Penyelesaian Permainan '*2048*' dengan Menggunakan Algoritma *Greedy*

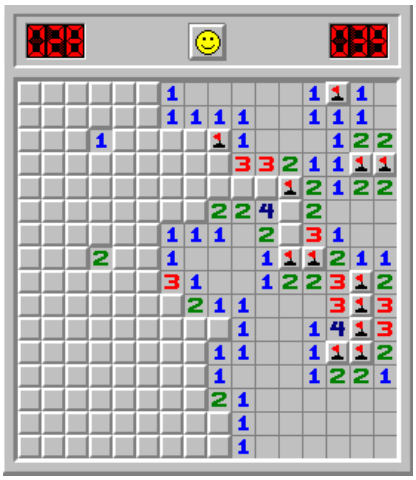
Sinaga Yoko Christoffel Triandi 13516052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516052@std.stei.itb.ac.id

***Abstrak* — Makalah ini akan menjelaskan penggunaan algoritma dalam menyelesaikan permainan '*2048*.' Algoritma yang digunakan berupa Algoritma *Greedy*. Algoritma ini merupakan algoritma yang tergolong *simple*, sehingga algoritma ini sangat sering digunakan. Algoritma *Greedy* merupakan algoritma yang memicu untuk mengambil keputusan terbaik di saat itu juga tanpa memikirkan konsekuensi atau bagaimana hasil ke depannya.**

***Keywords* — 2048, *Game*, *Greedy*, *Permainan*.**

I. PENDAHULUAN

Di zaman '*Millennial*' ini, sudah sangat banyak permainan yang beredar dalam dunia ini. Permainan-permainan yang ada dapat berupa permainan fisik, seperti petak umpet, anjing dan kucing, dan lari estafet, permainan menggunakan objek seperti congklak, lompat tali, dan ular tangga, maupun permainan dalam layar *gadget*, seperti *Minesweeper*, *Dota*, *Clash of Clans*, dan *Mobile Legends*. Permainan-permainan ini sudah menjadi suatu hiburan bagi semua kalangan, baik anak-anak maupun orang dewasa, juga laki-laki maupun perempuan. Tujuan sebuah permainan diciptakan adalah sebagai sarana hiburan untuk orang-orang di sela waktunya, juga bisa untuk mengasaha otak kita. Suatu permainan dapat dimainkan baik sendiri, maupun berkelompok, bergantung bagaimana cara kerja permainan tersebut.



Gambar 1.1. Permainan *Minesweeper*
(Sumber : https://exceptionnotfound.net/solving-minesweeper-with-c-sharp-and-linq/)

Salah satu permainan menggunakan layar *gadget* yang dapat mengasah otak adalah permainan '*2048*.' Permainan ini merupakan permainan yang cukup *simple* karena hanya perlu dimainkan oleh satu pemain, dan cara memainkannya hanya dengan menggeser sebuah jari pada layar. Permainan ini merupakan permainan yang ada dalam *smart phone* dengan berbagai macam *OS* seperti *Windows*, *Android*, dan *iOS*. Tak hanya dalam *smart phone*, permainan ini juga bisa dimainkan dalam komputer atau laptop secara *online*, yaitu dengan mengakses *website* http://2048game.com/.

Begitu banyak cara maupun metode yang digunakan dalam memenangkan permainan ini. Ada cara yang mengutamakan keamanan dalam bermain (cara agar tidak cepat kalah), ada juga cara yang mengutamakan kecepatan dalam memenangkan permainan tersebut. Dari berbagai cara maupun metode untuk memenangkan permainan ini, salah satu cara yang dapat digunakan adalah dengan algoritma *Greedy*. Dalam permainan ini, algoritma *Greedy* dapat diimplementasikan dengan selalu menggeser ke arah kotak yang dapat bergabung membentuk angka terbesar, sedangkan jika tidak ada yang bisa digabung, akan melakukan pergeseran asal. Cara tersebut akan terus dilakukan hingga permainan berhenti.

