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. **Lidya Khairunnisa (L0123075)**
2. **Lu'lu'a Lim'a Laila (L0123076)**

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

SURAKARTA

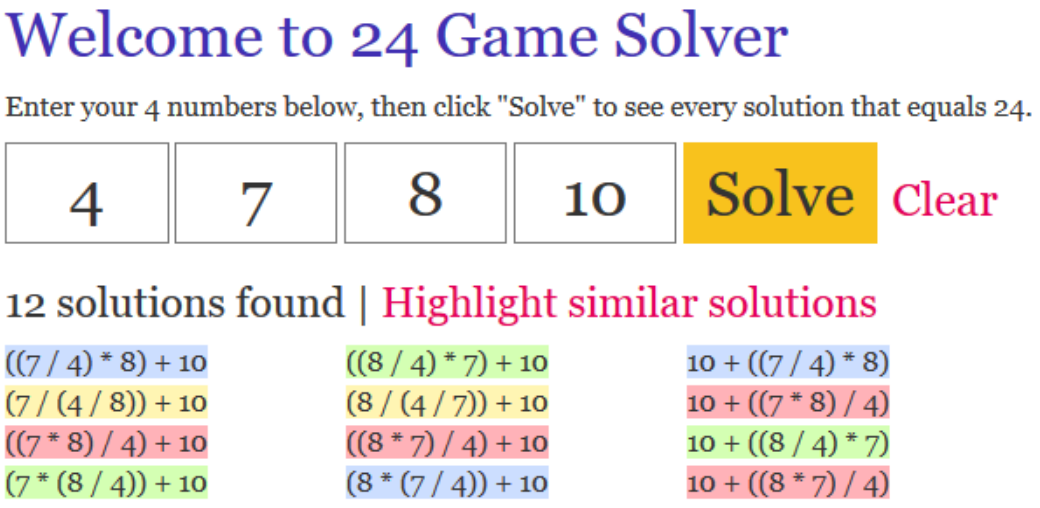
2024

BAB I
DESKRIPSI MASALAH

20 Game Solver adalah permainan aritmatika yang tujuannya mencari 4 integer non negatif dengan operator +, -, *, /, dan ( ) sehingga hasil akhirnya bernilai 20. 20 Game Solver adalah versi modifikasi dari permainan 24 Game Solver, di mana tujuan utamanya adalah mencari kombinasi operasi matematika yang menghasilkan angka 20, bukan 24, dengan menggunakan empat angka yang diberikan. Permainan ini merupakan permainan yang cukup populer di kalangan masyarakat karena permainan ini dapat membantu dalam mengasah logika matematika, meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Permainan ini juga melatih aritmatika mental kita, sehingga bermanfaat untuk semua orang di semua kategori usia.

Pada awal game, pemain akan diminta untuk memasukan 4 angka secara random dan kemudian berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 20. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (×), divisi (/) dan tanda kurung (()).

Tugas ini bertujuan untuk merancang algoritma brute force untuk menemukan solusi dari permainan ini.



Gambar 1.1 Permainan 20 Game Solver

BAB II
TEORI SINGKAT

Teori yang digunakan pada penerapan strategi algoritma dalam 20 Game Solver adalah algoritma *Brute Force* .

