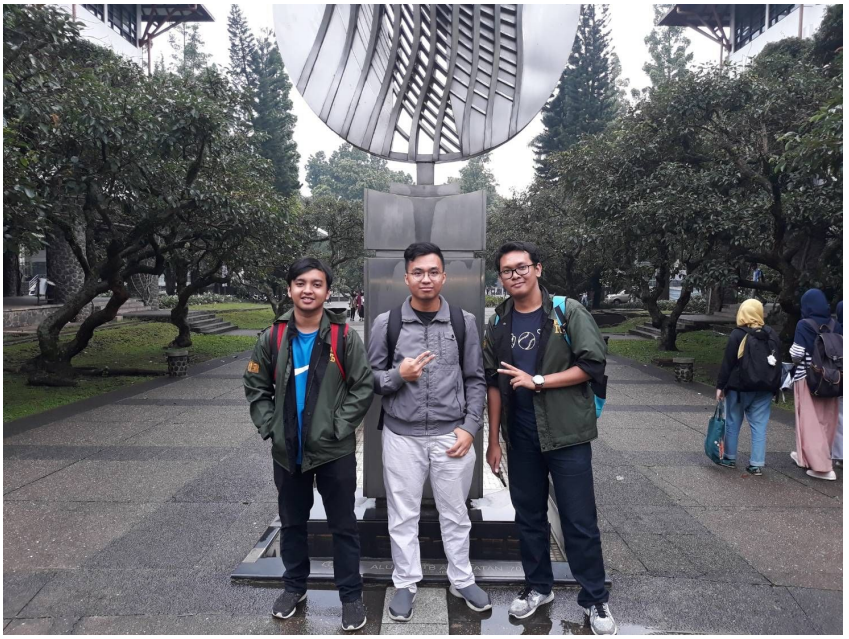
LAPORAN TUGAS BESAR I IF2211 STRATEGI ALGORITMA
PENYELESAIAN 24 GAME DENGAN ALGORITMA GREEDY

Laporan Ini Disusun Dalam Rangka Memenuhi
Tugas Besar 1 Mata Kuliah IF2211 Strategi Algoritma



Disusun Oleh :

13517032 - Putu Gde Aditya T. W
13517089 - Bram Musuko P
13517128 – Yudy Valentino

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2019

DAFTAR ISI

BAB I DESKRIPSI TUGAS BESAR

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

A. 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!”. Maksud dari prinsip tersebut adalah sebagai berikut : pada tiap langkah algoritma greedy, kita mengambil keputusan optimal yang bisa didapat saat itu tanpa mempertimbangkan konsekuensi selanjutnya. Pengambilan keputusan tersebut dinamakan optimum lokal, kemudian diharapkan pada setiap pengambilan optimum lokal dapat tercapainya optimum global, yaitu solusi optimum yang melibatkan keseluruhan langkah dari awal hingga akhir.

Algoritma greedy merupakan algoritma yang bersifat heuristik, mencari nilai maksimal sementara dengan harapan akan mendapatkan solusi yang cukup baik. Meskipun tidak selalu mendapatkan solusi terbaik (optimum), algoritma greedy umumnya memiliki kompleksitas waktu yang cukup baik, sehingga algoritma ini sering digunakan untuk kasus yang memerlukan solusi cepat meskipun tidak optimal seperti sistem real-time atau game.

Elemen-elemen algoritma greedy:

1. **Himpunan kandidat, C**
   Himpunan yang berisi elemen pembentuk solusi.
2. **Himpunan solusi, S**
   Himpunan yang terpilih sebagai solusi persoalan.
3. **Fungsi seleksi**
   Fungsi yang memilih kandidat yang paling mungkin untuk mencapai solusi optimal
4. **Fungsi kelayakan**
   Fungsi yang memeriksa apakah suatu kandidat yang dipilih dapat memberikan solusi yang layak. Maksudnya yaitu apakah kandidat tersebut bersama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada.
5. **Fungsi Objektif**
   Fungsi yang mengoptimalkan solusi.

