LAPORAN TUGAS KECIL I

IF2211 Strategi Algoritma

"Implementasi Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force"

Dosen:

Ir. Rila Mandala, M. Eng, Ph. D.

Monterico Adrian, S. T., M. T.

Disusun Oleh:

13522131 Owen Tobias Sinurat

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2023/2024

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa penulis ucapkan atas kesempatan dan keberhasilan dalam menyelesaikan Tugas Kecil 1 IF2211 Strategi Algoritma, Semester II tahun 2023/2024, yang berjudul "Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force". Laporan ini merupakan dokumentasi komprehensif dari proses pengembangan program yang dirancang untuk menemukan solusi paling optimal dalam permainan Breach Protocol dari https://cyberpunk-hacker.com/ menggunakan pendekatan algoritma *brute force*.

Tugas ini tidak hanya menantang penulis secara akademis, tetapi juga memberikan pengalaman belajar yang menarik dalam mengaplikasikan teori algoritma pada kasus nyata yang cukup kompleks. Penulis berusaha keras untuk mengembangkan solusi yang efisien dan efektif, mulai dari tahap *design* algoritma hingga implementasi *code* dan pengujian.

Dengan dukungan dari dosen, asisten kelompok, input dari rekan mahasiswa, dan berbagai sumber daya yang tersedia, penulis telah menyelesaikan program untuk tugas kecil ini yang diharapkan dapat bekerja dengan baik dan efisien. Akhir kata, penulis mengucapkan terima kasih kepada semua yang telah berpartisipasi dan mendukung. Penulis berharap agar tugas besar ini dapat memenuhi mata kuliah Strategi Algoritma dan dapat memberikan wawasan yang bermanfaat bagi pembaca serta menjadi sumber inspirasi untuk inovasi lebih lanjut.

Sumedang, 12 Februari 2024,

Owen Tobias Sinurat.

BAB I
DESKRIPSI MASALAH

1.1. Deskripsi Masalah

Dalam era digital yang terus berkembang ini, kita menyaksikan peningkatan eksponensial dalam produksi dan penyimpanan gambar. Fenomena ini menyebar luas di berbagai sektor, termasuk kehidupan kita secara pribadi yang mengabadikan kenangan melalui foto, sektor kesehatan yang bergantung pada gambar medis untuk diagnosis, dunia akademik yang berinteraksi dengan ilustrasi ilmiah untuk penelitian, dan industri komersial yang menggunakan gambar untuk mempromosikan produk dan layanan. Pada intinya, setiap gambar ini membawa data visual yang apabila dikelola dan diakses dengan tepat, dapat memberikan nilai yang luar biasa.

Dalam menghadapi kompleksitas dan volume gambar yang terus meningkat, sistem temu balik gambar merupakan solusi yang tepat. Sistem ini tidak hanya memfasilitasi proses pencarian dan akses cepat ke koleksi gambar yang luas, tetapi juga mengoptimalisasi pengelolaan aset digital tersebut. Fitur ini menjadi sangat *powerful* dalam konteks di mana kecepatan dan efisiensi dalam menemukan informasi visual menjadi kunci.

Maka dari itu, ntuk menjawab tantangan dalam Tugas Besar ini, kami telah mengembangkan sebuah sistem temu balik gambar yang inovatif melalui penerapan Aljabar Vektor yang terintegrasi dalam suatu *website*. Pendekatan ini adalah inti dari pemrosesan data dan pencarian informasi yang canggih. Dengan menggunakan teknik Content-Based Image Retrieval (CBIR), sistem kami mengidentifikasi dan mengklasifikasikan gambar berdasarkan karakteristik kontennya, yakni warna atau tekstur, sehingga memungkinkan pengguna untuk menggali dan menavigasi melalui kumpulan data visual mereka dengan cara yang intuitif dan efisien.

BAB II
ALGORITMA BRUTE FORCE

2.1 Cara Kerja

Secara garis besar cara pencarian solusi secara brute force pada program ini adalah:

1. Sequence dikumpulkan menjadi satu array untuk diproses lebih lanjut
2. Sequence diproses untuk mendapatkan segala kemungkinan solusi berdasarkan sequence yang ada. Disinilah letak brute force-nya, segala kemungkinan yang dimaksud bukanlah semua rute yang mungkin, melainkan semua rute yang mungkin dan mengandung kombinasi sequence atau sequence itu secara individual.
3. Setelah mendapatkan semua kombinasi sequence, di filter yang melewati batas buffer.
4. Setelah itu, kumpulan sequence tersebut akan dicari apakah ada rute yang mungkin memenuhi sequence terkait dengan aturan gerakan harus selang-seling antara vertikal dan horizontal.
5. Setelah mendapatkan solusi yang memenuhi, semua hasilnya di filter lagi apakah ada yang melewati batas buffer?
6. Setelah di filter, reward masing-masing solusi akan dihitung lalu di sorting berdasarkan reward tersebut secara descending.
7. Sudah didapat solusi paling optimal.

