkap nya. Ada dua komponen penting dalam keberhasilan penangkapan. Komponen tersebut adalah persentase dasar penangkapan yang tidak akan berubah dan persentase bonus penangkapan yang akan meningkat sesuai dengan pergerakan yang tersisa maupun waktu yang tersisa.

   Untuk meningkatkan penangkapan pokemon, pemain dapat membeli barang "Moves+5" ataupun "Time+10" yang disediakan sebelum *stage* tersebut dimulai. Penangkapan pertama ini, dilakukan dengan Poke Ball. Jika penangkapan pertama ini gagal, pemain dapat menggunakan Great Ball dengan membeli seharga 2500 koin pada 3DS dan 3500 koin pada *mobile*. Great Ball ini akan meningkatkan persentase keberhasilan sebanyak dia kali lipat. Selain itu, saat gagal menggunakan Poke Ball, akan ada kemungkinan untuk men-trigger Super Catch Power yang menambah persentase penangkapan sebesar 20% hingga 50%.[2]

2. Kekuatan Pokemon

   Seperti yang sudah disebutkan sebelumnya, setiap pokemon yang telah ditangkap, dapat digunakan untuk melawan pokemon liar lainnya. Semua pokemon tersebut, memiliki kekuatan yang berbeda-beda. Besar kekuatan ini berkisar antara 30 hingga 120. Besar dari kekuatan Pokemon ini meningkat sesuai dengan level dari pokemon tersebut. Normalnya semua pokemon memiliki maksimal level 10, tetapi beberapa pokemon dengan menggunakan barang tertentu dapat meningkatkan maksimal levelnya.

   Level pokemon dapat ditingkatkan dengan mendapatkan *experience* atau yang biasa disebut exp yang dapat diperoleh dengan mengalahkan pokemon liar di *main stage*, *special stage* ataupun *expert stage* dan dengan menggunakan barang. Exp pada setiap *stage* di *main stage* berbeda-beda sesuai dengan jumlah maksimal pergerakannya sedangkan pada *special stage* dan *expert stage* sudah ditentukan exp nya yaitu 10 dan 5. Jumlah exp tersebut hanya akan didapatkan sepenuhnya jika pemain dapat mengalahkan pokemon liar yang dilawannya. Jika tidak, maka perolehan exp akan disesuaikan dengan HP pokemon liar yang tersisa. [2]

3. *Damage* dari Pokemon

Damage atau kerusakan yang diberikan oleh pokemon bergantung pada beberapa hal, seperti Kekuatan dari pokemon itu sendiri, banyak kecocokan, combo, ke-efektif-an tipe pokemon, skill dan status kondisi musuh.

Unsur utama dari kerusakan ditentukan oleh kekuaran pokemon dan banyak kecocokan. Kekuatan pokemon adalah yang sudah disebutkan pada point 2 sebelumnya sedangkan banyak kecocokan adalah jumlah kecocokan yang pemain lakukan.

TABLE I. TABEL KERUSAKAN TERHADAP JUMLAH KECOCOKAN

Jumlah Kecocokan	Kerusakan
1	Kerusakan dasar x0.3
2	Kerusakan dasar x0.6
3	Kerusakan dasar x1
4	Kerusakan dasar x1.5
5	Kerusakan dasar x2
6	Kerusakan dasar x3

Fig. 4. Tabel Kerusakan Jumlah Kecocokan

Banyaknya combo yang sedang berlangsung jika menambahkan kerusakan yang diberikan. Berikut adalah kerusakan terhadap combo yang berlangsung.

TABLE II. TABEL KERUSAKAN TERHADAP COMBO

Combo	Kerusakan
1	Kerusakan x1
2-4	Kerusakan x1.1
5-9	Kerusakan x1.15
10-24	Kerusakan x1.2
25-49	Kerusakan x1.3
50-74	Kerusakan x1.4
75-99	Kerusakan x1.5
100-199	Kerusakan x2
200+	Kerusakan x2.5

Fig. 5. Tabel Kerusakan Terhadap Combo

d.