B. Permainan Kartu : 24 Game

Dalam permainan kartu 24, terdapat dek (tumpukan) 52 kartu remi. Nilai yang mungkin dari sebuah kartu adalah 1 (as), 2, …, 10, 11 (jack), 12 (queen), dan 13 (king).Permainan akan memilih 4 kartu secara acak, lalu setiap pemain akan mencari solusi 24 game dari ke-4 kartu tersebut. permainan kartu 24 ini memakai operator matematika dalam permainannya yaitu + - * /, pemain yang lebih dahulu menemukan solusi mendapatkan poin dan yang mempunyai poin paling sedikit akan menerima hukuman sesuai dengan kesepakatan tiap-tiap pemain.

BAB III PEMANFAATAN STRATEGI GREEDY

Implementasi greedy untuk permasalah 24 yang digunakan oleh kelompok kami adalah pertama-tama setelah mendapatkan inputan 4 buah angka, maka program akan melakukan fungsi sort bawaan library python. Sort disini disimpan di 2 array yang berbeda, satu array untuk menyimpan sort yang berisi terurut menaik dan yang satu lagi untuk menyimpan isi yang terurut mengecil. Hal ini dilakukan karena program ini akan melakukan proses greedy sebanyak 2 kali, walaupun strategi yang dilakukan sama, tetapi yang berbeda adalah data yang digunakan untuk memulai proses greedy.

Greedy yang pertama digunakan dengan memanfaatkan array sort yang terurut menaik. Pada putaran pertama program akan mengambil bilangan terbesar (indeks terakhir), setelah itu program akan mengecek berapa skor maksimal yang akan ia dapat jika menggunakan operator '+','-','*',-' dengan memakai operan bilangan terbesar ke 2. setelah itu program akan menentukan operan yang paling maksimal dan menyimpannya dalam array, hasil dari perhitungan itu akan digunakan lagi untuk menghitung bilangan selanjutnya (bilangan terbesar ke 3), hal tersebut akan diulangi sebanyak 3 kali. Khusus pada loop terakhir jika didapatkan perhitungan yang bisa menghasilkan nilai 24, maka program akan otomatis keluar dari loop, karena nilai 24 lebih diutamakan dari apapun di dalam game ini.

Pada greedy kedua, konsepnya sama dengan greedy pertama namun data yang digunakan terurut mengecil sehingga program akan memproses dari bilangan yang paling minimum. Untuk penggunaan operator kurung, operator tersebut hanya akan digunakan jika operator yang digunakan sebelumnya adalah '+' atau '-' dan operator yang digunakan sekarang '*' atau '/'. Sehingga jika ada kejadian tersebut maka boolean kurung akan dinyalakan dan poin akan dikurangi 1.

Setelah melakukan 2 greedy, program akan membandingkan mana hasil yang lebih besar. Jika ada yang memiliki nilai 24, maka persamaan itu akan diprioritaskan. Jika keduanya bernilai 24 atau tidak ada yang bernilai 24, maka hasil poin perhitungan akan diprioritaskan.