A. Definisi Algoritma *Brute Force*

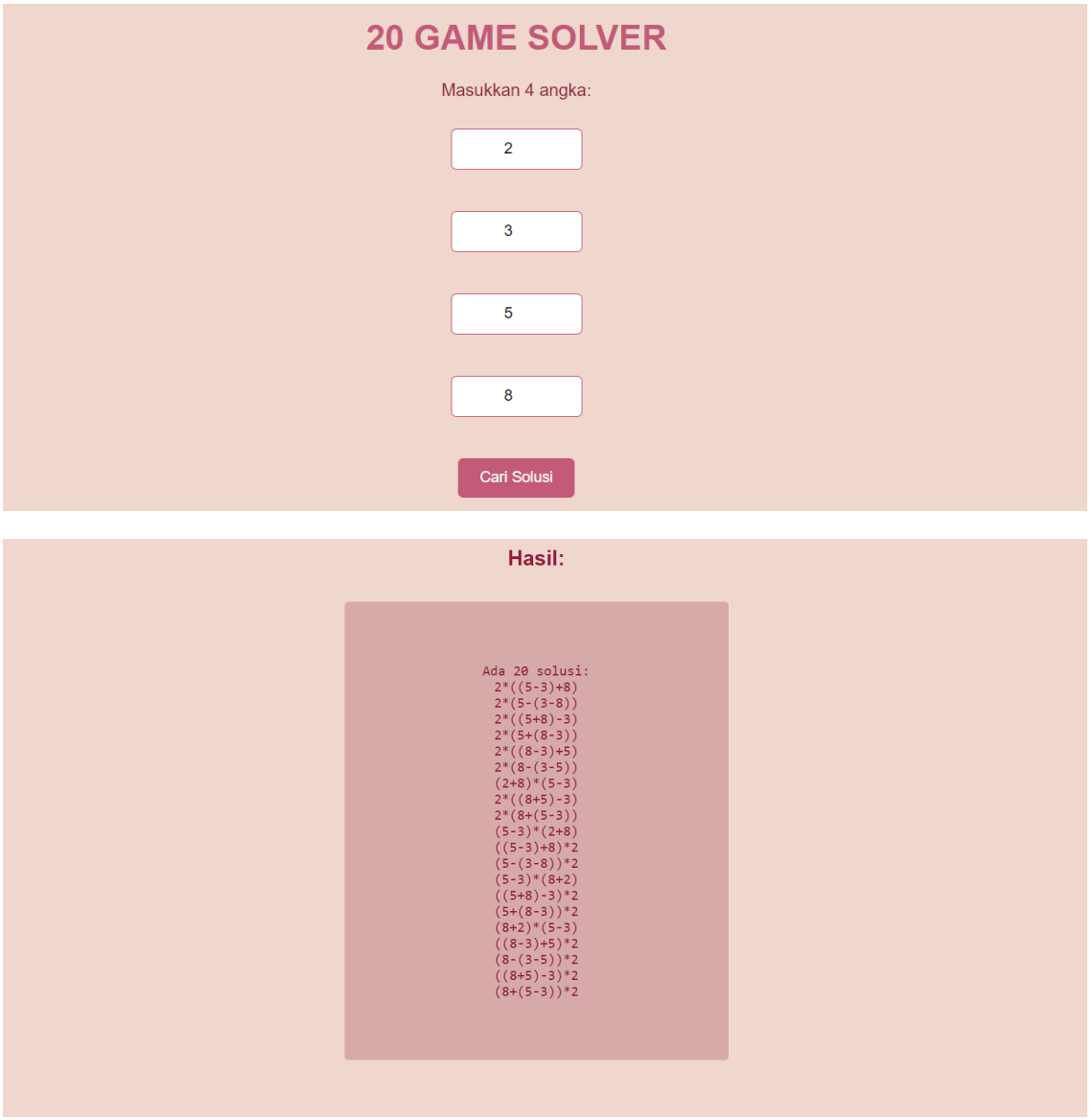
Brute Force merupakan algoritma yang dapat memecahkan permasalahan dengan sederhana (simple), langsung, dan jelas. Algoritma *Brute Force* melakukan pendekatan yang lempang *(straightforward)* dalam memecahkan suatu masalah. Oleh sebab itu, algoritma *Brute Force* sering sekali disebut dengan algoritma naïf *(Naïve algorithm)*. Algoritma *Brute Force* dapat digunakan untuk memecahkan hampir sebagian besar permasalahan yang ada karena memasukan dan mencoba semua isi dalam suatu data koleksi tanpa menggunakan cara-cara khusus. Teratur, sederhana dan mudah dimengerti adalah salah satu kelebihan dari algoritma *Brute Force*

