IF3051 Strategi Algoritma

Penerapan Algoritma Greedy untuk Peletakan Tanaman dalam *Game Harvest Moon: Back to Nature*

Nikodemus Adriel Limanthie/13510089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510089@stei.itb.ac.id

ABSTRAK

Algoritma Greedy adalah salah satu algoritma yang digunakan untuk mencari solusi optimum dalam pemecahan suatu masalah. Algoritma Greedy bekerja dengan mengambil suatu langkah pada saat tertentu tanpa memikirkan konsekuensi setelah pilihan tersebut diambil. Walaupun algoritma ini tidak selalu menghasilkan solusi yang optimum, namun solusi yang dihasilkan biasanya mendekati optimum. Permasalahan yang dibahas dalam makalah ini adalah mengenai peletakan tanaman dalam game Harvest Moon: Back to Nature agar penggunaan lahan tanaman optimum. Dengan menggunakan algoritma Greedy, saya akan menjelaskan pencarian solusi dari masalah tersebut.

Kata Kunci: algoritma Greedy, Harvest Moon: Back to Nature, peletakan tanaman

I. PENDAHULUAN

Harvest Moon: Back to Nature adalah sebuah permainan pada *platform* PlayStation yang menceritakan tentang seorang pemuda yang mendapatkan warisan sebuah lahan pertanian dari kakeknya yang sudah meninggal. Karena lahan tersebut tidak terurus, maka sang pemuda tersebut diminta untuk memperbaiki keadaannya dan diharapkan dapat mengembalikan pertanian tersebut seperti dulu.

Dalam permainan ini, pemain memainkan pemuda tersebut untuk memenuhi tujuan yang disebutkan di atas. Terdapat banyak hal yang dapat dilakukan pemain, seperti menanam berbagai macam tanaman di lahan yang disediakan, beternak hewan seperti sapi, ayam, dan domba, serta bersosialisasi dengan penduduk desa tempat lahan tersebut berada. Bahkan, pemain pun dapat mencari seorang calon istri dan menikahinya.

Dalam makalah yang saya buat kali ini, saya hanya akan membahas tentang penanaman tanaman pada lahan di permainan. Mekanisme dalam permainan ini mengharuskan pemain untuk menyiram tanaman agar tanaman tersebut dapat tumbuh. Untuk dapat menyiram tanaman, pemain harus berdiri di samping tanaman yang akan disiram, kemudian menyiramnya dengan alat yang dimiliki. Contohnya dapat dilihat pada gambar berikut.

Gambar 1 Menyiram Tanaman

Pada permainan ini terdapat beberapa jenis alat penyiram yang dapat digunakan, yaitu *Normal Watering Can*, *Copper Watering can*, *Silver Watering Can*, *Gold Watering Can*, dan *Mythril Watering Can*. Alat penyiram pada Gambar 1 adalah *Normal Watering Can* yang hanya dapat menyiram satu petak tanaman. Dengan semakin bagusnya alat penyiram yang dimiliki, semakin banyak pula petak yang dapat disiram sekaligus. Petak terbanyak yang dapat disiram dengan *Mythril Watering Can* adalah berukuran 3x5 petak. Contohnya dapat dilihat pada ilustrasi di bawah.

```
     OOOOO      menyiram ke
    POOOOO      arah kanan
     OOOOO

P = Karakter pemain
O = Petak yang tersiram
```

Total lahan dalam permainan ini berukuran 45x25 petak. Seluruh petak tersebut dapat ditanami tanaman jika sudah dibersihkan dari kayu dan batu. Namun, terdapat

masalah jika seluruh lahan dipenuhi tanaman karena akan ada tanaman yang tidak dapat disiram karena alat penyiram yang digunakan tidak dapat menyiram cukup jauh. Pemain juga tidak dapat berdiri di atas petak yang sudah terisi tanaman sehingga tidak memungkinkan jika semua petak lahan dipenuhi tanaman. Maka, dengan makalah ini saya akan berusaha mencari solusi agar pemain dapat menanam sebanyak mungkin tanaman pada lahan dan tetap bisa menyiramnya dengan menggunakan algoritma Greedy.

