Menyelesaikan *Puzzle* Matematika Braingle dengan Algoritma *Brute Force*

Drestanto Muhammad Dyasputro - 13514099
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
dyas@live.com

***Abstract*—Persoalan-persoalan sering sekali bermunculan pada hidup ini. Beberapa persoalan dapat diselesaikan, beberapa persoalan lain, mungkin tidak terselesaikan. Tidaklah mudah untuk menyelesaikan seluruh persoalan dengan baik dan benar. Berbagai metode untuk menyelesaikan masalah pun diusulkan. Namun, metode-metode yang dilakukan, tidak jarang, adalah metode abstrak yang tidak memiliki keterurutan (tidak sistematis). Komputer, adalah suatu benda, yang dapat menyelesaikan masalah secara terurut dan sistematis. Namun, komputer pun harus diberi tahu terlebih dahulu, bagaimana cara menyelesaikan masalahnya. Cara memberitahunya adalah dengan memasukkan algoritma ke dalam komputer tersebut. Algoritma-algoritma yang digunakan para programmer beragam bentuknya. Namun, ada algoritma yang sangat sederhana dan sering digunakan. Metode ini adalah metode mengenumerasi segala kemungkinan solusi untuk memecahkan masalah. Metode termudah dalam komputasi untuk menyelesaikan suatu persoalan adalah brute force.**

***Keywords*—persoalan, algoritma, brute force**

I. PENDAHULUAN

Braingle adalah sebuah situs matematika yang didalamnya terdapat beberapa persoalan matematika yang menarik.

II. DASAR TEORI

A. Algoritma

Dahulu, orang menghitung, mengkalkulasi, dan menyelesaikan persoalan secara langsung, spontan, dan gamblang. Hal ini tidaklah 100% buruk, namun pastinya memiliki kelemahan-kelemahan.

Adalah masalah apabila muncul persoalan yang serupa. Kita belum tentu dapat menyelesaikan sebagaimana layaknya dulu (karena penyelesaian dilakukan secara gamblang). Karena itulah, kita membutuhkan sesuatu metode penyelesaian masalah yang terstruktur. Dapat diulang-ulang. Hal ini pun sangat penting dalam proses pembuatan komputer yang sekarang sangat terkenal dan memiliki bentuk bermacam-macam (Handphone, Tab, Note, dll).

Dari latar belakang inilah, ditemukannya suatu algoritma. Menurut technopark.surakarta.go.id, algoritma adalah urutan langkah-langkah logis pada penyelesaian masalah yang disusun secara sistematis. Masalah dapat berupa apa saja, dengan catatan untuk setiap masalah ada syarat kondisi awal yang harus dipenuhi sebelum menjalankan algoritma.

Dan dari adanya usulan inipun, kita dapat membuat suatu sistem yang terstruktur (sistematis) pada kehidupan ini. Namun, yang akan difokuskan pada makalah ini adalah algoritma yang digunakan oleh komputer.

Berbagai usulan algoritma pun berdatangan bergantung pada persoalan apa yang mereka temui. Tentunya, algoritma menjadi popular apabila persoalan yang ditemui adalah persoalan yang universal dan popular juga (Seperti *Travelling Salesperson Problem*, *Searching*, *Sorting*, dsb), serta algoritma tersebut dapat menyelesaikan masalah tersebut secara umum.

Dari situ, muncullah banyak bentuk strategi penyelesaian masalah dengan algoritma brute force, greedy, divide and conquer, decrease and conquer, BFS DFS, A star, program dinamis dan masih banyak lagi.

Untuk lebih lanjutnya, algoritma akan semakin popular apabila penyelesaian masalah dilakukan secara cepat (mengefisiensi waktu) ataupun murah (mengefisiensi biaya / ruang).

B. Brute Force

Algoritma yang paling sering ditemukan dalam memecahkan masalah, tak lain adalah brute force. Mengapa brute force? Karena algoritma ini adalah algoritma paling sederhana dan paling mudah terpikirkan oleh banyak orang. Algoritma ini pun menjadi algoritma yang paling mudah dimenegerti.