B. Penerapan Algoritma *Brute Force*

Brute Force mencoba semua operator dari yang pertama sampai yang terakhir, apabila nilai sudah mencapai 20, maka *looping* akan memberikan *return* beserta operatornya. Langkah langkah yang dapat dilakukan dalam memecahkan 20 Game Solver dengan metode *Brute Force* adalah sebagai berikut:

- Misalkan terdapat empat buah angka yang akan mewakili entitas yang didapat (ex : 1,3,5,7). Dalam melakukan operasi aritmatika, akan digunakan 3 dari total 4 operator (+, -, *, /) yang nantinya akan digunakan untuk menghubungkan keempat variabel dalam sebuah operasi matematika tertentu.
- Pertama, program akan mencari seluruh permutasi yang mungkin dari nilai empat buah angka. Kombinasi empat kartu ini akan disimpan dalam sebuah vektor. Sedangkan, seluruh kombinasi disimpan dalam struktur data set. Penggunaan struktur data set dimaksudkan menghindari duplikat nilai.
- Selanjutnya mencari seluruh kombinasi tiga operator yang mungkin dari empat operator yang boleh digunakan. Dalam pencarian kombinasi operator, satu operator dapat digunakan berulang kali. Seluruh kombinasi ini disimpan dalam struktur data vektor.

- Untuk setiap kombinasi empat kartu dan tiga operator, lakukan pengecekan untuk lima buah ekspresi yang berbeda peletakkan tanda kurung. Langkah ini melakukan pengecekkan pada semua kombinasi yang mungkin.
- Jika didapatkan ekspresi yang menghasilkan 20, maka ekspresi tersebut merupakan salah satu solusi.



Gambar 2.1 dan 2.2 Contoh Permainan 20 Solver

BAB III
KODE DAN PENGUJIAN

A. Kode Program

1. Kode Program dalam HTML

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>20 GAME SOLVER</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4d9d0;
            color: #921a40;
            padding: 50px;
            text-align: center;
        }
        h1 {
            color: #c75b7a;
        }
```

Gambar 3.1 Elemen yang digunakan untuk menampilkan judul utama halaman

```
        input[type="number"] {
            padding: 10px;
            margin: 10px 0;
            width: 100px;
            font-size: 14px;
            border: 1px solid #c75b7a;
            border-radius: 5px;
            text-align: center;
        }
```

Gambar 3.2 Elemen yang akan digunakan oleh pengguna untuk memasukkan angka

```
        button {
            background-color: #c75b7a;
            color: white;
            padding: 10px 20px;
            margin-top: 10px;
            border: none;
            border-radius: 5px;
            font-size: 14px;
            cursor: pointer;
            transition: background-color 0.3s;
        }
        button:hover {
            background-color: #921a40;
        }
```

Gambar 3.3 Tombol untuk memulai pencarian solusi

```
41              #hasilOutput {
42                  margin-top: px;
43                  padding: 70px;
44                  background-color: #d9abab;
45                  border: 1px solid #white;
46                  border-radius: 5px;
47                  font-size: 16px;
48                  text-align: center;
49                  display: inline-block;
50                  min-width: 300px;
51              }
```

Gambar 3.4 Elemen HTML tempat hasil akan ditampilkan

```
52      </style>
53  </head>
54  <body>
55      <h1>20 GAME SOLVER</h1>
56      <p>Masukkan 4 angka:</p>
57      <input type="number" id="angka1" placeholder="Angka 1" /><br /><br />
58      <input type="number" id="angka2" placeholder="Angka 2" /><br /><br />
59      <input type="number" id="angka3" placeholder="Angka 3" /><br /><br />
60      <input type="number" id="angka4" placeholder="Angka 4" /><br /><br />
61      <button onclick="main()">Cari Solusi</button>
62
63      <h2>Hasil:</h2>
64      <pre id="hasilOutput"></pre>
```

Gambar 3.5 Elemen tempat input angka, tombol cari solusi dan mengeluarkan output hasil