BAB IV IMPLEMENTASI DAN PENGUJIAN

A. Implementasi Program

main2.py

```
import sys
import math

#Program ini bisa dipanggil dan menampilkan hasil hanya dengan menuliskan
py main2.py <namaFileInput.txt> <namaFileOutput.txt> pada command prompt

def count(x,y,i):#Prosedur untuk menghitung suatu ekspresi
     if(i == 3):
          return(x+y)
     elif (i == 2):
          return(x-y)
     elif (i == 1):
          return(x*y)
     else:
          return(x/y)

def convert(i):#Prosedur untuk mengonversi suatu operator menjadi nilainya
     if(i == 0):
          return("/")
     elif(i == 1):
          return("*")
     elif(i == 2):
          return("-")
     else:
          return("+")

def tulis(arrO,arrt,nilai):#Prosedur untuk menuliskan hasil jawaban
     if(arrO[1] < 2 and arrO[0] >1): # operasi berbentuk (a opr1 b) opr2 c
opr3 d
          jawaban                                                    =
"("+str(arrt[3])+convert(arrO[0])+str(arrt[2])+")"+convert(arrO[1])+str(ar
rt[1])+convert(arrO[2])+str(arrt[0])+" = "+str(nilai)
     elif (arrO[2] < 2 and arrO[1] > 1): # operasi berbentuk (a opr1 b
opr2 c) opr3 d
          jawaban                                                    =
"("+str(arrt[3])+convert(arrO[0])+str(arrt[2])+convert(arrO[1])+str(arrt[1
])+")"+convert(arrO[2])+str(arrt[0])+" = "+str(nilai)
     else :
          jawaban                                                    =
str(arrt[3])+convert(arrO[0])+str(arrt[2])+convert(arrO[1])+str(arrt[1])+c
```

```
onvert(arrO[2])+str(arrt[0])+" = "+str(nilai)
     return jawaban

def total(arrO,nilai,kurung):#Menuliskan score point dari suatu ekspresi
dalam bentuk string

     if (kurung):
          total = str(-abs(24-nilai)+sum(arrO)+6-1)
     else :
          total = str(-abs(24-nilai)+sum(arrO)+6)
     return total

def poin(arrO,nilai,kurung):#Menuliskan score point dari suatu ekspresi
dalam bentuk integer
     if(kurung):
          return(-abs(24-nilai)+sum(arrO)+6-1)
     else :
          return(-abs(24-nilai)+sum(arrO)+6)

def solveProblem(): #Prosedur untuk mencari solusi terbaik dari 4 kartu
yang telah diinput
        FInput = open(sys.argv[1],"r")
        arr = [] # menampung angka yang akan dihitung
        for line in FInput:
            arr.append(int(line))
        arr.sort()
        arrs = arr.copy()
        arrs.reverse()
        arrt2 = arrs.copy()
        arrt = arr.copy() # menampung isi array original
        operatorSebelum = 0
        arrO = []
        nilai = 0.0
        kurung = False
        for i in range(3):
            point = -100
            for j in range (3,-1,-1):
                kurungTemp = False
                x = count(arr[3-i],arr[2-i],j)
                temp = j-abs(24-x)+2
                if((j == 0 or j == 1) and (operatorSebelum >1)):
                    temp = temp - 1
                    kurungTemp = True
                if (x == 24 and i == 2):
                    point = temp
                    nilai = x
                    operator = j
                    if(kurungTemp):
```

```
                    kurung = True
                break
            elif((temp)>point):
                point = temp
                nilai = x
                operator = j
                if(kurungTemp):
                    kurung = True
        operatorSebelum = operator
        arr[2-i] = nilai
        arrO.append(operator)
    operatorSebelum = 0
    arrO2 = []
    nilai2 = 0.0
    kurung2 = False
    for i in range(3):
        point2 = -100
        for j in range (3,-1,-1):
            kurungTemp = False
            x = count(arrs[3-i],arrs[2-i],j)
            temp = j-abs(24-x)+2
            if((j == 0 or j == 1) and (operatorSebelum >1)):
                temp = temp - 1
                kurungTemp = True
            if (x == 24 and i == 2):
                point2 = temp
                nilai2 = x
                operator = j
                if(kurungTemp):
                    kurung2 = True
                break
            elif((temp)>point2):
                point2 = temp
                nilai2 = x
                operator = j
                if(kurungTemp):
                    kurung2 = True
        operatorSebelum = operator
        arrs[2-i] = nilai2
        arrO2.append(operator)
    point = poin(arrO,nilai,kurung)
    point2 = poin(arrO2,nilai2,kurung2)
    if(nilai == 24 and nilai2 == 24):
        if(point > point2):
            jawaban = tulis(arrO,arrt,nilai)
            hasil = total(arrO,nilai,kurung)
        else :
            jawaban = tulis(arrO2,arrt2,nilai2)
```

```
                hasil = total(arrO2,nilai2,kurung2)
        elif(nilai == 24):
            jawaban = tulis(arrO,arrt,nilai)
            hasil = total(arrO,nilai,kurung)
        elif(nilai2 == 24):
            jawaban = tulis(arrO2,arrt2,nilai2)
            hasil = total(arrO2,nilai2,kurung2)
        else :
            if(point > point2):
                jawaban = tulis(arrO,arrt,nilai)
                hasil = total(arrO,nilai,kurung)
            else:
                jawaban = tulis(arrO2,arrt2,nilai2)
                hasil = total(arrO2,nilai2,kurung2)
        FOutput = open(sys.argv[2],"w")
        FOutput.write(jawaban)
        FOutput.write("\nScore : ")
        FOutput.write(hasil)
        FOutput.write("\n")
        FInput.close()
        FOutput.close()

def main():
    solveProblem()

if __name__ == '__main__':
     main()
```

