Aplikasi Permainan Kartu 24 dengan Algoritma Greedy

LAPORAN TUGAS BESAR I

IF-2211 STRATEGI ALGORITMA

Disusun oleh :

Johanes	13517012
Steve Andreas Immanuel	13517039
Vinsen Marselino Andreas	13517054

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2019

BAB I
DESKRIPSI TUGAS

Dalam permainan kartu 24, terdapat dek (tumpukan) 52 kartu remi. Permainan akan memilih 4 kartu secara acak, lalu setiap pemain akan mencari solusi 24 game dari ke-4 kartu tersebut. Nilai yang mungkin dari sebuah kartu adalah 1 (as), 2, …, 10, 11 (jack), 12 (queen), dan 13 (king). Operator yang dapat dipilih + - * / ( ), dan hasil akhir sedekat mungkin dengan nilai 24. Selisih nilai ekspresi solusi dengan 24 akan menjadi pengurang.

Dalam tugas besar ini, setiap tim wajib merancang dan mengimplementasikan strategi greedy untuk memberikan solusi dalam permainan ini. Karena algoritma greedy membentuk solusi langkah per langkah (step by step), harus ditentukan urutan pemilihan operand, urutan pemilihan operator, dan penggunaan variasi kurung. Tidak boleh menggunakan strategi lain selain greedy.

Fungsi objektif persoalan ini adalah memaksimalkan skor utk ekspresi solusi yang dihasilkan. Seperti scrabble, setiap operator akan memiliki skor. Semakin kompleks operatornya, skor semakin kecil. Skor setiap operator didefinisikan 5 untuk +, 4 untuk -, 3 untuk *, dan 2 untuk /, serta -1 untuk setiap pasang kurung (). Selain skor, operator * dan / memiliki derajat lebih tinggi dibandingkan + dan -, artinya operator berderajat lebih tinggi akan diproses terlebih dahulu. Ekspresi a+b*c-d akan diproses seperti (a+(b*c))-d. Skor akhir setiap ekspresi adalah total skor dari operator dikurangi jumlah pasang kurung dan selisih dengan 24.

Setiap tim akan membuat satu engine backend yang menghasilkan ekspresi solusi berdasarkan masukan 4 angka, dan dua front-end. Front-end pertama berupa GUI yang mendemokan proses pengambilan 4 kartu untuk memberikan input, dan menampilkan hasilnya. Visualisasi kartu untuk demo boleh menggunakan library. Front-end kedua membaca file masukan, memproses 4 angka dari file masukan, dan menghasilkan file keluaran. Front-end kedua akan berinteraksi dengan lingkungan permainan untuk kompetisi antar tim.

Lingkungan permainan akan mengeluarkan 4 kartu secara acak. Setiap pemain akan memberikan jawaban masing-masing dan mendapatkan total skor berdasarkan operator yang digunakan. Jika tidak bisa diselesaikan, keempat kartu dikembalikan ke deck dengan urutan acak. Jika pemain memberikan ekspresi yang salah, akan diberikan nilai -10. Permainan diulang sampai dengan dek habis, dan tim pemenang adalah tim dengan skor tertinggi. Aplikasi pemain tidak diperbolehkan membuat cache solusi dari kombinasi supaya lebih cepat, karena akan dibandingkan waktu eksekusi dari implementasi strategi greedy.

BAB II
DASAR TEORI

2.1. Algoritma Greedy

Dalam memecahkan suatu persoalan algoritma, tidak ada teknik yang dapat digunakan untuk menyelesaikan semua persoalan di dunia. Beberapa teknik yang terkenal adalah *Divide and Conquer, Dynamic Programming, Backtracking, Greedy,* dan sebagainya. Kali ini, kami menggunakan salah satu teknik yang bernama greedy.

Algoritma greedy, seperti namanya, adalah algoritma yang rakus. Maksud dari rakus disini adalah dengan memilih suatu pilihan yang merupakan optimum lokal. Hal ini terus dilakukan dalam setiap langkah dengan harapan pada akhir nanti akan memperoleh hasil yang optimum global.

Dalam algoritma greedy, ada lima buah elemen utama, yaitu himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif. Algoritma greedy biasanya digunakan untuk mencari nilai optimasi(maksimum atau minimum) dari suatu permasalahan.

Beberapa pemanfaatan greedy adalah untuk mencari *minimum spanning tree(MST),* algoritma Dijkstra(mencari jarak terpendek), kompresi Huffman, dan masalah penjadwalan.