II. LANDASAN TEORI

Algoritma Greedy adalah sebuah algoritma yang bertujuan mencari solusi optimum dari sebuah masalah dengan cara mengambil langkah yang paling menguntungkan pada suatu tahap tertentu. Secara umum, elemen-elemen dalam algoritma Greedy dapat dibagi seperti berikut.

- Himpunan kandidat, dilambangkan dengan K
- Himpunan solusi, dilambangkan dengan S
- Fungsi seleksi, dilambangkan dengan SELEKSI()
- Fungsi kelayakan, dilambangkan dengan LAYAK()
- Fungsi obyektif, dilambangkan dengan SOLUSI()

Himpunan kandidat berisi elemen-elemen pembentuk solusi. Himpunan solusi berisi kumpulan elemen-elemen dari himpunan kandidat yang membentuk solusi. Fungsi seleksi adalah fungsi yang memilih kandidat yang paling mungkin menjadi solusi optimum. Fungsi kelayakan adalah fungsi yang memeriksa apakah kandidat yang dipilih memberikan solusi yang layak. Sedangkan fungsi obyektif adalah fungsi yang meminimumkan atau memaksimumkan nilai solusi.

Skema umum algoritma Greedy dapat dilihat pada *pseudo-code* berikut.

```
function greedy (input C: himpunan_kandidat) → himpunan_kandidat

Deklarasi
    X : kandidat
    S : himpunan_kandidat

Algoritma
    S ← {}    {inisialisasi S dengan kosong}
    while (not SOLUSI(S) and (C ≠ {})) do
        x ← SELEKSI(C)
        C ← C – {x}
        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
```

Pertama-tama, algoritma greedy akan melakukan iterasi terhadap semua elemen himpunan kandidat yang ada dan melakukan seleksi terhadap kandidat yang akan menghasilkan solusi optimum secara lokal. Lalu, dilakukan pengecekan dahulu terhadap kandidat tersebut menggunakan fungsi kelayakan. Jika sudah layak, maka kandidat dimasukkan ke dalam himpunan solusi. Hal ini dilakukan sampai elemen di himpunan kandidat habis atau solusi telah ditemukan.

III. METODE PENCARIAN SOLUSI

Lahan tanaman berukuran 45x25 akan direpresentasikan dengan sebuah matriks M. Pada matriks M, petak yang berisi tanaman dilambangkan dengan karakter 'O' dan petak yang kosong dilambangkan dengan karakter '_'.

Untuk mencari solusi dari persoalan ini, terdapat dua buah tahap algoritma secara umum. Yang pertama adalah algoritma greedy dalam penandaan petak yang akan diisi oleh tanaman. Yang kedua adalah penghapusan petak tanaman yang membuat petak kosong tidak dapat dijangkau dari luar lahan.

1. Algoritma Greedy dalam Penandaan Petak

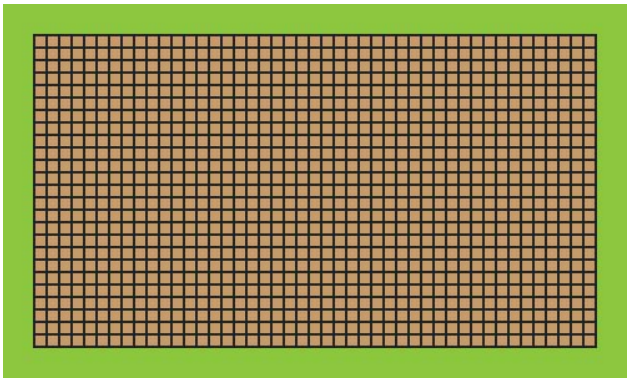
Skema algoritma Greedy untuk penandaan petak adalah seperti berikut.

```
function greedyPlanting (input width : int, input height : int) → matrix of char

Deklarasi
    S : matrix of char[width][height] (matriks solusi)
    i, j : integer

Algoritma
    S ← {}    {inisialisasi S dengan kosong}

    for i = 0 to width-1 do
        for j = 0 to height-1 do
            if PLANTABLE(S[i][j]) then
                S[i][j] ← 'O'     {isi dengan char O}
            else
                S[i][j] ← '_'     {isi dengan char _}
            endif
        endfor
    endfor

    return S
```