Dengan menggunakan algoritma *Greedy* dalam permainan '*2048*,' apakah permainan ini pasti bisa dimenangkan? Dan juga apakah algoritma *Greedy* merupakan cara yang paling optimal untuk memenangkan permainan '*2048*' ? Dalam makalah inilah akan dijelaskan permasalahan-permasalahan tersebut.

II. DASAR TEORI

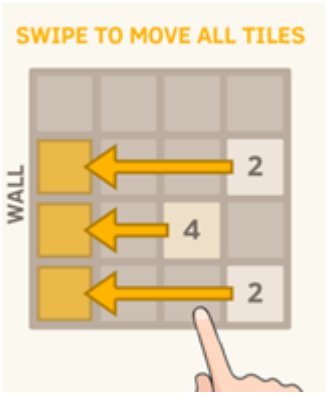
2.1. Permainan '*2048*'

'*2048*' adalah sebuah permainan yang diciptakan bulan Maret 2014 oleh Gabriele Cirulli, seorang pengembang *web* dari Italia yang berusia 19 tahun. Permainan ini berupa permainan sederhana yang dimainkan dengan cara menggeser blok-blok berangka hingga menghasilkan kotak berangka 2048. Permainan ini dirancang Cirulli hanya dalam satu pekan saja, dan tujuannya membuat permainan ini adalah untuk menguji apakah ia mampu merancang sebuah permainan dari awal. Ide permainan ini Cirulli tiru dari permainan *Veewo*

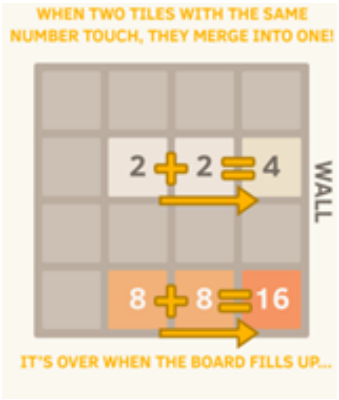
Studios 1024. Cirulli terkejut saat mengetahui bahwa permainan yang hanya dirancang untuk coba-coba ternyata sudah diunduh oleh 4 juta pengguna hanya dalam waktu kurang dari seminggu.

Gambar 2.1. Tampilan permainan '*2048*'
(Sumber : *http://2048game.com/*)

Cara memainkan permainan cukup sederhana dan mudah, karena hanya perlu menekan tombol arah pada keyboard (jika bermain dalam komputer atau laptop), atau bisa dengan menggeser jari pada layar (jika bermain dalam *smartphone*). Cara kerja permainan ini adalah jika kita menggeser ke suatu arah (atas, bawah, kiri, atau kanan), maka seluruh blok akan merapat ke sisi dari arah tersebut. Jika terdapat dua blok yang memiliki bobot sama, maka kedua blok tersebut akan menyatu dan membentuk blok baru dengan bobot penjumlahan dari dua blok tersebut. Misal terdapat dua blok berbobot sama, yaitu 2 (dua) bersebelahan kiri kanan, maka jika digeser ke kiri atau ke kanan, semua blok akan merapat sesuai arah pergeseran, dan kedua blok tersebut akan menjadi satu blok dengan bobot 4 (empat). Setelah melakukan pergeseran, sebuah blok baru akan muncul di sembarang tempat yang kosong, dengan bobot *random*.



Gambar 2.2.1. Cara gerak blok.
(Sumber : *http://www.ilovefreesoftware.com/07/iphone/2048-iphone-puzzle-game.html*)



Gambar 2.2.2. Cara blok bersatu.
(Sumber : *http://www.ilovefreesoftware.com/07/iphone/2048-iphone-puzzle-game.html*)

Tujuan dari permainan ini adalah untuk mendapatkan blok dengan bobot 2048. Permainan akan terus berjalan sampai seluruh ruang kosong dipenuhi oleh blok-blok dan tidak ada lagi blok yang bisa digabungkan (permainan berakhir dengan hasil kalah), atau jika terdapat sebuah blok yang berbobot 2048 (permainan berakhir dengan hasil menang), namun jika sudah terdapat blok berbobot 2048, permainan masih bisa dilanjutkan untuk mencapai bobot blok yang lebih tinggi lagi.

