Penerapan Algoritma Greedy pada Permainan Kartu 41

Nizamixavier Rafif Lutvie 13519085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: nizamilutvie@gmail.com

***Abstrak*—Algoritma greedy adalah salah satu strategi algoritma yang dapat digunakan untuk menyelesaikan persoalan optimalisasi. Makalah ini membahas penerapan algoritma greedy yang dapat digunakan sebagai strategi bermain pada salah satu jenis permainan kartu remi, yaitu 41.**

Kata kunci—permainan; kartu; 41; algoritma; greedy

I. PENDAHULUAN

Pada zaman sekarang, telah diciptakan berbagai barang sebagai sarana hiburan manusia untuk menghilangkan rasa kebosanan yang terdapat pada setiap orang, dengan salah satunya adalah kartu remi. Terdapat bermacam-macam variasi permainan yang dapat dilakukan dengan menggunakan kartu remi. Salah satu permainan yang cukup populer di Indonesia adalah permainan kartu 41. Jika dilihat sekilas, permainan kartu 41 memang sebagian besar mengandalkan keberuntungan untuk bisa memenangkannya. Akan tetapi, dapat dilakukan pendekatan yang lebih strategis untuk bisa meningkatkan kemungkinan seorang pemain untuk bisa mendapatkan hasil yang baik, bahkan maksimal.

II. DASAR TEORI

A. *Algoritma Greedy*

Algoritma greedy merupakan salah satu teknik untuk memecahkan persoalan optimasi agar menemukan solusi yang optimal. Algoritma memecahkan persoalan secara langkah per langkah, sehingga algoritma akan mengambil pilihan yang terbaik pada setiap langkah dan tidak bisa kembali pada langkah sebelumnya. Terdapat 6 komponen dalam algoritma greedy, yaitu:

1. Himpunan kandidat (C)

Himpunan kandidat adalah himpunan yang berisi kandidat yang akan dipilih pada setiap langkah.

2. Himpunan solusi (S)

Himpunan solusi adalah himpunan yang berisi kandidat yang sudah dipilih.

3. Fungsi solusi

Fungsi solusi adalah fungsi untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.

4. Fungsi seleksi (selection function)

Fungsi seleksi adalah fungsi untuk memilih kandidat berdasarkan strategi greedy tertentu.

5. Fungsi kelayakan (feasible)

Fungsi kelayakan adalah fungsi untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).

6. Fungsi obyektif

Fungsi obyektif adalah fungsi untuk mengoptimasi solusi, biasanya memaksimumkan atau meminimumkan.

Gambar 2.1. Ilustrasi Algoritma Greedy

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Terdapat beberapa langkah untuk menjalankan algoritma greedy. Pertama, perlu dibuat sebuah himpunan solusi kosong untuk menyimpan pilihan atau langkah yang telah diambil. Kemudian, perlu dicek apakah himpunan belum merupakan solusi permasalahan dan masih terdapat kandidat pilihan yang dapat diambil. Jika benar, akan dipakai fungsi seleksi untuk mencari kandidat pilihan yang akan diambil. Jika sudah ditemukan, keluarkan kandidat pilihan tersebut dari himpunan kandidat pilihan. Setelah itu, akan digunakan fungsi kelayakan untuk mengetahui apakah kandidat pilihan yang akan diambil dapat dimasukkan ke dalam himpunan solusi. Jika layak, kandidat akan dimasukkan ke dalam himpunan solusi. Proses tersebut akan berulang sampai himpunan solusi sudah merupakan solusi permasalahan atau sudah tidak terdapat kandidat yang dapat diambil. Jika himpunan solusi merupakan

solusi permasalahan, maka solusi telah ditemukan. Sedangkan jika himpunan solusi belum merupakan solusi permasalahan tetapi tidak terdapat solusi kandidat yang dapat diambil, maka tidak terdapat solusi bagi permasalahan.

