

# **Laporan Dokumentasi 20 Solver Desain dan Analisis Algoritma**

Disusun untuk Memenuhi Tugas Mata Kuliah

Desain Analisis Algoritma

Dosen Pengampu:

**Fajar Muslim S.T., M.T.**



Disusun Oleh:

1. Muhammad Farhan Khairullah (L0123093)
2. Muhammad Rifai Ageng Pambudi (L0123098)
3. Muiz Afif Mirza Lindu Aji (L0123099)

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

**2024**

## Pengertian Masalah

**20 Solver** merupakan sebuah program yang berguna untuk menyelesaikan suatu problem. Dalam kasus ini problemnya adalah bagaimana cara kita menyusun 4 angka input dengan 5 operator (+, -, \*, /, dan ‘()’) matematika sehingga kita bisa menghasilkan solusi akhir yang bernilai 20.

Akan tetapi, karena yang menentukan 4 angka yang di-input adalah kehendak pengguna, ini akan menimbulkan 2 kemungkinan penyelesaian. Penyelesaian bisa muncul kombinasi yang bisa menghasilkan nilai akhir 20, atau bahkan 4 angka yang ada tidak memungkinkan untuk mendapat solusi penyelesaian.

Saat program menemukan satu atau lebih solusi dari keempat angka yang di-input, program akan memberikan seluruh output solusi yang mungkin, serta menghitung jumlah solusi yang ditemukan. Namun, jika angka yang digunakan tidak memungkinkan untuk menghasilkan solusi yang bernilai 20 sama sekali, maka program akan mencetak tulisan “Tidak ada solusi yang ditemukan.”.

## Demonstrasi Program

Cara menjalankan program 20 Solver:

1. Masuk ke link :  
<https://muizz-farhankai-pampam.on.driv.tw/Test%20Hosting/Solver%202020.html>
2. Masukkan 4 angka pada kotak-kotak yang ada. (Misal: 2, 3, 5, 6)
3. Klik “Cari Solusi” maka program akan berjalan dan segala kemungkinan solusi untuk menghasilkan angka 20 akan keluar, beserta jumlahnya.

## Dokumentasi Program

### Analisis Kode

HTML

```
1 <!DOCTYPE html>
2 <html lang="id">
```

- **<!DOCTYPE html>** deklarasi type dokumen. Menandakan bahwa file ini adalah HTML.
- **<html lang="id">** lang=id adalah menunjukan bahasa yang digunakan adalah bahasa indonesia.

```
<head>
  <meta charset="UTF-8">
  <title>20 Solver</title>
  <link rel="stylesheet" href="styles.css">
</head>
```

- **<meta charset="UTF-8">** Menentukan set karakter yang digunakan untuk halaman, di sini menggunakan UTF-8, yang mendukung hampir semua karakter yang digunakan dalam berbagai bahasa.
- **<title>20 Solver</title>** Memberikan judul halaman yang muncul di tab browser
- **<link rel="stylesheet" href="styles.css">** menghubungkan file eksternal yang bernama "styles.css" untuk mengatur tampilan desain halaman browser.

```
<body>
  <div class="background"></div>
  <div class="container">
    <h1>20 Solver</h1>
```

- **<div class="background"></div>** Digunakan untuk mengatur latar belakang di file CSS
- **<div class="container">** Elemen ini membungkus seluruh konten utama, yang umumnya digunakan untuk pengaturan layout halaman.
- **<h1>20 Solver</h1>** untuk menampilkan judul halaman "20 Solver".

```
<form id="solver-form">
  <input type="number" id="num1" placeholder="Angka 1" required>
  <input type="number" id="num2" placeholder="Angka 2" required>
  <input type="number" id="num3" placeholder="Angka 3" required>
  <input type="number" id="num4" placeholder="Angka 4" required>
  <button type="submit">Cari Solusi</button>
</form>
```

