Perbandinganan Penggunaan Algoritma *Greedy* dan Modifikasi Algoritma *Brute Force* pada Permainan *Collapse XXL*

Rahadian Dimas Prayudha - 13509009

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509009@std.stei.itb.ac.id

***Abstract*—Permainan *puzzle* sering menjadi pilihan utama bagi banyak orang untuk mengisi waktu luang. Terdapat berbagai variasi dari permainan dengan tipe *puzzle*. Salah satunya, permainan *Collapse XXL* adalah salah satu jenis permainan yang juga bertipe *puzzle*. Permainan ini menampilkan empat ratus buah bujur sangkar berbagai warna yang tersusun menjadi sebuah bujur sangkar besar berukuran 20x20 bujur sangkar kecil. Permainan ini memiliki dua buah tujuan akhir, yaitu untuk mendapatkan nilai sebesar mungkin dan untuk membuat papan menjadi sekosong mungkin. Cara mendapatkan poin adalah dengan melakukan *clicking* pada sekumpulan bujur sangkar yang saling berdekatan. Semakin banyak bujur sangkar yang saling berdekatan, semakin besar nilai yang akan didapatkan. Semakin besar nilai yang didapatkan, maka sisa bujur sangkar pada papan permainan akan semakin sedikit. Untuk menyelesaikan permainan *Collapse XXL* ini, dapat digunakan algoritma *greedy*. Karena, algoritma *greedy* sudah sangat dikenal untuk menyelesaikan permasalahan optimasi yang merupakan permasalahan utama dari permainan *Collapse XXL* ini. Algoritma *greedy* dalam implementasinya bisa divariasikan sesuai dengan optimasi macam apa yang akan diharapkan oleh penggunanya. Misalnya, dalam permainan ini diinginkan nilai semaksimal mungkin dengan jumlah *clicking* sesedikit mungkin. Dalam makalah ini, akan dibahas secara lebih mendetail mengenai salah satu variasi algoritma *greedy* yang bisa digunakan untuk menyelesaikan masalah optimasi dalam permainan *Collapse XXL* ini.**

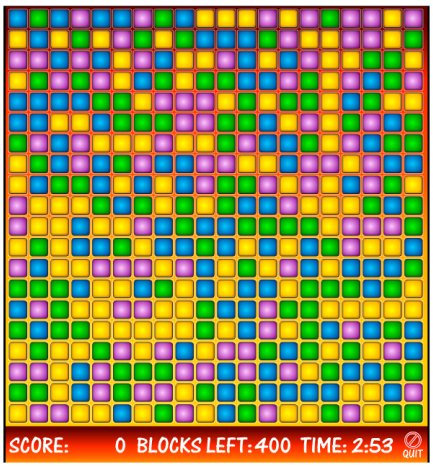
Index Terms*—*Collapse XXL*, algoritma *greedy*, *puzzle*, optimasi, *brute force

I. Introduction

Permainan dengan tipe *puzzle* adalah salah satu jenis permainan yang paling populer di masyarakat. Permainan bertipe ini dinilai cukup menghibur dan melatih ketajaman berpikir. Tidak hanya itu, permainan bertipe *puzzle* juga bisa dimainkan dengan logika sederhana, sehingga pemain tidak perlu kesulitan dalam memahami peraturan dan objektif dari permainan-permainan bertipe *puzzle*.

Salah satu contoh permainan bertipe *puzzle* adalah permainan dengan nama *Collapse XXL*. Permainan ini dapat dimainkan secara *online* melalui *website* http://www.flashrolls.com/puzzle-games/Collapse-XXL-Flash-Game.htm. Permainan ini menampilkan sebuah papan besar yang berbentuk bujur sangkar dan terdiri dari empat ratus buah bujur sangkar kecil beraneka warna yang disusun menjadi 20x20 bujur sangkar kecil. Berikut adalah tampilan halaman awal permainan dan papan awal permainan.