Fig. 6. Contoh Combo saat Pertarungan Berlangsung

Ke-efektif-an dari tipe pokemon juga terbagi tiga kategori yang akan dibahas pada point 4. Kerusakann ditentukan sesuai dengan kategori nya. Selain itu, pada permainan ini, setiap pokemon memiliki skill khusus yang akan aktif ketika dilakukan pencocokan. Ada beberapa skill pada pokemon yang meningkatkan kerusakan yang akan diberikan. Selain itu ada juga 6 status kondisi yang meningkatkan kekuatan serang, yaitu : *burn* (kerusakanx1.5 jika diserang dengan tipe api); *ghost* (kerusakanx1.5 jika diserang dengan tipe ghost); *poisoned* (kerusakanx1.5 jika diserang dengan tipe poison); *frozen* (kerusakanx1.2 jika diserang dengan tipe es);*asleep* (kerusakanx1.5 jika diserang).[2]

4. Tipe Pokemon

Ada 18 tipe pokemon yang ada pada permainan ini yaitu normal, fight, flying, poison, ground, rock, bug, ghost, steel,,fire, water, grass, electric, psychic, ice, dragon, dark, dan fairy. Sesuai dengan tipe nya, kerusakan yang diberikan dapat berubah. Pembagian kekuatan serang berdasarkan tipe pokemon ada 3, yaitu *super effective* (kerusakan x2), *effective* (kerusakan x1.5) dan *not very effective* (kerusakan x0.5).[2]

e.

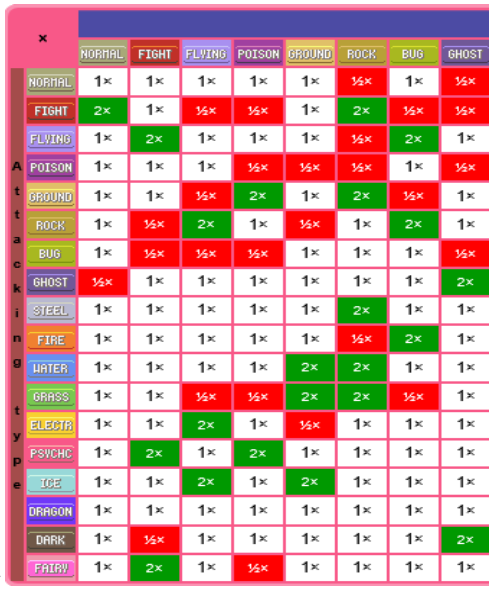


Fig. 7. Contoh Peningkatan atau Pengurangan Kerusakan Berdasarkan Tipe Pokemon[2]

Pada gambar di atas, kolom adalah tipe pokemon yang sedang dilawan, sedangkan baris adalah tipe pokemon yang memberikan serangan.

III. METODE PENGGUNAAN ALGORITMA BRUTE FORCE DALAM MENCARI PERGERAKAN PADA POKEMON SHUFFLE

A. *Permasalahan*

Tujuan dari permainan Pokemon Shuffle adalah untuk memperoleh semua pokemon yang ada. Dalam mencapai hal tersebut, pemain harus mengalahkan musuh terlebih dahulu

sebelum mencoba untuk menangkap. Sesuai yang sudah disebutkan di atas, persentase keberhasilan dalam penangkapan diperngaruhi oleh jumlah move yang tersisa saat HP dari pokemon liar habis. Semakin banyak jumlah move yang tersisa, semakin besar juga persentase penangkapan.

Salah satu cara dalam mencapai hal tersebut adalah dengan meningkatkan minimal combo yang terjadi saat pergerakan karena semakin besar combo yang terjadi semakin besar juga kerusakan yang diberikan yang mengakibatkan pokemon liar cepat kalah sehingga jumlah pergerakan yang tersisa lebih banyak.

f.

