Penerapan Algoritma Backtracking untuk Menyelesaikan Permainan Cryptarithmetic

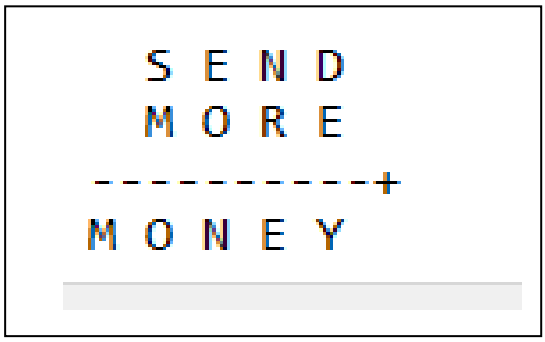
Prana Gusriana - 13519195
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519195@std.stei.itb.ac.id

***Abstract*—*Cryptarithmetic* merupakan *puzzle* matematika yang digit-digit dari suatu operasi aritmatika diganti dengan huruf. Operasi aritmatika yang dimaksud adalah penambahan, pengurangan, perkalian, dan pembagian. Persoalan ini merupakan persoalan kombinatorika sehingga dapat diselesaikan dengan menggunakan algoritma *brute force* ataupun algoritma *backtracking*. Pada makalah ini hanya akan dibahas *cryptarithmetic* dengan operasi penjumlahan dan diselesaikan dengan menggunakan algoritma *backtracking*.**

Keywords—cryptarithmetic; algoritma; backtracking; puzzle; pohon ruang status;

I. PENDAHULUAN

Persoalan *cryptarithmetic* merupakan *puzzle* matematika yang terdiri atas operasi aritmatika dengan digit-digitnya diganti dengan huruf atau simbol lain. Biasanya operasi aritmatika yang digunakan adalah penjumlahan. *Cryptarithmetic* sering disebut juga sebagai *verbal arithmetic*, *alphametic*, atau *word addition*. Nama "cryptarithm" diciptakan oleh Simon Vatriquant dalam Sphinx edisi Mei 1931, majalah belgia Mathematical Recreations dan diterjemahkan menjadi "cryptarithmetic" oleh Maurice Kraitchik pada tahun 1942. Persoalan ini sangat populer pada tahun 1930. Berikut adalah persoalan *alphametic* atau *cryptarithmetic* yang cukup terkenal dipublikasikan pada Juli tahun 1924 oleh H.E. Dudeney pada Strand Magazine



Gambar 1.1 Contoh persoalan cryptarithmetic

Pada dasarnya persoalan *cryptarithmetic* ini bertujuan untuk memetakan digit dari 0 sampai 9 ke huruf atau simbol sehingga setelah digit disubstitusikan ke huruf atau simbol tersebut maka operasi aritmatika dapat dibuktikan benar. Berikut adalah beberapa aturan dari permainan *cryptarithmetic* ini

1. Digit yang direpresentasikan oleh huruf atau simbol merupakan bilangan bulat antara 0 sampai 9 sehingga terdapat batasan jumlah huruf yang berbeda maksimal 10 huruf dalam satu persoalan *cryptarithmetic*
2. Setiap huruf atau simbol hanya merepresentasikan satu digit dan harus unik, yang berarti huruf atau simbol yang sama pada persoalan tidak boleh merepresentasikan digit yang berbeda dan tidak ada dua huruf atau simbol yang berbeda merepresentasikan digit yang sama.
3. Karena kata pada operand dan juga hasil akan merepresentasikan angka, maka huruf pertama dari kata yang panjangnya lebih sari satu tidak boleh nol
4. Setelah huruf atau simbol disubstitusi, maka penjumlahan operand harus sama dengan hasilnya