2.2. Permainan 24

Permainan 24 merupakan permainan yang bertujuan untuk menciptakan angka 24 dari empat buah angka. Dalam kasus ini, angka yang dipakai adalah dari angka satu(as) sampai tiga belas(king). Operator yang boleh dipakai pemain untuk membuat angka 24 adalah +,-,*,/,dan( ). Setiap angka harus dipakai tepat satu kali, dengan 3 buah operator biasa, dan operator kurung dapat sebanyak apapun. Sebagai contoh, saat mendapat angka 1,2,3, dan 4, alternatif jawaban yang dapat dipilih adalah 1 * 2 * 3 * 4 = 24.

BAB III
PEMANFAATAN STRATEGI GREEDY

3.1 Strategi Greedy

Seperti yang telah dijelaskan, algoritma greedy adalah algoritma yang mencari optimum lokal pada setiap langkah iterasinya dengan harapan akan mendapatkan optimum global di akhirnya. Terdapat lima elemen penting pada algoritma greedy, yaitu: himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif. Berikut ini adalah isi dari kelima elemen tersebut pada algoritma yang digunakan.

Himpunan kandidat = himpunan bilangan yang harus dioperasikan dan operatornya
Himpunan solusi = suatu persamaan matematika yang memenuhi fungsi kelayakan dan memiliki skor setinggi mungkin
Fungsi seleksi = memilih suatu angka dan operator dari himpunan kandidat
Fungsi kelayakan = menentukan apakah suatu persamaan layak dihitung
Fungsi objektif = mencari skor poin setinggi mungkin dengan hasil evaluasi sedekat mungkin dengan nilai 24

Algoritma greedy yang digunakan pada program ini adalah sebagai berikut.

1. Urutkan 4 buah bilangan yang dimasukkan menurun sehingga bilangan paling besar ada di paling depan
2. Ambil bilangan pertama dari urutan bilangan, hapus dari himpunan kandidat dan masukan ke himpunan solusi, dalam hal ini sebuah persamaan yang hanya berisi 1 angka
3. Ambil bilangan pertama dari urutan bilangan, coba membuat semua kemungkinan persamaan dengan bilangan tersebut dan semua operator yang ada (misalnya bilangan diletakan di depan, tengah, atau belakang dengan semua kemungkinan operator) dan disimpan pada suatu daftar
4. Cek apakah suatu persamaan dalam daftar tersebut valid dengan menggunakan fungsi kelayakan, jika layak maka carilah persamaan skor maksimum dari daftar tersebut berdasarkan skor yang telah ditentukan di deskripsi masalah. Namun pada algoritma ini terdapat suatu skalar pengali yang berfungsi untuk memperbesar bobot pengurangan pada bilangan yang tidak mencapai 24. Sehingga diharapkan skor maksimum yang terpilih merupakan persamaan yang lebih dekat dengan nilai 24
5. Simpan persamaan paling optimum sekarang
6. Hapus bilangan yang baru saja digunakan dari himpunan kandidat
7. Ulangi langkah 3 - 6, hingga himpunan kandidat telah habis
8. Pada saat mencari skor maksimum untuk bilangan keempat, ditambahkan semua kemungkinan posisi kurung, yang diharapkan dapat memperbesar skor yang ada

Selain greedy yang kami gunakan, awalnya kami mencoba greedy dengan berdasarkan kedekatannya dengan nilai 24. Namun jika menggunakan cara tersebut, kemungkinan besar persamaan akhir tidak akan dekat dengan nilai 24 sehingga kami mengganti algoritma tersebut. Ada pula ide awal kami tidak menggunakan kurung karena setelah menganalisis hasil brute force untuk masalah ini, kebanyakan solusi tidak menggunakan tanda kurung dan hanya memperkecil skor yang dihasilkan. Namun setelah diimplementasikan, ternyata penggunaan kurung di akhir dapat mengoptimasi nilai skor yang dihasilkan.

3.2 Spesifikasi Program

Program baru dicoba dijalankan pada sistem operasi berbasis Windows dan Linux dan dapat berjalan dengan lancar. Namun, pada sistem operasi Linux, ada bagian kode yang harus dihilangkan yaitu mengenai *icon* program. Hal ini dikarenakan pada Linux, *icon* program tidak dapat terlihat sehingga pengimplementasian *icon* program memang hanya dibuat untuk bekerja di Windows saja.

