**Laporan**

**Tugas Besar IF2211 Strategi Algoritma**

**Aplikasi Permainan Kartu 24 dengan Algoritma Greedy**



Oleh:

Muhammad Raihan Asyraf Desanto / 13517027

M. Rifky I. Bariansyah / 13517081

Muhammad Hanif Adzkiya / 13517120

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2019

**BAB I**

**DESKRIPSI PERSOALAN**

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.

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 ekspresi yang sama dengan 24 tetapi dengan nilai total operator yang lebih rendah akan memiliki bobot yang lebih tinggi daripada ekspresi dengan nilai total operator yang lebih tinggi tetapi tidak menghasilkan nilai 24. Semakin dekat skor akhir ekspresi dengan 24, dengan nilai total operator yang sama, bobotnya akan semakin tinggi.

Spesifikasi pada tugas besar ini adalah sebagai berikut:

1. Terdapat satu backend engine, dan dua front end: a. Backend menghasilkan ekspresi solusi berdasarkan masukan 4 angka, b. Front-end pertama berupa GUI yang mendemokan proses pengambilan 4 kartu untuk memberikan input, dan menampilkan hasilnya. Visualisasi kartu untuk demo boleh menggunakan library. c. 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.

2. Terdapat satu lingkungan permainan yang akan disiapkan oleh asisten untuk kompetisi, yang akan memanggil sejumlah program 24game solver dari tim pemain yang berbeda. Lingkungan permainan ini memiliki engine yang mengatur pemilihan 4 kartu secara acak, memberi masukan ke setiap program pemain, menerima solusi dari setiap program pemain, dan menghitung skor setiap pemain. Proses ini diulangi sampai dek 52 kartu habis. Aturan tambahan permainan adalah: a. Pemain dengan skor tertinggi akan menang. b. Pemain yang tidak menggunakan strategi greedy akan didiskualifikasi. c. Pemain yang melakukan kecurangan juga akan didiskualifikasi. d. Setiap kelompok hanya memiliki satu pemain. e. Pemenang akan dikompetisikan kembali dengan tim pemenang lainnya, sampai keluar 3 tim pemenang dari tim semua kelas (K1-K3).

3. Environment akan memanggil keempat program dengan argument nama file input dan nama file output. Contoh: "python batchXX\_kelompokYY.py AA BB " XX dan YY adalah nomor yang akan diberikan kepada setiap kelompok pada saat pengisian sheet kelompok. AA adalah nama file (termasuk ekstensi file) yang harus dibaca program, dan BB adalah nama file untuk menuliskan ekspresi yang dihasilkan. Perhatikan bahwa nama file berupa variabel, bukan nama file sesungguhnya. Jangan melakukan hardcode untuk melakukan operasi pada file "AA" dan "BB". a. File input terdiri dari 1 baris berisi 4 buah integer yang dipisahkan oleh whitespace. Integer memiliki domain [1..13] inklusif. b. File output berupa 1 baris ekspresi matematika yang diminta sesuai spek.

4. Strategi greedy yang diimplementasikan tiap kelompok harus mempertimbangkan fungsi objektifnya yaitu berusaha memaksimalkan skor. Strategi ini harus dituliskan secara eksplisit pada laporan, karena akan diperiksa pada saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan. Tiap kelompok dapat menggunakan kreativitas mereka dalam menyusun strategi greedy untuk memenangkan permainan.

5. Implementasi pemain 24game solver harus dapat dijalankan pada lingkungan permainan yang telah disediakan oleh asisten.

**BAB II**

**DASAR TEORI**

1. **Algoritma Greedy**

Algoritma untuk permasalahan optimisasi biasanya melalui urutan langkah, dengan serangkaian pilihan di tiap langkahnya. Algoritma greedy selalu melalui pilihan yang terlihat paling baik saat momen itu. Algoritma greedy tidak selalu memberikan hasil yang paling optimal, hanya pada kasus tertentu saja [1].

Algoritma ini memiliki elemen umum metode greedy:

1. Tentukan upa-struktur optimal dari masalah.
2. Kembangkan solusi rekursif.
3. Tunjukkan bahwa dengan sebuah pilihan greedy, maka hanya akan tersisa satu upa-masalah.
4. Buktikan akan aman dengan menggunakan pilihan greedy.
5. Kembangkan algoritma rekursif yang mengimplementasikan strategi greedy.
6. Ubah algoritma rekursif tersebut menjadi algoritma iteratif.

Kunci pertama dari algoritma ini adalah properti *greedy-choice* : kita dapat menyatukan solusi global optimal dengan menentukan solusi lokal optimal. Dalam kata lain ksaat menentukan sebuah pilihan, kita menentukan pilihan yang terlihat paling baik saat itu, tanpa mempertimbangkan apa yang dihasilkan upa-masalah. Jika sebuah masalah menunjukkan upa-struktur optimal bila sebuah solusi optimal untuk masalah tersebut berisi di dalamnya solusi optimal untuk upa-masalah [1].