Pada fungsi tersebut, algoritma akan mengiterasi satu persatu petak dari ujung kiri atas. Setiap petak akan dicek apakah dapat ditanami atau tidak dengan fungsi PLANTABLE(). Jika dapat ditanami, petak diisi dengan karakter 'O'. Jika tidak, petak diisi dengan karakter '_' sebagai tanda petak kosong.

Fungsi seleksi yang digunakan pada algoritma tersebut adalah iterasi petak per petak karena setiap kandidat memiliki kemungkinan solusi yang sama besarnya. Fungsi kelayakan di sini adalah fungsi PLANTABLE(). Fungsi ini melakukan pengecekan terhadap petak tersebut apakah dapat ditanami atau tidak. Kondisi petak yang dapat ditanami adalah petak harus berjarak ≤ 5 dari minimal salah satu petak kosong lain pada arah horizontal maupun vertikal. Kondisi ini dicantumkan dengan mengingat bahwa alat penyiram dapat menyiram pada ukuran 3x5. Maka, jika jarak petak A terhadap petak kosong lain berada pada ukuran 3x5 penyiraman tersebut, petak A dapat disiram dari petak kosong tersebut. Perlu diingat bahwa bagian luar dari

lahan termasuk dalam petak kosong. Contoh ilustrasi fungsi ini seperti berikut.

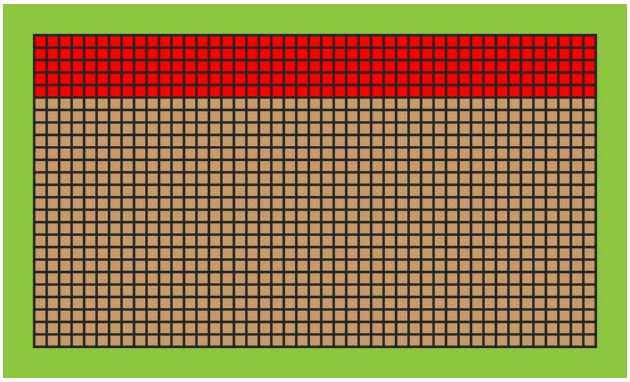


Gambar 2 Lahan Kosong

 = Satu petak kosong

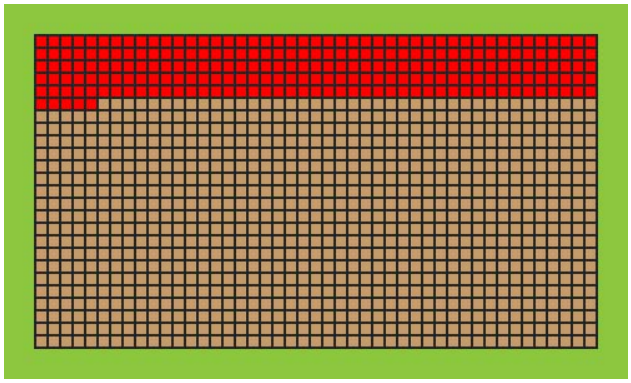
 = Satu petak berisi tanaman

Misalkan terdapat lahan kosong sebesar 45x25 petak seperti pada Gambar 2. Dengan menggunakan algoritma tersebut, akan dihasilkan tampilan lahan seperti berikut.



