Penyelesaian 2048 dengan Algoritma Greedy

Kevin Maulana (13512044)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512044@std.stei.itb.ac.id

ABSTRAK

2048 adalah sebuah permainan dalam papan berukuran 4x4, dimana pemain harus mendapatkan petak yang bernilai 2048 dengan menggeser petak-petak ke kiri, kanan, atas dan bawah, serta menjumlahkan dua angka yang sama dengan menggabungkannya.

Makalah ini akan menjelaskan aturan lengkap dalam permainan tersebut, serta menjelaskan penggunaan algoritma greedy untuk menentukan langkah dalam menyelesaikan permainan tersebut.

Kata Kunci – 2048, greedy.

I. PENDAHULUAN

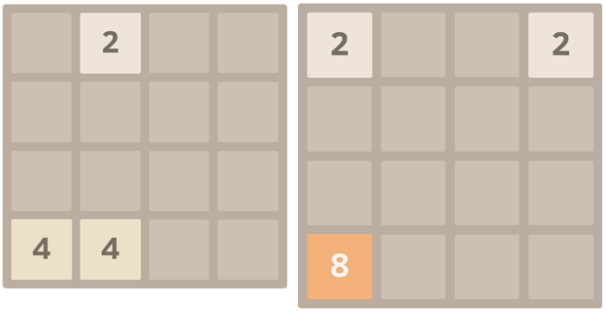
1.1 Sejarah Singkat 2048

2048 adalah suatu permainan dalam papan berukuran 4x4 dimana pemain harus mendapatkan petak yang bernilai 2048 untuk memenangkan permainan. Permainan ini dirilis pada tanggal 9 Maret 2014 oleh developer web asal Italia, Gabriele Cirulli. Permainan ini diadaptasi dari permainan-permainan yang telah mengusung konsep slide-block puzzle sebelumnya, seperti 1024 buatan Veewo Studios dan Threes! yang dibuat oleh Asher Vollmer dan dirilis sebulan sebelumnya.

Beberapa hari setelah 2048 diluncurkan, permainan ini mulai popular di internet, dan pada saat itu juga beredar banyak adaptasi 2048, seperti 4096, 2048 Numberwang, dan sebagainya. Wall Street Journal membuat suatu review tentang permainan ini yang mengatakan bahwa game ini seperti Candy Crush untuk matematikawan sejati.

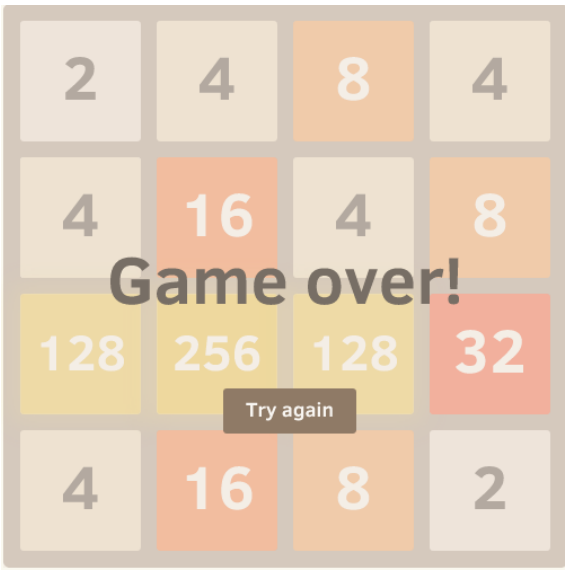
1.2 Cara Bermain 2048

Permainan ini menggunakan tombol panah keyboard untuk menggeser blok-blok berangka pada papan 4x4. Jika dua blok yang sama dan bersebelahan digeser sesuai arah, mereka bertabrakan dan membentuk sebuah blok baru yang nilainya dua kali lipat, dan pemain tersebut akan mendapatkan nilai sesuai dengan nilai blok baru tersebut.