```
66      <script>
67          // Menghasilkan semua permutasi dari sebuah array
68          function permutasi(arr) {
69              if (arr.length <= 1) return [arr];
70              let result = [];
71              for (let i = 0; i < arr.length; i++) {
72                  let currentNum = arr[i];
73                  let remainingNums = arr.slice(0, i).concat(arr.slice(i + 1));
74                  let remainingNumsPermuted = permutasi(remainingNums);
75                  for (let p of remainingNumsPermuted) {
76                      result.push([currentNum].concat(p));
77                  }
78              }
79              return result;
80          }
81
```

Gambar 3.6 Menghasilkan semua permutasi yang mungkin dari array input menggunakan rekursi

```
// Menghasilkan semua kombinasi operator
function kombinasiOperator(ops, k) {
    if (k === 1) return ops.map(op => [op]);
    let result = [];
    for (let i = 0; i < ops.length; i++) {
        let currentOp = ops[i];
        let remainingCombos = kombinasiOperator(ops, k - 1);
        for (let combo of remainingCombos) {
            result.push([currentOp].concat(combo));
        }
    }
    return result;
}
```

Gambar 3.7 Menghasilkan semua kombinasi operator sebanyak k dari array operator menggunakan rekursi

```
// Mengevaluasi lima jenis ekspresi berbeda dengan tanda kurung
function evaluasiEkspresi(a, b, c, d, ops) {
    let hasil = [];

    // ((a op1 b) op2 c) op3 d
    hasil.push(`((${a}${ops[0]}${b})${ops[1]}${c})${ops[2]}${d}`);
    // (a op1 (b op2 c)) op3 d
    hasil.push(`(${a}${ops[0]}(${b}${ops[1]}${c}))${ops[2]}${d}`);
    // (a op1 b) op2 (c op3 d)
    hasil.push(`(${a}${ops[0]}${b})${ops[1]}(${c}${ops[2]}${d})`);
    // a op1 ((b op2 c) op3 d)
    hasil.push(`${a}${ops[0]}((${b}${ops[1]}${c})${ops[2]}${d})`);
    // a op1 (b op2 (c op3 d))
    hasil.push(`${a}${ops[0]}(${b}${ops[1]}(${c}${ops[2]}${d}))`);

    return hasil;
}
```

Gambar 3.8 Menghasilkan semua kombinasi operator tanda kurung.

```
function solver(angka, hasil) {
    const operators = ['+', '-', '*', '/'];
    const permutations = permutasi(angka);
    const operatorCombinations = kombinasiOperator(operators, 3);

    for (let perm of permutations) {
        for (let ops of operatorCombinations) {
            let [a, b, c, d] = perm;
            let expressions = evaluasiEkspresi(a, b, c, d, ops);

            for (let expr of expressions) {
                try {
                    if (Math.abs(eval(expr) - 20) < 0.01) {
                        hasil.add(expr);
                    }
                } catch (e) {
                    // Skip if the expression is invalid (e.g., division by zero)
                }
            }
        }
    }
}
```

Gambar 3.9 Menguji semua permutasi angka dan kombinasi operator untuk menghasilkan ekspresi yang hasilnya mendekati 20

```
137             function main() {
138                 const angka1 = document.getElementById("angka1").value;
139                 const angka2 = document.getElementById("angka2").value;
140                 const angka3 = document.getElementById("angka3").value;
141                 const angka4 = document.getElementById("angka4").value;
142 
143                 const masukan = [angka1, angka2, angka3, angka4].map(Number);
144 
145                 const hasil = new Set();
146                 solver(masukan, hasil);
147 
148                 const hasilOutput = document.getElementById("hasilOutput");
149                 if (hasil.size === 0) {
150                     hasilOutput.textContent = "Tidak ada solusi";
151                 } else {
152                     hasilOutput.textContent =
153                         `Ada ${hasil.size} solusi:\n` + Array.from(hasil).join("\n");
154                 }
155             }
156         </script>
157     </body>
158     </html>
```

Gambar 3.10 Mengambil input angka dari pengguna, mencari solusi yang mungkin, dan menampilkannya di halaman jika ada yang sesuai.

2. Kode Program dalam JavaScript

```
1     const readline = require("readline");
2 
```

Gambar 3.11 Mengimpor modul readline yang merupakan modul bawaan di Node.js. Modul ini digunakan untuk membaca input dari pengguna melalui terminal (CLI).