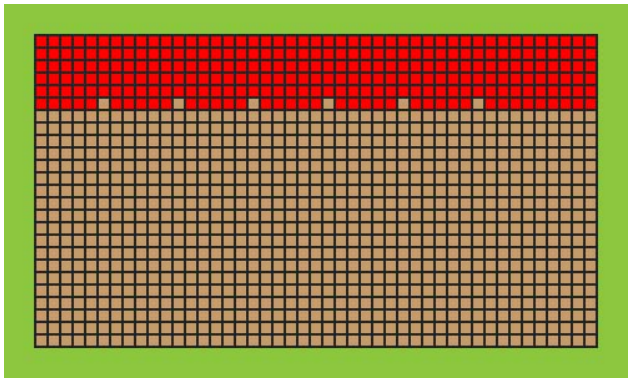
Gambar 3 Lahan Terisi pada Tahap Pertama Algoritma Greedy

Algoritma akan melakukan iterasi terhadap tiap baris dari lahan dan mengecek apakah tanaman dapat ditanam pada petak tersebut. Karena petak-petak pada lima baris pertama dapat disiram dari sisi luar lahan bagian atas, maka semua petak sampai baris kelima diberi tanda merah (dapat ditanami).



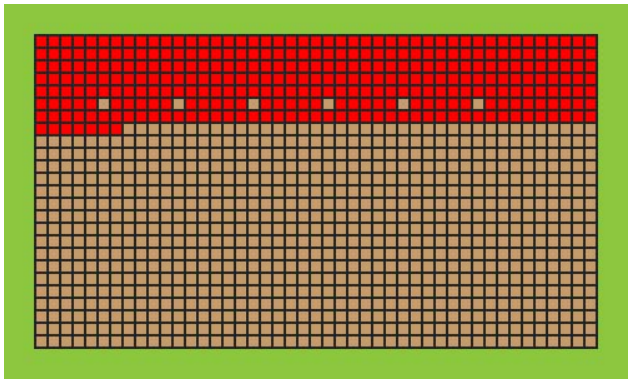
Gambar 4 Lahan Terisi pada Tahap Kedua Algoritma Greedy

Ketika algoritma sampai pada baris keenam, algoritma akan mengisi 5 baris pertama yang masih dapat dijangkau dari sisi luar lahan bagian kiri. Namun, ketika algoritma mengecek petak keenam pada baris keenam, petak tersebut diketahui tidak dapat disiram dari luar lahan. Maka, petak tersebut dikosongkan seperti pad gambar berikut.



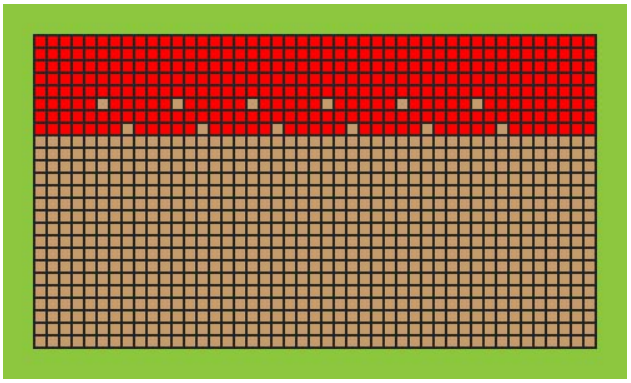
Gambar 5 Lahan Terisi pada Tahap Ketiga Algoritma Greedy

Demikian seterusnya, algoritma akan mengosongkan petak yang tidak dapat disiram dari luar lahan sehingga terdapat beberapa petak kosong pada lahan.



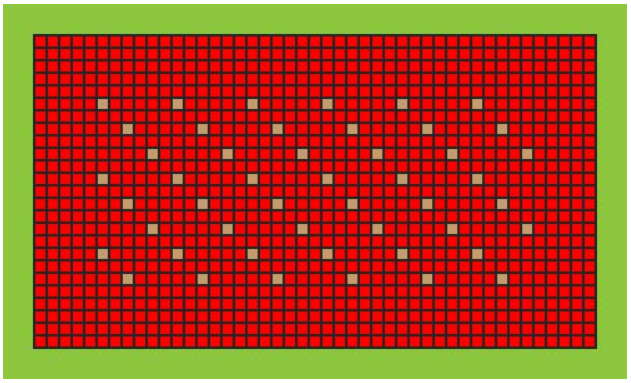
Gambar 6 Lahan Terisi pada Tahap Keempat Algoritma Greedy

Pada tahap berikutnya, seluruh baris ketujuh dapat diisi dengan tanaman karena setiap petak dapat disiram dari salah satu petak kosong. Namun, pada baris kedelapan petak kedelapan, algoritma akan menemukan bahwa petak tidak dapat disiram sehingga akan dikosongkan juga seperti pada gambar berikut.