Gambar 1: Blok-blok sebelum digeser (sisi kiri) dan blok-blok setelah digeser ke kiri (sisi kanan)

Permainan dinyatakan berakhir apabila sudah tidak ada langkah untuk melanjutkan (karena papan sudah penuh dan tidak ada blok yang dapat bertabrakan). Pemain dinyatakan menang apabila sudah mendapatkan blok dengan nilai 2048.



Gambar 2: Contoh berakhirnya permainan

1.3 Algoritma Greedy

Algoritma Greedy adalah salah satu algoritma yang dapat digunakan untuk memecahkan persoalan optimasi. Algoritma greedy mencari solusi optimum lokal disetiap langkah dengan harapan mendapat solusi optimum global diakhir langkah.

Masalah optimasi yang biasanya harus diselesaikan adalah maksimasi dan minimasi. Maksimasi adalah mendapatkan solusi yang hasilnya semaksimal mungkin dari suatu persoalan,sedangkan minimasi adalah mendapatkan solusi yang hasilnya seminimal mungkin dari suatu persoalan. Contoh persoalan maksimasi umumnya adalah memperolah keuntungan sebesar mungkin dari suatu persoalan,sedangkan contoh persoalan minimasi umumnya adalah masalah waktu,yaitu bagaimana cara mencapai tujuan dengan waktu paling sedikit atau menghemat waktu.

Prinsip greedy adalah "take what you can get now",artinya disetiap langkah algoritma ini akan mengambil langkah yang paling menguntungkan pada tahap itu tanpa perduli langkah selanjutnya. Solusi-solusi tersebut adalah solusi optimum lokal dengan harapan diakhir langkah akan mendapatkan solusi optimum global.

Adapun pseudo-code algoritma greedy adalah sebagai berikut :

```
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
        output('Tidak ada solusi')
endif
```

Penjelasan algoritma greedy :

- ➔ Ambil salah satu kandidat yang memenuhi kategori greedy, masukan kedalam himpunan solusi.
- ➔ Kurangi jumlah dari himpunan kandidat tersebut.
- ➔ Setelah diambil dan dimasukkan kedalam solusi.
- ➔ Jika layak, masukkan kedalam himpunan solusi.
- ➔ Jika tidak layak jangan masukkan kedalam himpunan solusi. (Definisi layak & tidak layak didefinisikan oleh fungsi kelayakan).
- ➔ Lakukan hingga sudah tidak ada makalah yang tersisa dari himpunan kandidat.

1.4.Elemen Algoritma Greedy

Dalam menyelesaikan persoalan,Algoritma Greedy memiliki 5 buah elemen,yaitu :

1. Himpunan Kandidat, C.

Berisi semua entitas yang membentuk solusi. Pada setiap langkah sebuah entitas akan dipilih dari himpunan kandidat.

2. Himpunan Solusi, S.

Himpunan solusi adalah entitas-entitas yang menjadi solusi permasalahan. Dalam proses membentuk himpunan solusi, pada setiap langkah kita membutuhkan fungsi seleksi dan fungsi kelayakan.

3. Fungsi Seleksi

Fungsi seleksi adalah fungsi yang menjelaskan bagaimana cara kita memilih solusi optimum lengkap.

4. Fungsi Kelayakan

Fungi kelayakan dapat dinyatakan dengan predikat lengkap. Fungsi kelayakan adalah fungsi syarat apakah solusi yang kita ambil layak untuk dimasukkan ke dalam himpunan solusi. Artinya solusi tersebut tidak melanggar *constraint* yang ada.

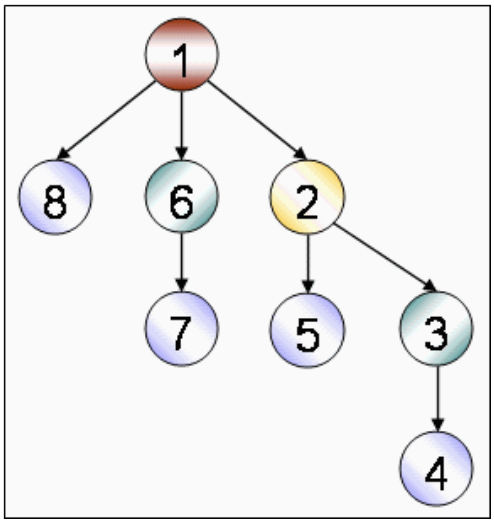
5. Fungsi Obyektif

Fungsi obyektif adalah fungsi yang dapat menjelaskan kondisi solusi sudah optimum atau tidak.

