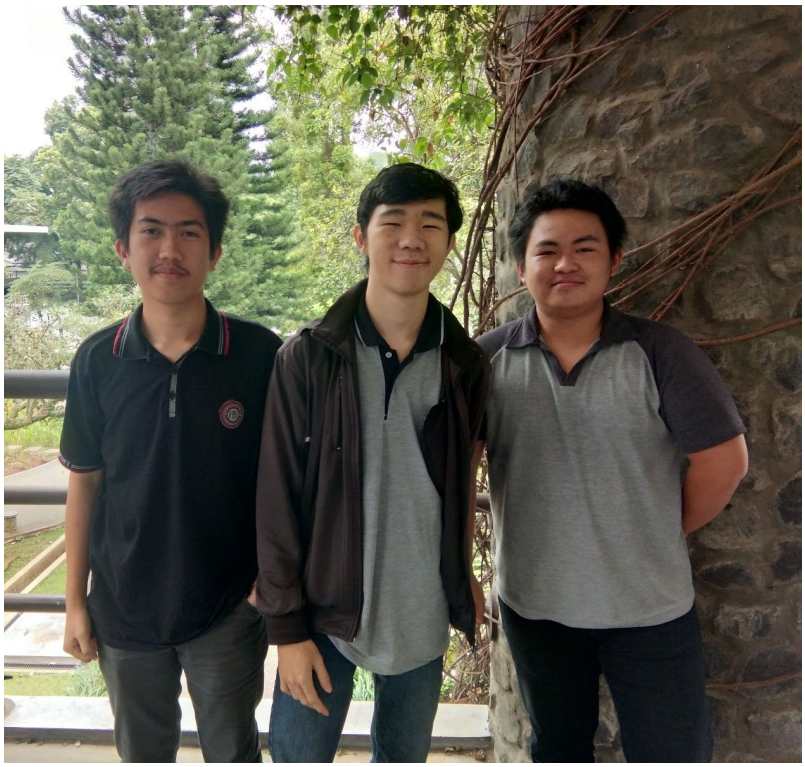
Laporan Tugas Besar 1
IF 2211 Strategi Algoritma
Aplikasi Permainan Kartu 24 dengan Algoritma *Greedy*



oleh
Suhailie - 13517045
Vijjasena - 13517084
Josep Andre Ginting - 13517108

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*.

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

Algoritma *Greedy*

Algoritma *greedy* adalah algoritma yang memecahkan suatu masalah secara per langkah dengan mencari solusi optimal tiap langkah tersebut. Solusi optimal tersebut berupa maksimasi nilai atau minimasi nilai. Hal tersebut dilakukan dengan tujuan agar tercapai solusi optimal global pada akhir algoritma.

Algoritma *greedy* memiliki elemen - elemen yang harus jelas :

a. Himpunan kandidat : Himpunan yang akan dipilih untuk mencari solusi
b. Himpunan solusi : Himpunan yang telah dipilih dan menyatakan solusi optimal
c. Fungsi seleksi : Langkah pemilihan solusi dari himpunan kandidat
d. Fungsi kelayakan : Syarat yang harus dipenuhi dalam pemilihan solusi
e. Fungsi objektif : Tujuan yang akan dicapai dari sebuah permasalahan

Contoh studi kasus :

Pada studi kasus pemilihan pelanggan, ada sejumlah pelanggan dengan lama waktu untuk melayani pelanggan tersebut :

T1 = 10 menit T2 = 3 menit T3 = 5 menit

Kita harus mencari urutan pelanggan yang akan dilayani sehingga total waktu pelayanan minimum.

a. Himpunan kandidat : Himpunan pelanggan beserta lama waktu pelayanannya : 10 menit, 3 menit, 5 menit
b. Himpunan solusi : Himpunan urutan pelanggan dengan waktu pelayanan minimum
c. Fungsi seleksi : Pilih pelanggan yang memiliki lama waktu pelayanan terkecil dari himpunan kandidat yang tersisa
d. Fungsi kelayakan : Pelanggan dilayani hanya sekali
e. Fungsi objektif : Urutan pelanggan sehingga waktu total pelayanan menjadi minimum

Langkah pengerjaan :

1. Urutkan himpunan kandidat secara membesar berdasarkan waktu pelayanan
2. Pilih pelanggan dengan waktu pelayanan terkecil dari himpunan kandidat tersisa
3. Tambahkan pelanggan tersebut ke dalam himpunan solusi.
4. Ulangi langkah no. 2 - 3 hingga himpunan kandidat habis.

Hasil Himpunan Solusi = {T2, T3, T1}
Total Waktu Pelayanan = 3 + (3 + 5) + (3 + 5 + 10) = 29 menit

Permainan Kartu 24

Permainan kartu 24 adalah sebuah permainan kartu di mana kita diberikan 4 angka yang berasal dari 4 kartu untuk dibuat sebuah operasi bilangan sehingga hasil dari operasi 4 angka tersebut bernilai 24.
Pemain masing - masing akan dibagikan sebuah kartu sebagai penanda, bila pemain telah mendapatkan operasi bilangan tersebut, maka pemain akan menutupi kartu mereka sebagai penanda bahwa mereka telah berhasil menemukan operasi tersebut.

Ada beberapa versi dari permainan kartu 24, yang paling umum dan sering digunakan adalah versi yang menggunakan operator tambah (+), kurang (-), kali (*), bagi (/) dan penggunaan tanda kurung. Beberapa versi lain ada menggunakan operator tambahan seperti perpangkatan, akar, dan operator lainnya.

Kartu yang biasanya digunakan adalah kartu remi, dengan nilai AS sebagai 1, dan kartu lainnya bernilai sesuai dengan angka yang tertulis di kartunya (Jack = 11, Queen = 12, King = 13).

Sebagai contoh :

Diberikan kartu dengan nilai 3, 5, 8, Jack.
Maka operasi 4 angka tersebut yang bernilai 24 adalah 3 * ((5 - 8) + 11).

BAB III
Pemanfaatan Strategi *Greedy*

Pada program permainan kartu 24, kami menggunakan algoritma *greedy* dengan pendekatan nilai operasi mendekati hasil 24.

1. 4 angka yang memiliki *range* [1..13] akan disusun secara mengecil di dalam sebuah list.

```
ltemp.sort(reverse = True)
```

2. Dua angka pertama yang telah disusun (x_1 ,x_2) akan dioperasikan dengan operator +, -, *, / yang bilamana hasilnya paling mendekati nilai 24.

```
s = l[0]
value = l[0]
s = s + cek(value,l[1])
s = s + (l[1])
oprb = cek(value,l[1])
value =  eval(s)
```

3. Hasil dari operasi pada no. 2 menjadi (x_{12}). Hasil ini akan dioperasikan dengan angka berikutnya (x_3).

```
s = s + cek(value,l[2])
s = s + (l[2])
if ((oprb == '+' or oprb == '-') and (cek(value,l[2]) == '*' or cek(value,l[2])
== '/')) :
       index = s.find(l[0])
       x = s[:index] + '(' + s[index:]
       s = x
       index = s.find(cek(value,l[2]))
       x = s[:index] + ')' + s[index:]
       s = x
oprbb = oprb
oprb = cek(value,l[2])
value = eval(s)
```

4. Ulangi langkah no. 2 - 3 hingga semua angka telah dioperasikan, maka akan ditampilkan hasil operasi tersebut beserta ekspresinya.

Pemanfaatan strategi algoritma *greedy* tersebut bertujuan agar nilai akhir dari hasil operasi tersebut dapat bernilai optimal atau mendekati nilai optimal.

BAB IV
Implementasi dan Pengujian

Program yang dibuat :

a. Program Front-End 1 : Program tersebut bertujuan sebagai penerapan GUI dari permainan kartu 24. Program menampikan hasil input dan output yang telah diproses.
b. Program Front-End 2 : Program tersebut menerima argumen untuk menerima input dari file eksternal yang kemudian diproses dan dituliskan ke dalam file eksternal lainnya sesuai dengan argumen yang diterima.
c. Program Back-End : Program berisi langkah - langkah pemrosesan input dengan algoritma *greedy*.

Proses jalannya program :

a. Input program merupakan list angka yang terdiri dari 4 angka dengan *range* [1..13] secara acak dari tumpukan kartu.
b. Input akan diproses oleh program dengan memanggil *back-end* program yang digunakan.
c. Hasil proses (output) akan ditampilkan ke layar *front-end* program.

Pseudocode dari *back-end* program :

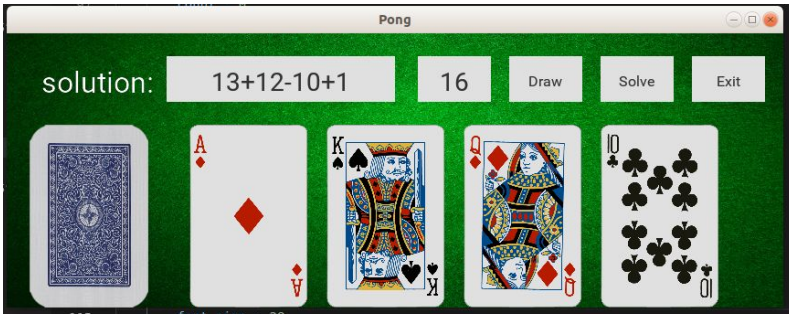
```
Function cek (val,n : string) : char
       c : array [0..3] of integer
       c[0] <- abs(24-(int(val)+int(n)))
       c[0] <- abs(24-(int(val)-int(n)))
       c[0] <- abs(24-(int(val)*int(n)))
       c[0] <- abs(24-(int(val)/int(n)))

       if (min(c) == c[0])
              -> '+'
       else if (min(c) == c[1])
              -> '-'
       else if (min(c) == c[2])
              -> '*'
       else (min(c) == c[3])
              -> '/'
       end if


Function hasil (l : array [0..3] of integer) : string
       l <- l.sort {sorting array dari angka terbesar ke yang terkecil}
       value : integer
       value <- 0
       s : string
       Oprb, oprbb : char
```

```
      s <- l[0]
      Value <- l[0]
      s <- s + cek(value,l[1])
      s <- s + l[1]
      oprb <- cek(value,l[1])
      value <- eval(s)
      s <- s + cek(value,l[2])
      s <- s + l[2]
      if ((oprb = ‘+’ or oprb = ‘-’) and (cek(value,l[2]) = ‘*’ or  cek(value,l[2]) =
‘/’))
            Index : integer
            Index <- s.find(l[0])
            s = s[:index] + ‘(’ + s[index:] {menyelipkan tanda kurung}
            Index <- s.find(l[2])
            s = s[:index] + ‘(’ + s[index:] {menyelipkan tanda kurung}
      Endif
      oprbb <- oprb
      oprb <- cek(value,l[2])
      value <- eval(s)
      s <- s + cek(value,l[3])
      s <- s + l[3]
      if (((oprbb='+' or oprbb='-') and (oprb='+' or oprb=='-')) and
(cek(value,l[3])='*' or cek(value,l[3])='/'))
            index <- s.find(l[0])
            s <- s[:index] + '(' + s[index:]
            index <- s.find(cek(value,l[3]))
            s <- s[:index] + ')' + s[index:]
      else ((oprb='+' or oprb='-') and (cek(value,l[3])='*' or cek(value,l[3])='/'))
            index <- s.find(l[1])
            s <- s[:index] + '(' + s[index:]
            index <- s.find(cek(value,l[3]))
            s <- s[:index] + ')' + s[index:]
   -> s
```

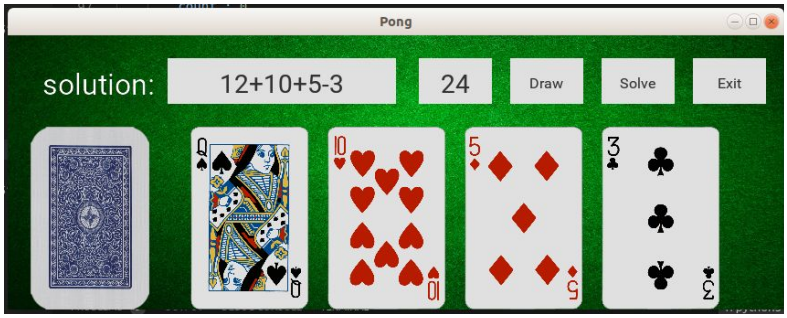
Pengujian :

Tampilan awal dari program

Pengambilan kartu dengan tombol Draw



Penyelesaian solusi dengan tombol Solve



Solusi yang didapatkan bernilai 24

Solusi yang didapatkan bernilai 25 (dekat dengan 24)

Pada hasil analisis, strategi algoritma *greedy* yang diimplementasikan dapat memberikan solusi dengan nilai optimal (nilai 24). Namun pada beberapa kasus, algoritma greedy tersebut tidak dapat memberikan solusi dengan nilai optimal.

Beberapa kasus tersebut memiliki kondisi jika ada 3 angka yang bernilai besar dan angka ke - 4 memiliki nilai yang kecil.

Contoh : 13, 12, 10, 1.

Program akan memberikan solusi 13+12-10+1, dimana hasil tersebut bernilai 16.

BAB V
Kesimpulan dan Saran

5.1. Kesimpulan

- Program dapat di-*compile* dan dieksekusi
- Dengan menggunakan algoritma *greedy*, dapat dibuat sebuah algoritma yang memiliki kompleksitas waktu yang jauh lebih cepat dalam pemecahan masalah, namun hasil yang didapatkan belum tentu hasil yang terbaik.

5.2. Saran

- Untuk strategi algoritma yang digunakan harus dikembangkan lebih baik sehingga memberikan operasi dengan nilai yang lebih optimal.

DAFTAR PUSTAKA

Munir, Rinaldi. 2017. Slide *Algoritma Greedy (2017)*.

https://en.wikipedia.org/wiki/24_Game (diakses pada 12 Februari 2019, pukul 17 : 36)