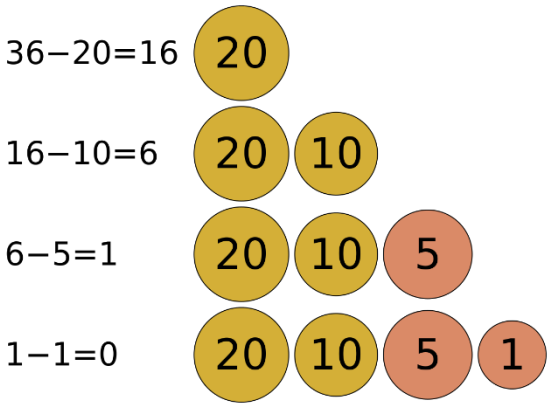
```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← {}      { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
      C ← C – {x}          { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
          S ← S ∪ {x}       { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then     { solusi sudah lengkap }
    return S
  else
    write('tidak ada solusi')
  endif
```

Gambar 2.2. Langkah Algortima Greedy

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Algoritma greedy bersifat heuristik, sehingga hanya akan membentuk solusi yang paling optimal secara lokal tetapi tidak selalu menghasilkan solusi yang paling optimal secara global. Hal ini disebabkan oleh strategi algoritma yang tidak beroperasi secara menyeluruh terhadap seluruh kemungkinan solusi yang ada serta adanya beberapa kemungkinan fungsi seleksi yang berbeda sehingga harus memilih fungsi yang tepat untuk mendapatkan solusi yang paling optimal.Algoritma greedy dapat digunakan untuk menghasilkan solusi hampiran atau solusi yang mendekati solusi optimal jika waktu komputasi yang lebih pendek lebih penting daripada mendapatkan solusi yang terbaik. Berikut adalah contoh-contoh persoalan yang dapat diselesaikan dengan menggunakan Algoritma greedy.

1. Persoalan penukaran uang (coin exchange problem)
2. Persoalan memilih aktivitas (activity selection problem)
3. Minimisasi waktu di dalam sistem
4. Persoalan knapsack (knapsack problem)
5. Penjadwalan Job dengan tenggat waktu (job schedulling with deadlines)
6. Pohon merentang minimum (minimum spanning tree)
7. Lintasan terpendek (shortest path)
8. Kode Huffman (Huffman code)
9. Pecahan Mesir (Egyptian fraction)



Gambar 2.3. Ilustrasi Persoalan Penukaran Koin Menggunakan Algoritma Greedy

Sumber : https://en.wikipedia.org/wiki/Greedy_algorithm

B. Permainan Kartu 41

Permainan kartu 41 adalah adalah salah satu permainan kartu remi yang cukup dikenal di Indonesia. Permainan kartu 41 dilakukan dengan menggunakan 1 set kartu remi yang berisi 52 kartu tanpa joker. Permainan dilakukan oleh 4 orang. Tujuan dari permainan ini adalah mendapatkan skor tertinggi yaitu 41. Setiap kartu memiliki nilai yang berbeda. Berikut adalah perhitungan skor masing-masing kartu.

- Kartu A bernilai 11 poin
- Kartu King, Queen, Jack dan, 10 bernilai 10 poin
- Kartu 2 sampai 9 bernilai sesuai dengan angkanya masing-masing

Selain nilai sebuah kartu, suit atau jenis kartu juga mempengaruhi penghitungan skor. Untuk bisa mendapatkan skor yang tinggi, diperlukan kartu dengan jenis yang sama yang kemudiann akan dijumlahkan. Jika terdapat kartu yang jenisnya bebeda, kartu tersebut justru akan menguangi nilai total kartu di tangan pemain. Contohnya, terdapat 5 wajik, 6 wajik, 7 wajik, dan 10 hati. Karena jenis utama adalah wajik dan 10 hati adalah jenis kartu yang berbeda, maka skor total pemain adalah 5 + 6 + 7 – 10 = 8.

