Pemanfaatan Algoritma *Greedy* Dalam Menyelesaikan Permasalahan *Elevators Logic*

Jundullah / 13518027
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail):
13518027@std.stei.itb.ac.id

***Abstract*—Permasalahan Elevators Logic merupakan permasalahan dimana terdapat lima buah elevator/lift. Di dalam kelima elevator tersebut ada orang yang terjebak karena pintu lift tidak dapat terbuka. Pintu lift akan terbuka jika dan hanya jika kelima elevator berada pada lantai 21 sampai 25. Untuk memindahkan elevator pada lantai tertentu dilakukan dengan cara memilih dua elevator dan ubah posisi lantai kedua elevator tersebut dengan menambahkan dengan 8 atau dikurangi dengan 13.**

Keywords—Knight Problem; Backtrack; Backtracking; Strategi Algoritma

I. Introduction

Permasalahan Elevators Logic dibuat oleh plastellina sebagai soal teka-teki untuk mengasah logika anak-anak pada tingkat sekolah dasar (SD) dan sekolah menengah pertama (SMP). Permasalahan ini melibatkan lima orang yang terjebak dalam lima buah lift berbeda yang saling terhubung.

Untuk menyelamatkan kelima orang tersebut, pemain harus mengarahkan sedemikian hingga kelima lift tersebut berada pada salah satu diantara lantai 21 sampai lantai 25.

Terdapat rule yang sangat merepotkan dalam pemindahan lantai dari setiap lift tersebut yaitu setiap pemindahan harus melibatkan tepat dua elevator dalam waktu yang bersamaan dan setiap pemindahan hanya boleh menaikkan elevator sebanyak delapan lantai atau menurunkan elevator sebanyak tiga belas lantai.

Gambar 1. Permasalahan Elevators Logic

II. Landasan Teori

A. *Definisi Algoritma Greedy*

Algoritma adalah langkah dalam mencari solusi atas sebuah masalah. banyak sekali algoritma yang dapat kita gunakan dalam membangun sebuah program , salah satunya adalah algoritma greedy.

Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Greedy sendiri diambil dari bahasa inggris yang artinya rakus, tamak atau serakah .Prinsip algoritma greedy adalah: "*take what you can get now!*".

Contoh masalah sehari-hari yang menggunakan prinsip greedy adalah memilih beberapa jenis investasi (penanaman modal), mencari jalur tersingkat dari Bandung ke Surabaya, memilih jurusan di Perguruan Tinggi, bermain kartu remi, dan masih banyak lagi.

Algoritma greedy membentuk solusi dengan cara langkah per langkah (*step by step*). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.

B. *Persoalan optimasi (optimization problems)*

Dalam setiap penggunaan langkah pengambilan pilihan dalam algoritma greedy, dibutuhkan persoalan optimasi. Persoalan optimasi adalah persoalan yang menuntut pencarian solusi optimum. Persoalan optimasi ada dua macam, yaitu:

- Maksimasi (maximization)
- Minimasi (minimization)

Solusi optimum (terbaik) adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang mungkin. Terdapat dua elemen persoalan optimasi, yaitu kendala (*constraints*) dan fungsi objektif atau fungsi optiamsi.

Solusi yang memenuhi semua kendala disebut solusi layak (*feasible solution*). Solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum.

C. *Kelebihan dan Kekurangan Algoritma Greedy*

Jika kita melihat definisi dan pengertian dari algoritma greedy, maka kita dapat menyimpulkan bahwa kelebihan dari algoritma greedy adalah cepat dalam bertindak alias respon cepat. Jika Anda perlu memecahkan masalah dengan cepat dan cepat, algoritma serakah adalah salah satu metode yang tepat.

Algoritma rakus tidak butuh waktu lama untuk memikirkan opsi lain yang bisa dilakukan, dan tidak perlu memperhitungkan baik buruk maupun konsekuensi dari apa yang diputuskan.

Meskipun merupakan ide pemecahan masalah yang logis, etapi algoritma serakah ini memiliki kelemahan dalam bentuk hasil akhir yang tidak sebaik algoritma brute force. Ini tentu saja disebabkan oleh pilihan opsi yang dikecualikan, sehingga isa negatif atau konsekuensi dari keputusan pemilihan tidak dapat sepenuhnya dipertanggungjawabkan.