Contoh penerapan aturan permainan *cryptarithmetic* pada persoalan SEND + MORE = MONEY adalah huruf S, E, N, D, M, O, R, dan Y hanya merepresentasikan digit diantara 0 sampai 9 dan untuk masing-masing huruf harus merepresentasikan digit yang berbeda, huruf S dan M tidak boleh nol, dan ketika huruf S, E, N, D, M, O, R, dan Y disubstitusi oleh digit maka penjumlahan SEND dan MORE harus sama dengan MONEY misalnya huruf S, E, N, D, M, O, R, dan Y akan disubstitusi oleh 9, 5, 6, 7, 1, 0, 8, dan 2, maka penjumlahan 9567 dan 1085 harus sama dengan 10652. Karena kombinasi digit tersebut telah memenuhi semua peraturan permainan cryptarithmetic (untuk setiap huruf merepresentasikan digit yang unik, huruf S dan M bukan nol, serta penjumlahan 9567 dan 1085 sama dengan 10652), maka kombinasi tersebut (S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, dan Y = 2) merupakan solusi untuk persoalan SEND + MORE = MONEY.

II. DASAR TEORI

A. *Algoritma Backtracking*

Backtracking dapat dipandang sebagai salah satu dari dua hal, yaitu sebagai sebuah fase di dalam algoritma traversal *Depth First Search* (DFS) dan sebagai sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis, baik untuk persoalan optimasi maupun non-optimasi. Algoritma runut-balik atau *backtracking* merupakan perbaikan dari *exhaustive search*. Pada *exhaustive search*, semua kemungkinan solusi dieksplorasi dan dievaluasi satu per saatu sedangkan pada algoritma *backtracking*, hanya pilihan yang mengarah ke solusi yang dieksplorasi, pilihan yang tidak mengarah ke solusi tidak akan dieksplorasi.

Pada umumnya, algoritma *backtracking* memiliki sifal-sifat berikut ini

1. Solusi Persoalan

   Solusi persoalan yang diharapkan dapat dinyatakan sebagai vektor dengan n-*tupple* bilangan bulat yang merupakan anggota dari himpunan solusi S

   $$X = (x_1, x_2, ..., x_n), \quad x_i \in S_i$$

2. Fungsi Pembangkit

   Fungsi pembangkit digunakan untuk membangkitkan nilai x_k, yang merupakan komponen vektor solusi.m Fungsi pembangkit dinyatakan sebagai predikat T() yang dapat berupa fungsi atau prosedur.

3. Fungsi Pembatas (*Bounding Function*)

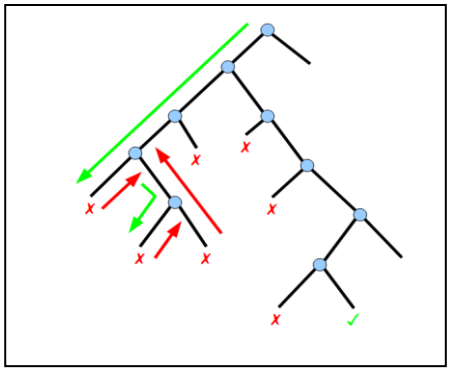
   Fungsi pembatas digunakan untuk mengecek apakah pembangkitan x_k oleh fungsi pembangkit mengarah ke solusi atau tidak. Jika pembangkitan x_k mengarah ke solusi atau dengan kata lain fungsi pembatasnya bernilai *True*, pembangkitan x_{k+1} akan dilanjutkan karena sampai pengisian x_k tidak ada *constraint* yang dilanggar. Jika pembangkitan x_k tidak mengarah ke solusi atau dengan kata lain fungsi pembatasnya bernilai *False*, pembangkitan x_{k+1} tidak akan dilanjutkan karena sampai pengisian xk terdapat *constraint* yang dilanggar sehingga pengisian komponen vektor selanjutnya tidak akan mengarah ke solusi dan nilai x_k akan dibuang lalu akan dilakukan *backtrack* untuk mencoba kemungkinan lainnya. Fungsi pembangkit dinyatakan dengan predikat B(x_1, x_2, ..., x_k).