Brute force adalah algoritma yang menggunakan pendekatan gamblang (spontan, cepat, langsung,

straightforward) dalam memecahkan suatu persoalan. Banyak persoalan-persoalan yang dipecahkan dengan brute force karena persoalan tidak terlalu rumit. Dan biasanya, jika kita melakukan coding program dengan sebuah algoritma, algoritma brute force adalah algoritma yang biasanya memiliki line coding yang paling sedikit (dengan kata lain, algoritma ini adalah algoritma yang paling sederhana).

Tentu saja, kesederhanaan dari algoritma ini memiliki konsekuensi. Beberapa persoalan diselesaikan dalam waktu yang relatif lama. Hal ini karena strategi pemecahan masalahnya masih gamblang dan belum terlalu dipikirkan. Berbeda dengan algoritma-algoritma lain yang biasanya lebih efisien.

C. Kompleksitas

Untuk membandingkan kemampuan suatu algoritma, secara nalar, kita akan berpikir untuk membandingkan waktunya. Namun, biasanya komputer melaksanakan tahapan algoritma dengan sangat ceapt sehingga kita pun kesulitan untuk mengukur waktunya.

Sekarang, hampir semua bahasa pemrograman menyediakan fungsi untuk menghitung waktu. Waktu yang dihitung cukup presisi, hingga milisekon. Bahkan ada program yang bisa menghitung hingga mikrosekon. Namun, ini pun masih memiliki masalah, karena kecepatan program akan terukur bergantung kepada komputernya. Kita tetap tidak dapat membandingkan.

Karena itu, diperkenalkanlah yang disebut dengan kompleksitas. Kompleksitas adalah tahapan komputasi dari sebuah algoritma. Kompleksitas yang akan difokuskan pada makalah ini adalah kompleksitas waktu. Dengan menggunakan kompleksitas, kita dapat membandingkan algoritma mana yang lebih baik dari yang lain. Biasanya dilambangkan dengan T(n), dimana n adalah jumlah data yang dijadikan masukkan. Contohnya adalah sebagai berikut :

$$T(n) = 2n - 2$$
$$T(n) = n^3 - 2n + 1$$
$$T(n) = 2^n - 2$$
$$T(n) = n!$$
$$T(n) = 2^n * n^2 + 5n^7$$
dan sebagainya

Notasi asimtotik yang sering digunakan untuk menyatakan kebaikan suatu algoritma adalah notasi algoritmik big-oh. Big-Oh menggambarkan batas atas suatu pertumbuhan fungsi apabila masukkan fungsi (n) lebih banyak. Contohnya adalah sebagai berikut :

$$T(n) = 2n - 2 = O(n)$$
$$T(n) = n^3 - 2n + 1 = O(n^3)$$
$$T(n) = 2^n - 2 = O(2^n)$$
$$T(n) = n! = O(n!)$$
$$T(n) = 2^n * n^2 + 5n^7 = O(2^n * n^2)$$
dan sebagainya

III. PERSOALAN DAN SOLUSI

A. Persoalan

Pada situs braingle, terdapat persoalan-persoalan yang menarik. Buksalah link di bawah ini

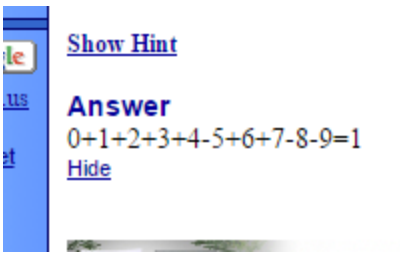
http://www.braingle.com/brainteasers/4603/0-1-2-3-4.html

Link tersebut menuju ke salah satu persoalan aritmatika pada situs tersebut. Persoalan aritmatika pada link tersebut adalah sebagai berikut :

Put the appropriate + or - signs between the numbers, in the correct places, to make this equation true:

0 1 2 3 4 5 6 7 8 9 = 1

Kita diminta untuk mengisi slot anatara angka-angka di atas dengan operator + atau - sehingga menghasilkan nilai 1. Hal ini mungkin tidaklah sulit dan kita dapat melihat jawabannya secara langsung pada situs tersebut :



Perhatikan bahwa jawaban di atas tepat, tapi apakah itu satu-satunya jawaban. Hal ini menarik untuk ditanyakan. Dan, bagaimana bila kita tidak hanya menghitung sampai 0, tapi juga menghitung sampai 10? Ini menjadi persoalan yang menarik.