1. **Permasalahan 24 Game**

24 Game adalah permainan aritmatika yang tujuannya adalah menemukan cara untuk memanipulasi 4 buah angka sehingga menghasilkan bilangan 24. Operator yang bisa digunakkan pada permainan adalah +, -, \*, dan / serta penggunaan kurung [2].

**BAB III**

**PEMANFAATAN STRATEGI GREEDY**

**3.1 Solusi Greedy**

Solusi greedy yang kami gunakan memprioritaskan hasil sedekat mungkin dengan 24. Jika ada solusi yang menghasilkan hasil yang sama, kami memilih untuk mengambil solusi dengan penggunan prioritas operator yang lebih tinggi. Berikut alur lengkap dari greedy yang kami gunakan :

1. Mengurutkan bilangan dari terbesar hingga terkecil
2. Pencarian solusi dilakukan secara urut dari angka terbesar hingga terkecil
3. Setiap langkah, dilakukan pemilihan operator yang memiliki nilai terdekat dengan 24 jika dilakukan operasi dengan operator tersebut
4. Urutan operasi yang dicoba terlebih dahulu adalah penjumlahan, pengurangan. Perkalian, dan pembagian. Pemilihan urutan ini agar operator yang memiliki point lebih tinggi dapat terpilih lebih dahulu.
5. Langkah diulangi hingga operasi terbentuk.

**3.2 Ilustrasi Algoritma Greedy**

Untuk ilustrasi, digunakan contoh bilangan yang diambil adalah 6-5-4-3.

1. Menyediakan variabel solution dan score.
2. Bilangan diurutkan dari yang terbesar, yaitu 6,5,4, dan 3.
3. Variabel solution diinisialisasi dengan bilangan terbesar yaitu 6.
4. Iterasi pertama, solution akan digabung dengan angka kedua yaitu 5. Diantara 6 dan 5 dicoba berbagai kemungkinan operator yang ada. Pertama dicoba operator tambah, ternyata menghasilkan angka 11. Solusi lokal diset dengan angka 11. Kedua, dicoba operator kurang, ternyata menghasilkan hasil 1. Karena selisih 1 dan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui. Ketiga, dicoba operator kali, ternyata menghasilkan hasil 30. Karena selisih hasil dengan 24 lebih kecil dari selisih solusi lokal dengan 24, maka solusi lokal diperbarui dengan 30. Keempat, dicoba operator bagi, ternyata menghasilkan hasil 1.2. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui.
5. Score ditambah dengan nilai dari operator yang digunakan (kali), yaitu sebesar 3.
6. Iterasi kedua, solution akan digabung dengan angka ketiga yaitu 4. Diantara solution dan 4 dicoba berbagai kemungkinan operator yang ada. Pertama, dicoba operator tambah, ternyata menghasilkan hasil 6\*5+4 yaitu sebesar 34. Solusi lokal diset dengan angka 34. Kedua, dicoba operator kurang, ternyata menghasilkan hasil 6\*5-4 yaitu sebesar 26. Karena selisih hasil dan 24 lebih kecil dari selisih solusi lokal dengan 24, maka solusi lokal diperbarui dengan 26. Ketiga, dicoba operator kali, ternyata menghasilkan hasil 6\*5\*4 yaitu sebesar 120. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui. Keempat, dicoba operator bagi, ternyata menghasilkan hasil 6\*5/4 yaitu sebesar 7.5. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui.
7. Score ditambah dengan nilai dari operator yang digunakan (kurang), yaitu sebesar 4.
8. Iterasi ketiga, solution akan digabung dengan angka keempat yaitu 3. Diantara solution dan 3 dicoba berbagai kemungkinan operator yang ada. Pertama, dicoba operator tambah, ternyata menghasilkan hasil 6\*5-4+3 yaitu sebesar 29. Solusi lokal diset dengan angka 29. Kedua, dicoba operator kurang, ternyata menghasilkan hasil 6\*5-4-3 yaitu sebesar 23. Karena selisih hasil dan 24 lebih kecil dari selisih solusi lokal dengan 24, maka solusi lokal diperbarui dengan 23. Ketiga, dicoba operator kali, karena operator sebelumnya adalah kurang, maka dicoba dua variasi, tanpa kurung dan menggunakan kurung. Dengan tanpa kurung ternyata menghasilkan hasil 6\*5-4\*3 yaitu sebesar 18. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui. Sedangkan apabila menggunakan kurung ternyata menghasilkan hasil (6\*5-4)\*3 yaitu sebesar 78. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui. Keempat, dicoba operator bagi, karena operator sebelumnya adalah kurang, maka dicoba dua variasi, tanpa kurung dan menggunakan kurung. Dengan tanpa kurung ternyata menghasilkan hasil 6\*5-4/3 yaitu sebesar 28.7. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui. Sedangkan apabila menggunakan kurung ternyata menghasilkan hasil (6\*5-4)/3 yaitu sebesar 8.7. Karena selisih hasil dengan 24 lebih besar dari selisih solusi lokal dengan 24, maka solusi lokal tidak diperbarui.
9. Score ditambah dengan nilai dari operator yang digunakan (kurang), yaitu sebesar 4.
10. Score dikurangkan dengan selisih hasil dengan 24, yaitu 1. Sehingga nilai akhir score adalah 10 dan hasil adalah 23 dengan operator berupa 6\*5-4-3