Gambar 1. Permainan Kartu 41

Sumber : https://www.bluestacks.com/id/blog/game-guides/higgs-domino-island/higgs-domino-island-kartu-41-id.html

Permainan dimulai dengan mengocok kartu lalu membagikan 4 kartu kepada masing-masing pemain. Sisa kartu pada dek diletakkan di tengah-tengah para pemain. Giliran pertama ditentukan secara acak dan pergantian giliran dilakukan searah dengan jarum jam. Pada setiap giliran, seorang pemain dapat mengambil sebuah kartu baru dari dek atau mengambil kartu yang dibuang oleh pemain lain pada giliran sebelumnya. Pemain biasanya akan mencoba mengumpulkan kartu-kartu dengan jenis yang sama yang memiliki nilai yang tinggi. Permainan dijalankan hingga terdapat pemain yang mencapai 41, yang artinya pemain tersebut memiliki 4 kartu di antara King, Queen, Jack, dan 10 serta sebuah kartu As dengan jenis yang sama. Jika kartu pada dek habis sebelum terdapat pemain yang mencapai 41, akan dilakukan penghitungan dari kartu setiap pemain dan pemenang ditentukan dari pemain yang memiliki skor yang tertinggi.

III. IMPLEMENTASI

Penulis menerapkan konsep Algoritma greedy pada permainan kartu 41. Ditentukan terlebih dahulu komponen pada algoritma greedy. Perlu diingat bahwa pada penerapan yang dilakukan, terdapat dua kali implementasi Algoritma greedy, yaitu pada saat mengmbil kartu dan pada saat membuang kartu

1. Himpunan kandidat (C)

Himpunan kandidat pada saat mengambil kartu adalah kartu buangan dan kartu pada dek, sedangkan himpunan kandidat pada saat membuang kartu adalah kartu yang ada di tangan.

2. Himpunan solusi (S)

Himpunan solusi pada saat mengambil kartu adalah kartu yang ada di tangan, sedangkan himpunan solusi pada saat membuang kartu adalah kartu -kartu yang telah dibuang.

3. Fungsi solusi

Fungsi solusi pada saat mengambil kartu sama dengan fungsi solusi pada saat membuang kartu, yaitu memeriksa apakah nilai kartu di tangan sudah mencapai 41.

4. Fungsi seleksi (selection function)

Fungsi seleksi pada saat mengambil kartu adalah memilih kartu yang memiliki jenis yang sama dengan *majority suit*, sedangkan fungsi solusi pada saat membuang kartu adalah memilih kartu yang paling tidak signifikan (kartu non-majority suit yang memiliki nilai terkecil).

5. Fungsi kelayakan (feasible)

Fungsi kelayakan pada saat mengambil kartu sama dengan fungsi kelayakan pada saat membuang kartu, yaitu kartu di tangan tidak “terbakar” atau memiliki kartu dengan jenis yang berbeda-beda pada akhir permainan.

6. Fungsi obyektif

Fungsi obyektif pada saat mengambil kartu sama dengan fungsi objektif pada saat membuang kartu, yaitu nilai kartu di tangan maksimum.

Penulis membuat algoritma penerapan strategi greedy menggunakan python 3. Pertama-tama, dilakukan inisiasi komponen-komponen seperti dek, kartu buangan, dan kartu di tangan.

```
# inisiasi

# deck adalah tumpukan kartu awal
# asumsi deck sudah diacak
deck = [[0,0] for i in range(52)]
# trash adalah kartu terbaru yang dibuang m
usuh yang dapat diambil oleh pemain
trash = [0,0]
# hand adalah array berisi kartu yang setia
p elemen kartu terdiri dari suit kartu dan
nilai kartu
# kamus suit kartu : sekop = 1, hati = 2, c
lub = 3, wajik = 4
# kamus nilai kartu : 2,3,4,5,6,...11 (J,Q,
K bernilai 10; As bernilai 11)
hand = []
# goal adalah suit kartu yang dijadikan tuj
uan pencarian
goal = 0

# inisiasi kartu pemain
for i in range(4):
    hand.append(deck[0])
    deck.pop(0)
```