Persoalan yang akan dibahas pada makalah ini pun adalah pengembangan dari persoalan di atas. Persoalannya kurang lebih sebagai berikut :

How many combination of appropriate + or - signs between the numbers that we can put, in the correct places, to make this equation true:

$$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10 = 1$$

Dari pertanyaan ini, kita tidaklah mungkin untuk mengerjakan secara manual. Kita harus mengenumerasi segala kemungkinan untuk menyelesaikan persoalan ini.

B. Solusi (metode brute force)

Persoalan di atas dapat diselesaikan dengan metode brute force. Caranya adalah, dengan mengenumerasi semua kemungkinan operator + dan - pada setiap slot operator. Karena ada 10 slot operator, dan setiap slot memiliki 2 kemungkinan operator. Maka, seluruh kemungkinannya adalah :

$$2*2*2*2*2*2*2*2*2*2 = 2^{10} = 1024$$

Mencoba seluruh kemungkinan (1024 kali percobaan) secara manual (dengan manusia, tanpa kalkulator ataupun komputer) sama saja dengan bunuh diri (sangatlah sulit). Namun, tidaklah sulit membuat program sederhana yang akan mengenumerasi seluruh kemungkinan jawaban dan mencetak ke layar kemungkinan-kemungkinan yang benar (merupakan solusi). Harusnya, hanya membutuhkan waktu kurang dari 1 detik untuk komputer menghitungnya.

Untuk membuat komputer dapat menghitung, kita harus mengajarkan komputer (memberitahu apa yang harus dia hitung). Caranya adalah dengan membuat program. Sebelum itu, marilah kita dekomposisi masalah ini sehingga pembuatan program dapat berjalan dengan lancar.

Pertama, kita harus menentukan, bagaimana data operator disimpan. Di sini, saya membuat array of boolean yang panjangnya adalah 10. Setiap Boolean merepresentasikan operator. True berarti + dan False berarti -. Setelah menyimpan operator dalam bentuk boolean, kita harus dapat mengkalkulasi persoalan di atas, apakah nilainya adalah 1?

Untuk melakukan kalkulasi, saya membuat fungsi baru. Fungsi ini saya sebut fungsi Solve(). Fungsi ini menerima array of boolean sebagai input, dan memberikan nilai sebagai output. Contohnya :

Apabila array of boolean bernilai true semua, maka :

$$0+1+2+\ldots+9 = 55$$

Fungsi Solve akan melemparkan nilai 55.

Intinya, fungsi ini akan menghitung nilai yang dihasilkan apabila operator yang digunakan seperti operator array of boolean tersebut.

Setelah membuat fungsi Solve, kita pun harus memiliki cara untuk menuliskan ke layar. Sehingga, kita membutuhkan suatu prosedur di sini. Prosedur ini saya beri nama Print. Menerima input sebuah array of boolean, dan menulis ke layar rentetan operator disertai dengan angka-angkanya. Contoh eksekusinya akan terlihat pada hasil eksekusi program.

Setelah membuat dua hal tersebut (satu fungsi Solve dan satu prosedur Print), maka kita siap membuat prosedur terakhir yang paling penting, yaitu mengenumerasi seluruh kemungkinan jawaban. Persoalan ini sesungguhnya mirip persoalan subset jika diperhatikan. Jika kita anggap indeks sebagai elemen himpunan, dan true berarti elemen himpunan ini termasuk dalam subset, maka, kita mengenumerasi seluruh subset dari himpunan bilangan 1 sampai 10. Karenanya, kita butuh mengenumerasi 2^{10} subset.