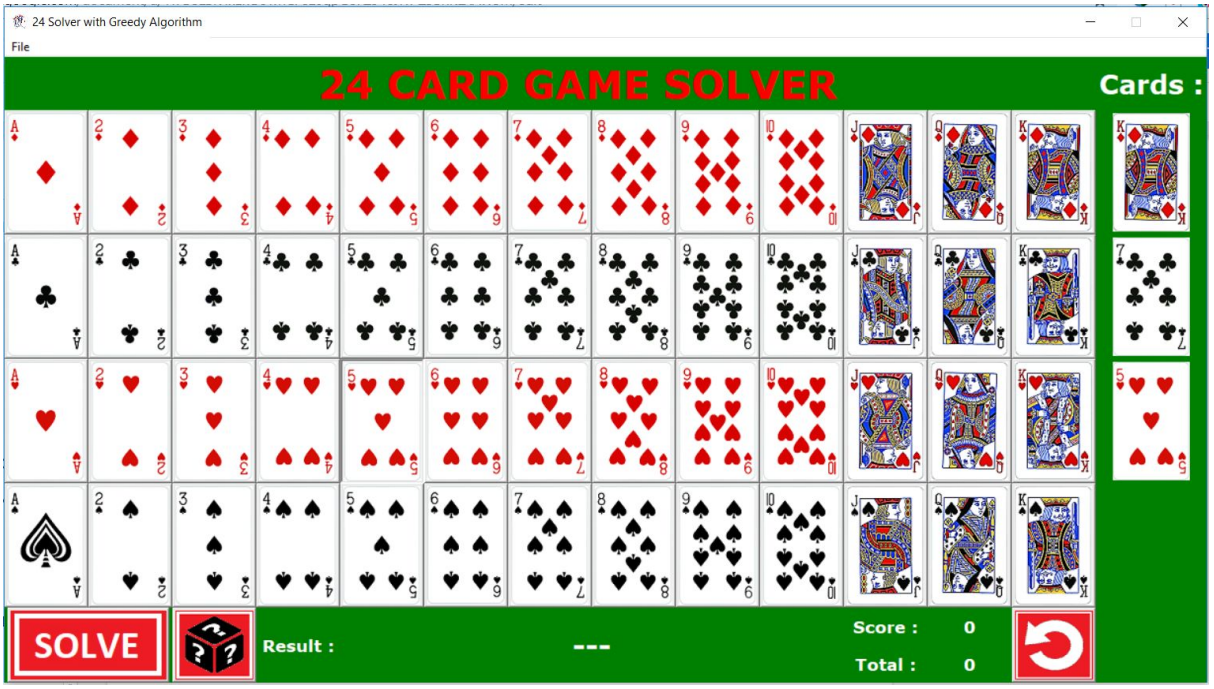
B. Pengujian Program

Pada bagian ini, kami akan menguji program ini baik dari bagian GUI, maupun bagian program dengan argumen file input dan file output. Berikut adalah tampilan dari GUI program kami :