**BAB IV**

**IMPLEMENTASI DAN PENGUJIAN**

**FUNCTION** Solver():

**global** solution #Solution Expression

**global** Score #Expression Score

**global** Eval #Expression Result

point <- 0

lastsymbol <- ""

List\_Num.sort(reverse <- **True**) #Sort Descending

solution <- str(List\_Num[0]) #Initialized with first largest number

point <- 0

lastsymbol <- ""

**for** i **in** range(1,4):

#Addition

maks\_equationlocal **<-** solution + "+" + str(List\_Num[i])

maks\_localpoint <- eval(maks\_equationlocal) #Store local result

lastsymbol\_local <- "+"

#Substraction

localequation <- solution + "-" + str(List\_Num[i])

localpoint <- eval(localequation)

#Check and store point and equation with closer result to 24

**IF**(abs(localpoint-24) < abs(maks\_localpoint-24)):

maks\_localpoint <- localpoint

maks\_equationlocal <- localequation

lastsymbol\_local <- "-"

**ENDIF**

#Multiplication

localequation <- solution + "\*" + str(List\_Num[i])

localpoint <- eval(localequation)

#Check and store point and equation with closer result to 24

**IF**(abs(localpoint-24) < abs(maks\_localpoint-24)):

maks\_localpoint <- localpoint

maks\_equationlocal <- localequation

lastsymbol\_local <- "\*"

**ENDIF**

#Parentheses correction

**IF**(lastsymbol = "+" **OR** lastsymbol = "-"):

localequation <- "(" + solution + ")\*" + str(List\_Num[i])

localpoint <- eval(localequation)

**IF**(abs(localpoint-24) < abs(maks\_localpoint-24)):

maks\_localpoint <- localpoint

maks\_equationlocal <- localequation

lastsymbol\_local <- "\*"

**ENDIF**

**ENDIF**

#Division

localequation <- solution + "/" + str(List\_Num[i])

localpoint <- eval(localequation)

#Check and store point and equation with closer result to 24

**IF**(abs(localpoint-24) < abs(maks\_localpoint-24)):

maks\_localpoint <- localpoint

maks\_equationlocal <- localequation

lastsymbol\_local <- "/"

**ENDIF**

#Parentheses correction

**IF**(lastsymbol = "+" OR lastsymbol = "-"):

localequation <- "(" + solution + ")/" + str(List\_Num[i])

localpoint <- eval(localequation)

#Check and store point and equation with closer result to 24

**IF**(abs(localpoint-24) < abs(maks\_localpoint-24)):

maks\_localpoint <- localpoint

maks\_equationlocal <- localequation

lastsymbol\_local <- "/"

**ENDIF**

**ENDIF**

#global solution and point

solution <- maks\_equationlocal

maks\_global <- maks\_localpoint

#Score counting

**IF(**lastsymbol\_local = "+"):

point += 5

**ELSEIF**(lastsymbol\_local = "-"):

point += 4

**ELSEIF**(lastsymbol\_local = "\*"):

point += 3

**ELSEIF**(lastsymbol\_local = "/"):

point += 2

**ENDIF**

**ENDFOR**

point -= abs(maks\_global - 24) - solution.count("(")

Score <- point

Eval <- eval(solution)

**BAB V**

**KESIMPULAN DAN SARAN**

**5.1 Kesimpulan**

Penggunaan algoritma greedy dalam menyelesaikan 24 game solver dapat mengurangi waktu eksekusi program dibandingkan dengan menggunakan strategi bruteforce. Namun, dalam beberapa kasus, strategi greedy mengeluarkan keluaran yang kurang optimal. Hal ini dikarenakan, pada penyelesaian bisa saja operator yang terpilih diawal justru menghasilkan hasil yang tidak optimal diakhir. Dan sebaliknya, operator yang tidak terpilih di awal bisa saja menghasilkan hasil yang optimal di akhir.

**5.2 Saran**

Persoalan yang memang harus mengiterasi seluruh kemungkinan, lebih baik menggunakan strategi bruteforce. Namun, apabila terdapat batasan waktu, dapat menggunakan algoritma gredy. Pengguanaan algoritma greedy ini harus dipilih solusi greedy yang paling optimal.

**REFERENSI**

[1] Cormen, Thomas H., *Introduction to Algorithms (3rd Ed.)*, The MIT Press, 2009.

[2] <https://www.pagat.com/adders/24.html>, diakses pada 8 Februari 2019.