

**LAPORAN DESAIN ANALISIS DAN ALGORITMA
WEBSITE 20 SOLVER MENGGUNAKAN ALGORITMA BRUTE
FORCE**



DOSEN PENGAJAR:

Fajar Muslim S.T., M.T.

DI SUSUN OLEH:

Moh Ferdinand Ramdhani L0123082

Muhammad Al Fathi Ayyash L0123088

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

2024

BAB I

PENDAHULUAN

A. Penjabaran dari problem

Program dirancang untuk dapat mencetak segala kemungkinan kombinasi solusi operasi dari 4 angka yang diberikan oleh *client* sama dengan 20. Jika tidak ada operasi yang memungkinkan, maka akan mencetak pesan “No solution found”.

Aturan dasar dari 20 Solver ini adalah:

1. Diberikan 4 angka positif dari 0-30,
2. Tujuannya adalah menghasilkan hasil 20 dari 4 angka ini,
3. Diantara 4 angka ini, digunakan operasi matematika seperti: penjumlahan (+), pengurangan (-), perkalian (*), dan pembagian (/),
4. Urutan pengoperasian dapat diatur menggunakan *parentheses* atau tanda kurung (()).

B. Pembagian tugas dalam kelompok

1. Moh Ferdinand Ramdhani: Membuat program (JS, HTML, CSS), menulis laporan
2. M Al Fathi Ayyash : Menulis laporan, membuat video demo

BAB II

IMPLEMENTASI

A. Penjelasan implementasi/source code

Link Github : <https://github.com/Alfaashh/20-Solver>

1. DOM

```
document
.getElementById('solverForm')
```

Menangkap element di file HTML dengan id “solverForm”

2. Event listener

```
.addEventListener('submit', function (event) {
    event.preventDefault();

    document.getElementById('solution').innerHTML = '';
    document.getElementById('solutionLabel').innerHTML = '';

    const number1 = document.getElementById('first').value;
    const number2 = document.getElementById('second').value;
    const number3 = document.getElementById('third').value;
    const number4 = document.getElementById('fourth').value;

    const numbers = [number1, number2, number3, number4];

    if (
        numbers.every((number) => !isNaN(number) && number < 31 &&
number >= 0)
    ) {
        findSolutions(numbers);
    } else {
        document.getElementById('solution').innerHTML = 'Invalid
number input.';
    }
}
```

Ketika form di submit maka fungsi akan dipanggil

a. event.PreventDefault()

Method ini digunakan untuk mencegah *default action* berupa refresh halaman pada suatu form.

b. Mengosongkan konten

Digunakan untuk mengosongkan konten dari elemen dengan ID “solution” dan “solutionLabel” untuk memastikan jika ada solusi sebelumnya ditampilkan, maka akan dihapus terlebih dahulu sebelum menampilkan hasil baru.

c. Menangkap dan menampung nilai input

Nilai yang diinputkan melalui elemen input HTML dengan id “first”, “second”, “third”, dan “fourth” akan ditampung pada variabel number1 hingga number4 dan dikumpulkan menjadi satu array dengan variabel numbers.

d. Validasi

Memeriksa setiap elemen dalam array numbers menggunakan method .Every() untuk memastikan :

- Nilai bukan berupa NaN (Not a Number)
- Nilai berada pada rentang 0 hingga 30.

Jika terpenuhi maka akan menjalankan function findSolutions dengan numbers sebagai parameternya. Namun jika tidak terpenuhi, akan menampilkan “Invalid number input.” pada elemen paragraf dengan id “solution”.

3. Deklarasi Operator

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

Deklarasi array operators yang menyimpan operator matematika.

4. Evaluasi bentuk kombinasi

```
function evaluateExpression(expression) {  
  try {  
    return Function('"use strict";return (' + expression + ')')();  
  } catch {  
    return NaN;  
  }  
}
```

Fungsi ini menangkap parameter expression yang digunakan untuk mengevaluasi bentuk kombinasi matematika yang diberikan dengan bentuk string. Menggunakan konstruktor Function untuk membuat fungsi baru dari string yang diberikan. Dengan menambahkan 'use strict'; di awal, kita memastikan bahwa kode tersebut dieksekusi dalam mode ketat, yang membantu

menghindari beberapa kesalahan umum dalam JavaScript dan mengembalikan hasil tersebut. Namun jika ada kesalahan fungsi akan mengembalikan nilai NaN (Not a Number).