1.5 Aplikasi Algoritma Greedy

Ada banyak penerapan algoritma greedy di dunia nyata, seperti:

- Pembentukan pohon Huffman dalam melakukan kompresi.
- Knapsack Problem.
- Pemilihan tugas berdasarkan tenggat waktu (deadline).
- Penentuan arah berdasarkan jarak terpendek untuk mencapai tujuan.



Gambar 3: Contoh persoalan algoritma greedy dalam menentukan nilai minimum

II. METODE

2.1 Implementasi Fungsi Umum

Subbab ini menjelaskan implementasi fungsi-fungsi yang terdapat pada permainan 2048 itu sendiri. Fungsi-fungsi umum yang terdapat pada permainan adalah sebagai berikut:

2.1.1 Deklarasi

```
{Kamus Global}
const N : integer=4
type Matriks : array[1..N,1..N] of integer
M : Matriks
score : integer
```

- Asumsi: papan memiliki ukuran 4x4, sesuai dengan ukuran papan dalam permainan aslinya.
- Matriks merupakan representasi papan dalam bentuk matriks. Pada Matriks, blok yang kosong diisi nilai 0.
- Score disini merupakan skor yang didapat oleh pemain dalam permainan yang sedang berlangsung.

2.1.2 Inisialisasi

```
procedure isiPapan(input,output M: Matriks)

N: integer

N ← panjangPapan(M)
for baris<=1 to N do
  for kolom <=1 to N do
    M[baris][kolom] ← 0
  endfor
endfor
```

Pada tahap inisialisasi, papan diisi dengan blok kosong yang direpresentasikan dengan angka 0.

2.1.3 Pengisian nilai secara random

```
procedure isiRandom(input,output M:Matriks)
{Prekondisi: Matriks tidak terisi penuh}

randbar: integer
randkol: integer

selectRandomTile(randbar,randkol)
M[randbar][randkol] = RandomNumber()
```

Pada prosedur isiRandom, baris random dan kolom random di-assign dengan prosedur selectRandomTile, yang menyeleksi blok kosong. Kemudian baris dan kolom random pada matriks di-assign dengan nilai 2 atau 4 dengan kemungkinan kemunculan angka 2 90%, dan angka 4 10%. Pada awal game dilakukan 2 kali pengisian secara random.

2.2 Implementasi Algoritma Greedy

Subbab ini menjelaskan implementasi fungsi-fungsi yang berhubungan dengan algoritma greedy, serta algoritma greedy itu sendiri.

2.2.1 Pengecekan pergeseran ke arah tertentu

```
function isAvailableLeft(input M:Matriks)
→boolean

empty: boolean
isi: boolean
i: integer
j: integer
temp: integer
```

```
i←1
j←1
while(not empty or not isi and not merge and
i<=4) do
  empty←false
  isi←false
  j ← 1
  while(not empty or not isi and not merge and
j<=4) do
    if(not isi and not empty) do
      if(M[i][j]!=0) then
        isi ← true
        temp ← M[i][j]
      else if(M[i][j]==0) then
        empty ← true
      endif
    else if(empty and not isi) do
      if(M[i][j]!=0) then
        →true
      endif
    else if(isi and not empty) do
      if(M[i][j]!=0 and M[i][j]=temp) then
       → true
      else if(M[i][j]!=0 and M[i][j]!=temp) then
       temp ← M[i][j]
      else
        isi ←false
        empty ← true
      endif
    endif
    j ← j+1
  endwhile
  i ← i+1
endwhile
```