Gambar 1 Halaman Awal Permainan *Collapse XXL*



Gambar 2 Tampilan Papan Permainan Awal

Permainan ini memiliki dua objektif, yaitu mendapatkan nilai sebesar mungkin dalam waktu tiga menit dan menyisakan sesedikit mungkin bujur sangkar pada papan permainan. Cara untuk mendapatkan nilai yang besar adalah dengan melakukan *clicking* pada sekelompok bujur sangkar kecil yang satu warna dan saling menempel atau saling berdekatan. Semakin banyak bujur sangkar kecil yang berdekatan, maka semakin besar nilai yang akan didapatkan. Rumus untuk nilai yang didapatkan adalah:

$$nilai = x^2$$

dengan x adalah jumlah bujur sangkar kecil sewarna yang di-*click*. Jumlah minimal bujur sangkar yang berdekatan yang bisa di-*click* adalah dua buah. Sedangkan, apabila bujur sangkar yang di-*click* tidak membentuk kumpulan bujur sangkar sewarna, maka nilai akan dikurangi sebesar lima poin.

Permainan akan berhenti apabila waktu permainan sebesar tiga menit telah habis atau apabila sudah tidak ada bujur sangkar yang berkelompok.

II. Dasar Teori

Algoritma *Greedy* merupakan metode yang paling populer untuk memecahkan persolan optimasi. Terdapat dua jenis perosalan optimasi, yaitu maksimasi dan minimasi.

Algoritma *greedy* adalah algoritma yang memecahkan masalah langkah per langkah dengan pada setiap langkah dilakukan hal berikut :

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now*!")
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Persoalan optimasi untuk suatu masalah dalam konteks algoritma *greedy* disusun oleh elemen-elemen sebagai berikut:

1. **Himpunan Kandidat (*C*)**
   Himpunan ini berisi elemen-elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya

2. **Himpunan Solusi (*S*)**
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi adalah himpunan bagian dari himpunan kandidat.

3. **Fungsi Seleksi**
   Fungsi ini dinyatakan dengan predikat seleksi. Merupakan dungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. **Fungsi Kelayakan**
   Fungsi ini dinyatakan dengan predikat layak. Fungsi kelayakan ini merupakan fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (constraints) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. **Fungsi Objektif**
   Fungsi objektif ini merupakan sebuah fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Berikut ini adalah skema umum 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 ∪ {x}) then
          S ← S ∪ {x}
      endif
    endwhile
    {SOLUSI(S) or C = {} }

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

Gambar 3* Skema Umum algoritma *greedy

Namun solusi optimum global yang diperoleh pada algoritma *Greedy* ini belum tentu merupakan solusi optimal, tetapi merupakan *sub-optimum* atau *pseudo-optimum*. Hal ini dikarenakan algoritma greedy tidak melakukan operasi secara menyeluruh pada semua alternatif solusi. Selain itu juga ada beberapa kasus yang mengakibatkan fungsi seleksi tidak bekerja dengan optimal.

III. Deskripsi Masalah

Dalam pencarian solusi untuk penyelesaian permainan *Collapse XXL* ini, tujuannya adalah untuk mendapatkan nilai sebesar mungkin yang bisa didapatkan. Nilai yang besar akan didapatkan apabila pemain memilih untuk memecahkan kelompok bujur sangkar yang berdekatan dengan warna yang sama dalam jumlah yang besar pula. Oleh karena itu, pemain harus selalu berusaha untuk memilih kelompok bujur sangkar sewarna dengan jumlah yang paling besar.

Jadi pada setiap kesempatan untuk melakukan *clicking* pada kelompok bujur sangkar sewarna tersebut, pemain

harus menemukan kelompok bujur sangkar sewarna yang paling besar dari seluruh kelompok bujur sangkar sewarna yang ada pada papan permainan. Program atau pencarian solusi ini berakhir apabila tidak ada lagi kelompok bujur sangkar sewarna yang memiliki posisi yang berdekatan, atau dengan kata lain tidak ada lagi kelompok bujur sangkar sewarna dengan jumlah bola pada rangkaian yang lebih dari satu. Kemudian permaian juga akan selesai dan nilai yang didapat sebelumnya akan diakumulasikan menjadi nilai akhir.