Program dapat dijalankan dengan menggunakan Python 3 di cmd atau dapat langsung menjalankan file .py yang ada. Program yang dapat langsung dijalankan adalah file GUI.py. Sedangkan untuk file batch03_kelompok31.py memerlukan 2 buah argumen lainnya yaitu file input yang berisi 4 buah bilangan yang dipisahkan dengan spasi dan file output yang akan menampilkan persamaan yang dihasilkan. Berikut adalah contoh pemanggilan file batch03_kelompok31.py.

```
python3 batch03_kelompok31.py input.txt output.txt
```

Untuk file GameEngine.py tidak dapat langsung dijalankan karena hanya merupakan back-end dari kedua file sebelumnya yang berisi fungsi-fungsi yang menghitung dan menjalankan algoritma greedy di atas.

Pastikan sebelum menjalankan GUI.py, sistem operasi yang digunakan telah ter-*install library* “tkinter” karena GUI.py menggunakan *library* tersebut.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

Berikut adalah *pseudocode* dari implementasi algoritma greedy yang digunakan :

```
{ Kelompok sin
  Kamis 31/1/19
  back-end
  menghitung solusi 24 solver dari input dengan algoritma greedy }

function calculate(input : list masukan) : list
   { fungsi mencari persamaan dengan point optimum untuk solusi
   permasalahan game 24 dari masukan yang ada }
   masukan.sort(reverse=True) { urutkan list masukan menurun }
   equation = [masukan[0]]
   { hapus masukan[0] dari daftar kandidat }
   masukan = masukan - masukan[0]

   while masukan != [] do
       maks = seleksi(masukan,equation)
       equation = maks[0]
       masukan.remove(maks[1])
   endwhile
   { yang direturn persamaan,total hasil evaluasi, dan point }
   equation = join(equation)
   { menggabungkan setiap karakter yang ada pada equation }
   return (equation,maks[2],maks[3])


function seleksi(input: list kandidat, input: list solusi_now):list
   { fungsi menyeleksi kandidat dan mengembalikan persamaan
   dengan point terbesar greedy by point }
   operator = '+-*/'
   angka = kandidat[0]
   daftar = list()
   for op in operator do  { coba untuk semua operator }
       for place in range(1,len(solusi_now)+1,2) do { operator,angka }
           { menggabungkan persamaan }
           temp_eq = solusi terbaru + operator + angka
           daftar = daftar U temp_eq
       endfor
       for place in range(0,len(solusi_now)+1,2) do { angka,operator }
           temp_eq = solusi terbaru + angka + operator
           daftar = daftar U temp_eq
```

```
        endfor
    endfor
    return greedy(daftar)


function countPoint(input : string exp, float result, float multiplier) :
float
    { menghitung point dari exp yang masuk dan result-nya, multiplier untuk
pengali selisih }
    point=-multiplier*abs(24-result)
    for char in exp: { nilai point untuk setiap operator }
        if char=='+' then
            point+=5
        else if char=='-' then
            point+=4
        else if char=='*' then
            point+=3
        else if char=='/' then
            point+=2
        else if char=='(' then
            point-=1
        endif
    endfor
    return point

function valid_eq(input : string equation) : boolean
    { mengecek apakah persamaan sudah valid yaitu yang
    tidak ada operator yang double tidak ada operator
    di depan dan tidak ada operator setelah ( }
    operator = '+-*/'
    flag = False
    if equation[0] in operator then { terdapat operator di awal }
        return False
    else
        for char in equation do
            if char in operator
                if flag then { terdapat operator setelah operator }
                    return False
                else
                    flag = True
                endif
            else if char == '(' then
                flag = True
            else
                flag = False
            endif
       endfor
```

```
    endif
    return True

function greedy(input : list daftar) : list {
    {
    fungsi yang mengembalikan persamaan dari daftar yang
    memiliki jumlah point terbesar
    dengan menggunakan kurung pada iterasi saat semua angka
    telah terpilih
    }
    braket2 = [()]
    braket4 = [(),(0, 4),(2, 6),(4, 8),(0, 6),(2, 8),(0, 4, 6, 10), (0, 5,1,
8), (0, 7,3, 8), (2, 7,3, 10), (2, 9,5, 10)] { menyimpan letak penyisipan
kurung }
    first = True
    for pasang in daftar do
        angka = pasang[1]
        for bracket in (braket4 if len(pasang[0])==7 else braket2) do
            exp=pasang[0].copy()
            for indexinsert,brac in zip(bracket,'()()') do { sisipkan kurung}
                exp.insert(indexinsert,brac)
            endfor
            equation = ''.join(exp)
            if (valid_eq(equation))
                hasil = eval(equation) { hitung hasil evaluasi ekspresi }
                if first then { iterasi pertama, inisialisasi maksimum }
                   max_point = countPoint(exp,hasil,3.6)
                   select_eq = exp
                   selected = angka
                   total = hasil
                   first = False
                else
                   temp_point = countPoint(exp,hasil,3.6)
                   { cek apakah solusi sekarang lebih baik dari solusi
                     Sebelumnya }
                   if (max_point<temp_point){
                      max_point = temp_point
                      select_eq = exp
                      selected = angka
                      total = hasil
                   endif
                endif
            else
                continue
            endif
        endfor
    endfor
```

```
    return [select_eq,selected,total,max_point]
}
```