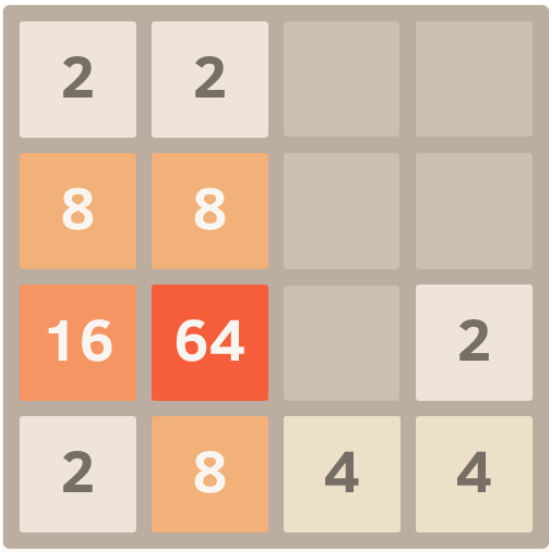
Fungsi pengecekan langkah ini mempunyai 2 variabel Boolean yang mengarah pada pengecekan blok sebelumnya dalam baris yang sama. Pada pengecekan kolom pertama dalam suatu baris, isi dan kosong diset false, karena belum ada blok yang mempunyai kolom yang lebih kecil. Ada tiga kasus pengecekan:

- Kolom sebelumnya tidak ada
  Pada tahap ini dicek apakah blok tersebut kosong atau tidak. Jika kosong, maka empty diset true, sedangkan jika tidak kosong, maka isi diset true, serta isinya diassign ke variable temp.
- Kolom sebelumnya tidak kosong
  Jika kolom yang dicek saat ini kosong maka kosong diset true dan isi diset false, sedangkan jika tidak kosong, dicek lagi apakah isi dari kolom yang dicek saat ini mempunyai nilai yang sama dengan variable temp. Jika sama maka return true, karena menandakan bahwa ada langkah yang dapat dilakukan, yaitu penggabungan blok. Sedangkan jika tidak sama maka isi temp diganti dengan isi kolom saat ini.
- Kolom sebelumnya kosong
  Jika kolom saat ini tidak kosong, maka fungsi langsung mengembalikan nilai true, karena menandakan bahwa kolom tersebut mempunyai blok yang dapat dipindahkan ke blok kosong.

```
function isAvailableRight(input M:Matriks)
→boolean
{ Sama seperti isAvailableLeft, hanya saja diiterasi
dari kolom terbesar terlebih dahulu }
```

```
function isAvailableUp(input M:Matriks)
→boolean
{ Sama seperti isAvailableLeft, hanya saja iterasi
baris dan kolom ditukar.}
```

```
function isAvailableDown(input M:Matriks)
→boolean
{ Sama seperti isAvailableUp, hanya saja diiterasi
dari baris terbesar terlebih dahulu}
```



Gambar 4

Diberikan contoh seperti Gambar 4, berikut hasil pengecekannya:

- isAvailableLeft: true
- isAvailableRight: true
- isAvailableUp: false
- isAvailableDown: false

Fungsi-fungsi tersebut dapat digabungkan menjadi sebuah fungsi untuk mengecek setiap langkah dalam permainan.

```
function IsAvailableSteps(input M:Matriks) → Boolean

→ isAvailableLeft(M) || isAvailableRight(M) || isAvailableUp(M) || isAvailableDown(M)
```

2.2.2 Penghitungan nilai tiap langkah

```
function HitungHorizontal(input M:Matriks) → integer

empty: boolean
isi: Boolean
i: integer
j: integer
temp: integer
score: integer

score ← 0
temp ← 0
for i=1 to 4 do
  isi ← false
  for j=1 to 4 do
    if (not isi and M[i][j]!=0) then
      isi ← true
      temp ← M[i][j]
    else if(isi and M[i][j]!=0) then
      if(M[i][j]=temp) then
        isi ← false
        score ← score + 2 * temp
        temp ← 0
      else
        temp ← M[i][j]
      endif
    endif
  endfor
endfor
→ score
```