BAB III
SOURCE PROGRAM

3.1 utilities.js

Modul berikut berfungsi sebagai alat-alat untuk tujuan general seperti mengonstruksi versi teks dari solusi yang berupa array.

```
export const constructSolutionText = (solution, matrix) => {
    const solutionText = solution.map((s) => matrix[s.y][s.x]).join(' ')
    return solutionText
}

export const isOneOfTheSolutions = (x, y, solution) => {
    return solution.some((s) => s.x === x && s.y === y)
}

export const generateInput = ({
    buffer,
    matrixSizer,
    token,
    totalSequence,
    sequenceMax,
}) => {
    const tokenArray = token.trim().split(' ')
        const matrix = Array.from({length: matrixSizer.row}, () =>
        Array.from({length: matrixSizer.column}, () => {
            const randomIndex = Math.floor(Math.random() * tokenArray.length)
            return tokenArray[randomIndex]
        })
    )
    const matrixString = matrix.map((row) => row.join(' ')).join('\n')
    let sequence = []
```

```
        for (let i = 0; i < totalSequence; i++) {
                                    const   sequenceLength   =
Math.floor(Math.random() * sequenceMax) + 1
                const sequenceArray = Array.from({length:
sequenceLength}, () => {
                                        const   randomIndex   =
Math.floor(Math.random() * tokenArray.length)
                 return tokenArray[randomIndex]
             })
                const reward = Math.floor(Math.random() *
100)
                sequence.push({token: sequenceArray.join('
'), reward})
          }
          console.log('matrix', matrixString)
          console.log('sequence', sequence)
          return {matrix: matrixString, sequence, buffer}
      }
```

3.2 processors.js

Modul ini berfungsi sebagai alat untuk memproses input, output, dan nilai-nilai yang harus diproses agar bisa dipakai di algoritma.

```
export const processInput = (matrix, sequence) => {
    return [
        matrix.split('\n').map((row) =>
            row
                .trim()
                .split(' ')
                .map((n) => parseInt(n, 16))
        ),
        sequence.split('\n').map((row) =>
            row
                .trim()
                .split(' ')
                .map((n) => parseInt(n, 16))
        ),
    ]
}
```

```
export const processFile = (text) => {
    const lines = text
        .trim()
        .split(/[(\n|\r\n)]/)
        .filter((s) => s !== '')
    const buffer = lines[0]
    const m = Number(lines[1].split(' ')[0])
    let matrix = ''
    let sequence = []
    for (let i = 2; i < m + 2; i++) {
        matrix = matrix.concat(lines[i])
        if (i + 1 !== m + 2) {
            matrix = matrix.concat('\n')
        }
    }
    const x = Number(lines[m + 2]) * 2
    for (let j = m + 3; j < x + m + 3; j = j + 2) {
        sequence.push({token: lines[j], reward:
Number(lines[j + 1])})
    }
    return {buffer, matrix, sequence}
}

function matchSequence(split, remain) {
    const children = remain.flatMap((remainder,
remainIndex) => {
        const childSplits = constructSequence(remainder,
split.result, [
            remainder,
            ...split.includes,
        ])
        const childRemains = remain.filter((_, i) => i !==
remainIndex)
        return childSplits.map((childSplit) =>
            matchSequence(childSplit, childRemains)
        )
    })
    return {
        children,
        value: split,
    }
```

```
}


export const processOutput = (matrix) => {
    return matrix.split('\n').map((row) =>
row.trim().split(' '))
}

export default function findSequences(candidates) {
    let rootNodes = []
    for (let i = 0; i < candidates.length; i++) {
        const candidate = candidates[i]
        const targets = candidates.filter((_, index) =>
index !== i)
        for (let j = 0; j < targets.length; j++) {
            const target = targets[j]
            const remain = targets.filter((_, index) =>
index !== j)
            const initialSplits =
constructSequence(candidate, target, [
                candidate,
                target,
            ])
            rootNodes = rootNodes.concat(
                initialSplits.map((split) =>
matchSequence(split, remain))
            )
        }
    }
    return rootNodes
}

function checkMatchR(offset, a, b) {
    for (let i = offset; i < b.length; i++) {
        if (a[i - offset] !== b[i]) {
            return false
        }
    }
    return true
}

function checkMatchL(offset, a, b) {
    for (let i = offset; i < a.length; i++) {
```

```
        if (a[i] !== b[i - offset]) {
            return false
        }
    }
    return true
}

function constructSequence(candidate, target, includes) {
    let sequences = []
    // gabungin 2 sequence dengan menghilangkan token yang
sama dan bertetangga, misal A B dengan B C menjadi A B C
    // bisa jadi tidak ada kombinasi yang mungkin
    for (let shiftRight = 0; shiftRight < target.length;
shiftRight++) {
        if (checkMatchR(shiftRight, candidate, target)) {
            sequences.push({shift: shiftRight, dir:
'right'})
        }
    }

    for (let shiftLeft = 1; shiftLeft < candidate.length;
shiftLeft++) {
        if (checkMatchL(shiftLeft, candidate, target)) {
            sequences.push({shift: shiftLeft, dir: 'left'})
        }
    }

    sequences.push({shift: target.length, dir: 'right'})
    sequences.push({shift: candidate.length, dir: 'left'})

    return sequences.map((s) => applyShift(candidate,
target, s, includes))
}

function applyShift(shiftee, target, shift, includes) {
    let result = []

    if (shift.dir === 'right') {
        const output = [...target]
        output.length = Math.max(shiftee.length +
shift.shift, target.length)
        for (let i = 0; i < shiftee.length; i++) {
            output[i + shift.shift] = shiftee[i]
```

```
        }
        result = output
    } else {
        const output = [...shiftee]
        output.length = Math.max(target.length +
shift.shift, shiftee.length)
        for (let i = 0; i < target.length; i++) {
            output[i + shift.shift] = target[i]
        }
        result = output
    }

    return {
        ...shift,
        result,
        shiftee,
        target,
        includes,
    }
}
```