```
3     // Menghasilkan semua permutasi dari sebuah array
4     function permutasi(arr) {
5       if (arr.length <= 1) return [arr];
6       let result = [];
7       for (let i = 0; i < arr.length; i++) {
8         let currentNum = arr[i];
9         let remainingNums = arr.slice(0, i).concat(arr.slice(i + 1));
10        let remainingNumsPermuted = permutasi(remainingNums);
11        for (let p of remainingNumsPermuted) {
12          result.push([currentNum].concat(p));
13        }
14      }
15      return result;
16    }
```

Gambar 3.12 Menghasilkan semua urutan (permutasi) yang mungkin dari elemen-elemen dalam array arr

```
17
18    // Menghasilkan semua kombinasi operator
19    function kombinasiOperator(ops, k) {
20      if (k === 1) return ops.map((op) => [op]);
21      let result = [];
22      for (let i = 0; i < ops.length; i++) {
23        let currentOp = ops[i];
24        let remainingCombos = kombinasiOperator(ops, k - 1);
25        for (let combo of remainingCombos) {
26          result.push([currentOp].concat(combo));
27        }
28      }
29      return result;
30    }
```

Gambar 3.13 Menghasilkan semua kombinasi operator dengan pengulangan sebanyak k dari array ops

```
31
32    // Mengevaluasi lima jenis ekspresi berbeda dengan tanda kurung
33    function evaluasiEkspresi(a, b, c, d, ops) {
34      let hasil = [];
35
36      // ((a op1 b) op2 c) op3 d
37      hasil.push(`((${a}${ops[0]}${b})${ops[1]}${c})${ops[2]}${d}`);
38      // (a op1 (b op2 c)) op3 d
39      hasil.push(`(${a}${ops[0]}(${b}${ops[1]}${c}))${ops[2]}${d}`);
40      // (a op1 b) op2 (c op3 d)
41      hasil.push(`(${a}${ops[0]}${b})${ops[1]}(${c}${ops[2]}${d})`);
42      // a op1 ((b op2 c) op3 d)
43      hasil.push(`${a}${ops[0]}((${b}${ops[1]}${c})${ops[2]}${d})`);
44      // a op1 (b op2 (c op3 d))
45      hasil.push(`${a}${ops[0]}(${b}${ops[1]}(${c}${ops[2]}${d}))`);
46
47      return hasil;
48    }
49
```

Gambar 3.14 Menghasilkan lima bentuk ekspresi aritmatika berbeda dengan tanda kurung menggunakan empat angka dan tiga operator.