2.2. Algoritma *Greedy*

Algoritma *Greedy* merupakan salah satu metode yang paling populer digunakan dalam memecahkan persoalan optimasi. Persoalan optimasi maksudnya adalah persoalan mencari solusi optimum. Persoalan optimasi hanya ada dua yaitu maksimasi (*maximization*) dan minimasi (*minimization*). Contoh persoalan optimasi adalah persoalan penukaran uang. Misal diberikan uang senilai A. uang senilai A tersebut ingin ditukar dengan koin - koin uang yang ada. Berapa jumlah minimum koin yang diperlukan untuk penukaran tersebut ? Dalam hal ini, persoalan yang ada adalah persoalan minimasi. Contoh soal tersebut adalah misal terdapat koin 1, 5, 10, 25, dan terdapat uang senilai 32 yang ingin ditukar dengan koin-koin senilai demikian, dengan banyak koin sesedikit mungkin. Dengan nilai 32, dapat ditukar dengan berbagai cara seperti :

1. 32 = 1 + 1 + 1 + 1 + ... + 1 (32 koin)
2. 32 = 5 + 5 + 5 + 5 + 10 + 1 + 1 (7 koin)
3. 32 = 25 + 5 + 1 + 1 (4 koin)
4. dst.

Dari banyak cara penukaran tersebut, ditemukan koin tersedikit yang dapat ditukarkan adalah 4 (empat) koin dengan bobot 25, 5, 1 dan 1.

Gambar 2.3. Persoalan pertukaran koin
(Sumber : http://www.csharpstar.com/csharp-coin-change-problem-greedy-algorithm/)

Seperti definisi algoritma *Greedy* yang sudah dipaparkan di atas, pada setiap langkahnya, algoritma ini akan selalu membuat pilihan optimum lokal, dengan harapan bahwa langkah sisanya akan mengarah ke solusi optimum global. Algoritma ini bekerja seperti namanya, *Greedy*, yang artinya adalah tamak, rakus, dan loba, dan algoritma ini bekerja dengan prinsip "*take what you can get now*," yang berarti ambil solusi terbaik dalam kondisi saat itu juga.

Seperti dalam persoalan sebelumnya, yaitu mengenai pertukaran koin, kita dapat mendapatkan jawaban persoalan itu menggunakan algoritma *Greedy*. Algoritma *Greedy* yang diimplementasikan dalam persoalan tersebut adalah *greedy by value*, di mana koin tukaran yang diprioritaskan adalah koin dengan bobot terbesar. Dalam persoalan tersebut, karena bobot koin penukaran terbesar adalah 25, maka langsung diambil sebuah koin berbobot 25, sehingga sisa bobot yang perlu ditukarkan adalah 7 (tujuh), lalu ambil bobot koin terbesar selanjutnya yang cukup untuk ditukarkan, dan dalam hal ini koin tersebut adalah koin berbobot 5 (lima). Setelah mengambil sebuah koin berbobot 5 (lima), tersisa 2 (dua) bobot yang perlu digantikan, dan koin penukaran yang bisa ditukarkan tinggal koin berbobot 1 (satu), sehingga diambil 2 (dua) koin berbobot 5, dan akhirnya didapat solusi 4 buah koin yang ditukarkan. Solusi tersebut berupa solusi optimum lokal, sekaligus merupakan solusi global.