Gambar 7 Lahan Terisi pada Tahap Kelima Algoritma Greedy

Maka akan terbentuk pola petak kosong seperti pada Gambar 7 jika algoritma diteruskan. Hasil akhir dari algoritma greedy tersebut seperti pada gambar berikut.



Gambar 8 Lahan Terisi pada Tahap Terakhir Algoritma Greedy

Dapat dilihat pada Gambar 8 bahwa terdapat petak-petak kosong dimana pemain dapat menyiram lahan. Dengan peletakan seperti ini, sebetulnya semua tanaman sudah dapat disiram. Namun, ditemukan masalah baru yaitu tidak adanya jalan masuk ke petak-petak kosong tersebut karena ditutupi oleh tanaman. Maka, ditambahkan sebuah algoritma greedy lain untuk membuat jalan bagi petak-petak kosong tersebut agar dapat diakses.

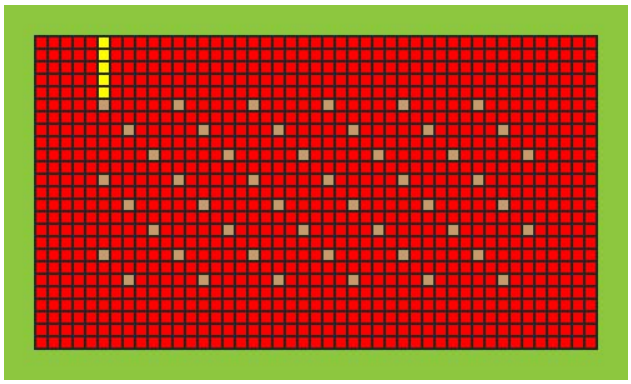
2. Algoritma Greedy untuk Penghapusan Petak

Skema algoritma greedy tersebut adalah seperti berikut.

```
function greedyRemoving (input M : matrix of char) →
matrix of char

Deklarasi
    i, j : integer
    dir : string

Algoritma
    for i = 0 to width-1 do
        for j = 0 to height-1 do
            if (M[i][j]) = '_' then {jika petak kosong}
                dir = SHORTESTDIR(M, i, ,j)
                REMOVEPLANT(M, i, j, dir)
            endif
        endfor
    endfor

    return S
```

Pada fungsi tersebut, algoritma akan mengiterasi satu persatu petak dari ujung kiri atas. Jika ditemukan petak yang kosong, variabel **dir** akan diisi dengan arah (*direction*) yang didapatkan dari fungsi SHORTESTDIR(). Fungsi ini akan mengecek petak pada (i,j) dan menghitung jarak dari petak tersebut menuju petak kosong pada arah atas, bawah, kiri, dan kanan. Arah yang menghasilkan jarak terpendek menjadi kembalian dari fungsi ini. Untuk arah atas, kembaliannya adalah "*up*", bawah adalah "*down*", kiri adalah "*left*" dan kanan adalah "*right*". Nilai **dir** akan digunakan oleh prosedur REMOVEPLANT() yang berfungsi menghapus semua petak tanaman (dengan menggantinya dengan karakter lain, misalnya 'X') mulai dari petak (i,j) sampai ditemukan petak kosong berikutnya pada arah **dir**.