Semua kemungkinan solusi dari persoalan disebut ruang solusi (*solution space*). Misal pada persoalan *knapsack* 0/1 untuk n = 3, solusi persoalannya dinyatakan sebagai X = (x_1, x_2, x_3) dengan x_i anggota dari {0, 1} dan ruang solusinya adalah {(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)}. Ruang solusi diorganisasikan ke dalam struktur pohon berakar. Simpul pohon menyatakan status persoalan sedangkan sisinya dilabeli dengan nilai x_i. Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*).

Berikut adalah beberapa prinsip pencarian solusi dengan menggunakan algoritma *backtracking*

1. Solusi dicari dengan membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari akar ke daun
2. Aturan pembangkitan simpul yang dipakai mengikuti aturan *Depth First Search* (DFS)
3. Simpul-simpul yang sudah dibangkitkan dinamakan simpul hidup, sedangkan simpul yang sedang diperluas dinamakan simpul-E *(Expand node*) dan setiap simpul-E diperluas, lintasan yang dibangun akan bertambah panjang
4. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi maka simpul-E tersebut akan dimatikan sehingga menjadi simpul mati *(dead node*) dan akan *backtrack* ke simpul sebelumnya untuk dicoba kemungkinan solusi yang lain. Fungsi yang digunakan untuk mematikan simpul-E adalah fungsi pembatas (*bounding function*)
5. Pencarian dihentikan jika simpul *goal* telah ditemukan atau tidak ada lagi simpul yang bisa dibangkitkan

Berikut adalah ilustrasi cara kerja algoritma backtracking



Gambar 2.1 Ilustrasi algoritma backtracking (Sumber: http://www.w3.org/2011/Talks/01-14-steven-phenotype/)

Algoritma *backtracking* dapat diimplementasikan dalam dua versi yaitu versi rekursif dan versi iteratif. Berikut adalah skema umum algoritma *backtracking* versi rekursif dan iteratif

1. Versi rekursif

procedure RunutBalikR(**input** k : **integer**)

{ Mencari **semua** solusi persoalan dengan metode runt balik; skema rekursif

Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1] x[2], ..., x[k-1] sudah ditentukan nilainya.

Luaran: semua solusi x = (x[1], x[2], ...,x[n])

}

Algoritma:

for setiap x[k] ∈ T(x[1], x[2], ..., x[k-1]) **do**

```
    if B(x[1], x[2], ..., x[k]) = true then
      if (x[1], x[2], ..., x[k]) adalah lintasan dari akar ke simpul solusi then
        write(x[1], x[2], ..., x[k]) { cetak solusi }
      endif
      if k < n then
        RunutBalikR(k+1) { tentukan nilai untuk x[k+1] }
      endif
    endif
  endfor
```

2. Versi iteratif

```
procedure RunutBalikI(input k : integer)
{ Mencari semua solusi persoalan dengan metode runt balik; skema iteratif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1]  x[2], ..., x[k-1] sudah ditentukan nilainya.
Luaran: semua solusi x = (x[1], x[2], ...,x[n])
}
Algoritma:
  while k ≠ 0 do
    if terdapat nilai x[k] yang belum dicoba sedemikian sehingga x[k] ∈ T(x[1], x[2], ..., x[k-1]) and B(x[1], x[2], ..., x[k]) = true then
      if (x[1], x[2], ..., x[k]) adalah lintasan dari akar ke simpul solusi then
        write(x[1], x[2], ..., x[k]) { cetak solusi }
      endif
      k ← k + 1 { tentukan nilai x[k] selanjutnya }
    else
      k ← k - 1
    endif
  endwhile
```

Algoritma *backtracking* dapat digunakan untuk menyelesaikan berbagai persoalan seperti persoalan N-ratu, persoalan *sum of subset*, pewarnaan graf, sirkuit hamilton, dan juga persoalan cryptarithmetic.