Pada algoritma *Greedy*, terdapat 5 (lima) elemen dalam penggunaan algoritma, yaitu himpunan kandidat (*C*), himpunan solusi (*S*), fungsi seleksi (*selection function*), fungsi kelayakan (*feasible*), dan juga fungsi obyektif. Dengan kata lain, algoritma *Greedy* melibatkan pencarian sebuah himpunan solusi (*S*) dari himpunan kandidat (*C*); yang dalam hal ini, *S* harus memenuhi beberapa kriteria yang ditentukan (*selection function and feasible*), yaitu menyatakan suatu solusi dan *S* dioptimasi oleh fungsi obyektif. Contoh kerja algoritma *Greedy* dalam *Pseudocode* dapat dilihat pada **Gambar 2.4.**

```
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 2.4. *Pseudocode* untuk algoritma *Greedy*
(Sumber : Slide kuliah algoritma *Greedy* oleh Rinaldi Munir)

Solusi algoritma *Greedy* belum tentu mendapatkan optimum global, sehingga algoritma ini tidak disarankan untuk persoalan yang mencari solusi optimum global. Contoh persoalan penukaran koin yang tidak cocok menggunakan algoritma *Greedy* adalah seperti berikut :

Koin : 5, 4, 3, 1	
Uang yang ingin ditukan = 7	
Solusi *Greedy* : 7 = 5 + 1 + 1	(3 koin; tidak optimal)
Solusi optimal : 7 = 4 + 3	(2 koin; optimal)

Algoritma *Greedy* untuk menyelesaikan persoalan pertukaran koin dapat diselesaikan menggunakan *pseudocode* seperti pada **Gambar 2.5.**

```
function CoinExchange(input C : himpunan_koin, A : integer) → himpunan_koin
{ mengembalikan koin-koin yang total nilainya = A, tetapi jumlah koinnya minimum }

Deklarasi
   S : himpunan_koin
   x : koin

Algoritma
  S ← {}
  while (∑(nilai semua koin di dalam S) ≠ A) and (C ≠ {} ) do
     x ← koin yang mempunyai nilai terbesar
     C ← C - {x}
     if  (∑(nilai semua koin di dalam S) + nilai koin x ≤ A then
       S ← S ∪ {x}
     endif
  endwhile