3.3 algorithm.js

Modul ini adalah fungsi kalkulasi utama dari program terkait.

```
import findSequencess from './processors'
import {constructSolutionText} from './utilities'

export function BFS(roots) {
    const seenNodes = []
    let queue = [...roots.map((n) => ({...n, depth: 1}))]
    while (queue.length > 0) {
        const node = queue.shift()
        seenNodes.push(node)
        queue = queue.concat(
            node.children.map((child) => ({...child, depth:
node.depth + 1}))
        )
    }
    return seenNodes
}

export function removeDuplicates(arr) {
```

```
    const keys = new Set()
    return arr.filter((seq) => {
        const key =
            seq.result.join(',') +
            '_' +
            seq.includes
                .map((incl) => incl.join(','))
                .sort((a, b) => a.localeCompare(b))
                .join('-')
        if (keys.has(key)) {
            return false
        }
        keys.add(key)
        return true
    })
}

export default function findSolution(
    matrix,
    sequences,
    bufferSize,
    rawSequence
) {
        // optimizesquence mengembalikan semua kombinasi
sequence yang mungkin menjadi solusi akhir
    const roots = findSequencess(sequences)
          const rootsValues = BFS(roots).map((node) =>
node.value)

        // ini menambah sequence secara individu karna
optimizer hanya return kombinasinya saja
    const values = [
        ...sequences.map((sequence) => ({
            result: sequence,
            includes: [sequence],
        })),
        ...rootsValues,
    ]
    const seqsThatFitInBuffer = values.filter(
        (r) => r.result.length <= bufferSize
    )
                          const      uniqueSequences      =
removeDuplicates(seqsThatFitInBuffer)
```

```
    const solutionsByDistance = uniqueSequences
        .flatMap((match) => {
            const pattern = match.result
                const solutions = findSolutions(pattern,
matrix, true)
                return solutions.map((solution) => ({match,
solution}))
        })
        .filter((seq) => seq.solution.length <= bufferSize)
        .map((s) => ({
            ...s,
              totalReward: calculateTotalReward(s.solution,
rawSequence, matrix),
        }))
        .sort((totalReward) => totalReward.reward)
        .reverse()
    return {solution: solutionsByDistance}
}

function      calculateTotalReward(route,      rawSequence,
rawMatrix) {
        const  solutionText  =  constructSolutionText(route,
rawMatrix)
    let totalReward = 0
    for (let i = 0; i < rawSequence.length; i++) {
        const sequence = rawSequence[i]
        const sequenceText = sequence.token
            .trim()
            .split(' ')
            .map((n) => parseInt(n, 16))
            .join(' ')
        const sequenceReward = sequence.reward
        if (solutionText.includes(sequenceText)) {
            totalReward += sequenceReward
        }
    }
    return totalReward
}

export const findSolutions = (pattern, matrix, findAll) =>
{
    const yLen = matrix.length
    const xLen = matrix[0].length
```