- **<form id="solver-form">** Formulir yang akan menerima input dari user dan di proses di JavaScript "app.js"
- **<input type="number" id="num1" placeholder="Angka 1" required>** Input angka pertama, dengan placeholder "Angka 1". Atribut "required" menandakan bahwa input ini wajib diisi.
- **<input type="number" id="num2" placeholder="Angka 2" required>** Input angka kedua, dengan placeholder "Angka 2". Atribut "required" menandakan bahwa input ini wajib diisi.
- **<input type="number" id="num3" placeholder="Angka 3" required>** Input angka ketiga, dengan placeholder "Angka 3". Atribut "required" menandakan bahwa input ini wajib diisi.
- **<input type="number" id="num4" placeholder="Angka 4" required>** Input angka keempat, dengan placeholder "Angka 4". Atribut "required" menandakan bahwa input ini wajib diisi.

```

    <div id="results"></div>
  </div>
  <script src="app.js"></script>
</body>
</html>

```

- **<div id="results"></div>** Div ini berfungsi untuk menampilkan hasil dari pencarian solusi yang dihasilkan oleh "JavaScript."
- **<script src="app.js"></script>** Memuat file "app.js", yang akan mengatur logika fungsionalitas aplikasi, dan akan ditampilkan di div "results".

```
body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  padding-top: 50px;
  background-color: #6A9C89;
}
```

“body” Mengatur tampilan halaman utama, dengan latar belakang hijau dan konten di tengah.

```
.container {
  background-color: #fff;
  padding: 20px 30px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

“.container” Membuat kotak yang bersih dengan sudut melengkung dan bayangan di sekitar Konten.

```
h1 {
  text-align: center;
}

form {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

input {
  padding: 10px;
  font-size: 16px;
}
```

- “h1.” Memusatkan teks judul "20 SOLVER"
- “form” Mengatur tata letak input secara vertikal dengan jarak antar elemen menggunakan
- “input” Mengatur ukuran dan padding input angka agar lebih nyaman digunakan.

```
button {
  padding: 10px;
  font-size: 16px;
  background-color: #16423C;
  color: #fff;
  border: none;
  cursor: pointer;
}
```

- “button” Tombol untuk mencari solusi ditata dengan warna hijau tua, dan akan berubah warna ketika di-hover.

```
#results {
  margin-top: 20px;
}
```

- “results” Mengatur jarak div hasil dari elemen di atasnya.

## JavaScript

```
document.getElementById('solver-form').addEventListener('submit', function(e) {
  e.preventDefault();
});
```

- Event listener ini mendengarkan event `submit` pada elemen form dengan ID `solver-form`.
- Fungsi “`e.preventDefault()`” digunakan untuk mencegah perilaku default dari form yang biasanya akan me-refresh halaman. Ini memungkinkan pengolahan data form dengan JavaScript tanpa reload halaman.

```
const num1 = parseInt(document.getElementById('num1').value);
const num2 = parseInt(document.getElementById('num2').value);
const num3 = parseInt(document.getElementById('num3').value);
const num4 = parseInt(document.getElementById('num4').value);

const nums = [num1, num2, num3, num4];
```

- Pada blok ini, `parseInt()` digunakan untuk mengubah input dari string (bentuk standar dari input HTML) menjadi angka integer. Input tersebut disimpan dalam variabel `num1`, `num2`, `num3`, dan `num4`, yang kemudian diubah menjadi array `nums`.

```
const target = 20;
const operators = ['+', '-', '*', '/'];
const solutions = [];
```

- “target = 20” mendefinisikan nilai target yang akan dicari oleh solver, yaitu 20.
- “operators” adalah array yang berisi operator aritmatika dasar yang digunakan dalam kombinasi ekspresi.
- “solutions” adalah array kosong untuk menyimpan solusi yang ditemukan.