  if (∑(nilai semua koin di dalam S) = A then
     return S
  else
     write('tidak ada solusi')
  endif
```

Gambar 2.5. *Pseudocode* untuk algoritma *Greedy* dalam persoalan penukaran koin
(Sumber : Slide kuliah algoritma *Greedy* oleh Rinaldi Munir)

III. ALGORITMA *GREEDY* DALAM MENYELESAIKAN PERMAINAN '*2048*'

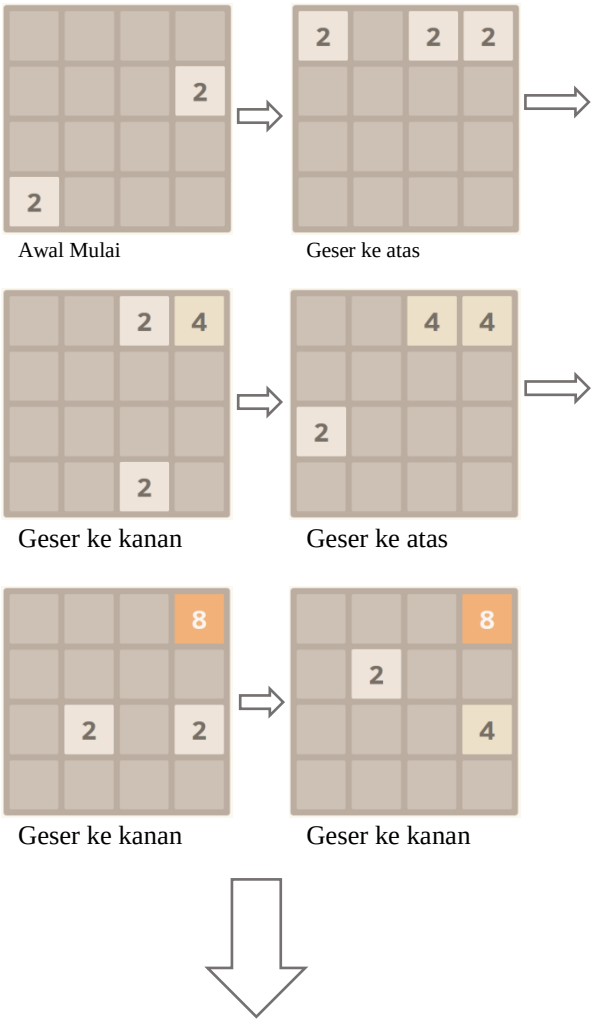
Algoritma *Greedy* yang digunakan dalam permainan '*2048*' merupakan penggunaan suatu strategi untuk memenangkan permainan tersebut. *Greedy* yang digunakan bisa dari berbagai aspek. Dalam makalah ini, ada dua strategi yang diambil. Strategi pertama adalah *Greedy* dengan cara selalu melakukan

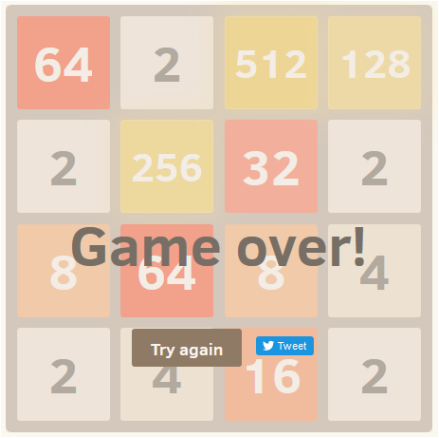
pergeseran ke arah di mana terdapat penggabungan blok yang menghasilkan bobot tertinggi dibanding arah-arah lainnya. Strategi kedua adalah *Greedy* dengan cara selalu melakukan penggeseran dengan memprioritaskan 2 arah, seperti atas dan kanan, kanan dan bawah, bawah dan kiri, atau atas dan kiri.

3.1. Strategi Pertama

Strategi Pertama, yaitu *Greedy* dengan cara selalu melakukan pergeseran ke arah di mana terdapat penggabungan blok dengan menghasilkan bobot tertinggi dibanding arah-arah lainnya. Strategi ini tergolong strategi *Greedy* yang bisa dibilang cukup *simple* dan mudah, karena hanya perlu mencari geseran arah yang menghasilkan penggabungan blok dengan bobot terbesar. Jika tidak ada pergeseran yang dapat menggabungkan dua blok, maka lakukan pergeseran asal.

Hasil permainan dengan menggunakan strategi ini dapat dilihat di bawah ini :





Akhir Permainan

Dengan menggunakan strategi pertama, hanya didapat blok dengan bobot maksimal 512, dan pergeseran blok dilakukan sebanyak 354 kali pergeseran.

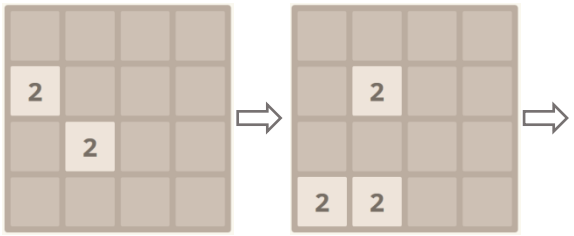
Kesimpulannya adalah dengan strategi pertama, belum tentu bisa memenangkan permainan ini, karena pergeseran yang dilakukan tergolong *random* berdasarkan hasil blok penggabungan terbaik.

3.2. Strategi Kedua

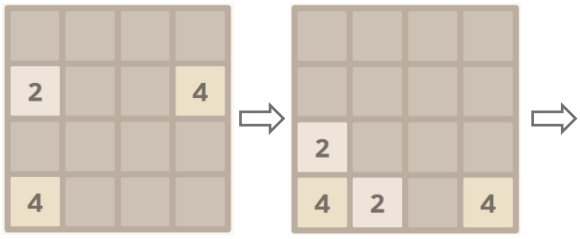
Strategi kedua, yaitu *Greedy* dengan cara selalu melakukan penggeseran dengan memprioritaskan 2 arah, seperti atas dan kanan, kanan dan bawah, bawah dan kiri, atau atas dan kiri. Strategi ini merupakan strategi pertama yang lebih teratur atau terstruktur, karena cara kerjanya juga mencari pergeseran yang dapat menggabungkan dua blok yang menghasilkan bobot tertinggi, namun pergeseran tersebut juga dibatasi dengan memprioritaskan suatu arah. Dalam percobaan ini, arah yang diprioritaskan adalah arah bawah, dan prioritas kedua adalah arah kiri. Jika tidak bisa dilakukan pergeseran ke bawah maupun ke kiri, maka dilakukan pergeseran ke kanan, dan jika tidak bisa juga dilakukan pergeseran ke kanan juga, maka dilakukan pergeseran ke atas, namun harus segera melakukan pergeseran ke bawah untuk mengembalikan blok dengan bobot maksimal kembali ke sudut.