D. *Skema Umum Algoritma Greedy*

Algoritma greedy melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif.

```
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. Skema Umum Algoritma Greedy (Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf)

III. Implementasi Algoritma Greedy Terhadap Penyelesaian Elevators Logic

Penyelesaian dari tahap ke tahap membutuhkan beberapa fungsi seperti fungsi kelayakan:

```
def inRange(i):
    return i <= 25 and i >= 21


def finish():
    for i in arr:
        if not(inRange(i)):
            return False
    return True
```

Selain itu juga dibutuhkan fungsi seleksi. Dalam hal ini seleksi yang dilakukan terbagi menjadi 3 tahap, yaitu:

A. Pilih dua elevator yang lebih kecil dari 21

```
    elevator1 = -1
    elevator2 = -1

    for i in range(5):
        if(arr[i]<21 and elevator1<i):
            elevator1 = i
            break

    for i in range(elevator1+1, 5):
        if(arr[i]<21 and elevator2<i):
            elevator2 = i
            break

    if(elevator1 != -1 and elevator2
!= -1):
        arr[elevator1] += 8
        arr[elevator2] += 8
        swap = True
        print('  8', end = '   ')
        print(elevator1,elevator2, '
', arr)
```

B. Pilih dua elevator yang lebih dari 25

```
    elevator1 = 50
    elevator2 = 50

    for i in range(5):
        if(arr[i]>25 and elevator1>i):
            elevator1 = i
            break

    for i in range(elevator1+1, 5):
        if(arr[i]>25 and elevator2>i):
            elevator2 = i
            break

    if(elevator1 != 50 and elevator2
!= 50):
        arr[elevator1] -= 13
        arr[elevator2] -= 13
        swap = True
        print('-13', end = '   ')
        print(elevator1,elevator2, '
', arr)
```

C. Pilih satu elevator yang lebih kecil dari 21 bersama dengan elevator yang lebih dari 25

```
elevator1 = -1

    for i in range(5):
        if(arr[i]<21 and elevator1<i):
            elevator1 = i
            break

    elevator2 = 50

    for i in range(5):
        if(arr[i]>25 and elevator2>i):
            elevator2 = i
            break

    if(elevator1 != -1 and elevator2
!= 50):
        if elevator1 > 13:
            arr[elevator1] -= 13
            arr[elevator2] -= 13
            print('  8', end = '   ')
        else:
            arr[elevator1] += 8
            arr[elevator2] += 8
            print('  8', end = '   ')
        swap = True
        print(elevator1,elevator2, '
', arr)
```

D. Lakukan algoritma swap ketika hanya tersisa satu elevator yang tidak berada pada posisi yang benar

```
if(not(swap)):
    i = 5
    count = 0
    while (i > 0 and count < 2):
        i -= 1
        if(inRange(arr[i])):
            if(count == 0):
                if(arr[i] < 21):
                    count = 8
                else:
                    count = -13
            arr[i] += count
```

IV. Pengujian Algoritma Greedy Terhadap Penyelesaian Permasalahan Elevators Logic

Contoh pengujian berikut untuk permasalahan original dari plastellina yang melibatkan lima elevator yang nilai awal dari tiap elevator secara berturut-turut adalah 17, 26, 20, 19, dan 31.

A. Pengujian algoritma secara tertulis menggunakan terminal

Dengan menggunakan algoritma *greedy* dibutuhkan 16 langkah pemindahan elevator yang dilakukan agar mendapatkan solusi agar kelima elevator yang tidak dapat dibuka, sehingga kelima orang yang tadinya terjebak akhirnya dapat keluar.