```
50    function solver(angka, hasil) {
51      const operators = ["+", "-", "*", "/"];
52      const permutations = permutasi(angka);
53      const operatorCombinations = kombinasiOperator(operators, 3);
54
55      for (let perm of permutations) {
56        for (let ops of operatorCombinations) {
57          let [a, b, c, d] = perm;
58          let expressions = evaluasiEkspresi(a, b, c, d, ops);
59
60          for (let expr of expressions) {
61            try {
62              if (Math.abs(eval(expr) - 20) < 0.01) {
63                hasil.add(expr);
64              }
65            } catch (e) {
66              // Lewati jika ada ekspresi yang tidak valid (misalnya pembagian dengan nol)
67            }
68          }
69        }
70      }
71    }
72
```

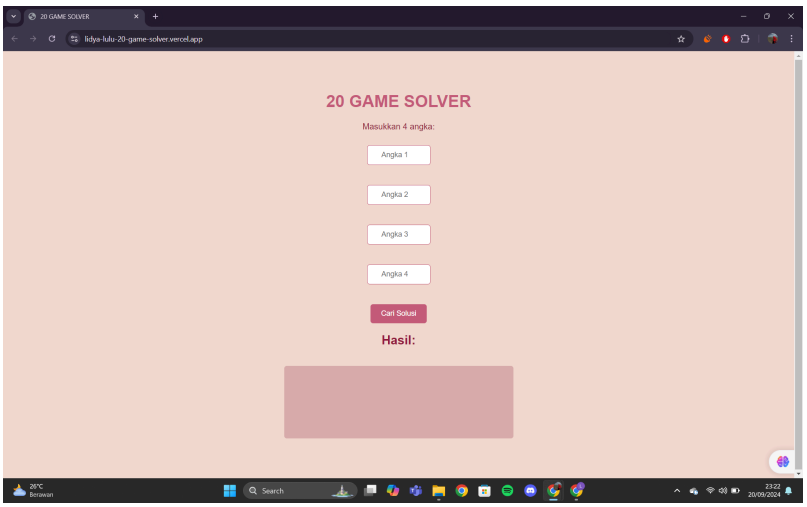
Gambar 3.15 Mencari semua ekspresi aritmatika dari permutasi angka dan kombinasi operator yang hasilnya mendekati 20.