Tujuan dari strategi ini adalah untuk menyimpan blok dengan bobot maksimal di sudut, sehingga penambahan blok tertata rapi dan pergeseran blok tidak asal-asalan atau berantakan.

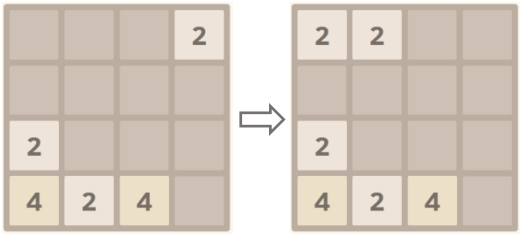
Hasil permainan dengan menggunakan strategi ini dapat dilihat di bawah ini :



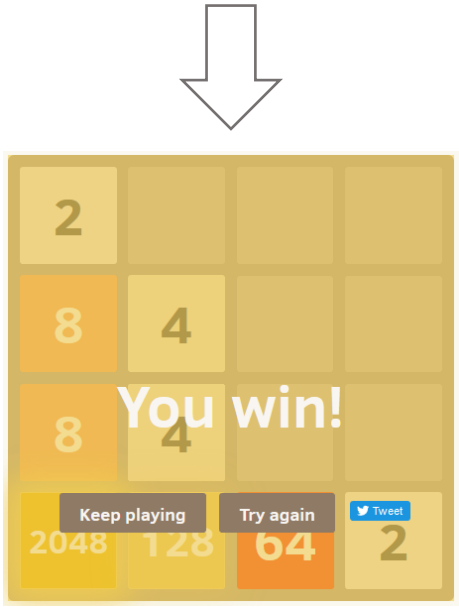
Awal mulai Geser ke bawah



Geser ke kiri Geser ke bawah



Geser ke kiri Geser ke kiri



Akhir permainan

Dengan menggunakan strategi kedua, blok dengan bobot 2048 bisa didapat dengan melakukan pergeseran sebanyak 894 kali pergeseran.

Kesimpulannya adalah dengan strategi kedua, permainan dapat dimenangkan, walaupun banyak pergeseran yang didapat belum tentu banyak pergeseran yang minimum. Namun strategi ini memungkinkan untuk mendapatkan banyak pergeseran minimum, karena pergeseran yang dilakukan terstruktur dan teratur, sehingga tidak ada pergeseran yang mubazir atau membuang-buang urutan.

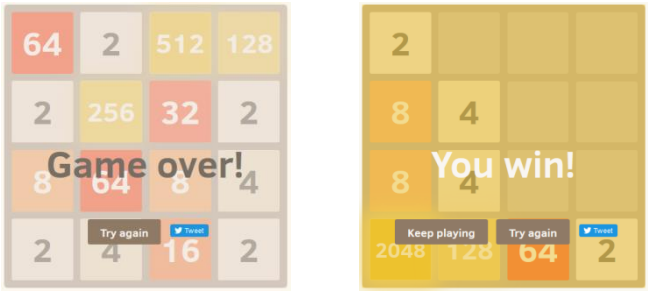
IV. ANALISIS PERBANDINGAN DUA STRATEGI DALAM ALGORITMA *GREEDY*

Pembandingan kedua strategi dilakukan untuk mengetahui strategi *Greedy* yang mana terbaik dilakukan untuk memenangkan permainan '*2048*.'

Untuk aspek kemudahan, tentu strategi pertama lebih mudah dibanding strategi kedua, karena strategi pertama hanya perlu melihat pergeseran mana yang menghasilkan bobot blok tertinggi, tanpa perlu memprioritaskan suatu arah. Untuk strategi kedua lebih rumit karena walaupun caranya hampir sama dengan strategi pertama, strategi kedua perlu memprioritaskan suatu arah agar bobot terbesarnya berada di sudut, sehingga penambahan blok lebih teratur dan terstruktur.

Untuk aspek keberhasilan, strategi kedualah yang memiliki kesempatan untuk menang lebih besar dibanding strategi pertama. Strategi kedua memiliki kesempatan lebih besar untuk menang karena pergeseran yang dilakukan lebih terstruktur, sehingga kesempatan untuk melakukan pergeseran yang merugikan lebih kecil dibanding strategi pertama.

Berdasarkan Percobaan yang dilakukan menggunakan dua strategi tersebut, didapat data seperti yang ada pada **Tabel 4.1.** Percobaan yang dilakukan merupakan percobaan pertama dari penggunaan strategi tersebut.



Gambar 4.1. Hasil strategi pertama (kiri) dan hasil strategi kedua (kanan)

	Strategi Pertama	Strategi Kedua
Bobot Maksimal	512	2048
Banyak Pergeseran	354	894

Tabel 4.1. Perbandingan data strategi satu dengan strategi dua

V. KESIMPULAN