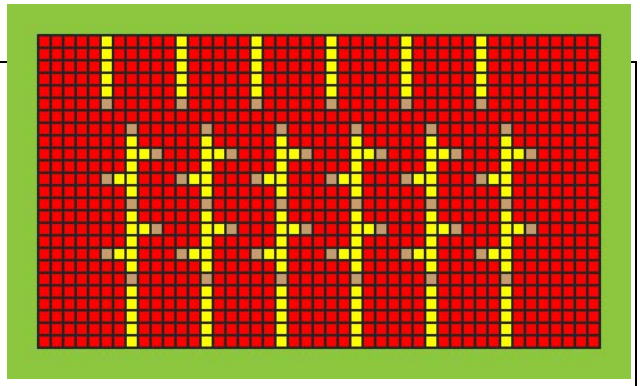
Secara sederhana, algoritma akan mencari petak kosong terdekat lain dari suatu petak kosong (dalam arah vertikal atau horizontal) dan membuat jalan ke petak kosong tersebut. Jika algoritma tersebut diaplikasikan pada hasil peletakan tanaman yang pertama, maka akan dihasilkan tampilan lahan seperti berikut.



Gambar 9 Tampilan Lahan Setelah Dibuat Jalan Tahap Pertama

 = Petak tanaman yang dihapus

Untuk petak kosong pertama, ditemukan salah satu arah yang paling pendek jaraknya adalah atas. Maka, lima buah petak di bagian atas dari petak kosong pertama dihapus.



Gambar 10 Tampilan Lahan Setelah Dibuat Jalan

Setelah algoritma selesai, maka penghapusan petak-petak pada lahan akan menghasilkan tampilan lahan seperti pada Gambar 10. Dengan peletakan seperti itu, semua tanaman dapat disiram dengan menggunakan *Mythril Watering Can*.

IV. Analisis

Jika dihitung efisiensi penggunaan lahannya, maka solusi dari algoritma greedy tersebut mempunyai efisiensi:

$$\frac{\text{Petak keseluruhan} - \text{Petak kosong}}{\text{Petak keseluruhan}} = \frac{45x25 - 192 \text{ petak kosong}}{45x25} = 82.9\%$$

Nilai efisiensi tersebut dapat dinilai relatif cukup besar. Walaupun begitu, jika dicari solusi lain dari permasalahan ini, dapat ditemukan solusi yang memiliki efisiensi lebih besar dibandingkan dengan solusi ini. Hal ini disebabkan banyaknya peletakan petak kosong yang terkesan bertumpuk sehingga tidak efisien dalam pembagian tempat penyiraman tanamannya.

Secara keseluruhan, algoritma greedy yang diaplikasikan dalam permasalahan ini sedikit berbeda dengan algoritma greedy pada umumnya. Perbedaan terletak pada kesamaan prioritas dari tiap kandidat petak lahan yang ada. Pada umumnya, setiap kandidat pada algoritma greedy memiliki prioritas yang berbeda baik dilihat dari sisi nilai, maupun ukuran. Namun, tiap petak di lahan ini memiliki prioritas yang sama karena tidak memiliki atribut nilai maupun ukuran. Oleh karena itu, pemilihan kandidat dilakukan dengan iterasi satu persatu.

V. Kesimpulan

Kesimpulan yang didapatkan dari permasalahan ini adalah algoritma greedy yang digunakan tidak memberikan solusi yang paling optimum yang bisa ditemukan. Namun, solusi yang dihasilkan memiliki efisiensi yang relatif cukup besar sehingga dapat dianggap sudah mendekati nilai optimum. Maka, dapat disimpulkan bahwa algoritma greedy tidak selalu memberikan solusi yang paling optimum, namun solusinya mendekati optimum.

Referensi

[1] Cormen, T. H. (1990). *Introduction to Algorithms.* USA: MIT press.

[2] Munir, R. (2009). *Strategi Algoritma.* Bandung: Institut Teknologi Bandung.

[3] Render, Sky. (2007, Maret 06). *GameFAQs: Harvest Moon: Back To Nature (PS) FAQ/Walkthrough by Sky Render*. Retrieved Desember 20, 2012, from Video Game Cheats, Reviews, FAQs, Message Boards, and More - GameFAQs: www.gamefaqs.com/ps/446412-harvest-moon-back-to-nature/faqs/26294

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, 29 April 2010

Nikodemus Adriel Limanthie
13510089