B. Persoalan Cryptarithmetic sebagai Constraints Satisfaction Problem

Constraint Satisfaction Problem (CSP) merupakan pencarian nilai untuk setiap variabel persoalan sehingga memenuhi semua *constraint* yang ada. Tujuan dari CSP ini adalah mencari kombinasi nilai untuk setiap variabel persoalan sehingga kombinasi tersebut memenuhi semua *constraint* yang ada. Ide dari CSP ini adalah untuk memperkecil ruang pencarian dengan cara mengeliminasi kombinasi variabel yang tidak memenuhi *constraint*.

Constraint Satisfaction Problem terdiri dari tiga komponen <X, D, C> yaitu

1. X adalah himpunan variabel {X1, ..., Xn}

2. D adalah himpunan dari domain {D1, ..., Dn}, Di terdiri dari himpunan nilai yang diperbolehkan untuk variabel Xi

3. C adalah himpunan *constraint* atau batasan yang menentukan kombinasi nilai yang diizinkan

Persoalan *cryptarithmetic* merupakan persoalan CSP karena persoalan ini merupakan persoalan untuk menentukan digit pada setiap huruf sehingga tidak melanggar *constraint* yang ada. Berikut adalah contoh persoalan cryptarithmetic

$$\begin{array}{cccc} C_3 & C_2 & C_1 & C_0 \\ & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

Gambar 3.1 Contoh persoalan cryptarithmetic

Pada persoalan tersebut terdefinisi komponen CSP sebagai berikut

1. Variabel dari persoalan tersebut adalah X = {F, T, U, W, R, O, C_0, C_1, C_2, C_3}
2. Domain untuk setiap variabel adalah D_i = {0, 1, ..., 9}
3. Constraint yang terdefinisi untuk persoalan ini adalah
   - alldiff (F, T, U, W, R, O) {variabel yang terlibat harus mempunyai nilai yang berbeda}
   - $(O + O + C_0) \bmod 10 = R$
   - $(W + W + C_1) \bmod 10 = U$
   - $(T + T + C_2) \bmod 10 = O$
   - $C_3 = F$
   - dengan $C_0 = 0$, $C_1 = (O + O + C_0)$ div 10, $C_2 = (W + W + C_1)$ div 10, dan $C_3 = (T + T + C_2)$ div 10

III. IMPLEMENTASI

Persoalan *cryptarithmetic* dapat diselesaikan dengan menentukan digit pada setiap huruf sehingga tidak ada *constraint* yang dilanggar. Solusi dari persoalan tersebut dapat dinyatakan sebagai vektor X = (x1, x2, ..., xn), xi ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, dengan n merupakan jumlah huruf unik. Untuk dapat menyelesaikan persoalan *cryptarithmetic* dengan menggunakan algoritma *backtracking*, definisikan terlebih dahulu fungsi untuk mengecek apakah x_k dapat dibangkitkan dengan suatu nilai, kemudian definisikan juga fungsi pembatas, lalu definisikan fungsi *backtracking* nya, dan definisikan fungsi untuk memanggil fungsi *backtracking*

Berikut adalah *pseudo-code* dari fungsi tempat(i, k) yang merupakan fungsi untuk mengecek apakah x[k] dapat dibangkitkan oleh nilai i dengan cara mengecek apakah nilai i pernah dibangkitkan pada x[0] ... x[k-1] atau tidak, jika nilai i belum pernah dibangkitkan maka nilai i dapat di *assign* ke x[k]

```
function tempat (i, k: integer) → boolean
{ Fungsi untuk mengecek apakah x[k] dapat dibangkitkan oleh nilai i }
Algoritma:
  j traversal [0...k-1]
    if (x[j] = i) then
      return False
    endif
  return True
```

