Penyelesaian *Verbal Arithmetic* dengan Algoritma *Brute Force*

Luthfi Chandra Fibrian - 13510047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Luthfi.chandra@s.itb.ac.id

***Abstract*—*Verbal arithmetic* adalah salah satu permainan matematika yang menyerupai aljabar. Teka-teki ini mensubtitusi semua angka dengan huruf. Untuk memecahkan teka-teki ini bisa digunakan sifat-sifat bilangan dan teknik aljabar. *Puzzle* ini dapat diselesaikan dengan memanfaatkan berbagai algoritma. Salah satu algoritma yang bisa digunakan adalah algoritma *brute force*. Algoritma *brute force* dapat menghasilkan solusi paling optimal dan cocok untuk permasalahan yang relatif kecil seperti pada persoalan verbal arithmetic. Algoritma *brute force* untuk persoalan ini dapat memanfaatkan berbagai *heuristic* yang berdasarkan teknik-teknik aljabar.**

***Index Terms*— *Verbal arithmetic, brute force, exhaustive search, heuristic,* aljabar.**

I. PENDAHULUAN

Verbal arithmetic adalah salah satu permainan matematika yang terdiri dari persamaan matematika sederhana dengan angka-angka yang tidak diketahui yang dinyatakan dengan huruf-huruf. Tujuan dari permainan ini adalah menemukan nilai dari masing-masing huruf.

Persamaan matematika dalam permainan ini adalah persamaan matematika yang sederhana, seperti penjumlahan, pengurangan, perkalian, dan pembagian. Nilai dari setiap angka tidak harus unik tetapi akan lebih menarik jika nilai dari setiap angka unik.

Peraturan umum *verbal arithmetic puzzle*:

- Tujuan dari permainan ini adalah mencari nilai masing-masing huruf.
- Setiap huruf merepresentasikan sebuah nilai yang merupakan angka 0 sampai angka 9.
- Operasi pada soal adalah pertambahan, pengurangan, perkalian, dan pembagian.
- Tidak ada angka yang dimulai oleh angka 0. Contoh: PLAY = 0123 tidak valid karena dimulai dengan angka 0.

Berikut adalah contoh dari permainan verbal arithmetic:

	P	L	A	Y
+		T	H	E
	G	A	M	E

Solusi:
P=2, L =8, A =4, Y =0, T=6, H =1, E =9, G =3, M =5

	2	8	4	0
+		6	1	9
	3	4	5	9

Contoh lainnya:

	F	E	A	R
+			N	O
	E	V	I	L

Solusi:
0= V,1= I,2= O,3= R, 4= N,5= L,7= A,8= F,9= E.

	8	9	7	3
+			4	2
	9	0	1	5

Verbal arithmetic bisa digunakan untuk pengajaran aljabar. Pengsubtitusian angka dengan huruf dalam teka-teka ini serupa dengan aljabar. Lebih jauh lagi, *verbal arithmetic* juga bisa digunakan untuk memodelkan pemecahan kode enkripsi dari suatu *ciphertext.*

II. DASAR TEORI

Algoritma *brute force* adalah algoritma yang paling sederhana. Algoritma *brute force* menggunakan pendekatan yang paling *straightforward* untuk mencari solusi dari suatu permasalahan.

Pada umumnya algoritma *brute force* tidak efisien dalam pemecahan suatu masalah. Algoritma ini memerlukan komputasi yang sangat besar sehingga akan lebih baik jika algoritma ini digunakan untuk masalah yang berukuran kecil.

Algoritma *brute force* dapat digunakan untuk menyelesaikan berbagai persoalan. Hampir tidak ada persoalan yang tidak bisa diselesaikan oleh algoritma *brute force*. Algoritma *brute force* selalu menghasilkan solusi paling optimal.

Keunggulan algoritma *brute force*:

- Hampir semua persoalan bisa diselesaikan.
- Mudah dimengerti karena menggunakan pendekatan yang *straightforward*.

- Selalu menghasilkan solusi paling optimal.

Kelemahan algoritma *brute force*:

- Sangat jarang menghasilkan algoritma yang mangkus.
- Untuk permasalahan yang besar, algoritma *brute force* bisa berjalan sangat lambat.
- Kompleksitasnya sangat besar.