Alternatif pemecahan masalah lain adalah dengan menggunakan modifikasi dari algoritma *brute force*. Pertama, program akan menghitung masing-masing jumlah bujur sangkar berwarna tertentu. Dari hasil tersebut kemudian ditentukan bujur sangkar berwarna tertentu yang jumlahnya paling banyak. Kemudian, algoritma *brute force* akan melakukan penelusuran satu per satu terhadap isi papan permainan. Apabila terdapat kelompok bujur sangkar sewarna, maka kelompok tersebut akan dipilih. Pemilohan dilakukan dengan kondisi bahwa warna dari kelompok bujur sangkar sewarna tersebut bukan warna terbanyak yang telah dicatat sebelumnya.

Diharapkan dengan menggunakan alternatif ini, nilai yang didapat bisa lebih besar karena kelompok bujur sangkar sewarna yang jumlahnya lebih banyak akan menghasilkan nilai yang jauh lebih besar.

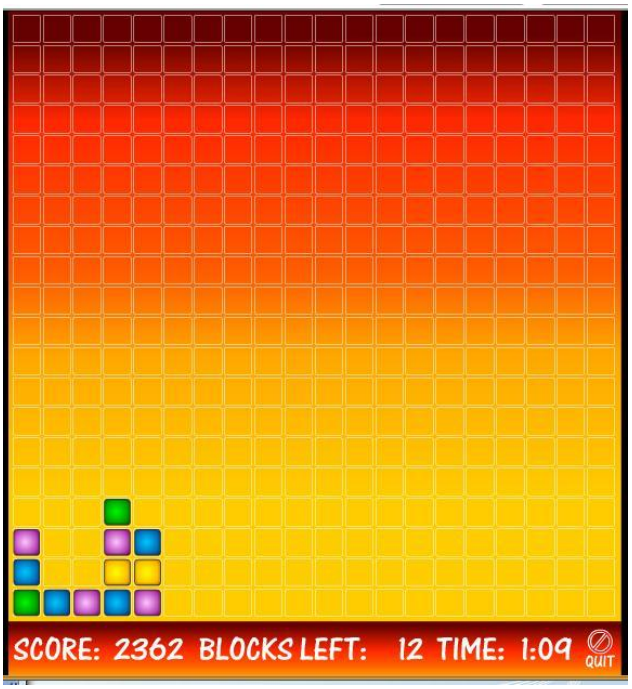
Perlu diingat bahwa apabila kita memilih salah satu bujur sangkar yang membentuk kelompok bujur sangkar sewarna, berarti kita akan memilih kelompok tersebut untuk dihilangkan. Apabila terdapat empat bujur sangkar sewarna yang berdekatan dan misalnya diberi nama bujur sangkar tersebut *a*, *b*, *c*, dan *d*, hasilnya akan sama saja apabila kita memilih bujur sangkar bernama *a*, *b*, *c*, atau *d* tersebut. Bujur sangkar manapun yang kita pilih dari keempat bujur sangkar tersebut akan menyebabkan kita menghilangkan keempat bujur sangkar tersebut dari papan permainan.

IV. IMPLEMENTASI

A.Alternatif Pemecahan I

Sebelumnya telah diuraikan bahwa, untuk memperoleh hasil yang optimal (walaupun untuk beberapa kasus tidak akan memperoleh hasil yang maksimal) dapat digunakan algoritma *greedy* untuk melakukan pemilihan kelompok bujur sangkar sewarna mana yang akan di-*click*.

Cara kerja algoritma *greedy* untuk menyelesaikan permasalahan dalam permainan *Collapse XXL* ini secara optimal adalah dengan pertama–tama melihat keseluruhan bujur sangkar beraneka warna yang ada dalam papan permainan. Kemudian program akan memilih kelompok bujur sangkar yang memiliki warna yang sama yang jumlah bujur sangkar kecilnya paling besar. Kemudian program akan memilih kelompok bujur sangkar sewarna tersebut sehingga seluruh bujur sangkar kecil yang membentuk kelompok akan hilang dari papan permainan dan pemain mendapatkan nilai yang optimal. Kemudian, program akan kembali melakukan hal yang sama yaitu memilih kelompok bujur sangkar yang memiliki warna yang sama dengan rangkaian terbesar di dalam papan permainan yang tersisa. Kemudian menghilangkan kelompok tersebut. Begitu seterusnya dan program akan melakukan hal yang sama sampai tidak ada lagi bujur sangkar yang warnanya sama dalam posisi yang berdekatan.