Berikut adalah *pseudo-code* dari fungsi boundingfunction(k) untuk mengecek apakah kombinasi digit yang telah dibangkitkan sampai x_k memenuhi *constraint* yang ada atau tidak

```
function boundingFunction(k : integer) → boolean
{ fungsi untuk mengecek apakah xk yang dibangkitkan mengarah ke solusi atau tidak (memenuhi semua constraint) }
Algoritma
  Ci = 0 {carry awal = 0}
  for setiap kolom penjumlahan dari kanan ke kiri do
    if operand yang dijumlahkan pada suatu kolom ada nilai yang belum dibangkitkan then
      return True { Bangkitkan terlebih dahulu agar constraintnya bisa dicek }
    if hasil pada suatu kolom ada nilai yang belum dibangkitkan then
      return True { Bangkitkan terlebih dahulu agar constraintnya bisa dicek }
    if bukan kolom paling kiri then
      if (penjumlahan operand + carry) mod 10 != result pada suatu kolom then
        return False { tidak memenuhi constraint }
      else
        if (penjumlahan operand + carry) != result pada suatu kolom then
          return False { tidak memenuhi constraint }
      Update Ci { Penjumlahan operand dan carry dibagi 10 }
  for operand dan hasil do
    if panjang operan atau hasil > 1 and nilainya = 0 then
      return False { Huruf awal jika panjangnya > 1 maka tidak boleh nol }
  return True {Memenuhi semua constraint}
```

Berikut adalah *pseudo-code* dari fungsi backTracking(k) untuk melakukan pencarian solusi dengan algoritma *backtracking*

```
procedure backTracking(input k : integer)
{ Mencari solusi persoalan cryptarithmetic dengan menggunakan algoritma backtracking
Masukan: k yaitu indeks komponen vektor solusi
Keluaran: sol yang merupakan himpunan solusi yang memenuhi semua constraint berbentuk X = (x1, x2, ..., xn) dengan n merupakan banyak huruf unik
}
Algoritma
  for i ∈ {0, 1, ..., 9} do
    if (tempat(i, k)) then
      x[k] = i
      if(boundingFunction(k)) then
        if k = n then { jika x sudah terisi semua }
          tambahkan x sekarang ke solusi
        else
          backTracking(k+1)
    { Jika boundingFunction false maka akan backtrack }
```

Persoalan *cryptarithmetic* dapat diselesaikan jika jumlah huruf atau simbol unik pada suatu persoalan paling banyak 10 huruf atau simbol dan persoalan *cryptarithmetic* mungkin memiliki solusi jika panjang hasil tidak lebih kecil dari panjang operand maksimal dan tidak lebih besar dari panjang operand maksimal ditambah satu. Berikut adalah *pseudo-code* dari fungsi solve() untuk memanggil fungsi *backtracking* dan sebelumnya akan dilakukan pengecekan beberapa *constraint*

yang menentukan apakah suatu persoalan *cryptarithmetic* dapat diselesaikan atau tidak

```
procedure solve()
{ Mencari solusi dari suatu persoalan cryptarithmetic dengan menggunakan algoritma backtracking }
Algoritma:
  output(Persoalan) { Tuliskan persoalannya }
  if jumlah huruf unik < 10 then
    if (isSolvable()) then { Mengecek apakah persoalan cryptarithmetic mungkin diselesaikan}
      printSolusi() {Jika ada solusi tampilkan dan jika tidak tampilkan pesan tidak ada solusi}
    else
      output("Persoalan tidak dapat diselesaikan")
  else
    output("Tidak bisa diselesaikan karena jumlah huruf unik lebih dari 10")
```

Berikut adalah strategi penyelesaian persoalan *cryptarithmetic* dengan menggunakan algoritma *backtracking* secara umum

