LAPORAN TUGAS KECIL 1
STRATEGI ALGORITMA
BRUTE-FORCE ALGORITHM

Disusun Oleh :

Nama : Maulvi Ziadinda Maulana

Kelas : K-03

NIM : 13522122

Institut Teknologi Bandung

2024

A. Algoritma Brute-Force

Berikut adalah gambaran dari cara kerja algoritma yang telah saya buat.

1. Program akan memulai dengan menggunakan pointer yang menunjuk pada buffer yang memiliki menunjuk ke elemen dalam array dengan nilai pointer yang dimulai dari 0 hingga bufferSize.
2. Program akan memulai pencarian dengan cara membuat kombinasi dalam buffer dari elemen-elemen yang ada didalam matriks. Pembuatan kombinasi ini akan dimulai dengan bergerak secara horizontal lalu vertikal secara bergantian dimulai dari ujung kiri atas matriks.
3. Program akan membuat kombinasi-kombinasi dengan ukuran array kombinasi yang dimulai dari minimal panjang seqeunce yang ada ditambah 1 hingga buffer terisi penuh. Misalnya apabila panjang sequence terpendek adalah 3, maka program akan membuat kombinasi array dengan panjang 3, melakukan pengecekan untuk setiap kombinasi, lalu dilanjutkan dengan panjang 4, dan seterusnya hingga buffer penuh.
4. Ada beberapa kondisi yang dapat menghentikan proses pembuatan kombinasi. Yang pertama apabila pointer saat ini sedang menunjuk ke posisi ke-1 (elemen dimulai dari 0) dan elemen yang ditunjuk dalam matriks bukanlah awal dari sebuah sequence, maka kombinasi pada elemen buffer selanjutnya tidak akan dilanjutkan, namun akan dilakukan kombinasi elemen yang lain pada pointer saat ini. Hal ini dikarenakan setiap elemen dalam matriks dipastikan dapat diakses maksimal dengan 2 langkah, dan tentu sangat logis apabila elemen pada pointer posisi ke-1 bukanlah start dari sequence, maka proses akan dilanjutkan untuk kombinasi selanjutnya. Selainjutnya apabila posisi pointer saat ini adalah panjang maksimal sequence yang ada ditambah 1 dan tidak ada sequence yang terdeteksi, maka kombinasi saat ini tidak akan dilanjutkan untuk elemen selanjutnya. Yang ketiga adalah ketika score saat ini sudah maksimal, maka pencarian akan dihentikan, kombinasi array yang didapatkan dapat dipastikan memiliki langkah yang minimal karena proses pembuatan kombinasi dimulai dari panjang minimal. Yang terakhir adalah ketika elemen yang ditunjuk saat ini sudah ada didalam buffer.

5. Pengecekan score akan dilakukan setiap gerakan horizontal ataupun vertikal dilakukan. Proses pengecekan score dilakukan dengan algoritma yaitu melakukan loop dari awal buffer, jika elemen saat ini adalah start dari sebuah sequence maka akan dilanjutkan pengecekan kesamaan elemen sequence selanjutnya, jika tidak maka akan dilanjutkan elemen buffer selanjutnya untuk dicek. Apabila sudah ditemukan kecocokan, maka proses pencarian akan dihentikan.