Gambar 4 Kondisi Papan Permainan Sebelum Kondisi Berhenti

Pada masalah pencarian solusi permainan *Collapse XXL* ini, himpunan kandidat adalah semua bujur sangkar yang terdapat di dalam papan permainan. Implementasi dari algoritma *greedy* ini adalah sebagai berikut :

- Himpunan solusi dalam permainan *Collapse XXL* ini adalah kelompok bujur sangkar yang memiliki warna yang sama dan paling banyak jumlah bujur sangkar kecil penyusunnya.
- Fungsi seleksi dalam permasalahan pencarian solusi permainan *Collapse XXL* ini adalah fungsi yang akan memilih kelompok bujur sangkar sewarna yang tersusun dari bujur sangkar kecil terbanyak di dalam papan permainan.
- Fungsi kelayakan dalam permasalahan pencarian solusi permainan *Collapse XXL* ini adalah fungsi yang memastikan bujur sangkar yang dipilih membentuk kelompok bujur sangkar sewarna yang dibangun dengan lebih dari satu bujur sangkar warna tersebut.
- Fungsi objektif dalam permasalahan pencarian solusi permainan *Collapse XXL* ini bertugas untuk memastikan semua bujur sangkar yang dapat dipilih sudah dipilih. Artinya, sudah tidak ada lagi kelompok bujur sangkar sewarna yang bisa di-*click* dan dihilangkan dari papan permainan.

Dari hasil analisis di atas, maka dapat dirumuskan *pseudocode*-nya sebagai berikut :

```
function samaWarna (input b: board, pos: point) → integer
{menerima input berupa keadaan papan permainan dan koordinat pada papan, kemudian mengembalikan jumlah bujur sangkar yang sama warnanya dan berdekatan posisinya dengan bujur sangkar tersebut. Apabila pada posisi tertentu kosong, maka akan dikembalikan nilai 0.}
```

```
procedure pilih (input b: board, pos: point)
{menerima input keadaan papan dan koordinat tertentu pada papan, kemudian dilakukan pemilihan (clicking) pada kelompok bujur sangkar sewarna yang memuat bujur sangkar pada posisi tersebut. Kemudian, kelompok bujur sangkar tersebut akan dihilangkan dari papan permainan dan poin akan ditambahkan.}
```

```
function isGroup (input b: board, pos: point) → boolean
{menerima input keadaan papan dan koordinat tertentu pada papan, kemudian melakukan pengecakan pada isi dari koordinat tersebut apakah terbentuk kelompok bujur sangkar sewarna atau tidak. Nilai true dikembalikan jika pada posisi tersebut terdapat kelompok bujur sangkar sewarna, sebaliknya nilai false dikembalikan jika pada posisi tersebut tidak terdapat kelompok bujur sangkar sewarna.}
```

```
function isEnded (input b: board) → boolean
{menerima masukan berupa keadaan papan permainan dan seluruh isinya, kemudian nilai true akan dikembalikan oleh prosedur ini apabila sudah tidak ada kelompok bujur sangkar sewarna (tidak ada bujur sangkar sewarna yang berdekatan lagi), sebaliknya nilai false akan dikembalikan jika masih ada kelompok bujur sangkar sewarna yang masih bisa dipilih.}
```