1. Baca input nama file persoalan cryptarithmetic yang ingin diselesaikan
2. Baca file persoalan cryptarithmetic dari user
3. Proses file sehingga operand dan hasil dapat teridentifikasi
4. Buat solusi persoalan terurut berdasarkan kemunculan huruf dari kolom kanan ke kiri dan baris awal ke bawah
5. Cek constraint apakah persoalan bisa dan mungkin dicari solusinya
6. Jika persoalan bisa dan mungkin dicari solusinya, cari solusi persoalan dengan menggunakan algoritma backtracking dan jika ada solusi tampilkan seluruh solusinya dan jika tidak tampilkan pesan bahwa tidak ada solusi
7. Jika persoalan tidak bisa dicari solusinya, tampilkan bahwa persoalan tidak dapat diselesaikan

IV. PENGUJIAN

Berikut ini akan dilakukan pengujian untuk menyelesaikan beberapa input persoalan *cryptarithmetic*. Pertama program akan meminta input nama file persoalan *cryptarihmetic* yang akan diujikan, kemudian program akan menyelesaikan persoalan tersebut dan akan menampilkan solusi ataupun pesan jika tidak terdapat solusi atau tidak bisa diselesaikan. Pengujian pertama akan dicoba dengan input SEND + MORE = MONEY, pengujian kedua akan dicoba dengan input THREE + THREE + TWO + TWO + ONE = ELEVEN, pengujian ketiga akan dicoba dengan input JAMES + LILY = HARRY, pengujian keempat akan dicoba dengan input ABCD + ABC = EFG, pengujian kelima akan dicoba dengan input AAA + AAA = AAA, dan pengujian keenam akan dicoba dengan input ABCDE + FGHIJ = KLMNO

1. Pengujian Pertama (SEND + MORE = MONEY)

```
D:\Fileq\IF\Semester 4\Strategi Algoritma\ProgramMakalah\T1\src>python Cryptarithmetic.py
Masukkan nama file: pengujian1.txt
==================================================
                    PERSOALAN:
==================================================
SEND
MORE
--------- +
MONEY
================================================================================================================
                    Terdapat 1 Solusi dari permainan cryptarithmetic ini, yaitu:
================================================================================================================
********* Solusi ke-[1] : ['D', 'E', 'Y', 'N', 'R', 'O', 'S', 'M'] = [7, 5, 2, 6, 8, 0, 9, 1] *********
9567
1085
---------+
10652

================================================================================================================
Persoalan ini dapat diselesaikan dengan menggunakan algoritma backtracking dalam waktu 0.1716508865356445 3 detik
================================================================================================================
```

Gambar 4.1 Hasil pengujian1

Pengujian pertama berhasil menemukan satu-satunya solusi dari persoalan tersebut karena hanya terdapat satu kombinasi digit yang dapat memenuhi semua constraint dari persoalan tersebut

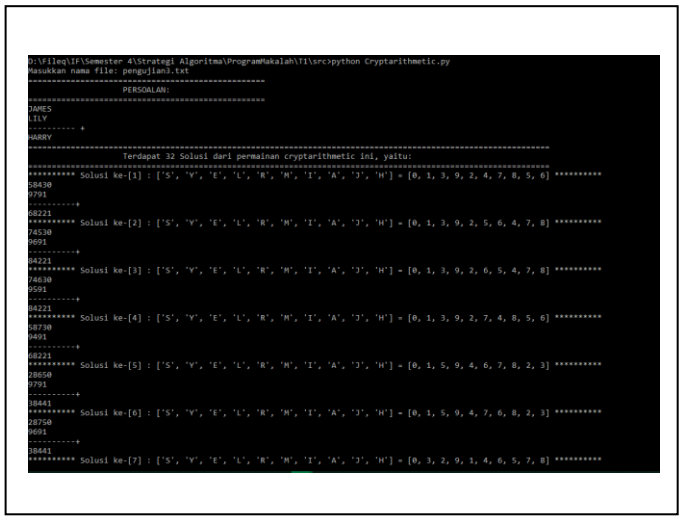
2. Pengujian Kedua (THREE + THREE + TWO + TWO + ONE = ELEVEN)