```
function permute(arr) {
  const results = [];

  function backtrack(current, remaining) {
    if (remaining.length === 0) {
      results.push(current);
      return;
    }
    for (let i = 0; i < remaining.length; i++) {
      backtrack(
        current.concat(remaining[i]),
        remaining.slice(0, i).concat(remaining.slice(i + 1))
      );
    }
  }

  backtrack([], arr);
  return results;
}
```

**Tujuan:** Menghasilkan semua permutasi dari array nums yang berisi angka-angka input. results menyimpan semua permutasi yang dihasilkan.

**Fungsi Rekursif backtrack:**

- “current” menyimpan kombinasi sementara selama proses rekursif.
- “remaining” berisi elemen-elemen yang belum dipilih dalam permutasi saat ini.
- Jika “remaining.length” bernilai 0, artinya permutasi lengkap telah dibuat, dan ditambahkan ke dalam results.

- Fungsi ini secara rekursif mencoba semua kombinasi dengan menambahkan satu elemen dari remaining ke dalam current, kemudian mengulangi proses dengan menghapus elemen tersebut dari remaining.

Setelah semua permutasi dihasilkan, array results dikembalikan.

```
function evaluateExpression(nums, ops, arrangement) {
  let a = nums[0], b = nums[1], c = nums[2], d = nums[3];
  let result;
  let expr = '';

  try {
    switch(arrangement) {
      case 1: // ((a op1 b) op2 c) op3 d
        result = compute(compute(compute(a, ops[0], b), ops[1], c), ops[2], d);
        expr = `(${a} ${ops[0]} ${b}) ${ops[1]} ${c} ${ops[2]} ${d} = ${result}`;
        break;
      case 2: // (a op1 (b op2 c)) op3 d
        result = compute(compute(a, ops[0], compute(b, ops[1], c)), ops[2], d);
        expr = `${a} ${ops[0]} (${b} ${ops[1]} ${c}) ${ops[2]} ${d} = ${result}`;
        break;
      case 3: // a op1 ((b op2 c) op3 d)
        result = compute(a, ops[0], compute(compute(b, ops[1], c), ops[2], d));
        expr = `${a} ${ops[0]} (${b} ${ops[1]} ${c}) ${ops[2]} ${d} = ${result}`;
        break;
      case 4: // a op1 (b op2 (c op3 d))
        result = compute(a, ops[0], compute(b, ops[1], compute(c, ops[2], d)));
        expr = `${a} ${ops[0]} (${b} ${ops[1]} ${c} ${ops[2]} ${d}) = ${result}`;
        break;
      case 5: // (a op1 b) op2 (c op3 d)
        result = compute(compute(a, ops[0], b), ops[1], compute(c, ops[2], d));
        expr = `(${a} ${ops[0]} ${b}) ${ops[1]} (${c} ${ops[2]} ${d}) = ${result}`;
        break;
      default:
        return null;
    }
    if (Math.abs(result - target) < 1e-6) {
      return expr;
    }
  } catch (error) {
    // Menghindari error seperti pembagian dengan nol
    return null;
  }
  return null;
}
```

- **Tujuan:** Mengevaluasi ekspresi matematis berdasarkan urutan angka “nums”, kombinasi operator “ops”, dan susunan tanda kurung yang ditentukan oleh “arrangement”.



- Terdapat lima susunan tanda kurung yang berbeda untuk mengevaluasi ekspresi, dijelaskan oleh kasus 1 hingga 5 dalam switch statement.
- Setiap hasil perhitungan dievaluasi menggunakan fungsi “compute()”, yang melakukan operasi dasar matematika.
- Hasil yang mendekati target (20) dalam toleransi kecil ( $1e-6$ ) akan dikembalikan sebagai string yang menggambarkan ekspresi, lengkap dengan hasilnya.

```
function compute(a, op, b) {  
  switch (op) {  
    case "+":  
      return a + b;  
    case "-":  
      return a - b;  
    case "*":  
      return a * b;  
    case "/":  
      if (b === 0) throw "Division by zero";  
      return a / b;  
    default:  
      return 0;  
  }  
}
```

- **Tujuan:** Melakukan operasi matematika antara dua angka, **a** dan **b**, berdasarkan operator **op**.
- Operasi yang didukung: penjumlahan (+), pengurangan (-), perkalian (\*), dan pembagian (/).
- Ada penanganan kesalahan untuk pembagian dengan nol, yang menghasilkan error.