Kemudian dibuat fungsi untuk menghitung jumlah kartu dengan jenis yang sama serta fungsi untuk menghitung jumlah nilai kartu dengan jenis yang sama.

```
# menghitung total nilai pada sebuah suit
def countValue(x,suit):
    total = 0
    for i in range(len(x)):
        if(x[i][0] == suit):
            total = total + x[i][1]
    return total



# menghitung jumlah kartu pada sebuah suit
def countSuit(x,suit):
```

```
    total = 0
    for i in range(len(x)):
        if(x[i][0] == suit):
            total = total + 1
    return total
```

Setelah itu, dibuat sebuah fungsi untuk menentukan suit atau jenis kartu mana yang akan menjadi target pemain untuk didapatkan.

```
# mengembalikan suit yang merupakan majority suit,
# yaitu suit memiliki nilai yang paling besar
# jika terdapat suit dengan nilai yang sama,
# diambil suit yang memiliki jumlah kartu terbanyak
def majoritySuit(x):
    hasil = []
    maksV = max(countValue(x,1),countValue(x,2),countValue(x,3),countValue(x,4))
    for i in range(1,5):
        if(maksV == countValue(x, i)):
            hasil.append(i)
    if(len(hasil) > 1):
        jumlahhasil = []
        for i in range(len(hasil)):
            jumlahhasil[i] = countSuit(x,hasil[i])
        maksS = max(jumlahhasil)
        for i in range(len(jumlahhasil)):
            if(jumlahhasil[i] == maksS):
                return hasil[i]
    else:
        return hasil[0]
```

Selanjutnya, dibuat sebuah fungsi untuk menentukan kartu manakah dari tangan pemain yang harus dibuang.

```
# mengembalikan kartu non-majority suit yang memiliki nilai paling kecil
# jika suit semua kartu sama, dibuang kartu yang jumlah kartu suit nya paling kecil
def removeCard(x):
    removed = [0,12] #nilai sebagai awalan
    major = majoritySuit(x) #kalkulasi majority suit saat ini
    for i in range(len(x)):
        if(x[i][0] != major and x[i][1] <= removed[1]):
            if(x[i][1] == removed[1]):
                if(countSuit(x,x[i][0]) < countSuit(x,removed[0])):
                    removed = x[i]
            else:
                removed = x[i]
    if(removed[0] == 0):
        for i in range(len(x)):
            if(x[i][1] < removed[1]):
                removed = x[i]
    return removed
```

Barulah dimulai penggunaan fungsi-fungsi tersebut di dalam algoritma pengoptimalan.

```
# jika masih ada kartu pada deck dan jumlah nilai di tangan belum 41
while(deck >=0 and count(hand) != 41):
    goal = majoritySuit(hand)
    if(trash[0] == goal): # jika kartu buangan yang dapat diambil sesuai dengan suit goal
        hand.append(trash)
        trash.pop(0)
    else:
        hand.append(deck[0])
        deck.pop(0)

    # membuang kartu non-majority suit yang memiliki nilai paling kecil
    # jika suit semua kartu sama, membuang kartu yang bernilai paling kecil
    hand.remove(removeCard(hand))
```

IV. KESIMPULAN

Algoritma greedy merupakan strategi pengoptimalan persoalan yang cukup sederhana dan mudah ntuk diimplementasikan. Algoritma greedy dapat digunakan untuk menjalankan permainan kartu 41 agar mendapatkan hasil yang mendekati maksimal.

V. UCAPAN TERIMA KASIH

Puji syukur terhadap Tuhan Yang Maha Esa karena berkat rahmat-Nya, penulis dapat menyelesaikan tugas ini dengan baik. Penulis juga mengucapkan terima kasih kepada Ibu Nur Ulfa Maulidevi selaku dosen pembimbing dalam mata pelajaran Strategi Algoritma. Tidak lupa juga penulis mengucapkan terima kasih kepada keluarga penulis atas bantuan dan dukungan yang diberikan selama proses pembuatan tugas.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf diakses pada 10 Mei 2021 pukul 15.32 WIB.

[2] https://www.bluestacks.com/id/blog/game-guides/higgs-domino-island/higgs-domino-island-kartu-41-id.html diakses pada 10 Mei 2021 pukul 16. 37 WIB.

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.

Surabaya, 10 April 2021

Nizamixavier Rafif Lutvie 13519085