Salah satu varian dari algoritma *brute force* adalah *exhaustive search*. *Exhaustive search* bisa digunakan untuk mencari solusi untuk masalah-masalah kombinatorik.

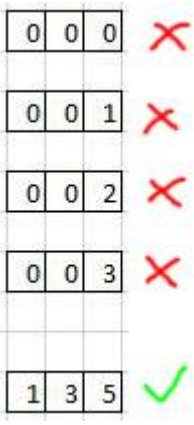
Langkah-langkah umum *exhaustive search*:

- Enumerasi setiap solusi yang mungkin.
- Evaluasi setiap kemungkinan solusi dan simpan hasil terbaiknya.
- Pilih solusi terbaik dari hasil evaluasi.

Contoh *exhaustive search*:

Misalkan terdapat sebuah kunci gembok yang dikunci menggunakan kombinasi angka sebanyak 3 digit.

Solusi:
Mencoba satu per satu kombinasi kunci dimulai dari 000 sampai hasilnya didapatkan.



Didapatkan solusi yaitu 135

Exhaustive search dapat dipercepat dengan menggunakan *heuristic*. Teknik ini digunakan untuk mengeliminasi kemungkinan-kemungkinan yang mustahil tanpa harus mengujinya dengan *exhaustive search*.

III. Penerapan Brute Force pada Verbal Arithmetic *Puzzle*

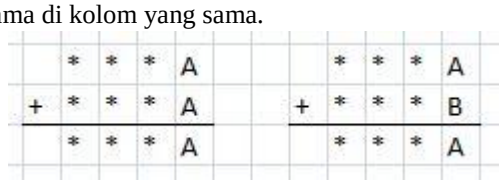
Verbal arithmetic puzzle dapat diselesaikan dengan algoritma *brute force*. Pencarian solusi dapat dilakukan dengan cara *exhaustive search*. Semua kemungkinan solusi dienumerasi lagi dicek satu per satu sampai ditemukan solusi yang valid.

3.1 ***Heuristic***

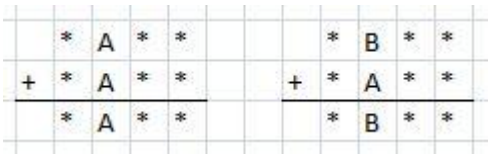
Semakin banyak jumlah huruf, maka semakin lama proses pencarian solusinya. Oleh karena itu, penyelesaian sebaiknya tidak menggunakan *brute force* secara murni. Akan lebih baik jika menerapkan *heuristic* untuk mempercepat pencarian solusi.

3.1.1 Mencari "0" dan "9" di Penjumlahan dan Pengurangan

Angka 0 dan 9 bisa dicari jika ada dua atau tiga huruf yang sama di kolom yang sama.



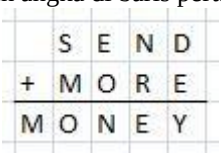
Dari kolom A + A = A dan A + B = B dapat disimpulkan bahwa nilai adalah 0 karena angka berapapun jika ditambahkan atau dikurangi 0 pasti tidak akan berubah.



Pada contoh di atas bisa didapatkan A = 9 atau A = 0. Ini tergantung oleh adanya carry 1 dari hasil operasi kolom sebelumnya.

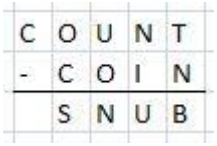
3.1.2 Mencari "1" di Penjumlahan atau Pengurangan

Angka 1 bisa ditemukan jika jumlah angka di baris pertama = jumlah angka dibaris kedua dan jumlah baris hasil operasi = jumlah angka di baris pertama + 1.



M pasti bernilai 1 karena menerima carry dari penjumlahan S+M=O dan nilai S+M ≥ 10.

Pada operasi pengurangan, angka 1 bisa ditemukan jika jumlah angka di baris kedua = jumlah angka di baris pertama – 1 dan jumlah angka di baris kedua = jumlah angka di baris hasil operasi.



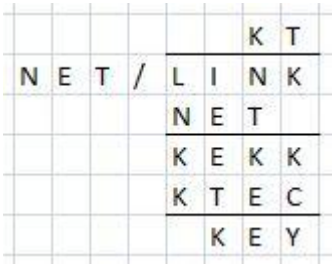
Pada contoh di atas, C pasti bernilai 1.