Fungsi penghitungan skor ini mempunyai variabel Boolean yang mengarah pada pengecekan blok sebelumnya, apakah sudah terisi atau belum. Pada pengecekan kolom pertama dalam suatu baris, isi dan kosong diset false, karena belum ada blok yang mempunyai kolom yang lebih kecil. Ada dua kasus pengecekan:

- Kolom sebelumnya tidak ada/kosong
  Pada tahap ini dicek apakah blok tersebut kosong atau tidak. Jika blok yang dicek mempunyai nilai, maka isi diset true dan variable temp di-assign dengan nilai blok yang dicek.
- Kolom sebelumnya mempunyai nilai
  Jika kolom yang dicek saat ini mempunyai nilai, dicek terlebih dahulu apakah kolom tersebut mempunyai nilai yang sama, jika sama skor ditambahkan dengan 2 kali nilai temp (nilai hasil penggabungan), sedangkan jika tidak sama maka variable temp diisi dengan nilai blok saat ini.

```
function HitungVertikal(input M:Matriks) → integer

{Sama dengan hitungHorizontal, hanya saja iterasi pada baris dan kolom ditukar}
```

Diberikan contoh seperti Gambar 4, berikut hasil pengecekannya:

- HitungHorizontal: 28
- HitungVertikal: 0

2.2.3 Penghitungan blok yang dapat dikosongkan

```
function HitungEmptyHorizontal(input M:Matriks) → integer

isi: Boolean
i: integer
j: integer
temp: integer
count: integer

count ← 0
temp ← 0
for i=1 to 4 do
  isi ← false
  for j=1 to 4 do
    if (not isi and M[i][j]!=0) then
      isi ← true
      temp ← M[i][j]
    else if(isi and M[i][j]!=0) then
      if(M[i][j]=temp) then
        isi ← false
        count ← count+1
        temp ← 0
      else
        temp ← M[i][j]
      endif
    endif
  endfor
endfor
→ count
```

Fungsi penghitungan langkah ini mempunyai pengecekan yang sama dengan fungsi penghitungan skor, hanya saja nilai count nya ditambah 1 setiap ditemukan 2 blok yang dapat digabungkan.

Diberikan contoh seperti Gambar 4, berikut hasil pengecekannya:

- HitungEmptyHorizontal: 3
- HitungEmptyVertikal: 0

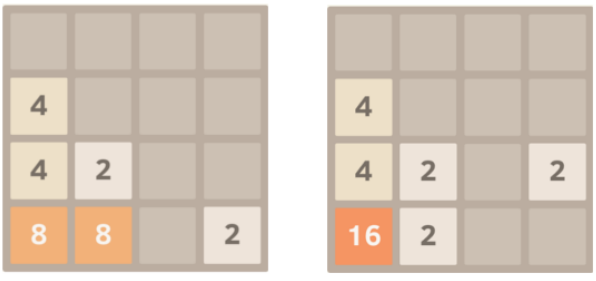
2.2.4 Algoritma Greedy by Value

```
Program GreedyByValue2048

{Kamus Global}
const N : integer=4
type Matriks : array[1..N,1..N] of integer
M : Matriks
score : integer
hightile : integer
{1 menujukkan arah kiri, 2 menujukkan arah bawah, 3
menujukkan arah kanan, 4 menunjukkan arah atas}

repeat 2 times
  isiRandom(M)
while(isAvailableSteps(M) and hightile!=2048) do
  if(HitungHorizontal(M)!=0) then
    MoveLeft(M)
  else if(HitungVertikal(M)!=0) then
    MoveDown(M)
  else if(isAvailableLeft(M))
    MoveLeft(M)
  else if(isAvailableDown(M))
    MoveDown(M)
  else if(isAvailableRight(M))
    MoveRight(M)
  else
    MoveUp(M)
  endif
endwhile
```

Pada program dengan algoritma greedy by value, program menentukan langkah apa yang harus ditempuh melalui skor tertinggi yang dapat dicapai dalam satu langkah. Pada program ini, peletakan blok difokuskan pada ujung kiri bawah, sehingga jika langkah horizontal lebih besar, program mengambil langkah kiri, sedangkan jika langkah vertical lebih besar, program mengambil langkah kanan. Tujuan dari pendekatan ini adalah untuk mendapatkan skor setinggi-tingginya dalam permainan dan mendapatkan blok baru yang mempunyai nilai tertinggi dengan cepat.