Fungsi seleksi:

```
function mostGroup (input b: board) → point
{menerima input keadaan papan permainan dan akan mengembalikan koordinat tertentu pada papan permainan yang berisi kelompok bujur sangkar sewarna yang jumlah bujur sangkar penyusunnya paling banyak.}

KAMUS
maks: integer {jumlah maksimum bujur sangkar penyusun kelompok bujur sangkar sewarna}
x: integer
y: integer
pos: point

ALGORITMA
for (x = 1 to 20)
 for (y = 1 to 20)
   pos.x = x
   pos.y = y
   if (maks < samaWarna(b,pos)) then
     maks = samaWarna(b,pos)
   endif
 endfor
endfor
```

```
procedure greedy (input b: board)
{menerima input keadaan papan permainan, kemudian akan dilakukan pemilihan (clicking) terhadap seluruh kelompok bujur sangkar sewarna yang bisa dipilih dimulai dari kelompok bujur sangkar sewarna dengan jumlah bujur sangkar penyusun terbanyak}

KAMUS
pos: point

ALGORITMA
while (not isEnded(b)) do
    pos = mostGropup(b)
    if (isGroup(b, pos)) then
      pilih (b,pos)
    endif
endwhile
```

B. Alternatif Pemecahan II

Untuk alternatif pemecahan II, digunakan algoritma *brute force* yang dimodifikasi dengan pemberian syarat atau pengecekan pada kondisi bujur sangkar pada papan. Untuk alternatif pemecahan II berikut ini definisi fungsinya:

```
function maksimum (input a, b, c, d, e: integer) → integer
{menerima input lima buah integer dan mengembalikan nilai integer paling besar.}
```

```
function mostColor (input b: board) → integer
{menerima input keadaan papan permainan dan akan mengembalikan warna yang jumlah bujur sangkar dengan warna tersebut paling banyak. Warna direpresentasikan dengan integer.}

KAMUS
warna1: integer
warna2: integer
warna3: integer
warna4: integer
warna5: integer
warna: integer
pos: pointc {berisi koordinat x, y, dan warna pada posisi tersebut}

ALGORITMA
for (x = 1 to 20)
 for (y = 1 to 20)
   if (pos[x][y].color = 1) then
     warna1 = warna1+1
   else if (pos[x][y].color = 2) then
     warna2 = warna2+1
   else if (pos[x][y].color = 3) then
     warna3 = warna3+1
   else if (pos[x][y].color = 4) then
     warna4 = warna4+1
   else if (pos[x][y].color = 5) then
     warna5 = warna5+1
   endif
 endfor
endfor
warna = maksimum (warna1, warna2, warna3, warna4, warna5)
```

```
procedure BF (input b: board)
{menerima input keadaan papan permainan, kemudian akan dilakukan pemilihan (clicking) terhadap seluruh kelompok bujur sangkar sewarna yang bisa dipilih dimulai dari kelompok bujur sangkar sewarna dengan jumlah bujur sangkar penyusun terbanyak}

KAMUS
pos: point

ALGORITMA
while (not isEnded(b)) do
 for (x = 1 to 20)
   for (y = 1 to 20)
    if (pos[x][y].color != mostColor(b)) then
     pilih(b,pos)
    endif
   endfor
  endfor
endwhile
```

V. Analisis Solusi

A. Alternatif Pemecahan I

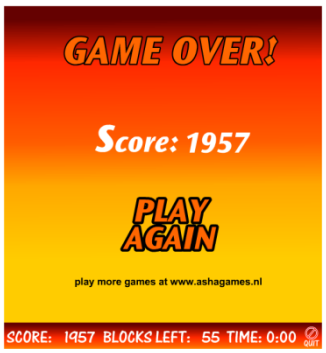
Dalam algoritma *greedy* yang digunakan untuk penyelesaian masalah permainan *Collapse XXL*, terdapat dua fungsi utama, yaitu fungsi mostGroup dan prosedur greedy.

Fungsi mostGroup digunakan untuk melakukan pencarian kelompok bujur sangkar dengan warna sama dengan jumlah bujur sangkar penyusun paling banyak. Fungsi ini membantu fungsi greedy untuk memudahkan pemecahan masalah.