5. Membentuk bentuk kombinasi yang unik

```
function generateUniqueExpressions(nums, ops) {
  const expressions = new Set();
  const plusExpression = new Set();

  function addExpressions(a, b, c, d) {
    ops.forEach((op1) => {
      ops.forEach((op2) => {
        ops.forEach((op3) => {
          if (op1 === op2 && op2 === op3 && op1 === '+') {
            if (plusExpression.size === 0) {
              plusExpression.add(`${a}${op1}${b}${op2}${c}${op3}${d}`);
            } else {
              return;
            }

            for (const expression of plusExpression) {
              expressions.add(expression);
            }
          } else {
            // ((n0 op0 n1) op1 n2) op2 d3
            expressions.add(`((${a}${op1}${b})${op2}${c})${op3}${d}`);
            // (n0 op0 (n1 op1 n2)) op2 d3
            expressions.add(`(${a}${op1} (${b}${op2}${c})) ${op3}${d}`);
            // // (n0 op0 n1) op1 (n2 op2 n3)
            expressions.add(`(${a}${op1}${b}) ${op2} (${c}${op3}${d})`);
            // n0 op0 ((n1 op1 n2) op2 n3)
            expressions.add(`${a}${op1}${b}) ${op2} ((n1 op1 n2) op2 n3)`);
            // n0 op0 (n1 op1 (n2 op2 n3))
            expressions.add(`${a}${op1} ((n1 op1 (n2 op2 n3))`);
            expressions.add(`${a}${op1} (${b}${op2} (${c}${op3}${d}))`);
          }
        });
      });
    });
  }
}
```

```

    });
  }

  for (let i = 0; i < nums.length; i++) {
    for (let j = 0; j < nums.length; j++) {
      if (j === i) continue;
      for (let k = 0; k < nums.length; k++) {
        if (k === i || k === j) continue;
        const l = nums.findIndex(
          (_, index) => index !== i && index !== j && index !==
k
        );
        addExpressions(nums[i], nums[j], nums[k], nums[l]);
      }
    }
  }
  return Array.from(expressions);
}

```

Fungsi tersebut digunakan untuk menghasilkan seluruh bentuk kombinasi yang unik dari sejumlah angka (nums) menggunakan operator yang ada (ops).

1. Inisialisasi variabel penampung

expressions: Set ini digunakan untuk menyimpan ekspresi unik yang dihasilkan.

plusExpression: Set ini hanya digunakan untuk menyimpan ekspresi yang terdiri dari penjumlahan (+).

2. Fungsi addExpressions()

Fungsi ini menangkap 4 angka dan akan menghasilkan kombinasi ekspresi dengan berbagai operator.

- Looping operator

Tiga loop forEach() digunakan untuk iterasi melalui semua kombinasi operator yang ada.

- Penjumlahan

Jika semua operator merupakan penjumlahan (+) dan variabel plusExpression tidak bernilai, maka akan hanya ada satu bentuk kombinasi dan akan ditampung ke dalam variabel plusExpression, kemudian setiap ekspresi pada plusExpression ditambahkan ke variabel expressions

- Kombinasi

Jika seluruh operator tidak berupa penjumlahan (+), berbagai kombinasi ekspresi akan ditambahkan ke expressions dengan menggunakan berbagai cara pengelompokan (menggunakan tanda kurung).

3. Penentuan angka

Tiga loop digunakan untuk memilih empat angka dari array nums. Loop pertama (i) bertugas memilih angka pertama, diikuti oleh loop kedua (j) yang memilih angka kedua, dan seterusnya, dengan pengecekan yang memastikan bahwa angka yang dipilih tidak sama. Untuk mendapatkan indeks angka keempat (l), digunakan findIndex, yang membantu menemukan angka yang tidak dipilih oleh loop sebelumnya. Setelah keempat angka berhasil dipilih, fungsi addExpressions dipanggil untuk menghasilkan berbagai kombinasi ekspresi matematika berdasarkan angka-angka tersebut.