3.1.3 Mencari "1" di Perkalian atau Pembagian

Semua angka jika dikalikan dengan 1 maka hasilnya akan tetap angka itu. Contoh:

MAKE * T = MAKE. Dapat disimpulkan bahwa nilai T = 1.

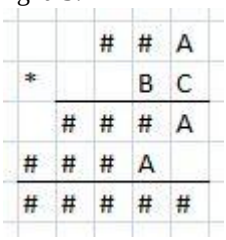
Pada operasi pembagian juga berlaku hal yang sama. Contoh:



Pada operasi pertama, didapatkan K * NET = NET. Dapat disimpulkan nilai K = 1.

3.1.4 Mencari "0" dan "5" di Operasi Perkalian atau Pembagian

Angka apapun jika dikali dengan 0 pasti hasilnya adalah 0. Angka ganjil apapun jika dikalikan dengan 5 pasti menghasilkan angka 5.



Dari contoh di atas didapatkan:
##A * C = ###A
##A * B = ###A
Dapat disimpulkan A = 0 atau A = 5.

3.2 **Brute Force**

Jika solusi belum ditemukan setelah menerapkan *heuristic* maka langkah selanjutnya adalah melakukan *brute force*.

Struktur data yang digunakan adalah sebagai berikut:

```
struct Soal {
        char[] operand1;
        char[] operand2;
        char[] hasil;
        }
integer[] kemungkinan;
```

Setiap element dari operand1, operand2, dan hasil memiliki array kemungkinan yang menyatakan angka-angka apa saja yang mungkin menjadi solusi. Pada awal program array kemungkinan diinisialisasi dengan {0,1,2,..,.9}.

Untuk menguji semua kemungkinan digunakan iterasi yang mirip dengan iterasi pada odometer. Ilustrasi:

Tahap selanjutnya:

Ketika operand1 sudah diuji semua kemungkinannya, operand2 dilakukan increment. Setelah semua kemungkinan operand2 diuji, hasil dilakukan increment. Iterasi dilakukan berulang-ulang sampai semua kemungkinan dicoba atau sampai solusi didapatkan. Pada saat pengujian setiap huruf harus memiliki angka yang unik.

Berikut ini adalah *pseudo-code* dari program *verbal arithmetic solver*:

Prosedur terapkan_*heuristic*:

```
  procedure terapkan_heuristic(input/output
Soal soal)
  {menerapkan heuristic untuk mengeliminasi
kemungkinan yang sudah pasti salah.
Angka yang dianggap tidak mungkin dihapus
dari array kemungkinan}
```

Prosedur cari_solusi:

```
procedure cari_solusi(input/output
soal:Soal)
{mencari solusi dari persoalan dengan
mencoba semua kemungkinan}

DEKLARASI
boolean found

ALGORITMA
found <-- false
while(not found) do
       foreach(kemungkinan_h in
                     array_hasil) do
              foreach(kemungkinan2 in
operand2) do
                     foreach(kemungkinan1 in
operand1) do

       increment(kemungkinan1)
                            if(semua angka
unik and operand1 + operand2 = array_hasil)
then
                                   found <--
true
                            endif
                     endfor
                     increment(kemungkinan2)
              endfor
```

```
                increment(kemungkinan_h)
        endfor
endwhile
```

Berikut ini adalah contoh eksekusi dari pseudocode tersebut:

Soal:

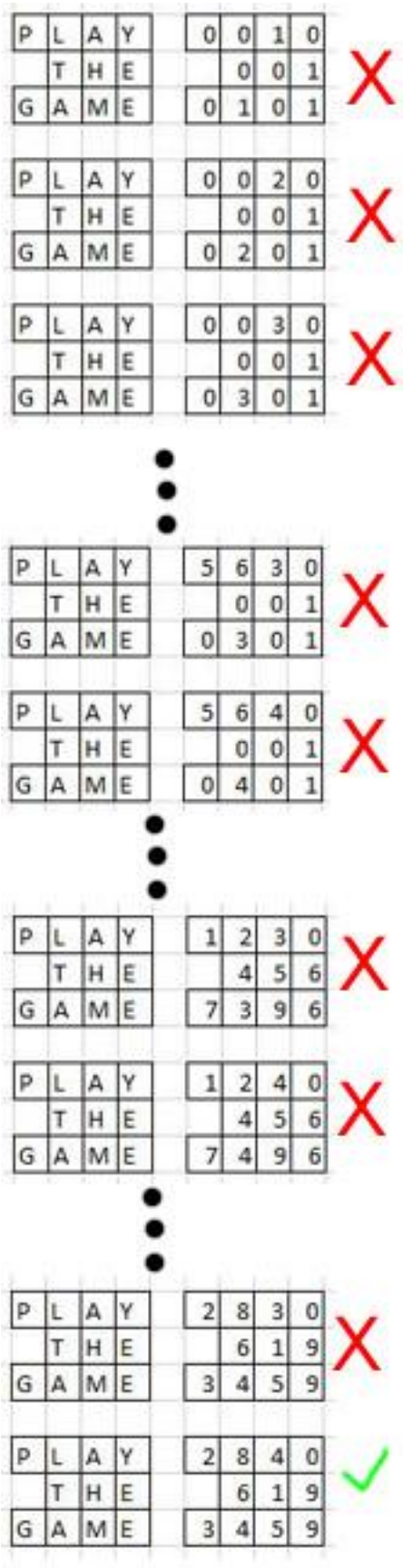
	P	L	A	Y
+		T	H	E
	G	A	M	E

Setelah dilakukan prosedur terapkan_*heuristic*:

kemungkinan1	P	0	1	2	3	4	5	6	7	8	9
	L	0	1	2	3	4	5	6	7	8	9
	A	0	1	2	3	4	5	6	7	8	9
	Y	0									
kemungkinan2	T	0	1	2	3	4	5	6	7	8	9
	H	0	1	2	3	4	5	6	7	8	9
	E	0	1	2	3	4	5	6	7	8	9
kemungkinan_h	G	0	1	2	3	4	5	6	7	8	9
	A	0	1	2	3	4	5	6	7	8	9
	M	0	1	2	3	4	5	6	7	8	9
	E	0	1	2	3	4	5	6	7	8	9

Pada tahap ini huruf Y memenuhi *heuristic*. Pada persoalan ini hanya satu huruf yang memenuhi *heuristic* sehingga jumlah kemungkinan solusinya besar. Hal ini akan menyebabkan lamanya proses pencarian solusi.

Selanjutnya dilakukan evaluasi untuk semua kemungkinan sampai didapatkan solusinya.



Setelah dilakukan evaluasi, didapatkan solusi sebagai berikut:

- P = 2

- L = 8
- A = 4
- Y = 0
- T = 6
- H = 1
- E = 9
- G = 3
- M = 5

IV. Analisis

Persoalan *verbal arithmetic* dapat diselesaikan dengan algoritma *brute force* tetapi memerlukan komputasi yang sangat besar. Pada contoh di atas jumlah evaluasi yang dilakukan sangat banyak dan memakan waktu yang cukup lama.

Semakin banyak huruf dan bilangan yang digunakan, semakin banyak evaluasi yang dilakukan. Pada contoh terdapat sembilan huruf dan bilangan yang digunakan dimulai dari 0 sampai 9.

Kompleksitas algoritma *brute force* untuk persoalan ini adalah:

$$O(n^m)$$

m adalah banyaknya huruf

n adalah banyaknya bilangan.

V. Kesimpulan

Persoalan verbal arithmetic dapat diselesaikan dengan algoritma *brute force* tetapi membutuhkan waktu yang cukup lama. Meskipun sudah diterapkan beberapa *heuristic*, tetap saja waktu yang dibutuhkan untuk pencarian solusi cukup lama. Untuk mempercepat waktu eksekusi algoritma, perlu ditambahkan lagi *heuristic-heuristic* lainnya untuk mengeliminasi kemungkinan-kemungkinan yang mustahil.

References

[1] *Rinaldi Munir, Diktat Kuliah IF3051 Strategi Algoritma*, Program Studi Teknik Informatika ITB, 2005

[2] http://www.*puzzle*s-on-line-niche.com/math-games.html#sthash.KZ0IwxKa.dpbs waktu akses: 20 Desember 2012 pukul 19.00

[3] http://cryptarithms.awardspace.us/primer.html#05M waktu akses: 20 Desember 2012 pukul 20.00

[4] http://www.mathematik.uni-bielefeld.de/~sillke/PUZZLES/ALPHAMETIC/ waktu akses: 20 Desember 2012 pukul 20.00

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, 21 Desember 2012

Luthfi Chandra Fibrian
13510047