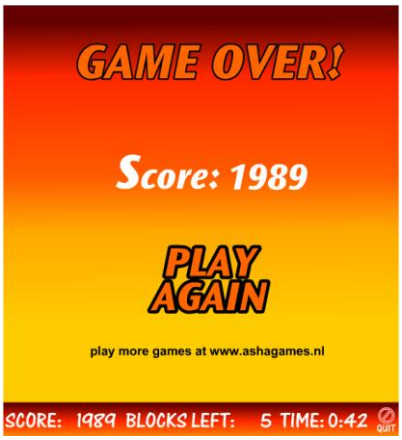
Fungsi greedy berguna untuk menyelesaikan permasalahan pada permainan *Collapse XXL* ini secara keseluruhan, yaitu memanggil fungsi mostGroup untuk mendapatkan kelompok bujur sangkar dengan warna sama yang paling banyak bujur sangkar penyusunnya, dan kemudian menghilangkan kelompok bujur sangkar yang sama warnanya tersebut. Tujuan dari fungsi ini adalah tujuan algoritma *greedy* secara keseluruhan, yaitu mendapatkan nilai yang optimal.

Salah satu tes uji untuk implementasi algoritma *greedy* dalam penyelesaian masalah permainan *Collapse XXL* ditunjukkan pada gambar 5 dan gambar 6.

Pada gambar 5 ditunjukkan hasil dari permainan yang dimainkan dengan cara normal, yaitu dengan menelusuri dari baris paling atas ke baris di bawahnya. Sedangkan pada gambar 6 ditunjukkan hasil permainan yang dimainkan dengan menggunakan algoritma *greedy* sebagaimana telah diuraikan di dalam bagian implementasi di atas. Untuk kasus permainan ini, algoritma *greedy* memberikan hasil yang lebih optimal. Buktinya, dengan nilai yang lebih besar daripada saat memainkan permainan dengan menggunakan cara normal.



Gambar 5 Hasil Permainan dengan Menggunakan Cara Normal



Gambar 6 Hasil Permainan dengan Menggunakan Algoritma *Greedy* Alternatif I

Nilai untuk permainan normal adalah 1957, sedangkan nilai untuk permainan yang diselesaikan dengan algoritma *greedy* adalah 1989. Memang nilai hasil permainan dengan menggunakan kedua metode tidak berbeda jauh. Namun, dapat dilihat dari jumlah bujur sangkar sisa (*blocks left*) untuk metode penyelesaian biasa adalah 55 buah, sedangkan untuk algoritma *greedy* adalah 5 buah. Hal ini membuktikan bahwa hasil dari algoritma *greedy* lebih optimal dalam kasus ini.

B. Alternatif Pemecahan II

Nilai untuk permainan normal adalah 1957, sedangkan nilai untuk permainan yang diselesaikan dengan algoritma alternatif II adalah 10875. Dapat dilihat perbedaan nilai yang begitu jauh antara hasil permainan normal, permainan dengan alternatif I, dan permainan dengan alternatif II. Permainan dengan alternatif II menghasilkan nilai yang jauh lebih besar dibandingkan dengan penggunaan metode lainnya.

Gambar 7 Hasil Permainan dengan Menggunakan Algoritma Alternatif II

Hal ini membuktikan bahwa optimalisasi yang dilakukan algoritma *greedy* tidak selalu menghasilkan solusi optimal global, tetapi selalu optimal lokal. Pada permainan *Collapse XXL* ini, penggunaan algoritma pemecahan masalah II yang memanfaatkan modifikasi dari algoritma *brute force* ternyata memberikan hasil yang lebih optimal.

VI. KESIMPULAN

Algoritma *greedy* pada alternatif pemecahan I dapat digunakan untuk mencari solusi dalam permainan *Collapse XXL*. Solusi yang diberikan melalui algoritma *greedy* dapat dikatakan cukup optimal dan dapat menghasilkan poin yang cukup besar dan menyisakan bujur sangkar dalam jumlah yang lebih sedikit.

Akan tetapi, masih terdapat cara lain yang dapat digunakan untuk menyelesaikan permainan *Collapse XXL* dengan nilai yang lebih besar, yaitu dengan menggunakan algoritma modifikasi dari algoritma *brute force* yang menghasilkan nilai yang lebih besar.

REFERENSI

Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Informatika Sekolah Teknik Elektro dan Informatika ITB, 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, 4 Desember 2011

ttd

Rahadian Dimas Prayudha
NIM. 13509009