```

const permutations = permute(nums);
const operatorCombos = [];

// Mengenerate semua kombinasi operator
for (let op1 of operators) {
  for (let op2 of operators) {
    for (let op3 of operators) {
      operatorCombos.push([op1, op2, op3]);
    }
  }
}

// Mengenerate semua penempatan tanda kurung (1-5)
const arrangements = [1, 2, 3, 4, 5];

// Iterasi melalui semua kombinasi
for (let perm of permutations) {
  for (let ops of operatorCombos) {
    for (let arrangement of arrangements) {
      const expr = evaluateExpression(perm, ops, arrangement);
      if (expr && !solutions.includes(expr)) {
        // Ganti '*' dengan 'x' untuk tampilan
        solutions.push(expr.replace(/\*/g, "x"));
      }
    }
  }
}

```

- **Permutasi:** Menghasilkan semua permutasi dari angka input.
- **Kombinasi Operator:** Menghasilkan semua kombinasi tiga operator dengan melakukan iterasi di atas operator (+, -, \*, /).
- **Iterasi:** Untuk setiap permutasi angka, kombinasi operator, dan susunan tanda kurung, ekspresi dievaluasi dengan memanggil “evaluateExpression()”. Jika hasilnya memenuhi syarat (mendekati target 20), ekspresi tersebut ditambahkan ke dalam array “solutions”.
- Hasil perkalian diganti dari \* menjadi x untuk tampilan yang lebih familiar di UI.

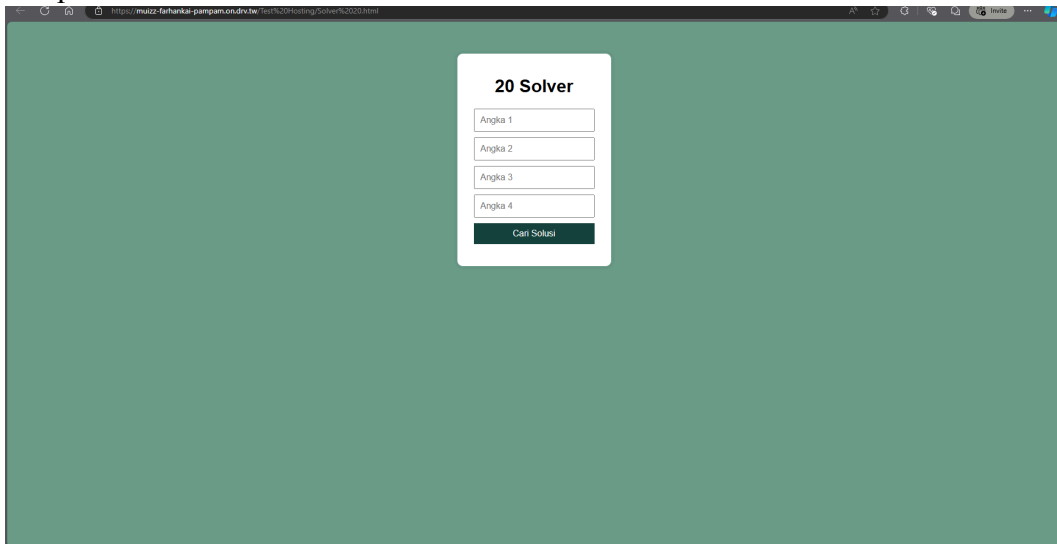
```
const resultsDiv = document.getElementById("results");
resultsDiv.innerHTML = "";