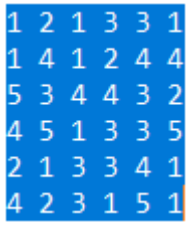
Fig. 8. Contoh Area Puzzle

Masalah yang akan penulis bahas akan dicontohkan dengan Fig. 8. Dalam area puzzle tersebut terdapat 6z6 petak dengan 5 jenis pokemon. Agar dapat mengalahkan pokemon liar dengan sisa pergerakan maksimal, pemain harus memilih 2 petak pokemon yang akan ditukar posisinya sehingga menghasilkan combo yang paling banyak.

B. Ide Penyelesaian

Dalam menyelesaikan masalah ini, penulis memilih algoritma *brute force* untuk menemukan pergerakan dengan combo yang paling banyak (hanya menghitung yang ada di area sekarang, tidak dengan kemungkinan-kemungkinan pokemon-pokemon lain yang akan muncul selanjutnya). Nilai yang akan dioptimalisasi pada kasus ini adalah jumlah kecocokan yang akan terjadi jika suatu pergerakan (pertukaran 2 petak) dilakukan. Setelah pergerakan dilakukan, pokemon yang ada di atasnya akan jatuh ke bawah.

Langkah pertama dari penyelesaian masalah ini adalah dengan mengubah semua data-data pokemon tersebut menjadi sebuah matriks. Setiap pokemon dilambangkan dengan sebuah angka atau simbol sehingga dapat dibedakan dengan pokemon lainnya. Jika tidak ada pokemon pada koordinat tersebut, maka akan dilambangkan dengan suatu simbol, dalam kasus kali ini, penulis melambangkan pokemon dengan angka sedangkan koordinat yang kosong dilambangkan dengan '-'. Berikut adalah contoh translasi Fig. 8. menjadi sebuah matriks.



g.

Fig. 9. Matriks dari hasil Fig. 8.

Dari gambar diatas, angka 1 melambangkan Zapdos (pokemon kuning banyak lancip-lancip), angka 2 melambangkan Sceptile (pokemon hijau), angka 3 melambangkan Ampharos (pokemon kuning bulat)), angka 4 melambangkan Raikou (pokemon putih kumis biru) dan), angka melambangkan Primarina (pokemon putih dengan moncong pink).

Langkah selanjutnya adalah bagian *brute force* nya, yaitu memeriksa nilai untuk setiap petak jika ditukar dengan petak lainnya (jumlah kecocokan yang akan terjadi).

h.

Fig. 10. Contoh Pertukaran 1

i.

Fig. 11. Contoh Pertukaran 2

Pada Fig. 10. Pergerakan dilakukan dari posisi 2,2 ke posisi 2,4. Dengan perhitungan nilai sesuai dengan jumlah kecocokan yang ada, nilai dari pergerakan ini adalah 1 karena hanya ada satu kecocokan yang terbentuk saat pergerakan itu dilakukan.

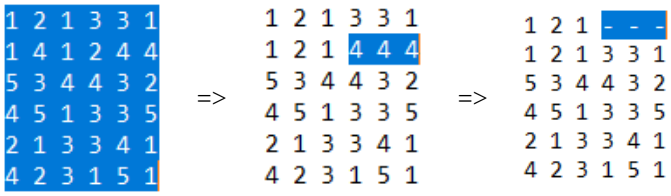


Fig. 12. Matriks pertukaran 1

Pada Fig. 11. Pergerakan dilakukan dari posisi 2,5 ke posisi 3,5. Dengan perhitungan nilai sesuai dengan jumlah kecocokan yang ada, nilai dari pergerakan ini adalah 3.