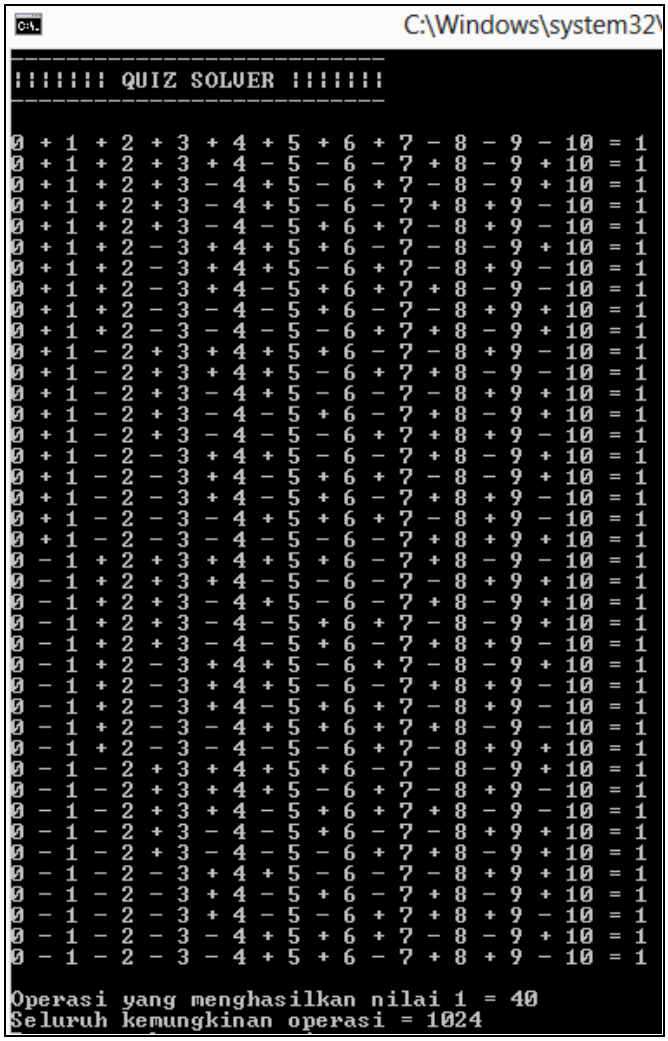
Membuat algoritma untuk mengenumerasi subset sesungguhnya tidak semudah membuat algoritma untuk melakukan traversal biasa. Pendekatan yang saya lakukan untuk mengenumerasi seluruh kemungkinan jawaban adalah dengan pendekatan rekursif.

Algoritmanya ditulis dalam bahasa pascal sebagai berikut :

```
procedure LetsGo(var op : Tabel;
    i, j : integer;
    var jmlOp, totOp : integer);
begin
    if (i=j) then
    begin
        op[j] := true;
        totOp := totOp +1;
        if (Solve(op)=1) then
        begin
            jmlOp := jmlOp+1;
            Print(op);
        end;
        op[j] := false;
        totOp := totOp + 1;
        if (Solve(op)=1) then
        begin
            jmlOp := jmlOp+1;
            Print(op);
        end;
    end else //i!=j
    begin
        op[i] := true;
        LetsGo(op,i+1,j,jmlOp,totOp);
        op[i] := false;
        LetsGo(op,i+1,j,jmlOp,totOp);
    end;
end;
```

Algoritma ini memanggil fungsi dan prosedur yang telah dijelaskan sebelumnya yaitu Solve() dan Print(). Perhatikan bahwa algoritma ini rekursif dengan basis jika i=j.

Pada algoritma di atas, dicatat juga jmlOp dan totOp untuk menghitung jumlah solusi dan jumlah enumerasi. Dari algoritma di atas, program pun dapat dijalankan. Hasil aksekusi (running) program tersebut adalah sebagai berikut :



Dari hasil di atas, kita mendapatkan 40 kemungkinan solusi. Dan kita telah mengenumerasi 1024 kemungkinan. Hal ini tentu saja sulit untuk dilakukan secara manual oleh manusia. Komputer telah menyelesaikannya!

Tentunya, kita tidak akan meninggalkan solusi seperti di atas. Hal yang selalu penting untuk diketahui setelah kita menyelesaikan persoalan-persoalan dengan suatu algoritma adalah kompleksitas.