```
72
73  function main() {
74    const rl = readline.createInterface({
75      input: process.stdin,
76      output: process.stdout,
77    });
78
79    rl.question(
80      "Masukkan 4 angka dipisah dengan spasi (misal: 1 3 4 6): ",
81      (input) => {
82        const masukan = input.split(" ").map(Number);
83
84        if (masukan.length !== 4 || masukan.some(isNaN)) {
85          console.log("Input tidak valid. Harap masukkan 4 angka.");
86        } else {
87          const hasil = new Set();
88          solver(masukan, hasil);
89
90          if (hasil.size === 0) {
91            console.log("Tidak ada solusi");
92          } else {
93            console.log(
94              `Ada ${hasil.size} solusi:\n` + Array.from(hasil).join("\n")
95            );
96          }
97        }
98
99        rl.close();
100     }
101   );
102 }
103
104 main();
```

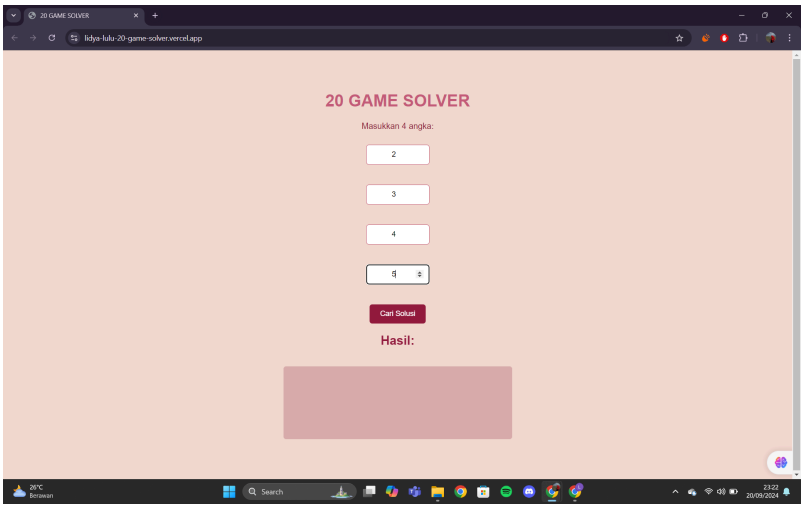
Gambar 3.16 Mengambil input 4 angka dari pengguna, memeriksa validitasnya, lalu mencari dan menampilkan ekspresi yang hasilnya mendekati 20.

B. Testing dan Hasil Tangkap Layar Input Output

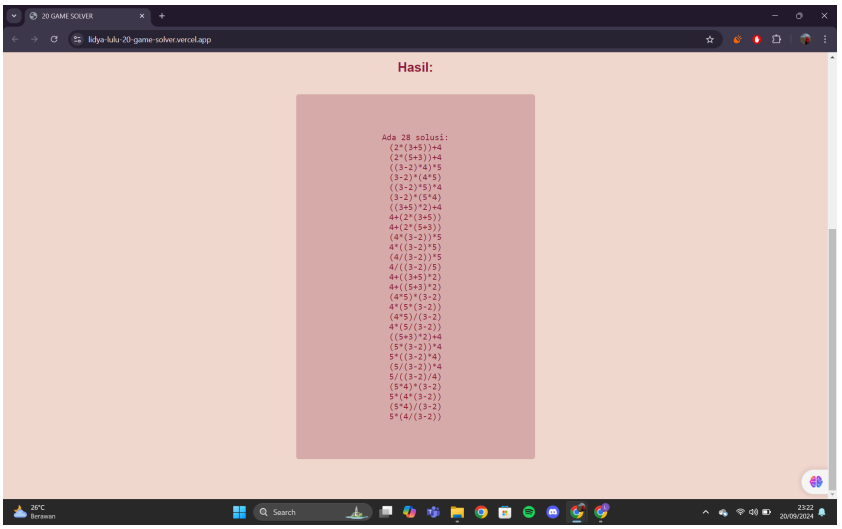
1. Input Output HTML



Gambar 3.17 Tampilan awal saat pertama kali membuka website

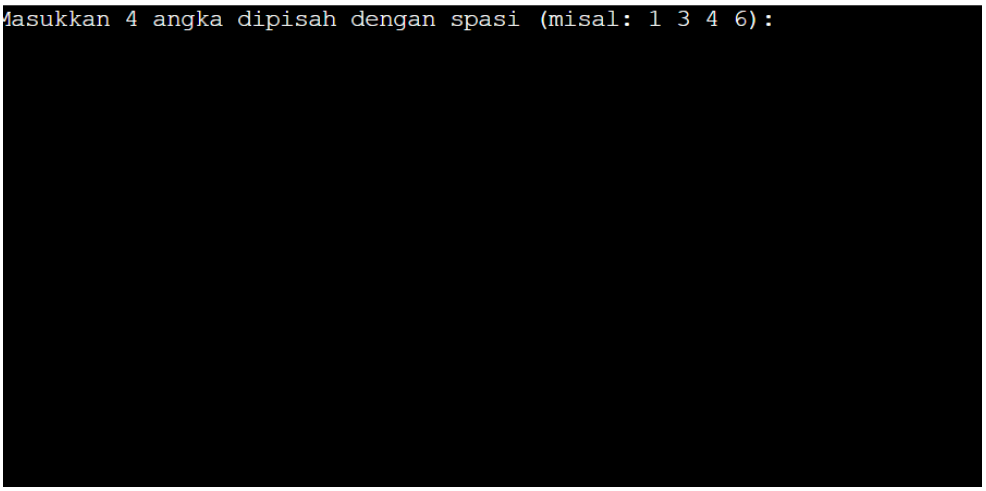


Gambar 3.18 Memasukkan angka ke bagian input di website

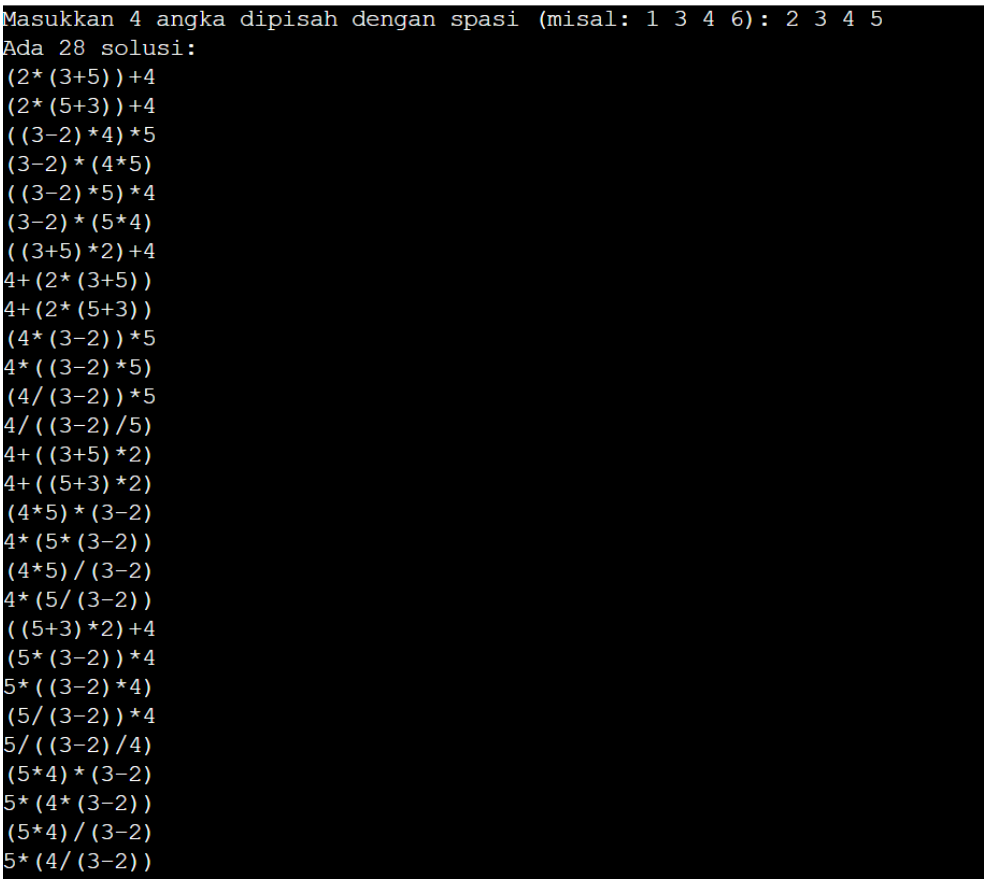


Gambar 3.19 Hasil output pada website

2. Input Output dalam JavaScript