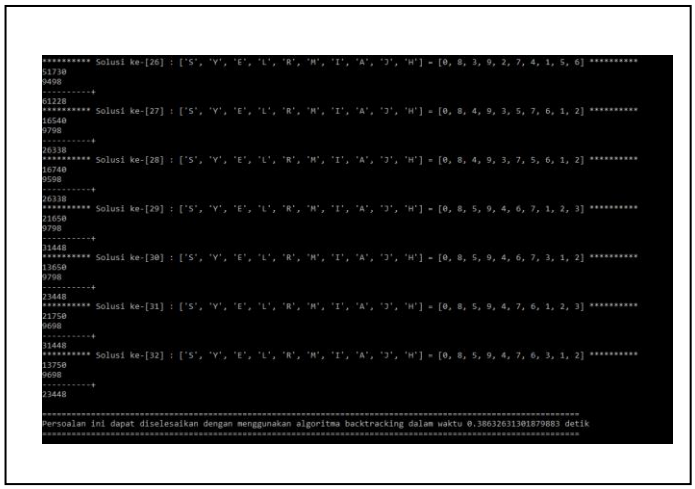
```
D:\Fileq\IF\Semester 4\Strategi Algoritma\ProgramMakalah\T1\src>python Cryptarithmetic.py
Masukkan nama file: pengujian2.txt
==================================================
                    PERSOALAN:
==================================================
THREE
THREE
TWO
TWO
ONE
---------- +
ELEVEN
================================================================================================================
                    Terdapat 1 Solusi dari permainan cryptarithmetic ini, yaitu:
================================================================================================================
********* Solusi ke-[1] : ['E', 'O', 'N', 'W', 'R', 'T', 'V', 'H', 'L'] = [1, 3, 9, 0, 6, 8, 2, 4, 7] *********
84611
84611
803
803
891
----------+
171219

================================================================================================================
Persoalan ini dapat diselesaikan dengan menggunakan algoritma backtracking dalam waktu 0.4023010730743408 detik
================================================================================================================
```

Gambar 4.2 Hasil Pengujian 2

Pengujian kedua berhasil menemukan satu-satunya solusi dari persoalan tersebut karena hanya terdapat satu kombinasi digit yang dapat memenuhi semua constraint dari persoalan tersebut

3. Pengujian Ketiga (JAMES + LILY = HARRY)

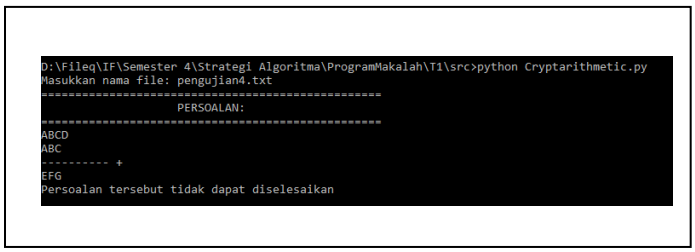




Gambar 4.3 Hasil pengujian 3

Pengujian ketiga berhasil menemukan 32 solusi dari persoalan tersebut karena terdapat tiga puluh dua kombinasi digit yang dapat memenuhi semua constraint dari persoalan tersebut

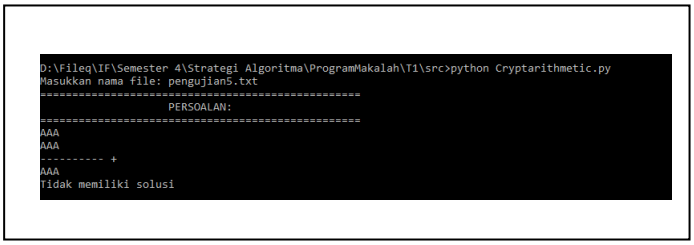
4. Pengujian Keempat (ABCD + ABC = EFG)