Kompleksitas algoritma persoalan ini dapat dihitung sebagai berikut :

1. Karena LetsGo mengenumerasi seluruh kemungkinan, maka :

$$T(n) = 2^n$$

2. Di dalam prosedur LetsGo, setiap tahapan pasti memanggil fungsi Solve (perhatikan pada bagian basis). Maka :

$$T(n) = n$$

3. Jika kasus solve masuk ke dalam if, maka, program harus memanggil prosedur Print yang pastinya juga memiliki kompleksitas :

$$T(n) = n$$

4. Sehingga, jika digabung seluruhnya, ambil kasus terburuk, dimana prosedur Print selalu terpanggil :

$$T(n) = 2^n * (n + n)$$
$$= 2^n * (2n)$$
$$T(n) = O(n*2^n)$$

Mungkin, kompleksitas ini terasa buruk. Namun, penyelesaian secara brute force tidak memerlukan kita sebagai manusia untuk berpikir terlalu rumit. Kelebihan-kelebihan brute force pun akan dijelaskan lebih lanjut pada bagian V.

IV. PENGEMBANGAN PERSOALAN

Setelah menyelesaikan persoalan dan menentukan kompleksitas algoritma solusi pun, bukan berarti kita dapat meninggalkan program begitu saja (melupakannya dan membuangnnya dari ingatan). Untuk mengembangan diri kita lebih baik lagi, persoalan yang sudah kita selesaikan, tidak kita tinggalkan begitu saja. Cobalah untuk membuat persoalan semakin kompleks dan levelnya semakin tinggi. Cobalah untuk menyelesaikan persoalan itu lagi.

Persoalan yang menarik untuk jadi pengembangan adalah :

1. Apabila slot operator tidak hanya bisa diisi operator + dan -, tapi juga bisa diisi dengan *. Apakah solusinya akan bertambah? Jika iya, tuliskan semua pertambahan solusinya.

   Ini tentu adalah peningkatan level persoalan. Karena, fungsi Solve dan Print harus diperbaiki. Untuk fungsi Print, mungkin tidaklah rumit untuk memperbaikinya. Namun, untuk fungsi Solve, tidaklah mudah memperbaikinya, mengingat adanya precedence antara * dan + atau -.

2. Persoalan 1 juga bisa dikembangkan dengan menambah operator lain seperti / (bagi) ataupun ^ (pangkat).

3. Atau, persoalan juga bisa dikembangkan apabila slot operator dibiarkan kosong sehingga angka yang berdekatan menjadi satu kesatuan
   Misalnya, slot operator antara 7 dan 8 dibiarkan kosong, sehingga, setelah angka 6, terdapat 78, kemudian terdapat 9. Hal ini juga menjadi persoalan yang lebih menarik dan tentunya lebih sulit.

Segala bentuk pengembangan dapat kita lakukan agar kita lebih tangkas dalam memecahkan persoalan. Perlu diingat, bahwa persoalan tidak akan pernah selesai. Jika persoalan selesai, sesungguhnya bukan karena persoalan itu selesai, namun karena kita memutuskan untuk selesai.

V. WHY BRUTE FORCE?

Hal menarik yang sering ditanyakan orang banyak adalah

"*Why Brute Force?*"

Maksud dari pertanyaan ini adalah. Kita memiliki banyak cara lain seperti Divide and Conquer, DFS BFS, dan lain-lain. Sebenarnya, hal yang ingin digarisbawahi pada makalah ini bukan tentang brute force-nya, namun tentang bagaimana kita dapat menyelesaikan persoalan dengan sebuah algoritma dan membiarkan komputer menyelesaikanna untuk kita. Namun, masih banyak orang yang tidak menyukai metode brute force. Dianggap metode ini terlalu buruk karena secara gamblang mengenumerasi segala kemungkinan secara langsung satu persatu.

Namun, brute force memiliki kelebihan-kelebihan dibanding algoritma yang lain.

1. Metode brute force dapat menyelesaikan seluruh persoalan komputer
   Jika ada persoalan yang bisa diselesaikan dengan metode selain brute force, maka pastilah brute force dapat menyelesaikan.
   Tapi, tidak semua persoalan bisa diselesaikan dengan cara non-brute force. Misalnya TSP (*Travelling Salesperson Problem*). Sampai sekarang, masih belum ada orang yang menemukan cara yang jika terjadi kasus terburuk (worst case), memiliki kompleksitas waktu yang lebih baik.