Kesimpulan yang didapat dari percobaan ini adalah bahwa ada banyak algoritma yang dapat diimplementasikan dalam menyelesaikan permainan '*2048*.' Salah satu algoritma yang dapat digunakan adalah algoritma *Greedy*. Algoritma *Greedy* merupakan salah satu algoritma yang belum tentu dapat memenangkan permainan ini, karena algoritma *Greedy*, seperti artinya, hanya melihat kondisi di saat itu juga tanpa mementingkan bagaimana kondisi ke depannya, sehingga algoritma *Greedy* hanya akan mendapatkan solusi lokal, belum tentu mendapat solusi global. Penggunaan algoritma *Greedy* dalam permainan ini berupa penggunaan suatu strategi.

Penggunaan strategi merupakan contoh implementasi algoritma *Greedy* karena penyelesaian permainan dilakukan sesuai dengan keinginan pemain.

Strategi yang dilakukan dalam penyelesaian permainan '*2048*' contohnya adalah strategi selalu menggeser ke arah yang dapat menggabungkan dua blok dan menghasilkan bobot terbesar dibanding arah-arah lainnya. Contoh strategi lain adalah strategi menggeser dengan memprioritaskan suatu arah, agar blok dengan bobot terbesar tersimpan di sudut. Dengan melakukan percobaan, terbukti bahwa strategi kedua lebih ampuh dan optimal digunakan untuk memenangkan permainan ini, karena pergeseran yang dilakukan lebih terstruktur dan teratur.

VII. UCAPAN TERIMA KASIH

Pada akhir makalah ini, saya ingin mengucapkan terima kasih kepada Bapak Rinaldi Munir yang telah menyediakan *slide* kuliah yang dapat melancarkan saya dalam mengerjakan makalah ini. Saya juga mengucapkan terima kasih kepada Ibu Masayu karena telah mengajar saya selama ini sehingga saya dapat mengerti dan dapat mengerjakan makalah ini dengan baik. Yang terakhir saya berterima kasih kepada teman saya Audya Qintari, Teknik Sipil, 15016041, karena telah membantu saya dengan memainkan dan memenangkan permainan '*2048*' sesuai dengan strategi yang saya berikan.

REFERENSI

Slide kuliah algoritma *Greedy* oleh Rinaldi Munir

http://www.latimes.com/business/technology/la-fi-tn-2048-hit-game-creator-gabriele-cirulli-20140327-story.html#axzz2yiskNuep/

www.cityam.com/blog/1395045472/how-one-19-year-old-won-internet-addictive-puzzler-2048/

www.techcrunch.com/2014/03/24/clones-clones-everywhere-1024-2048-and-other-copies-of-popular-paid-game-threes-fill-the-app-stores/

https://exceptionnotfound.net/solving-minesweeper-with-c-sharp-and-linq/

http://2048game.com/

http://www.ilovefreesoftware.com/07/iphone/2048-iphone-puzzle-game.html/

http://www.csharpstar.com/csharp-coin-change-problem-greedy-algorithm/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 13 Mei 2018

Sinaga Yoko Christoffel Triandi
13516052