Pada GUI kami terdapat beberapa elemen, yaitu :

a. Tombol untuk memilih kartu
   Tombol-tombol ini adalah tombol untuk memilih kartu yang diinginkan. Ada 52 kartu yang bisa dipilih dalam permainan ini.
b. Tombol random
   Tombol ini berfungsi untuk memilih 4 kartu secara acak
c. Tombol solve
   Tombol ini berfungsi untuk mencari solusi paling efektif dari 4 kartu yang telah dipilih.
d. Tombol reset
   Tombol ini berfungsi untuk mengulang kembali permainan dari awal
e. Tombol import dan export
   Tombol ini berfungsi untuk menerima masukan dari file eksternal dan memasukkan hasilnya pula pada file eksternal. Tombol ini berada pada menu File di sebelah kiri atas

Berikut adalah beberapa screenshot mengenai GUI kami :



Ini adalah pemilihan kartu secara manual oleh user.

Ini adalah 4 kartu yang dipilih secara acak oleh program.

Ini adalah tampilan dari solusi 4 kartu yang telah dipilih sebelumnya.

Berikut adalah hasil dari random beberapa angka acak beserta hasilnya :

Kartu : 5, Jack, 9, Jack
Nilai : 5, 11, 9, 11

Pada kasus ini, menurut kami sudah cukup efektif, karena sebetulnya tidak ada solusi untuk kombinasi angka 5 11 9 11, sehingga program mencarikan ekspresi dengan hasil perhitungan terdekat dari 24 dengan poin tertinggi, yaitu 11+11+9-5 = 26 dengan poin 12.

Kartu : 8, Ace, 2, 6
Nilai : 8, 1, 2, 6

Pada kasus ini, menurut kami ekspresi (1+2)*6+8 = 26 kurang efektif, karena kombinasi 8 1 2 6 masih bisa menghasilkan 24 dengan operator yang sama, yaitu ekspresi ((8 + 1) * 2) + 6 = 24. Algoritma greedy ini kurang efektif pada kasus ini karena algoritma greedy yang kami buat melakukan sorting terlebih dahulu, sehingga jika solusinya merupakan angka dengan urutan secara acak (solusi ini menggunakan angka 8 kemudian angka 1), maka akan sulit bagi greedy kami untuk menemukannya.

Kartu : 4, 3, 7, King
Nilai : 4, 3, 7, 13

Pada kasus ini, menurut kami ekspresi 13+7+4+3 = 27 sudah cukup efektif, karena sebetulnya tidak ada solusi untuk kombinasi angka 4 3 7 13, sehingga program mencarikan ekspresi dengan hasil perhitungan terdekat dari 24 dengan poin tertinggi, yaitu 13+7+4+3 = 27 dengan poin 12.

BAB V KESIMPULAN DAN SARAN

A. Kesimpulan

Dalam menyelesaikan tugas besar ini, kami berhasil membuat program penyelesaian permaian kartu 24 dengan mengimplementasikan algoritma greedy dalam mencari solusi yang mendekati hasil solusi 24 dengan total nilai operator yang maksimum. Kami juga berhasil membuat sebuah *Graphical User Interface* program yang mampu untuk melakukan random kartu dari 52 kartu yang ada lalu melakukan solve, program kami juga mampu membaca import file eksternal dan melakukan solve terhadap file tersebut dan membuat file eksternal untuk hasil-hasil penyelesaian permainan kartu 24. Program kami juga dapat dipanggil dengan environment argumen untuk nama file input dan nama file output.

B. Saran

Melalui tugas besar ini kami mendapat kesempatan belajar untuk mengimplementasikan materi kuliah yang sudah kami dapat di kelas IF 2211 Strategi Algoritma. Untuk proses pengerjaan, bagian management waktu menjadi bagian yang cukup sulit karena jadwal yang sudah mulai padat di minggu-minggu awal perkuliahan ini, tapi selain memperdalam mengenai algoritma greedy kami juga belajar akan management waktu dengan baik, sehingga kami mampu menyelesaikan tugas ini dengan tepat waktu dan juga kami lebih terdorong untuk mengggali lebih jauh mengenai library dalam python untuk menyelesaikan permasalahan mengenai *user interface.*

DAFTAR PUSTAKA

https://bertzzie.com/knowledge/analisis-algoritma/Greedy.html
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf