Laporan Tugas Kecil I

Algoritma *Brute Force* pada 24 *Game*

Disusun untuk memenuhi tugas mata kuliah

IF2211 Strategi Algoritma pada Semester II 2018/2019

Disusun oleh :

Nama : Steve Andreas Immanuel

NIM : 13517039

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2019

Algoritma *Brute Force*

Algoritma *brute force* pada 24 *Game* yang digunakan adalah sebagai berikut:

- Buat *array* kosong (misalkan bernama `solutions`) untuk menampung solusi yang ditemukan.
- Ambil empat bilangan dari *input* pengguna dan bentuk semua permutasi yang mungkin dari empat bilangan tersebut.
- Bentuk semua permutasi berulang dengan tiga elemen dari semua operator yang mungkin yaitu +, -, *, dan /.
- Bentuk ekspresi matematika dari semua hasil permutasi bilangan dan operasi serta letakkan tanda kurung di posisi yang mungkin. Misalkan empat bilangan a, b, c, d, dan operator +, maka pada algoritma yang digunakan, tanda kurung disisipkan di tempat sebagai berikut:
  - (a+b)+(c+d)
  - ((a+b)+c)+d
  - (a+(b+c))+d
  - a+((b+c)+d)
  - a+(b+(c+d))
- Evaluasi hasil ekspresi matematika yang dibentuk dari bilangan, operator, dan tanda kurung.
- Jika hasil evaluasi adalah 24, simpan ekspresi matematika pada *array* `solutions`. Jika hasil evaluasi 24 maupun bukan 24, lanjutkan iterasi ke kemungkinan ekspresi matematika berikutnya hingga habis.
- Terakhir, setelah semua kemungkinan ekspresi matematika dievaluasi, tuliskan semua isi dari *array* `solutions` beserta banyak isi *array* untuk menyatakan solusi yang ditemukan dan banyaknya.

Source Code Program

Untuk mengimplementasikan algoritma *brute force* pada 24 *Game* ini, digunakan bahasa Python 3.7. Pada *source code* digunakan *library time* untuk mencatat waktu eksekusi program. Berikut adalah *source code* dari program:

```
import time

def generatePerm(L):
    if len(L)==0:
        return set()
    elif len(L)==1:
        return [(L[0],)]
    else:
        result=set()
        for i in range(0,len(L)):
            first=(L[i],)
            L_rem=L[:i]+L[i+1:]
            for tempres in generatePerm(L_rem):
                result.add(first+tempres)
        return result

def generateOps():
    op='+-*/'
    result=[]
    for i in range(4):
        for j in range(4):
            for k in range(4):
                result.append([op[i]]+[op[j]]+[op[k]])
    return result

def operate(digit1,op,digit2):
    if op=='+':
        return float(digit1)+float(digit2)
    elif op=='-':
        return float(digit1)-float(digit2)
    elif op=='*':
        return float(digit1)*float(digit2)
    elif op=='/':
        return float(digit1)/float(digit2)

def makeExp(perm_digit,ops,variation):
    if variation==1:
      result=['('+perm_digit[0]+ops[0]+perm_digit[1]+')'+ops[1]+'('+perm_di
      git[2]+ops[2]+perm_digit[3]+')']

      result.append(operate(operate(perm_digit[0],ops[0],perm_digit[1]),ops
      [1],operate(perm_digit[2],ops[2],perm_digit[3])))
    elif variation==2:
      result=['(('+perm_digit[0]+ops[0]+perm_digit[1]+')'+ops[1]+perm_digit
      [2]+')'+ops[2]+perm_digit[3]]

      result.append(operate(operate(operate(perm_digit[0],ops[0],perm_digit
      [1]),ops[1],perm_digit[2]),ops[2],perm_digit[3]))
```

```
    elif variation==3:
      result=['('+perm_digit[0]+ops[0]+'('+perm_digit[1]+ops[1]+perm_digit[
      2]+'))'+ops[2]+perm_digit[3]]

      result.append(operate(operate(perm_digit[0],ops[0],operate(perm_digit
      [1],ops[1],perm_digit[2])),ops[2],perm_digit[3]))
    elif variation==4:
      result=[perm_digit[0]+ops[0]+'(('+perm_digit[1]+ops[1]+perm_digit[2]+
      ')'+ops[2]+perm_digit[3]+')']

      result.append(operate(perm_digit[0],ops[0],operate(operate(perm_digit
      [1],ops[1],perm_digit[2]),ops[2],perm_digit[3])))
    else:
      result=[perm_digit[0]+ops[0]+'('+perm_digit[1]+ops[1]+'('+perm_digit[
      2]+ops[2]+perm_digit[3]+'))']

      result.append(operate(perm_digit[0],ops[0],operate(perm_digit[1],ops[
      1],operate(perm_digit[2],ops[2],perm_digit[3]))))
    return result

if __name__=='__main__':
    user_input = input('Input 4 numbers : ')
    digits=user_input.split(' ')
    start=time.time()
    solutions=[]
    epsilon=1e-12
    for perm_digit in generatePerm(digits):
        for ops in generateOps():
            for i in range(1,6):
                try:
                    result=makeExp(perm_digit,ops,i)
                except:
                    continue
                if abs(result[1]-24)<=epsilon:
                    solutions.append(result[0])
    end=time.time()
    solutions=sorted(solutions)
    for solution in solutions:
        print(solution)
    print('Found',len(solutions),'solution(s).')
    print('Time Execution :',end-start,'s')
```

Contoh *Input Output*

Berikut adalah contoh *input output* dari program pada berbagai kondisi beserta waktu eksekusinya.

```
>> Input 4 numbers : 1 8 9 10
Found 0 solution(s).
Time Execution :
0.06300926208496094 s

>> Input 4 numbers : 1 1 1 11
(1+1)*(1+11)
(1+1)*(11+1)
(1+11)*(1+1)
(11+1)*(1+1)
Found 4 solution(s).
Time Execution :
0.01764535903930664 s

>> Input 4 numbers : 8 3 8 3
8/(3-(8/3))
Found 1 solution(s).
Time Execution :
0.021140575408935547 s

>> Input 4 numbers : 1 5 7 9
(1-7)*(5-9)
(5-9)*(1-7)
(7-1)*(9-5)
(9-5)*(7-1)
Found 4 solution(s).
Time Execution :
0.053389549255371094 s

>> Input 4 numbers : 12 13 15 17
((13+17)*12)/15
((13+17)/15)*12
((17+13)*12)/15
((17+13)/15)*12
(12*(13+17))/15
(12*(17+13))/15
(12/15)*(13+17)
(12/15)*(17+13)
(13+17)*(12/15)
(13+17)/(15/12)
(17+13)*(12/15)
(17+13)/(15/12)
12*((13+17)/15)
12*((17+13)/15)
12/(15/(13+17))
12/(15/(17+13))
Found 16 solution(s).
Time Execution :
0.0533900260925293 s
```

```
>> Input 4 numbers : 4 7 8 28
((28*4)/7)+8
((28*7)-4)/8
((28/7)*4)+8
((4*28)/7)+8
((4/7)*28)+8
((7*28)-4)/8
((7*8)-28)-4
((7*8)-4)-28

((7-8)*4)+28
((8*7)-28)-4
((8*7)-4)-28
((8-7)*28)-4
(28*(4/7))+8
(28*(8-7))-4
(28-4)*(8-7)
(28-4)/(8-7)
(28/(7/4))+8
(28/(8-7))-4
(4*(28/7))+8
(4*(7-8))+28
(4-28)*(7-8)
(4-28)/(7-8)
(4/(7-8))+28
(4/(7/28))+8
(7*8)-(28+4)
(7*8)-(4+28)
(7-8)*(4-28)
(8*7)-(28+4)
(8*7)-(4+28)
(8-7)*(28-4)
28+((7-8)*4)
28+(4*(7-8))
28+(4/(7-8))
28-((8-7)*4)
28-(4*(8-7))
28-(4/(8-7))
8+((28*4)/7)
8+((28/7)*4)
8+((4*28)/7)
8+((4/7)*28)
8+(28*(4/7))
8+(28/(7/4))
8+(4*(28/7))
8+(4/(7/28))
Found 44 solution(s).
Time Execution :
0.0533905029296875 s
```

Hasil Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi.	✓	
2. Program berhasil dieksekusi.	✓	
3. Solusi 24 *Game* benar untuk semua data tes. Bandingkanlah hasilnya dengan aplikasi 24*solver* yang sudah ada.	✓	

Solusi 24 *Game* yang dihasilkan pada program mungkin sedikit berbeda dengan yang terdapat pada aplikasi-aplikasi 24*solver* yang sudah ada. Hal ini dikarenakan solusi yang dianggap valid di program adalah solusi yang tidak mengandung pembagian terhadap nol dan selisih hasilnya dengan bilangan 24 adalah kurang dari atau sama dengan 10^{-12}. Selain itu, variasi peletakkan tanda kurung yang digunakan pada program adalah seperti yang telah dijelaskan pada algoritma *brute force* poin ke-4.