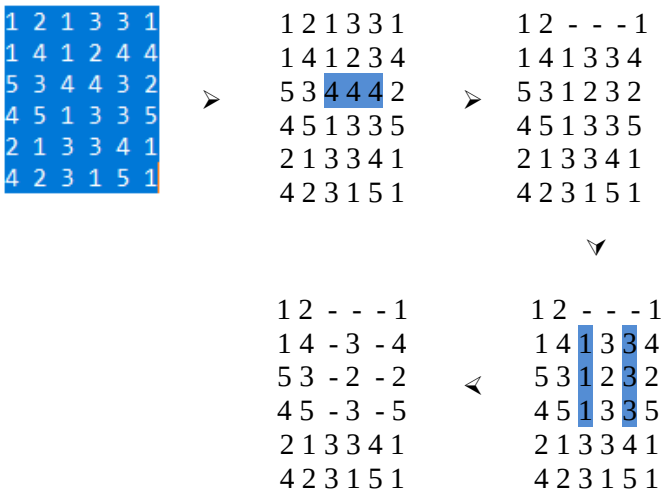


Fig. 13. Matriks pertukaran 2

Ketika sudah ada 1 jawaban, nilai tersebut dijadikan nilai tertinggi sementara. Jika ada nilai yang lebih tinggi, maka pertukaran yang disimpan digantikan dengan pertukaran yang nilainya lebih tinggi. Hal ini dilakukan hingga semua pertukaran sudah dilakukan yaitu sebanyak 36!. Perhitungan nilai dilakukan secara rekursif yaitu menghitung yang sekarang, lalu menghitung kembali hasil perubahan jika masih ada yang kecocokan yang terjadi. Jika tidak, maka perhitungan pun dihentikan.

C. Implementasi (Pseudo Code)

Dalam pengimplementasian algoritma brute force ini, ada beberapa fungsi dan prosedur yang harus diimplementasikan, yaitu :

- Fungsi Hitung yang menerima masukan matriks yang melambangkan area puzzle, koordinat awal dan koordinat akhir perpindahan. Fungsi ini akan menukar isi kedua koordinat masukan lalu menghitung berapa banyak kecocokan yang akan terjadi jika melakukan perpindahkan tersebut dengan cara memeriksa setiap posisi apakah ada kecocokan. Jika ada kecocokan maka pokemon-pokemon yang bersangkutan akan ditandai sehingga tidak akan di periksa kembali (tetapi jika dia terkena 2 kali kecocokan, pokemon ini tidak akan di periksa, tapi pokemon lain yang menjadi kecocokannya akan diperiksa). Setiap menemukan kecocokan, maka jumlah kecocokan akan bertambah. Jika ada kecocokan, maka pokemon yang ditandai akan “dihapus” dari matriks dan pokemon pun diturunkan. Fungsi ini akan mengembalikan jumlah kecocokan yang terjadi (jumlah combo yang terbentuk).

```
int Hitung(Matriks M, Point p1, Point p2) {
   int temp = M[p1.x][p1.y];
   M[p1.x][p1.y] = M[p2.x][p2.y];
   M[p2.x][p2.y] = temp;
   int jumlah = 0;
   for (int i = 1; i <= 6; i++) {
      for (int j = 1; j <= 6; j++) {
         if(M[i][j] belum ditandai) {
            if(ada kecocokan dengan minimal 3) {
               tandai pokemon;
               jumlah++;
            }
         }
      }
   }
   if(jumlah != 0) {
      HapusPokemon(M);
      TurunkanPokemon(M);
      jumlah += Hitung(M);
   }
   return jumlah;
}
```

Fig. 14. Fungsi Hitung

- Prosedur HapusPokemon yang menerima masukan matriks yang melambangkan area puzzle dan memeriksa semua koordinat yang ada. Jika isi matriks pada koordinat itu sudah ditandai dengan sebuah tanda, maka matriks pada koordinat tersebut akan diisikan dengan sebuah nilai yang melambangkan bahwa pada koordinat tersebut tidak ada isinya.

```
void HapusPokemon (Matriks& M) {
  for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= 6; j++) {
      if (M[i][j] ditandai) {
        M[i][j] dikosongkan;
      }
    }
  }
}
```