Gambar 4.4 Hasil pengujian 4

Pengujian keempat tidak berhasil menemukan solusi, persoalan tersebut tidak dapat diselesaikan karena panjang kata pada hasil lebih kecil dari panjang kata maksimal dari operand (3 < 4)

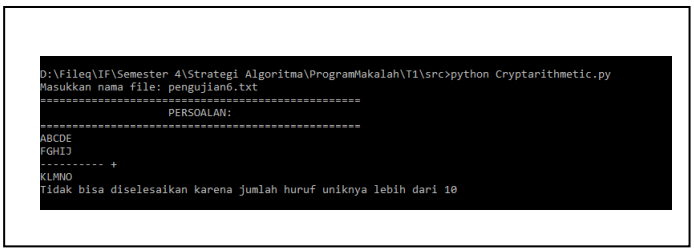
5. Pengujian Kelima (AAA + AAA = AAA)



Gambar 4.5 Hasil pengujian 5

Pengujian kelima tidak berhasil menemukan solusi dari persoalan tersebut karena meskipun persoalan tersebut dimungkinkan dilakukan pencarian solusi dengan menggunakan algoritma backtracking tapi ternyata tidak ada kombinasi digit yang dapat memenuhi semua constraint persoalan tersebut

6. Pengujian Keenam (ABCDE + FGHIJ = KLMNO)



Gambar 4.6 Hasil pengujian 6

Pengujian keenam tidak berhasil menemukan solusi, persoalan tersebut tidak dapat diselesaikan karena jumlah huruf unik lebih dari sepuluh

V. KESIMPULAN

Persoalan cryptarithmetic dapat diselesaikan dengan menggunakan algoritma backtracking. Penggunaan algoritma backtracking untuk menyelesaikan persoalan tersebut lebih efektif daripada menggunakan algoritma brute force karena pada algoritma backtracking tidak mengeksplorasi kemungkinan yang tidak menuju solusi dan jika pada saat pencarian tidak memenuhi constrant maka akan dipotong dan dilakukan backtrack.

VIDEO LINK AT YOUTUBE

https://youtu.be/mP5AxMMxDmk

SOURCE CODE LINK

https://github.com/pranagusriana/cryptarithmetic-using-backtracking

UCAPAN TERIMA KASIH

Terima kasih kepada Allah swt. karena senantiasa memberikan rahmat dan karunianya sehingga penulis dapat menyelesaikan makalah ini dengan tepat waktu. Penulis juga berterima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT., selaku dosen mata kuliah IF2211 Strategi Algoritma atas segala bimbingan dan ilmu yang telah diberikan kepada penulis. Penulis juga mengucapkan terima kasih kepada kedua orangtua penulis dan semua pihak yang telah membantu penulis baik secara langsung maupun tidak langsung. Terakhir, penulis mengucapkan terima kasih kepada diri penulis sendiri karena telah mencoba menyelesaikan makalah ini dengan semaksimal mungkin. Dalam pengerjaan makalah ini dimungkinkan terdapat kesalahan yang tidak disengaja. Oleh sebab itu, penulis sangat terbuka untuk menerima kritik, saran, serta komentar dari berbagai pihak agar kedepannya penulis dapat mengerjakannya dengan lebih baik lagi. Semoga dengan adanya makalah ini dapat bermanfaat bagi orang banyak.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf . Diakses pada 11 Mei 2021

[2] Russel S.J., Norvig P. Artificial Intelligence: A Modern Approach Third Edition, Prentice Hall, Englewood Cliffs, 2010

[3] https://www.basic-mathematics.com/cryptarithms.html . Diakses pada 11 Mei 2021

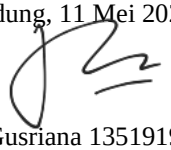
[4] https://educalingo.com/en/dic-en/cryptarithm . Diakses pada 11 Mei 2021

[5] https://www.codeproject.com/Articles/176768/Cryptarithmetic . Diakses pada 11 Mei 2021

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Mei 2021



Prana Gusriana 13519195