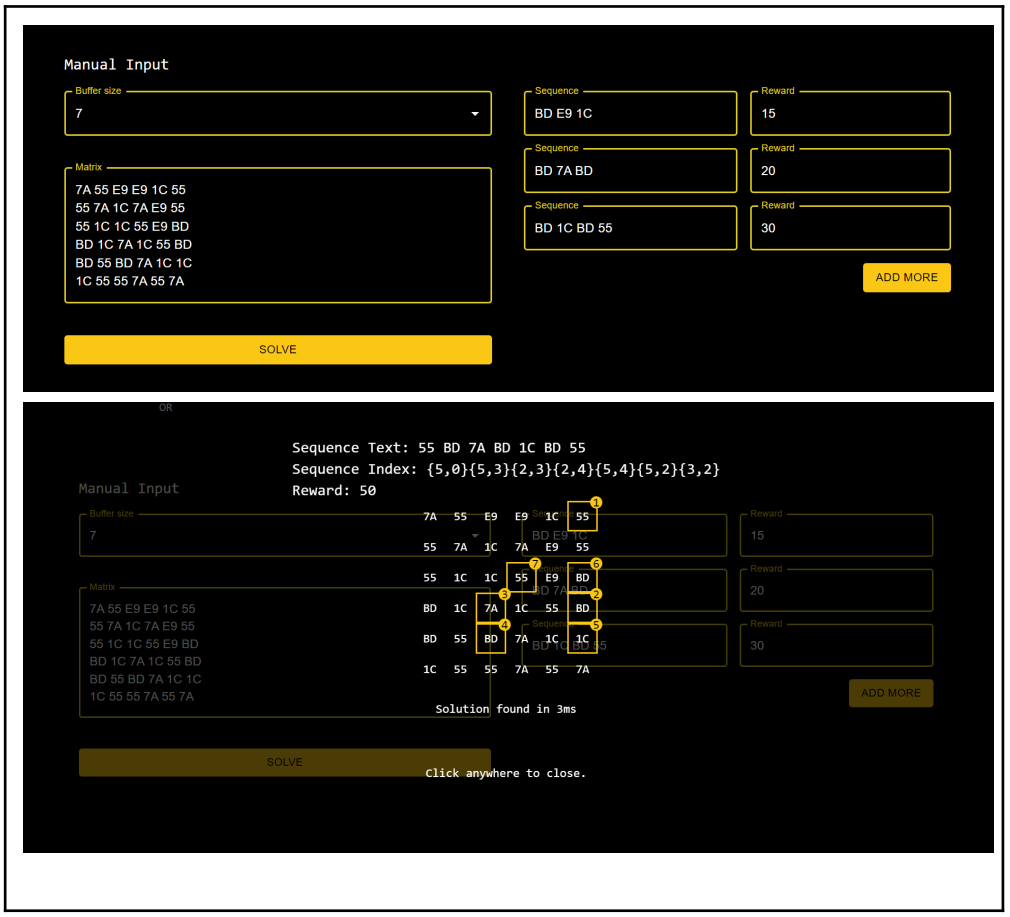
```
    // queue untuk bfs
    const queue = [
        {
            patternPtr: 0,
            used: constructArray(yLen, xLen, false),
            stepsSoFar: [],
            x: 0,
            y: 0,
            allowedDir: 'horizontal',
        },
    ]

    let isInitial = true
    const solutions = []

    while (queue.length > 0) {
        const searchPoint = queue.shift()
         const {patternPtr, used, stepsSoFar, allowedDir} =
searchPoint

        if (patternPtr === pattern.length) {
            if (!findAll) {
                return [stepsSoFar]
            }
            solutions.push(stepsSoFar)
        }

          for (const {x, y} of walkAllowedDir(searchPoint,
yLen, xLen)) {
            if (matrix[y][x] === pattern[patternPtr]) {
                queue.push({
                    patternPtr: patternPtr + 1,
                    used: markUsed(used, x, y),
                    stepsSoFar: stepsSoFar.concat({x, y}),
                     allowedDir: allowedDir === 'vertical' ?
'horizontal' : 'vertical',
                    x,
                    y,
                })
            }
                 // baris pertama bukan? kalo iya maka di
toleransi walaupun ga sesuai sequence
```

```
            else if (isInitial) {
                queue.push({
                    patternPtr: patternPtr,
                    used: markUsed(used, x, y),
                    stepsSoFar: stepsSoFar.concat({x, y}),
                     allowedDir: allowedDir === 'vertical' ?
'horizontal' : 'vertical',
                    x,
                    y,
                })
            }
        }
        isInitial = false
    }
    return solutions
}

function* walkAllowedDir(searchPoint, yLen, xLen) {
    const {used, allowedDir} = searchPoint
    if (allowedDir === 'vertical') {
        const {x} = searchPoint
        for (let y = 0; y < yLen; y++) {
            if (used[y][x]) {
                continue
            }
            yield {x, y}
        }
    } else {
        const {y} = searchPoint
        for (let x = 0; x < xLen; x++) {
            if (used[y][x]) {
                continue
            }
            yield {x, y}
        }
    }
}

function markUsed(arr, x, y) {
    const copy = cloneArray(arr)
    copy[y][x] = true
    return copy
}
```

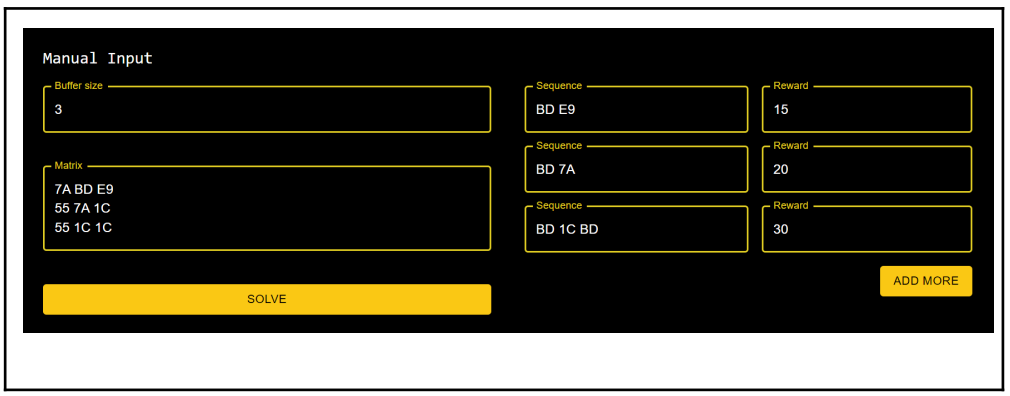
```
function cloneArray(arr) {
    return arr.map((subarr) => subarr.slice())
}

function constructArray(yLen, xLen, fillValue) {
    const arr = new Array(yLen)
    for (let y = 0; y < yLen; y++) {
        arr[y] = new Array(xLen).fill(fillValue)
    }
    return arr
}
```

