

**PROJECT DESAIN ANALISIS ALGORITMA**  
**20 GAME SOLVER DENGAN ALGORITMA *BRUTE FORCE***

Laporan Desain Analisis Algoritma  
Disusun untuk Memenuhi Tugas Mata Kuliah Desain Analisis Algoritma  
Dosen Pengampu:  
**Fajar Muslim, S.T, M.T.**



Disusun Oleh:

- |                    |            |
|--------------------|------------|
| 1. Mutiara Hasanah | (L0123102) |
| 2. Nayla Amira     | (L0123108) |

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA**  
**UNIVERSITAS SEBELAS MARET**  
**SURAKARTA**  
**2024**

## BAB 1 PENDAHULUAN

Permasalahan 20 solver adalah tantangan yang penyelesaiannya dengan mengkombinasikan beberapa angka dan operasi untuk mencapai hasil akhir 20. Dalam kasus ini, kita harus mencari sebanyak mungkin kombinasi operasi matematika seperti penambahan, pengurangan, perkalian, pembagian dan tanda kurung pada angka acak dengan setiap angka harus digunakan satu kali. Urutan angka dan jenis operasi yang digunakan dapat bervariasi, yang berarti ada banyak kemungkinan kombinasi yang harus dieksplorasi. dan urutan serta jenis operasinya dapat bervariasi. Selain itu, kita harus mempertimbangkan urutan operasi matematika (prioritas) dan bagaimana penggunaan tanda kurung dapat mengubah hasil perhitungan.

Tantangan utama dari masalah ini adalah kompleksitas dalam eksplorasi semua kemungkinan kombinasi angka dan operasi. Sebab, semakin banyak angka yang digunakan, semakin besar pula jumlah kombinasi yang mungkin dihasilkan serta memastikan bahwa hasil akhirnya adalah angka 20. Pendekatan untuk menyelesaikan masalah ini menggunakan metode algoritma *brute-force*, yaitu mencoba semua kemungkinan solusi untuk menemukan hasil yang valid.

## BAB 2 DASAR TEORI

Landasan teori yang akan digunakan pada penerapan strategi algoritma dalam kasus ‘20 solver’ ada dua, yaitu operasi aritmatika dasar dan *Brute-Force*.

### A. Definisi *Brute-Force*

*Brute-Force* merupakan sebuah pendekatan langsung (straight forward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Algoritma ini memecahkan masalah dengan sederhana, langsung, dan dengan cara yang jelas (obvious way). Algoritma ini juga sering kali disebut dengan algoritma naif (naive algorithm).

### B. Karakteristik Algoritma *Brute-Force*

Algoritma *brute-force* merupakan algoritma yang tidak “cerdas” dan tidak mangkus dikarenakan algoritma ini membutuhkan jumlah komputasi yang besar dan waktu yang lama. Pada masalah kecil, algoritma ini lebih dipertimbangkan karena kesederhanaan dan implementasinya yang mudah. Algoritma *brute-force* sering digunakan sebagai basis pembandingan dengan algoritma yang lebih mangkus. Walaupun kurang mangkus, hampir semua persoalan dapat diselesaikan dengan algoritma *brute-force*.

### C. Keunggulan dan Kekurangan Algoritma *Brute-Force*

Ada beberapa keunggulan dan kekurangan pada algoritma *Brute-Force*. Tidak seperti algoritma lain, algoritma ini dapat diaplikasikan pada hampir semua masalah sehingga keunggulan *brute-force* yakni penerapannya yang sangat luas. Algoritma ini juga mudah dipahami dan cenderung sederhana. Algoritma ini layak untuk masalah seperti pencarian, pengurutan pencocokan string, dan perkalian matriks. Tetapi di samping itu, algoritma *brute-force* jarang menghasilkan algoritma yang mangkus dan efektif. Selain itu, algoritma ini juga menghabiskan memori dan waktu eksekusi yang banyak. Metode pemecahan masalah lainnya juga tidak kreatif.