4.2. Pengujian

Dalam mencari algoritma *greedy* yang paling menguntungkan, kami melakukan pengujian yaitu dengan men-*generate* semua solusi terbaik untuk semua kombinasi angka dengan algoritma *brute-force*. Setelah itu, kami mencari pola yang terdapat pada solusi-solusi *brute-force* dan mengimplementasikannya pada algoritma *greedy* yang akhirnya digunakan. Setelah dilakukan analisis dan perbandingan antara solusi *brute-force* dan *greedy* dari semua kombinasi angka yang mungkin, strategi *greedy* yang paling menguntungkan adalah *greedy by point*. Artinya dalam setiap langkah akan dipilih solusi optimum lokal yang menghasilkan poin paling besar.

Strategi greedy dalam pemecahan masalah 24 Game ini tidak akan selalu menghasilkan solusi paling baik. Hal ini dikarenakan tidak dimungkinkannya melakukan *backtrack* pada algoritma *greedy* sehingga ketika sudah didapatkan solusi optimum lokal, solusi tersebut tidak dapat diubah dan akan berdampak pada solusi optimum global di akhir. Contoh kasusnya adalah ketika angka yang diberikan adalah 1, 6, 6, dan 8. Pada solusi secara *greedy* akan didapatkan 8+1+6+6 dengan hasil evaluasi 21. Hal ini dikarenakan algoritma *greedy* akan memilih angka dan operator yang menghasilkan poin terbesar. Padahal secara *brute-force* dapat diperoleh solusi 6/(1-6/8) yang menghasilkan nilai 24 dan poin yang juga lebih tinggi.

Salah satu contoh kumpulan angka yang menghasilkan nilai sama dengan metode brute force adalah 1, 2, 3, dan 4. Pertama, algoritma greedy akan memilih angka 4 dan mengalikannya dengan 3 karena perhitungan itulah yang menghasilkan angka paling dekat dengan 24. Setelah itu, program akan mencari hubungan antara angka 12 dari 4*3 dengan bilangan lain. Maka dipilihlah 4*2*3 yang menghasilkan 24, dan terakhir dipilih 4*1*2*3 yang bernilai 24.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kelompok kami sudah berhasil menyelesaikan Tugas Besar ini dengan cukup baik. Walaupun apabila dibandingkan dengan strategi *brute-force* yang akan menemukan seluruh jawaban dengan benar masih cukup banyak perbedaannya. Selain dari *game engine* yang sudah cukup baik, kami juga sudah menyelesaikan *frontend* berupa GUI (*Graphical User Interface)* yang dapat menampilkan visual berupa kartu yang dapat bergerak. Kami juga sudah melakukan penggabungan antara *game engine* dan GUI dengan baik.

Frontend 2 dapat dijalankan dengan baik, yaitu membaca suatu masukan dari sebuah file untuk kemudian dituliskan hasil dari *game engine* ke dalam sebuah file yang lain.

5.2. Saran

- Untuk pengembangan selanjutnya, strategi algoritma greedy yang dipakai dapat dikembangkan dengan lebih baik dan dengan pendekatan yang berbeda. Seperti *greedy by operator*, ataupun *greedy by value*.
- Selain itu, dapat pula dikembangkan rumus terbaik yang dapat dipakai untuk memaksimalkan greedy yang dipakai dan memperjelas objektif yang akan dicapai.

Daftar Pustaka

http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf

https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/

https://www.tutorialspoint.com/python/python_gui_programming.htm