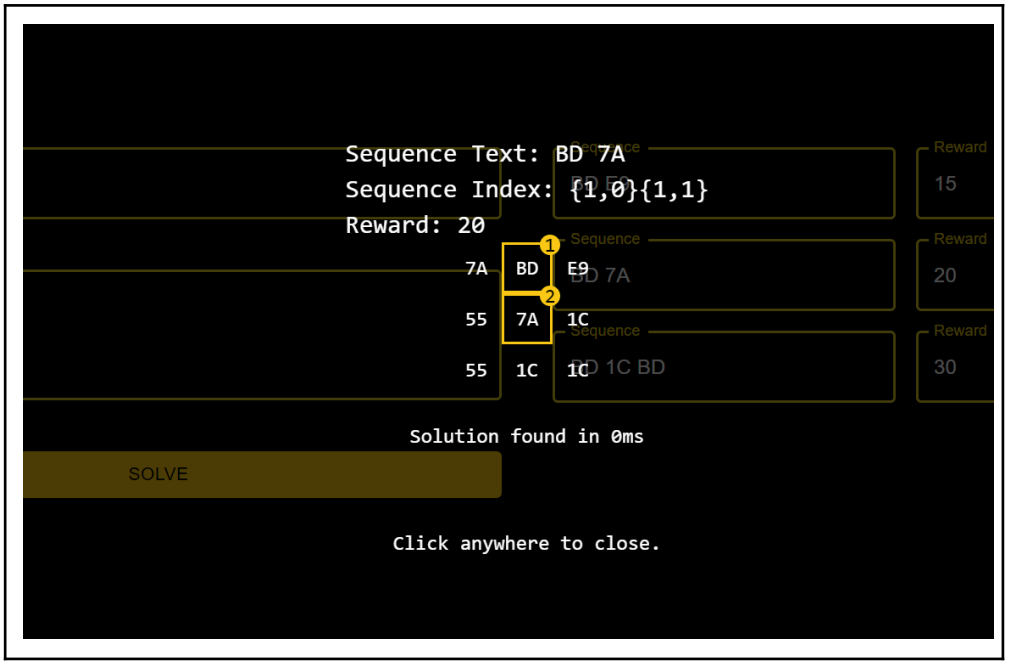
BAB IV
HASIL EKSEKUSI PROGRAM

4.1 Hasil 1



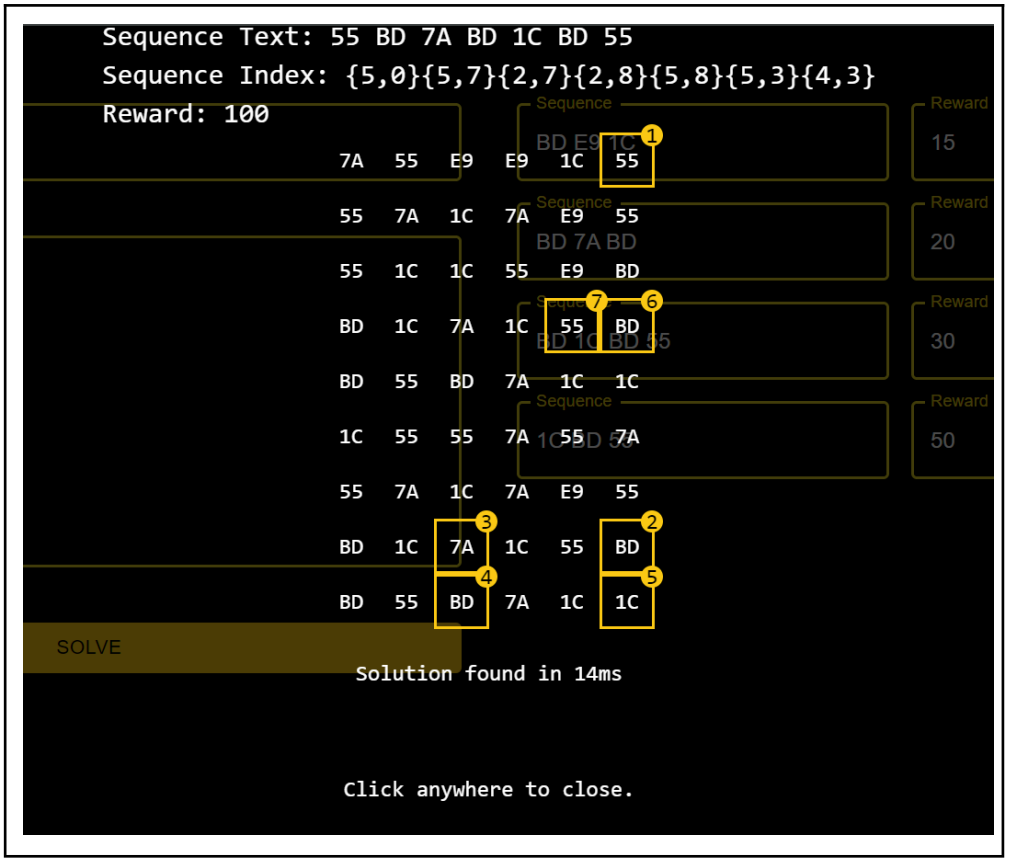
4.2 Hasil 2



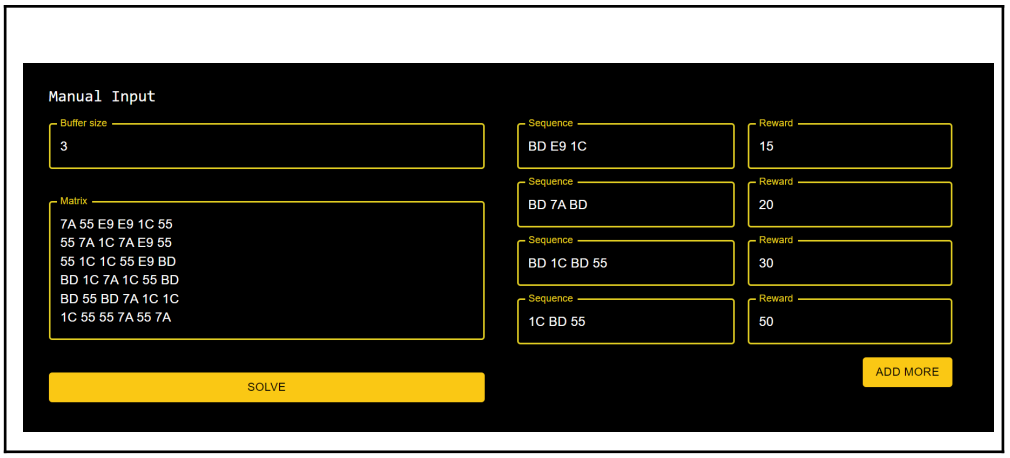

Sequence Text: BD 7A
Sequence Index: {1,0}{1,1}
Reward: 20
7A BD E9
55 7A 1C
55 1C 1C
Solution found in 0ms
SOLVE
Click anywhere to close.