Gambar 5: Contoh persoalan algoritma greedy by value dalam permainan

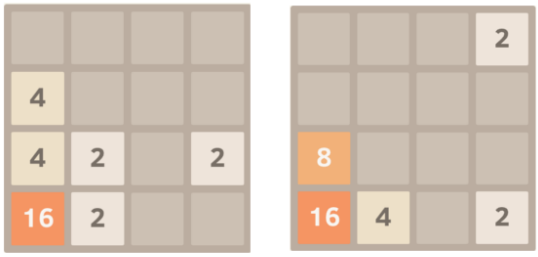
2.2.4 Algoritma Greedy by Free Space

```
Program GreedyByFreeSpace2048

{Kamus Global}
const N : integer=4
type Matriks : array[1..N,1..N] of integer
M : Matriks
score : integer
hightile : integer
{1 menujukkan arah kiri, 2 menujukkan arah bawah, 3
menujukkan arah kanan, 4 menunjukkan arah atas}

repeat 2 times
  isiRandom(M)
while(isAvailableSteps(M) and hightile!=2048) do
  if(HitungEmptyHorizontal(M)!=0) then
    MoveLeft(M)
  else if(HitungVertikal(M)!=0) then
    MoveDown(M)
  else if(isAvailableLeft(M))
    MoveLeft(M)
  else if(isAvailableDown(M))
    MoveDown(M)
  else if(isAvailableRight(M))
    MoveRight(M)
  else
    MoveUp(M)
  endif
endwhile
```

Pada program dengan algoritma greedy by free space, program menentukan langkah apa yang harus ditempuh melalui blok kosong yang dapat dicapai dalam satu langkah. Tujuan dari pendekatan ini adalah untuk dapat bermain dengan langkah yang lebih banyak sehingga banyak kesempatan untuk memperoleh nilai dan mendapatkan blok baru.



Gambar 6: Contoh persoalan algoritma greedy by free space dalam permainan

III. UCAPAN TERIMA KASIH

Terkait dengan penulisan makalah ini, penulis ingin berterimakasih kepada:

- Allah SWT, atas segala nikmat dan karunia-Nya.
- Pak Rinaldi Munir, yang telah berjasa dalam membimbing saya selama kuliah di jurusan Teknik Informatika ITB.
- Keluarga saya, yang telah memberikan semangat untuk terus menimba ilmu.
- Gabriele Cirulli, Asher Vollmer, dan beberapa kru yang terlibat dalam pembuatan game 2048 dan Threes!
- Internet, karena tanpa internet game tersebut mungkin tidak populer bahkan tidak akan ada saat ini.

REFERENSI

[1] Munir, Rinaldi, Diktat Kuliah Strategi Algoritma, Institut Teknologi Bandung, 2009.

[2] http://www.nouse.co.uk/2014/03/18/game-review-2048/ (diakses tanggal 18 Mei 2013 pukul 17:33)

[3] http://gabrielecirulli.github.io/2048/ (diakses tanggal 18 Mei 2013 pukul 17:28)

[4] http://techcrunch.com/2014/03/24/clones-clones-everywhere-1024-2048-and-other-copies-of-popular-paid-game-threes-fill-the-app-stores/ (diakses tanggal 18 Mei 2013 pukul 17:35)

[5] http://abcnews.go.com/Technology/2048-mobile-game-eat-time/story?id=23037239 (diakses tanggal 18 Mei 2013 pukul 17:40).

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, 19 April 2014

Kevin Maulana - 13512044