```
Start       [17, 26, 20, 19, 31]
  8   0 2   [25, 26, 28, 19, 31]
-13   1 2   [25, 13, 15, 19, 31]
  8   1 4   [25, 21, 15, 19, 39]
  8   2 3   [25, 21, 23, 27, 39]
-13   3 4   [25, 21, 23, 14, 26]
  8   3 4   [25, 21, 23, 22, 34]
  8   0 1   [20, 16, 10, 9, 34]
  8   0 4   [28, 16, 10, 9, 42]
  8   1 2   [28, 24, 18, 9, 42]
-13   0 4   [15, 24, 18, 9, 29]
  8   0 4   [23, 24, 18, 9, 37]
  8   2 3   [23, 24, 26, 17, 37]
-13   2 4   [23, 24, 13, 17, 24]
  8   2 3   [23, 24, 21, 25, 24]
```

Gambar 3. Hasil pengujian algoritma penyelesaian permasalahan Elevators Logic

Pada Gambar 3 hanya ditampilkan 14 pemindahan lift, hal ini disebabkan oleh adanya fungsi swap yang bukan merupakan bagian dari algoritma greedy sehingga tidak ikut ditampilkan. Dua pemindahan yang dilakukan saat menjalankan fungsi swap adalah -13 0 1 dan -13 2 3.

Solusi ini merupakan solusi yang dapat digunakan, namun bukanlah sebuah solusi optimum, karena terdapat solusi yang lebih cepat yaitu 8 kali pemindahan berdasarkan video *youtube* di channel Vincelin Gino yang berjudul “How to solve it – Elevators problem”.

B. Pengujian algoritma secara visual menggunakan Shockwave Flash

Shockwave Flash adalah format yang digunakan oleh plastellina untuk melakukan *development* dari logic game yang dibuatnya. Elevators Logic hanyalah salah satu dari sekian banyak logic game yang dibuat oleh plastellina.

Gambar 4. Tampilan awal saat Elevators Logic dimulai

Sebagaimana yang terlihat pada Gambar 4, posisi awal dari kelima elevator yang terlibat adalah 17,26,20,19, dan 31 secara berturut-turut. Hal ini sudah disesuaikan dengan poin sebelumnya sehingga dalam memainkannya hanya perlu mengikuti jawaban yang ada pada soal sebelumnya.

Gambar 5. Tampilan setelah menyelesaikan permasalahan dalam 16 langkah pemindahan

C. Solusi alternatif yang lebih efisien oleh Vincelin Gino

Dalam makalah ini hanya ditampilkan kondisi awal dan kondisi akhir dari solusi yang ditawarkan oleh Vincelin Gino. Untuk langkah lengkapnya dapat dilihat di channel youtube Vincelin Gino.

Pada channel tersebut, tidak disebutkan algoritma yang digunakan oleh Vincelin Gino. Namun jika diasumsikan bahwa delapan langkah pemindahan lift merupakan solusi yang mangkus, dapat disimpulkan bahwa algoritma yang digunakan Vincelin Gino adalah algoritma *bruteforce* atau algoritma apapun yang dapat menghasilkan solusi yang terjamin mangkus.

Gambar 6. Pemilihan dua elevator pertama dengan solusi oleh Vincelin Gino (Sumber : https://www.youtube.com/watch?v=z9VN5fFo0QU)

Karena masalah ini melibatkan n yang kecil, tidak masalah menggunakan algoritma yang kompleksitas waktunya cukup tinggi karena pengaruhnya akan sangat kecil.

Gambar 7. Solusi akhir yang hanya membutuhkan delapan kali pemindahan elevator. (Sumber : https://www.youtube.com/watch?v=z9VN5fFo0QU)

V. KESIMPULAN

Permasalahan *Elevators Logic* dapat diselesaikan dengan menggunakan algoritma *greedy*. Algoritma greedy sangat mudah untuk diimplementasikan dan memiliki kompleksitas waktu yang jauh lebih cepat dibandingkan algoritma naif (*brute force*), namun memiliki kekurangan yang cukup besar yaitu tidak selalu menemukan solusi optimum.

VIDEO LINK AT YOUTUBE

Simulasi : https://youtu.be/wna2di13wN8

Penjelasan : https://youtu.be/XOMQ09jMUl4

UCAPAN TERIMA KASIH

Puji syukur dipanjatkan atas kehadirat tuhan YME, atas karunia dan nikmat-Nya penulis dapat menyelesaikan makalah ini. Penulis juga ingin menyampai terima kasih yang sebesar-besarnya kepada orang tua dan teman-teman serta dosen mata kuliah IF2211 Strategi Algoritma karena bimbingan dan ilmu yang disampaikan sangat bermanfaat dalam mewujudkan
makalah ini.

REFERENCES

[1] http://smartmanoj.blogspot.com/2014/05/solution-to-plastelina-family-crisis.html

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/stima19-20.htm

[3] https://www.youtube.com/watch?v=z9VN5fFo0QU

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, 3 Mei 2020

Ttd

Jundullah

1351802