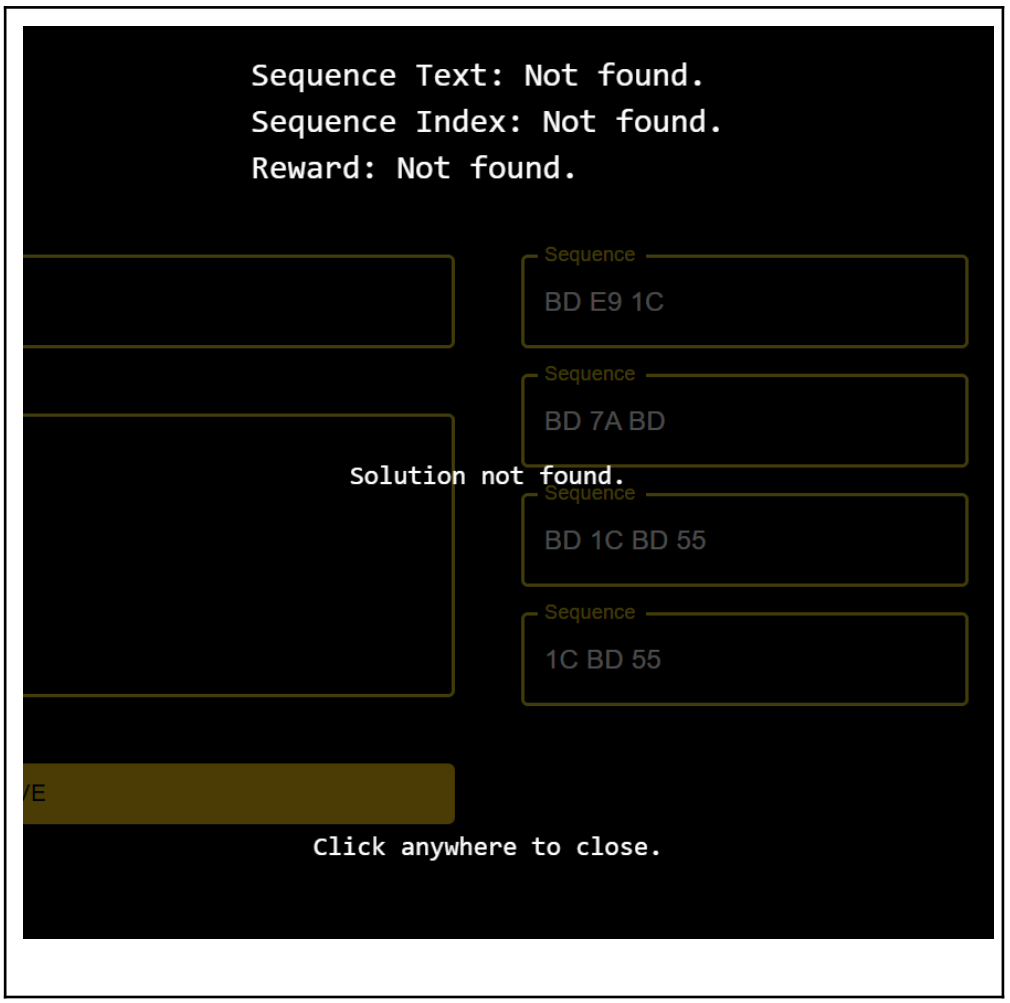
4.3 Hasil 3


Manual Input
Buffer size
7
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
55 7A 1C 7A E9 55
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
SOLVE
Sequence
BD E9 1C
Reward
15
Sequence
BD 7A BD
Reward
20
Sequence
BD 1C BD 55
Reward
30
Sequence
1C BD 55
Reward
50
ADD MORE