Fig. 15. Prosedur HapusPokemon

- Prosedur TurunkanPokemon yang menerima masukan matriks yang melambangkan area puzzle dan memeriksa ada tidaknya isi matriks yang tidak ada isinya pada setiap kolom. Jika pada kolom tersebut ada nilai yang kosong, maka matriks tidak berubah. Tetapi, jika ditemukan nilai yang kosong pada kolom tersebut, maka pokemon akan ditata ulang sedemikian rupa sehingga tidak ada lagi nilai kosong.

```
void TurunkanPokemon (Matriks& M) {
  for(int i = 1; i <= 6; i++) {
    boolean ada = false;
    int j = 1;
    int perpindahan = 0;
    while ((j <= 6) && (!ada)) {
      if (M[i][j] kosong) {
        ada = true;
        perpindahan = j;
      }
      j++;
    }
    if (ada) {
      turunkan pokemon di atas perpindahan;
    }
  }
}
```

Fig. 16. Prosedur TurunkanPokemon

- Fungsi CariPergerakan yang menerima masukan matriks yang merupakan area puzzle yang akan dicari pergerakannya. Fungsi ini akan mengecek semua kemungkinan pergerakan yang dapat terjadi (bahkan pergerakan kepada dirinya sendiri yang sudah pasti total pergerakannya tidak ada). Keluaran dari fungsi ini adalah array of point yang hanya berisi 2 point. Point pada indeks perama adalah koordinat awal, point pada indeks kedua adalah koordinat akhr perpindahan.

```
Point[] CariPergerakan(Matriks M) {
  int maks = 0;
  Point * p = new Point[2]
  for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= 6; j++) {
      for (int k = 1; k <= 6; k++) {
        for (int l = 1; l <= 6; l++) {
          Point p1 = (i,j);
          Point p1 = (k,l);
          int temp = HitungPerpindahan(M,p1,p2);
          if (maks < temp) {
            maks = temp;
            p[0] = p1;
            p[1] = p2;
          }
        }
      }
    }
  }
  return p;
}
```

Fig. 17. Fungsi CariPergerakan

D. Kekurangan dan Kelebihan

Kelebihan dari algoritma ini adalah kepastiannya dalam menemukan pergerakan yang menghasilkan combo yang maksimal karena semua kemungkinan yang ada diperiksa jumlah kecocokan yang terbentuknya (combonya).

Kekurangan terbesar dari algoritma ini adalah waktu eksekusinya karena akan ada 36! kemungkinan yang dapat terjadi sehingga akan ada 36! perhitungan pada fungsi Hitung dan yang lainnya.

Algoritma ini dapat diperbaiki dan dibuat lebih mangkus dengan cara memberikan batasan-batasan seperti membuang perhitungan terhadap posisi sendiri, membuang perhitungan terhadap posisi yang berbeda tetapi mengandung pokemon yang sama, dan juga tidak menghitung dua kali perpindahan yang sama (perpindahan antara koordinat 1 dengan koordinat 2 sama saja dengan perpindahan antara koordinat 2 dengan koordinat 1).

IV. KESIMPULAN

Dengan menggunakan algoritma *brute force*, pergerakan dengan combo terbanyak dapat ditentukan. Walaupun tidak mangkus karena kejadian yang tidak seharusnya dicoba tetap dicoba pada algoritma ini, algoritma ini pasti memberikan hasil dengan minimal combo terbanyak. Jika algoritma ini ditambahkan aturan-aturan lain seperti membatasi pergerakan yang tidak perlu (yang tidak mengubah jumlah kecocokan, yaitu tetap tidak ada kecocokan) maka algoritma ini akan lebih mangkus.

REFERENSI

[1] Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau Munir, Rinaldi, Diktat Kuliah IIF2211 Strategi Algoritma, Institut Teknologi Bandung

[2] http://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_Shuffle

[3] https://id.techinasia.com/review-pokemon-shuffle-mobile

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, 18 Mei 2017

Kevin Iswara
135150