Gambar 3.20 Output yang pertama kali keluar saat program JavaScript dijalankan



Gambar 3.20 Output keluar setelah memasukkan angka yang diinginkan

BAB 4
KESIMPULAN

Dalam penggunaannya, algoritma *Brute Force* mampu menyelesaikan permasalahan dan menemukan solusi yang tepat pada 20 Game Solver. Algoritma *brute force* dapat mencoba semua kemungkinan operasi aritmatika yang mungkin dibentuk oleh 4 angka dengan nilai yang bervariasi dan menghasilkan hasil yang sama, yakni 20.

Di dalam proyek Desain Analisis Algoritma ini, kami berhasil membuat sebuah web untuk menyelesaikan 20 Game Solver yang ditulis dalam HTML. *Website* kami berhasil melakukan penyelesaian 20 Game Solver dengan cukup baik.

REFERENSI

Ndaumanu, R. I. (2020). Studi Komparatif Algoritma Fisher Yates dengan Brute Force pada Permainan Kartu 24. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *6*(1), 95. https://doi.org/10.26418/jp.v6i1.35948

Munir, R. (n.d.). *IF2211 Strategi Algoritma - Semester II Tahun 2021/2022*. https://informatika.stei.itb.ac.id. Retrieved September 18, 2024, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm

KONTRIBUSI

Nama	Peran
Lidya Khairunnisa (L0123075)	● Berpartisipasi dalam diskusi kelompok ● Membuat kode JavaScript dan HTML ● Membuat dokumentasi ● Membuat video
Lu'lu'a Lim'a Laila (L0123076)	● Berpartisipasi dalam diskusi kelompok ● Membuat kode JavaScript dan HTML ● Mendeploy website ● Membuat video