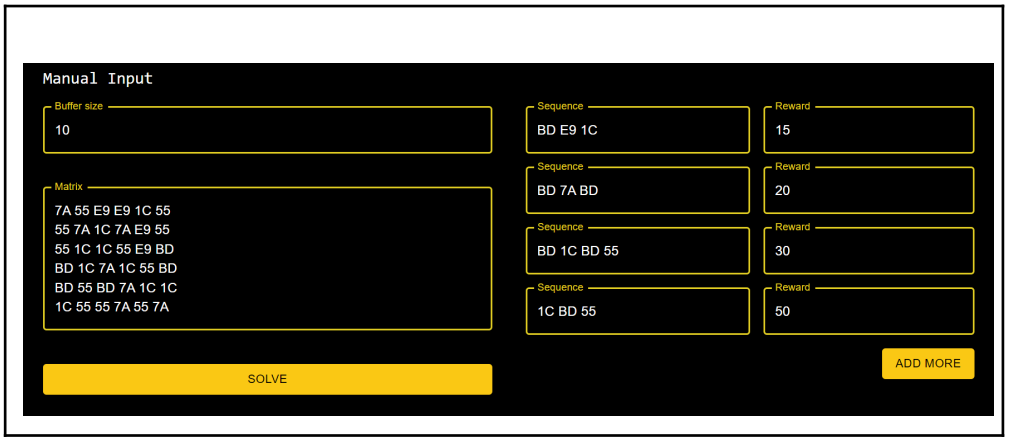

Sequence Text: 55 BD 7A BD 1C BD 55
Sequence Index: {5,0}{5,7}{2,7}{2,8}{5,8}{5,3}{4,3}
Reward: 100
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
55 7A 1C 7A E9 55
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
Sequence
Reward
15
20
30
50
SOLVE
Solution found in 14ms
Click anywhere to close.

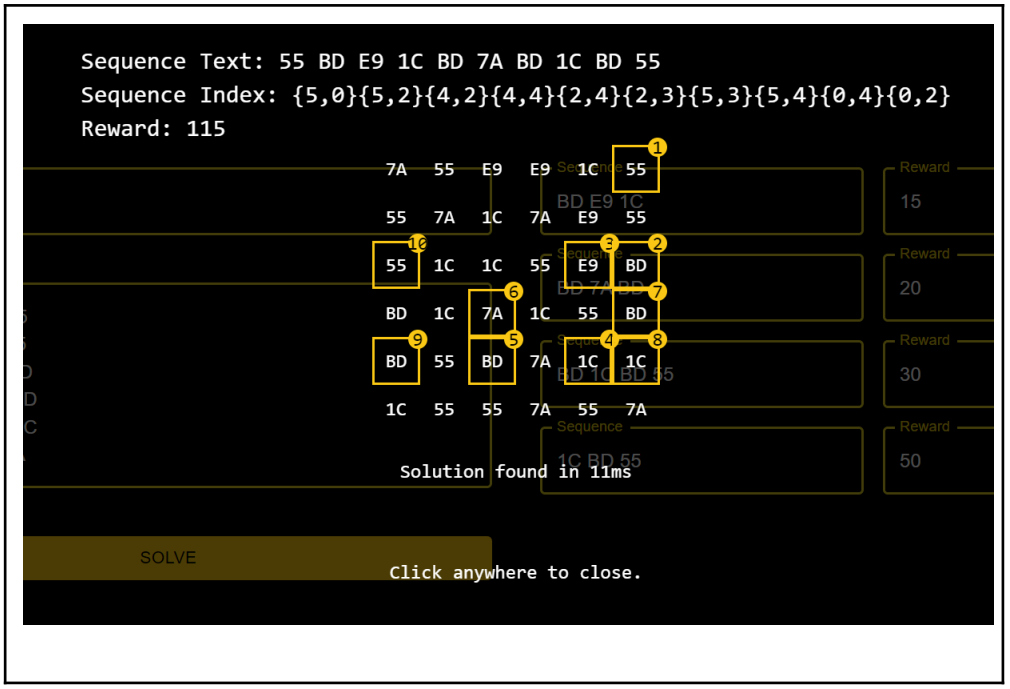
4.4 Hasil 4


Manual Input
Buffer size
3
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
SOLVE
Sequence
BD E9 1C
Reward
15
Sequence
BD 7A BD
Reward
20
Sequence
BD 1C BD 55
Reward
30
Sequence
1C BD 55
Reward
50
ADD MORE