4. Mengembalikan nilai

Fungsi mengembalikan array dari set expressions, yang berisi semua ekspresi unik yang dihasilkan.

6. Mencari solusi penyelesaian

```
function findSolutions(nums) {
  const expressions = generateUniqueExpressions(nums,
operators);
  const solutions = new Set();

  for (const expr of expressions) {
    if (evaluateExpression(expr) === 20) {
      const formattedExpr = expr.replace(/([+ \- * /])/g, ' $1
').trim();
      solutions.add(formattedExpr);
    }
  }

  const solutionElement = document.getElementById('solution');
  const solutionLabel =
document.getElementById('solutionLabel');

  if (solutions.size > 0) {
```

```

        solutionElement.innerHTML =
Array.from(solutions).join('<br>');

        solutionLabel.innerHTML =
            solutions.size + ' solution' + (solutions.size > 1 ? 's' :
'');
    } else {
        solutionElement.innerHTML = 'No solution found';
        solutionLabel.innerHTML = '';
    }
}

```

Fungsi ini bertujuan untuk menemukan semua kombinasi ekspresi yang dapat dihasilkan dari angka yang diberikan (dalam parameter `nums`) yang hasilnya sama dengan 20.

1. Mengenerate hasil yang unik

Memanggil `generateUniqueExpressions` untuk mendapatkan semua kombinasi ekspresi yang mungkin menggunakan angka-angka dari `nums` dan operator yang didefinisikan sebelumnya dan hasilnya disimpan dalam variabel `expressions`.

2. Inisialisasi variabel `solution`

Variabel `solution` digunakan untuk menyimpan ekspresi yang menghasilkan nilai 20

3. Mengevaluasi bentuk kombinasi

Setiap bentuk ekspresi dalam variabel `expressions` akan dievaluasi menggunakan fungsi `evaluateExpression`. Jika hasil evaluasi sama dengan 20, maka ekspresi tersebut akan dilanjutkan dengan :

- Ekspresi diformat untuk menambahkan spasi di sekitar operator. Misalnya, $3+4*2$ menjadi $3 + 4 * 2$, sehingga lebih mudah dibaca.
- Ekspresi yang diformat kemudian ditambahkan ke `solutions`.

4. Menampilkan hasil

Setelah semua ekspresi dievaluasi, fungsi ini memeriksa apakah ada solusi yang ditemukan. Jika `solutions` memiliki elemen, maka hasilnya ditampilkan di elemen HTML yang sesuai, dengan jumlah solusi juga ditampilkan. Jika tidak ada solusi yang ditemukan, maka

akan ditampilkan pesan "No solution found".

BAB III

HASIL

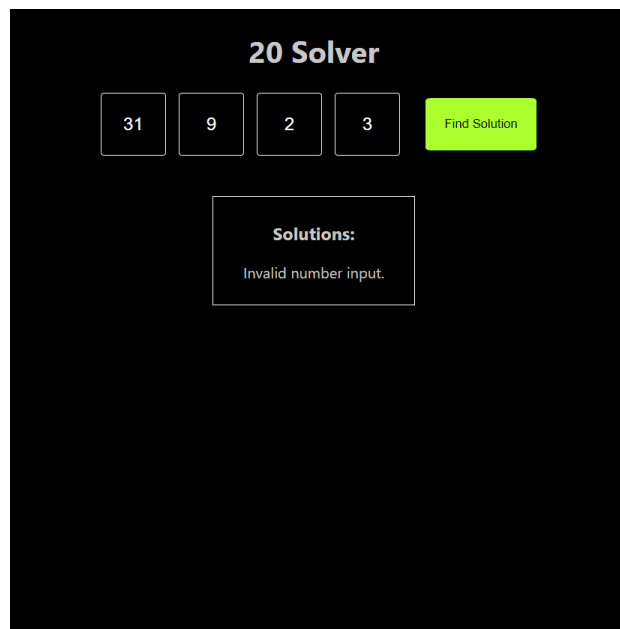
A. Pengujian

Link website : <https://20-solver-uns.vercel.app/>

Beberapa bentuk input untuk menguji program, diantaranya:

1. *Invalid case input*

- a. Input angka lebih dari 30,



The screenshot shows the '20 Solver' web application interface. At the top, the title '20 Solver' is displayed. Below it, there are four input boxes containing the numbers '31', '9', '2', and '3'. To the right of these boxes is a green button labeled 'Find Solution'. Below the input boxes, there is a box labeled 'Solutions:' which contains the text 'Invalid number input.'.

```
if (
  numbers.every((number) => !isNaN((number)) && number < 31
&& number >= 0)
)
```

- b. Input angka kurang dari 0,

20 Solver

Find Solution

Solutions:
Invalid number input.

```
if (
  numbers.every((number) => !isNaN((number)) && number <
31 && number >= 0)
)
```

- c. Kurang dari 4 angka input,

20 Solver

Find Solution

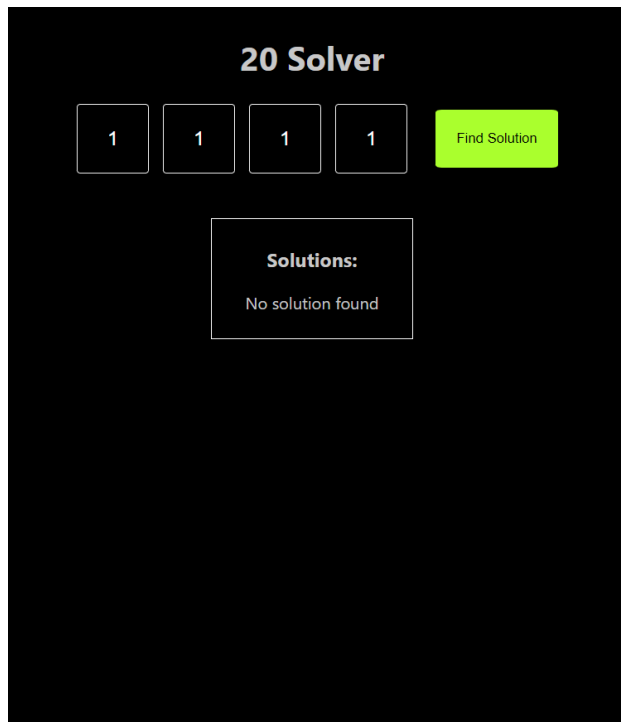
Solutions:

Please fill out this field.

Terdapat 4 elemen input yang setiap elemen input terdapat atribut *required*, yang menunjukkan bahwa elemen tersebut wajib diisi.

```
<input type="text" id="" placeholder="" required />
```

- d. Input kombinasi angka yang tidak mungkin menghasilkan 20,



```
if (evaluateExpression(expr) === 20) {  
  const formattedExpr = expr.replace(/([+\-*\/])/g, ' $1  
').trim();  
  solutions.add(formattedExpr);  
}
```

Maka variabel solutions memiliki size == 0, dan jalankan perintah else

```
if (solutions.size > 0) {  
  solutionElement.innerHTML =  
Array.from(solutions).join('<br>');  
  
  solutionLabel.innerHTML =  
    solutions.size + ' solution' + (solutions.size > 1 ? 's' :  
'');  
} else {  
  solutionElement.innerHTML = 'No solution found';  
  solutionLabel.innerHTML = '';  
}
```

- e. Input dengan bentuk *string* dan *special character*;

20 Solver

test 3 @ 2 Find Solution

Solutions:
Invalid number input.

```
if (
    numbers.every((number) => !isNaN((number)) && number <
31 && number >= 0)
)
```

2. Valid input

- a. Angka bilangan bulat,

20 Solver

1 5 3 9 Find Solution

Solutions:
8 solutions
(1 + 9) * (5 - 3)
(1 + (9 / 3)) * 5
5 * (1 + (9 / 3))
(5 - 3) * (1 + 9)
(5 - 3) * (9 + 1)
5 * ((9 / 3) + 1)
(9 + 1) * (5 - 3)
((9 / 3) + 1) * 5

20 Solver

27

3

5

4

Find Solution

Solutions:

16 solutions

$(27 - (3 * 4)) + 5$
 $27 - ((3 * 4) - 5)$
 $(27 + 5) - (3 * 4)$
 $27 + (5 - (3 * 4))$
 $(27 + 5) - (4 * 3)$
 $27 + (5 - (4 * 3))$
 $(27 - (4 * 3)) + 5$
 $27 - ((4 * 3) - 5)$
 $(5 + 27) - (3 * 4)$
 $5 + (27 - (3 * 4))$
 $(5 + 27) - (4 * 3)$
 $5 + (27 - (4 * 3))$
 $(5 - (3 * 4)) + 27$
 $5 - ((3 * 4) - 27)$
 $(5 - (4 * 3)) + 27$
 $5 - ((4 * 3) - 27)$

b. Angka bilangan desimal,

20 Solver

0.5

10

2

4

Find Solution

Solutions:

25 solutions

$((0.5 + 2) * 4) + 10$
 $10 + ((0.5 + 2) * 4)$
 $10 + ((2 + 0.5) * 4)$
 $((10 + 2) / 0.5) - 4$
 $((10 - 2) / 0.5) + 4$
 $(10 + 2) + (4 / 0.5)$
 $10 + (2 + (4 / 0.5))$
 $10 + (4 * (0.5 + 2))$
 $(10 + (4 / 0.5)) + 2$
 $10 + ((4 / 0.5) + 2)$
 $10 + (4 * (2 + 0.5))$
 $((2 + 0.5) * 4) + 10$
 $((2 + 10) / 0.5) - 4$
 $(2 + 10) + (4 / 0.5)$
 $2 + (10 + (4 / 0.5))$
 $(2 + (4 / 0.5)) + 10$
 $2 + ((4 / 0.5) + 10)$
 $((4 / 0.5) + 10) + 2$

c. Edge case

20 Solver

17

1

1

1

Find Solution

Solutions:

1 solution

$17 + 1 + 1 + 1$

BAB IV

KESIMPULAN

A. Implementasi Teknologi

1. Proyek 20 Solver diimplementasikan menggunakan kombinasi teknologi web standar:

- JavaScript (JS) untuk logika dan fungsionalitas
- HTML untuk struktur dan konten
- CSS untuk styling dan tata letak

2. Penggunaan DOM dan Event Listener menunjukkan integrasi yang baik antara JavaScript dan HTML, memungkinkan interaktivitas pada halaman web.

B. Responsivitas

Meskipun tidak disebutkan secara eksplisit dalam laporan, penggunaan teknologi web standar memungkinkan responsivitas di berbagai perangkat. Untuk memastikan responsivitas optimal, beberapa langkah tambahan yang dapat diambil meliputi:

- Penggunaan CSS media queries untuk menyesuaikan tata letak pada berbagai ukuran layar
- Implementasi desain fluid grid
- Penggunaan ukuran font yang responsif
- Optimalisasi gambar dan elemen lain untuk kecepatan loading di perangkat mobile

C. Algoritma Brute Force: Kelebihan dan Kekurangan

Proyek ini menggunakan pendekatan brute force untuk menyelesaikan masalah 20 Solver. Berikut adalah kelebihan dan kekurangan dari penggunaan algoritma ini:

Kelebihan:

1. Jaminan menemukan semua solusi, algoritma brute force memeriksa semua kemungkinan kombinasi, memastikan tidak ada solusi yang terlewat.
2. Mudah diimplementasikan, kode untuk algoritma brute force relatif sederhana dan mudah dipahami.

3. Efektif untuk dataset kecil, untuk jumlah input yang terbatas (dalam hal ini 4 angka), algoritma ini dapat memberikan hasil dengan cepat.

Kekurangan:

1. Kurang efisien untuk dataset besar: Jika jumlah input atau kompleksitas masalah meningkat, waktu eksekusi dapat meningkat secara eksponensial.
2. Penggunaan sumber daya tinggi: Algoritma ini dapat memakan banyak memori dan daya komputasi, terutama untuk masalah yang lebih kompleks.
3. Tidak optimal untuk masalah optimisasi: Dalam kasus di mana hanya solusi terbaik yang diperlukan, brute force mungkin tidak efisien karena memeriksa semua kemungkinan.