### BAB 3 PENERAPAN BRUTE FORCE

#### 1. Algoritma *Brute-Force* pada permasalahan 20 solver

Algoritma *Brute-Force* untuk masalah ini melibatkan pencarian seluruh kombinasi operasi aritmatika yang memungkinkan menggunakan empat angka yang diberikan, dengan tujuan menghasilkan angka 20. Langkah-langkah implementasi nya sebagai berikut:

##### a. Inisialisasi Angka

Input empat angka yang akan dioperasikan, misalnya a, b, c, dan d

##### b. Penggunaan Operator Aritmatika

- Operator dasar aritmatika yang digunakan adalah penjumlahan (+), pengurangan (-), perkalian (\*), dan pembagian (/)
- Setiap dua angka akan dikombinasikan berbagai operasi

##### c. Mencari Kombinasi Operasi

- Ketika dua angka pertama dioperasikan dengan satu operator, hasilnya akan disimpan, kemudian operasikan hasil tersebut dengan angka berikutnya hingga seluruh angka habis
  - a op1 b op2 c op3 d
- Mencoba berbagai urutan operasi dengan letak tanda kurung
  - ((a op1 b) op2 c) op3 d
  - (a op1 (b op2 c)) op3 d
  - a op1 ((b op2 c) op3 d)
  - a op1 (b op2 (c op3 d))

##### d. Pengulangan Kombinasi

- Mencoba berbagai urutan angka untuk menemukan kombinasi yang cocok

a, b, c, d  
a, b, d, c  
a, c, b, d  
a, c, d, b  
a, d, b, c  
a, d, c, b  
b, a, c, d  
b, a, d, c  
b, c, a, d

dst

- Setiap kombinasi angka dan operasi dicoba satu per satu dalam pendekatan *brute-force*

+++

- + -

+ - \*

/ ++

e. Memeriksa Solusi

- Setelah setiap operasi aritmatika selesai, hasil dari ekspresi diperiksa apakah mendekati angka 20 (dalam range 19.99 hingga 20.01 untuk menghindari kesalahan pembulatan).
- Jika hasil mendekati 20, solusi akan disimpan dan ditampilkan

f. *Worst-case Scenario*

Dalam *worst case*, seluruh kombinasi dari 4 angka dan 4 operator harus dicoba. Dengan melihat semua permutasi dari 4 angka dan berbagai kombinasi operator (dengan tanda kurung), jumlah total operasi yang perlu dihitung bisa mencapai ribuan, namun ini tetap terhitung dalam cakupan brute force.

2. Implementasi Solusi pada Pemrograman

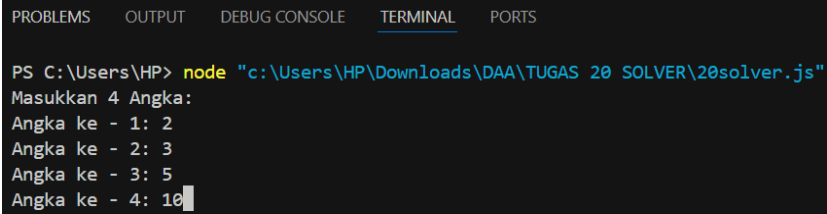
Pada pemrograman ini terdapat beberapa implementasi yaitu function yang memiliki tujuan memeriksa suatu nilai, melakukan operasi aritmatika, mencoba semua kemungkinan kombinasi, dan mengelola interaksi dengan pengguna serta memulai proses perhitungan.

Alur Umum pada program ini yaitu :

1. Pengguna memasukkan empat angka secara satu per satu.
2. Program akan mencoba semua kombinasi operasi aritmatika di antara angka - angka yang sudah diinput dengan menggunakan rekursi.
3. Jika terdapat kombinasi operasi yang menghasilkan 20, maka program akan menampilkan solusi.
4. Jika tidak ditemukan kombinasi operasi yang menghasilkan 20, maka program akan memberi tahu bahwa tidak ada solusi dengan angka - angka yang ada.

### 3. Testing dan Hasil Tangkapan Layar Input dan Output

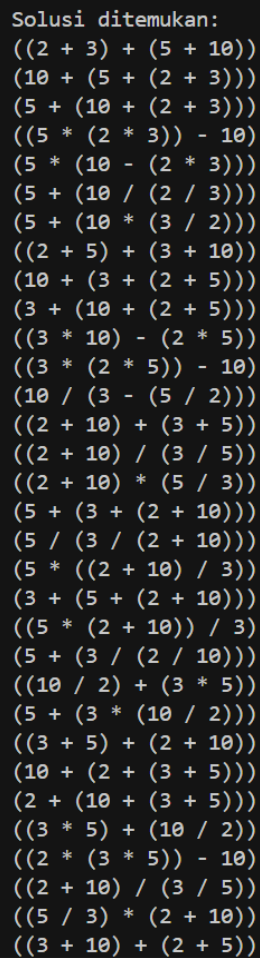
#### a. Input pada terminal



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\HP> node "c:\Users\HP\Downloads\DAA\TUGAS 20 SOLVER\20solver.js"
Masukkan 4 Angka:
Angka ke - 1: 2
Angka ke - 2: 3
Angka ke - 3: 5
Angka ke - 4: 10
```

#### b. Output pada terminal