if (solutions.length === 0) {
  resultsDiv.innerHTML = `<p>Tidak ada solusi yang ditemukan.</p>`;
} else {
  solutions.forEach((sol, index) => {
    resultsDiv.innerHTML += `<p>Solusi ke-${index + 1}: ${sol}</p>`;
  });
  resultsDiv.innerHTML += `<p>Total Solusi: ${solutions.length} buah.</p>`;
}
```

- Membersihkan elemen “results” dari solusi sebelumnya.
- Jika tidak ada solusi ditemukan, pesan "Tidak ada solusi yang ditemukan" ditampilkan.
- Jika ada solusi, setiap solusi ditampilkan dalam urutan bersama dengan jumlah total solusi.

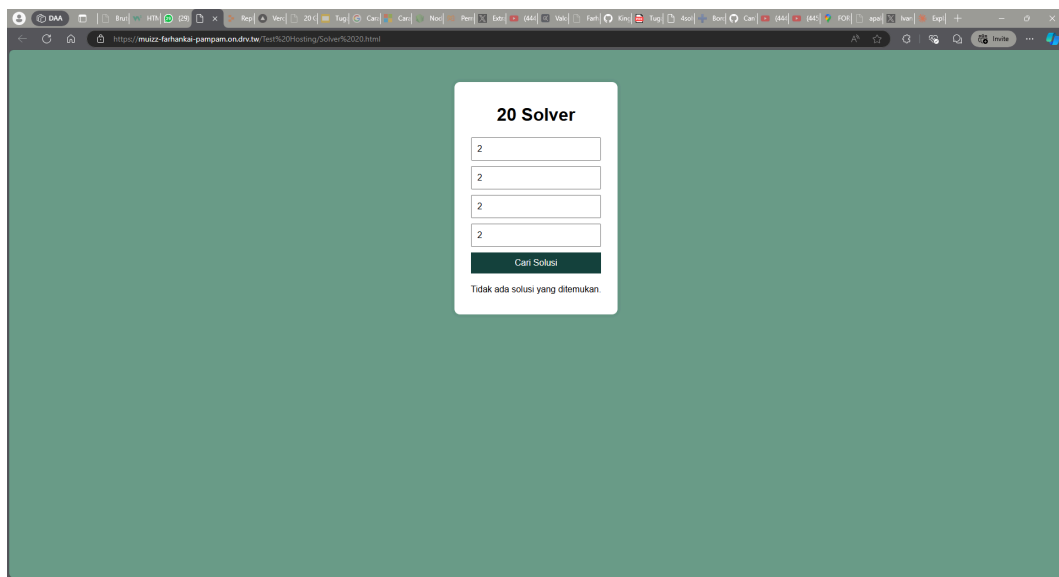
## Pengujian Kode

### 1. Tampilan Web

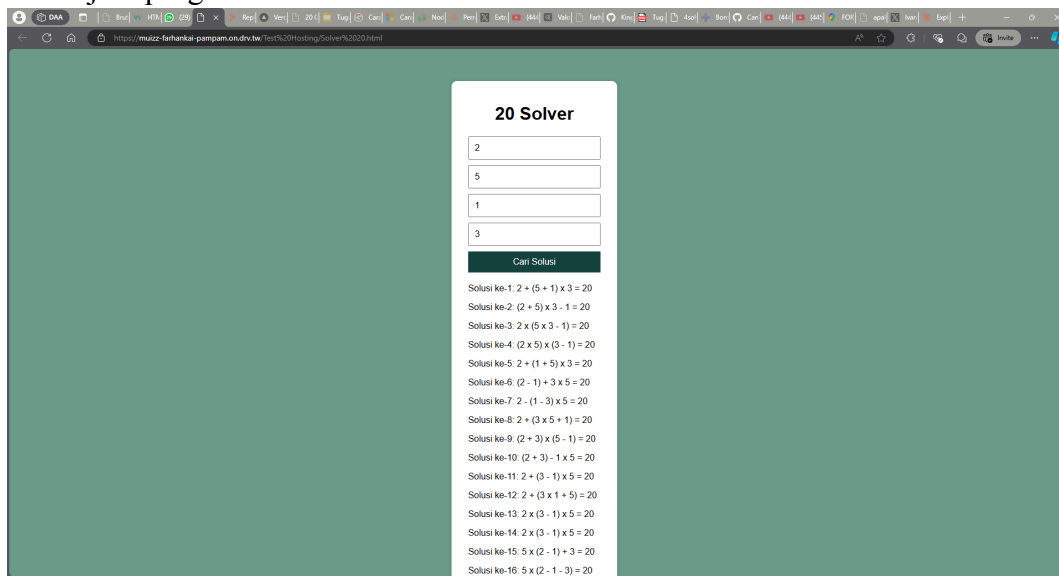


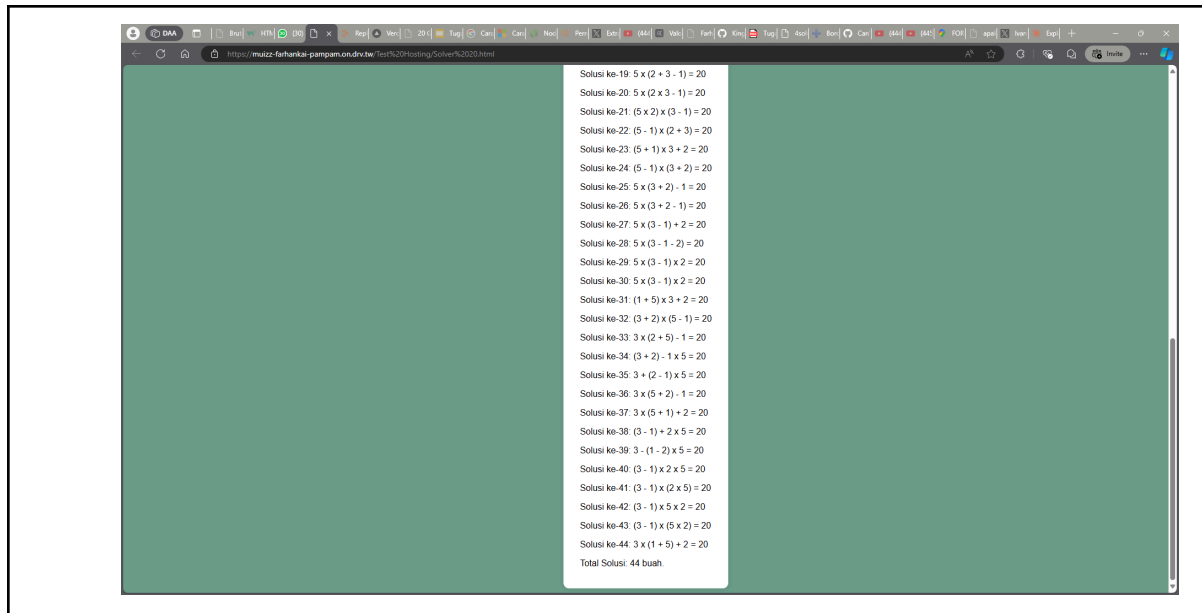
The screenshot shows a web browser window with a URL that includes '20SolvingSolver/2020.html'. The main content area has a green background. In the center, there is a white box titled '20 Solver'. Inside this box, there are four text input fields, each with a placeholder label: 'Angka 1', 'Angka 2', 'Angka 3', and 'Angka 4'. Below these fields is a dark green button with the text 'Cari Solusi' in white.

### 2. Kasus jika program tidak menemukan solusi yang mungkin



### 3. Kasus jika program menemukan solusi





### Peran Anggota Kelompok

1. Muhammad Farhan Khairullah: Membantu dalam penulisan laporan dokumentasi, error testing dalam logika di JavaScript, merangkai repository github.
2. Muhammad Rifai Ageng Pambudi: Mendesain HTML, CSS, serta dokumentasi Video, membantu dalam penulisan laporan dokumentasi
3. Muiz Afif Mirza Lindu Aji: Membantu dalam penulisan laporan dokumentasi, dan JS developer