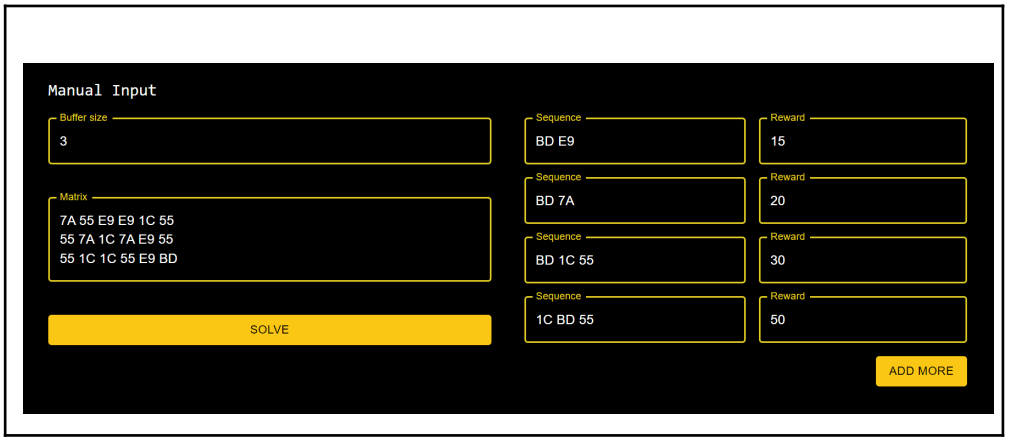

Sequence Text: Not found.
Sequence Index: Not found.
Reward: Not found.
Sequence
BD E9 1C
Sequence
BD 7A BD
Solution not found.
Sequence
BD 1C BD 55
Sequence
1C BD 55
Click anywhere to close.

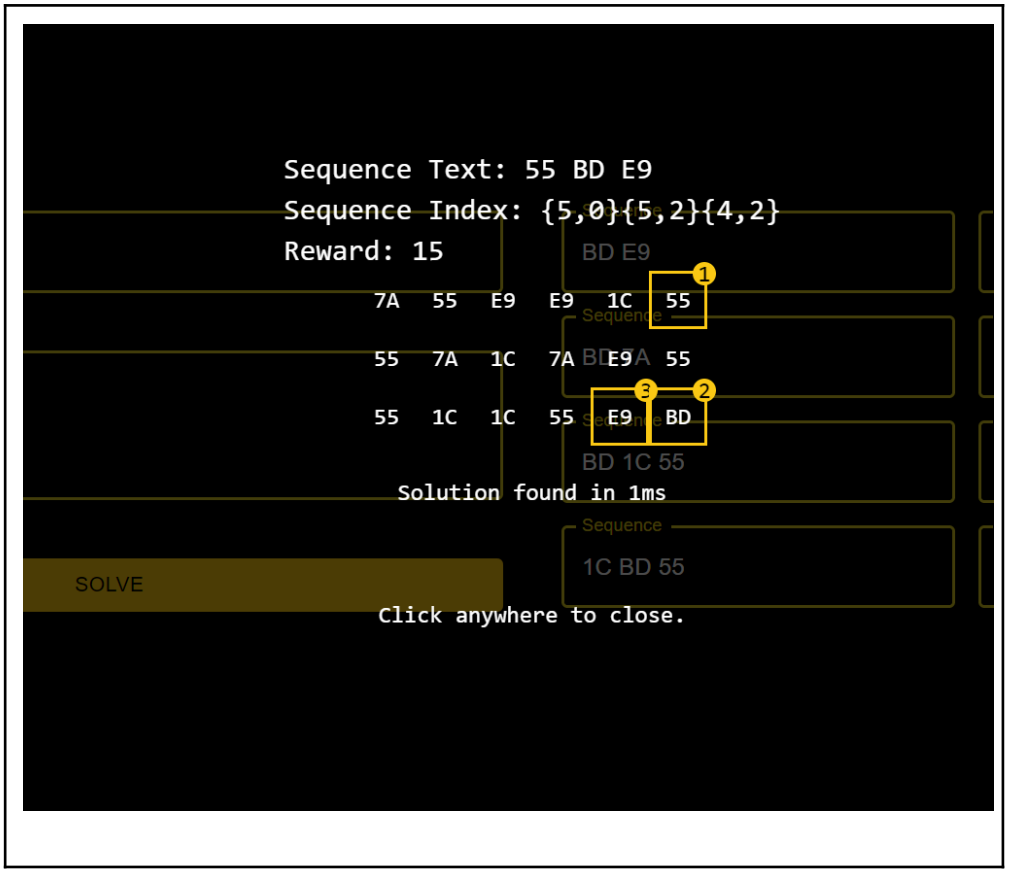
4.5 Hasil 5


Manual Input
Buffer size
10
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
SOLVE
Sequence
BD E9 1C
Reward
15
Sequence
BD 7A BD
Reward
20
Sequence
BD 1C BD 55
Reward
30
Sequence
1C BD 55
Reward
50
ADD MORE


Sequence Text: 55 BD E9 1C BD 7A BD 1C BD 55
Sequence Index: {5,0}{5,2}{4,2}{4,4}{2,4}{2,3}{5,3}{5,4}{0,4}{0,2}
Reward: 115
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
Solution found in 11ms
SOLVE
Click anywhere to close.

4.6 Hasil 6


Manual Input
Buffer size
3
Matrix
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
SOLVE
Sequence
BD E9
Reward
15
Sequence
BD 7A
Reward
20
Sequence
BD 1C 55
Reward
30
Sequence
1C BD 55
Reward
50
ADD MORE


Sequence Text: 55 BD E9
Sequence Index: {5,0}{5,2}{4,2}
Reward: 15
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
1
2
3
Solution found in 1ms
Click anywhere to close.
BD E9
BD 1C 55
1C BD 55
SOLVE

BAB V
LAMPIRAN

GitHub Repository

https://github.com/owenthe10x/Tucil1_13522131

Tabel Pencapaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca masukan berkas .txt	✓	
4. Program dapat menghasilkan masukan secara acak	✓	
5. Solusi yang diberikan program optimal	✓	
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI	✓	