```
Solusi ditemukan:
((2 + 3) + (5 + 10))
(10 + (5 + (2 + 3)))
(5 + (10 + (2 + 3)))
((5 * (2 * 3)) - 10)
(5 * (10 - (2 * 3)))
(5 + (10 / (2 / 3)))
(5 + (10 * (3 / 2)))
((2 + 5) + (3 + 10))
(10 + (3 + (2 + 5)))
(3 + (10 + (2 + 5)))
((3 * 10) - (2 * 5))
((3 * (2 * 5)) - 10)
(10 / (3 - (5 / 2)))
((2 + 10) + (3 + 5))
((2 + 10) / (3 / 5))
((2 + 10) * (5 / 3))
(5 + (3 + (2 + 10)))
(5 / (3 / (2 + 10)))
(5 * ((2 + 10) / 3))
(3 + (5 + (2 + 10)))
((5 * (2 + 10)) / 3)
(5 + (3 / (2 / 10)))
((10 / 2) + (3 * 5))
(5 + (3 * (10 / 2)))
((3 + 5) + (2 + 10))
(10 + (2 + (3 + 5)))
(2 + (10 + (3 + 5)))
((3 * 5) + (10 / 2))
((2 * (3 * 5)) - 10)
((2 + 10) / (3 / 5))
((5 / 3) * (2 + 10))
((3 + 10) + (2 + 5))
```

### c. Input pada web

The screenshot shows a web browser window with the URL <https://tugas-daa.vercel.app>. The page title is "20 Game Solver". Below the title, there is a section labeled "Masukkan Angka" (Enter Numbers). It contains four input fields labeled "Angka ke-1", "Angka ke-2", "Angka ke-3", and "Angka ke-4". The values entered in these fields are 2, 3, 5, and 10, respectively. Below the input fields is a button labeled "Cari Solusi" (Find Solution).

### d. Output pada web

The screenshot shows the same web browser window, but now the "Cari Solusi" button has been clicked. The output is displayed in a yellow box. It starts with the label "Hasil :" followed by a list of 20 mathematical expressions, each representing a possible solution to the 20 Game puzzle. The expressions are:

- $((2 + 3) + (5 + 10))$
- $(10 + (5 + (2 + 3)))$
- $(5 + (10 + (2 + 3)))$
- $((5 * (2 * 3)) - 10)$
- $(5 * (10 - (2 * 3)))$
- $(5 + (10 / (2 / 3)))$
- $(5 + (10 * (3 / 2)))$
- $((2 + 5) + (3 + 10))$
- $(10 + (3 + (2 + 5)))$
- $(3 + (10 + (2 + 5)))$
- $((3 * 10) - (2 * 5))$
- $((3 * (2 * 5)) - 10)$
- $(10 / (3 - (5 / 2)))$
- $((2 + 10) + (3 + 5))$
- $((2 + 10) / (3 / 5))$
- $((2 + 10) * (5 / 3))$
- $(5 + (3 + (2 + 10)))$
- $(5 / (3 / (2 + 10)))$
- $(5 * ((2 + 10) / 3))$
- $(3 + (5 + (2 + 10)))$
- $((5 * (2 + 10)) / 3)$
- $(5 + (3 / (2 / 10)))$
- $((10 / 2) + (3 * 5))$
- $(5 + (3 * (10 / 2)))$
- $((3 + 5) + (2 + 10))$
- $(10 + (2 + (3 + 5)))$
- $(2 + (10 + (3 + 5)))$

## BAB 4 PROGRAM DALAM BAHASA JAVA SCRIPT

Berikut ini adalah kode program yang dapat menyelesaikan permasalahan 20 solver dengan metode Brute Force dan ditulis dalam bahasa pemrograman Java Script.

```
1  const readline = require('readline');
2  ....
3  function duaPuluh(nilai) {
4  |   return nilai > 19.99 && nilai < 20.01;
5  }
6
7  function hitung(a, b, op) {
8  |   switch(op) {
9  |       case '+': return a + b;
10 |      case '-': return a - b;
11 |      case '*': return a * b;
12 |      case '/':
13 |          if (b !== 0) return a / b;
14 |          return 0;
15 |      default: return 0;
16 |   }
17 }
18
19 function cobaKombinasi(angka, ekspresi) {
20 |   if (angka.length === 1) {
21 |       if (duaPuluh(angka[0])) {
22 |           return ekspresi[0] + "\n";
23 |       }
24 |       return "";
25 |   }
```