2. Metode brute force adalah metode yang paling sederhana
   Kenapa hal ini baik? Karena, semakin sederhana suatu algoritma, maka semakin mudah dimengerti pula algoritmanya.

3. Algoritma-algoritma seperti searching, sorting, perkalian matriks, dll masih layak untuk diselesaikan dengan brute force.
   Memang masih ada cara lain untuk menyelesaikan persoalan-persoalan di atas. Namun, kompleksitas brute force tidak begitu buruk untuk dipakai dan algoritmanya jauh lebih sederhana.

4. Algoritma brute force dijadikan standar untuk membandingkan komputasi suatu algoritma
   Suatu algoritma yang baik dikatakan baik karena kompleksitasnya lebih baik dibanding brute force. Hal ini membuktikan bahwa brute force menjadi acuan segala penyelesaian persoalan.

Hal yang penting dari kelebihan-kelebihan di atas adalah poin nomor 2. Poin ini sangatlah dijadikan pertimbangan dalam merancang suatu OS (sistem operasi).

Kita tahu, dalam membuat sistem operasi, pastilah terdapat banyak sekali algoritma-algoritma di dalamnya demi kinerja (performansi) komputer. Penting untuk diketahui, bahwa banyak algoritma-algoritma pada OS menggunakan brute force. Penggunaan brute force di sini bukanlah karena tidak ada cara lain untuk menyelesaikan persoalan OS. Namun, penggunaan brute force justru disarankan untuk megnurangi bug pada OS (adanya bug pada OS sangat berbahaya karena OS, sebagai peneglola sistem-sistem, tidak boleh sampai merusak sistem yang dikelola) dan memudahkan debugging apabila masih ada permasalahan pada OS. Perhatikan bahwa dalam perancangan OS, brute force **disarankan.**

Mungkin, masalah yang timbul berikutnya adalah bagaimana caranya untuk merancang OS yang cepat? Hal ini bisa disiasati dengan menggunakan prosesor yang lebih baik. (Penyiasatan dilakukan secara hardware)

VI. KESIMPULAN

Ternyata metode Brute Force dalam memecahkan masalah bukan metode yang buruk. Terkadang, metode ini dihindari demi kinerja (performansi) program yang kita miliki. Kita menggunakan algoritma-algoritma lain seperti Divide and Conquer, Branch and Bound, BFS DFS, dan lain-lain. Membandingkan algoritma lain dengan brute force, tentu terasa bahwa brute force bukanlah algoritma yang baik. Namun, di sisi lain, brute force menggunakan algoritma yang simple, sehingga tidak sulit untuk debugging.

Inti dari penggunaaan algoritma dalam makalah ini adalah untuk mempercepat proses pemikiran. Manusia, memiliki kecepatan berpikir yang sangat terbatas, atau bisa dikatakan lambat. Berbeda dengan komputer, yang dapat

berpikir jauh lebih cepat, sehingga, setelah diprogram, komputer pun dapat menghitung segala sesuatu dengan sangat cepat.

Dan terakhir, persoalan braingle hanya dapat menjawab satu buah solusi saja. Dan dia pun hanya mempertanyakan satu buah solusi. Namun, permasalahan ini bisa menjadi permasalahan yang lebih menarik apabila kita expand permasalahannya menjadi seperti pada makalah ini. Dan meng-expand masalah seperti makalah ini adalah cara berpikir yang baik dan kreatif untuk terus berkembang.

REFERENCES

[1] http://www.braingle.com/brainteasers/4603/0-1-2-3-4.html

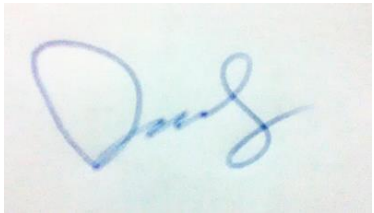
[2] Christof Paar, Jan Pelzl, Bart Preneel (2010). Understanding Cryptography: A Textbook for Students and Practitioners

[3] https://www.quora.com/What-is-a-brute-force-way-of-solving-problems

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, 10 Desember 2015



Drestanto Muhammad Dyasputro - 13514099