```
26
27     let hasil = "";
28     for (let i = 0; i < angka.length; ++i) {
29         for (let j = i + 1; j < angka.length; ++j) {
30             let angkaBaru = [];
31             let ekspresiBaru = [];
32             for (let k = 0; k < angka.length; ++k) {
33                 if (k !== i && k !== j) {
34                     angkaBaru.push(angka[k]);
35                     ekspresiBaru.push(ekspresi[k]);
36                 }
37             }
38
39             const operasi = ['+', '-', '*', '/'];
40             for (let op of operasi) {
41                 if (op === '/' && angka[j] === 0) continue; // Avoid division by zero
42                 let hasilHitung = hitung(angka[i], angka[j], op);
43                 let ekspresiHitung = `(${ekspresi[i]} ${op} ${ekspresi[j]})`;
44                 angkaBaru.push(hasilHitung);
45                 ekspresiBaru.push(ekspresiHitung);
46                 hasil += cobaKombinasi(angkaBaru, ekspresiBaru);
47                 angkaBaru.pop();
48                 ekspresiBaru.pop();
49             }
```



```

49
50         if ((op === '-' || op === '/') && angka[i] !== 0) { // Avoid division by zero in reverse
51             hasilHitung = hitung(angka[j], angka[i], op);
52             ekspresiHitung = `(${ekspresi[j]} ${op} ${ekspresi[i]})`;
53             angkaBaru.push(hasilHitung);
54             ekspresiBaru.push(ekspresiHitung);
55             hasil += cobaKombinasi(angkaBaru, ekspresiBaru);
56             angkaBaru.pop();
57             ekspresiBaru.pop();
58         }
59     }
60 }
61 }
62 return hasil;
63 }
64
65 // Fungsi utama
66 function main() {
67     const rl = readline.createInterface({
68         input: process.stdin,
69         output: process.stdout
70     });
71
72     let angka = [];
73     let ekspresi = [];
74     let inputCount = 0;
75

```

```

76     function getInput() {
77         if (inputCount < 4) {
78             rl.question(`Angka ke - ${inputCount + 1}: `, (input) => {
79                 let num = parseFloat(input);
80                 angka.push(num);
81                 ekspresi.push(num.toString());
82                 inputCount++;
83                 getInput();
84             });
85         } else {
86             let hasil = cobaKombinasi(angka, ekspresi);
87             if (!hasil) {
88                 console.log("Tidak ada solusi ditemukan untuk 20.");
89             } else {
90                 console.log("Solusi ditemukan:\n" + hasil);
91             }
92             rl.close();
93         }
94     }
95
96     console.log("Masukkan 4 Angka:");
97     getInput();
98 }
99
100 main();

```

## **BAB 5 TABEL PENILAIAN**

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil running.	✓	
Program dapat membaca input / generate sendiri dan memberikan luaran	✓	
Solusi yang diberikan program memenuhi (berhasil mencapai 20)	✓	

## BAB 6 KESIMPULAN

Dalam melakukan penyelesaian permasalahan 20 solver menggunakan algoritma *Brute-Force*, yaitu: dengan mencoba semua kemungkinan kombinasi operasi aritmatika yang terdiri dari empat angka (dengan nilai bervariasi). Pada Laporan Desain Analisis Algoritma ini, penulis membuat program untuk menyelesaikan permasalahan 20 solver yang ditulis dalam bahasa pemrograman Java Script.

Algoritma brute force pada solver 20 sangat efektif untuk mengeksplorasi seluruh kemungkinan kombinasi angka dan operator. Meskipun kompleksitasnya tinggi, ia menawarkan solusi yang pasti, dan merupakan pendekatan sederhana namun lengkap untuk masalah ini. Karena algoritma ini mencoba semua kemungkinan kombinasi, kompleksitas waktu bisa sangat besar, terutama ketika jumlah angka dan operator bertambah. Namun, dengan brute force, kita dapat menjamin bahwa solusi ditemukan jika ada, meskipun waktu yang diperlukan bisa lebih lama dibandingkan dengan pendekatan yang lebih efisien seperti backtracking atau dynamic programming.

## BAB 7 REFERENSI

[Algoritma-Brute-Force-\(2022\)-Bag1.pdf \(itb.ac.id\)](#)

Sinaga, A., & Nuraisana. (2021). Implementasi algoritma brute force dalam pencarian menu

pada aplikasi pemesanan Coffee (Studi kasus : Tanamera Coffee). *Jurnal Ilmu Komputer*

*Dan Sistem Informasi (JIKOMSI)*, 4(1), 6–15. <https://doi.org/10.9767/jikoms.v4i1.129>

[https://github.com/kimpoter/Tucil1\\_13521150/tree/main](https://github.com/kimpoter/Tucil1_13521150/tree/main)