B. Source Code Program

Untuk membuat struktur program menjadi lebih baik, saya membagi fungsi-fungsi yang diperlukan menjadi beberapa file. Yang pertama ada class Solver untuk menyelesaikan masalah, ada main progam, ada data structure, ada input/output, dan file utils untuk membantu proses yang dilakukan dalam program. Untuk membuat laporan ini lebih efisien untuk dibaca, maka yang saya sertakan hanyalah source code untuk solver secara spesifik pada algortima brute force yang diterapkan. Source code untuk file-file lain dapat dilihat pada repository yang terlampir di poin D. Berikut adalah source code algoritma brute force pada Solver.

```
void BreachProtocolSolver::Solve() {
    auto startTime = chrono::high_resolution_clock::now();

    for (int i = minSequnceLength - 1; i < bufferSize; i++) {
        if (result.score == maxScore) {
            break;
        }

        HorizontalMove(0, i);
    }

    auto endTime = chrono::high_resolution_clock::now();

    auto processingTime = chrono::duration_cast<chrono::milliseconds>(endTime
- startTime);

    result.time = processingTime.count();

    showResult();
}

void BreachProtocolSolver::HorizontalMove(int bufferPointer, int maxPointer) {
    if (result.score == maxScore || bufferPointer > maxPointer + 1) {
        return;
    }

    int currentRow = 0;
    if (bufferPointer == 0) {
        currentRow = 0;
```

```
    } else {
        currentRow = sequenceString[bufferPointer - 1].row;
    }

    if (bufferPointer <= bufferSize) {

        int tempPointer;

        if (bufferPointer < bufferSize) {
            tempPointer = bufferPointer;
        } else if (bufferPointer >= bufferSize){
            tempPointer = bufferSize - 1;
        }

        array<int, 2> temp = CheckScore(tempPointer);

        if (temp[0] > result.score || (temp[0] == result.score && (temp[1] <
result.last || result.last == -1))){
            result.sequenceResult = sequenceString;
            result.score = temp[0];
            result.last = temp[1];
        }


        if (bufferPointer >= bufferSize || (bufferPointer >= maxSequnceLength
+ 1  && temp[0] == 0) || result.score == maxScore) {
            return;
        }
    }

    for (int i = 0; i < matrixCol; i++) {
        sequenceString[bufferPointer] = Point(currentRow, i);

        if (bufferPointer == 1 && !CheckStartOfSequence(Point(currentRow, i))
&& !CheckStartOfSequence(sequenceString[0])) {
            continue;
        }

        if (CheckPoint(Point(currentRow, i), sequenceString, bufferPointer)) {
            continue;
        }

        VerticalMove(bufferPointer + 1, maxPointer);
    }
}

void BreachProtocolSolver::VerticalMove(int bufferPointer, int maxPointer) {
    if (result.score == maxScore || bufferPointer > maxPointer + 1) {
        return;
    }

    int currentCol = 0;
    if (bufferPointer == 0) {
        currentCol = 0;
    } else {
        currentCol = sequenceString[bufferPointer - 1].col;
    }

    if (bufferPointer <= bufferSize) {
        int tempPointer;

        if (bufferPointer < bufferSize) {
```

```
            tempPointer = bufferPointer;
        } else if (bufferPointer >= bufferSize){
            tempPointer = bufferSize - 1;
        }

        array<int, 2> temp = CheckScore(tempPointer);

        if (temp[0] >= result.score || (temp[0] == result.score && (temp[1] <
result.last || result.last == -1))){
            result.sequenceResult = sequenceString;
            result.score = temp[0];
            result.last = temp[1];
        }

        if (bufferPointer >= bufferSize || (bufferPointer >= maxSequnceLength
+ 1  && temp[0] == 0) || result.score == maxScore) {
            return;
        }
    }

    for (int i = 0; i < matrixRow; i++) {
        sequenceString[bufferPointer] = Point(i, currentCol);

        if (bufferPointer == 1 && !CheckStartOfSequence(Point(i, currentCol))
&& !CheckStartOfSequence(sequenceString[0])) { // cek yang 1 iya atau ga
            continue;
        }

        if (CheckPoint(Point(i, currentCol), sequenceString, bufferPointer)) {
            continue;
        }

        HorizontalMove(bufferPointer + 1, maxPointer);
    }
}

bool BreachProtocolSolver::CheckPoint(Point point, vector<Point>
sequenceString, int bufferPointer) {
    for (int i = 0; i < bufferPointer; i++) {
        if (sequenceString[i].row == point.row && sequenceString[i].col ==
point.col) {
            return true;
        }
    }

    return false;
}

bool BreachProtocolSolver::CheckStartOfSequence(Point point) {
    for (set<string>::iterator it = startOfSequences.begin(); it !=
startOfSequences.end(); it++) {
        if (matrix[point.row][point.col] == *it) {
            return true;
        }
    }
    return false;
}

array<int, 2> BreachProtocolSolver::CheckScore(int bufferPointer) {
    array<int, 2> scoreAndLastPointer = {0, -1};

    for (int i = 0; i < sequences.size(); i++) {
```

```
        if (sequences[i].data.size() > bufferPointer + 1) {
            continue;
        }

        bool isMatch;
        int LastPointer = 0;
        for (int j = 0; j <= bufferPointer + 1 - sequences[i].data.size();
j++) {

            isMatch = true;

            int k = 0;
            LastPointer = j;

            while (k < sequences[i].data.size()) {
                if (sequenceString[LastPointer].row == -1) {
                    isMatch = false;
                    break;
                }

                if (sequences[i].data[k] !=
matrix[sequenceString[LastPointer].row][sequenceString[LastPointer].col]) {
                    isMatch = false;
                    break;
                }

                k++;
                LastPointer++;
            }

            if (isMatch) {
                scoreAndLastPointer[0] += sequences[i].value;
                if (LastPointer > scoreAndLastPointer[1]){
                    scoreAndLastPointer[1] = --LastPointer;
                }
                break;
            }
        }
    }

    return scoreAndLastPointer;
}
```

C. Testing Program

Progam akan ditest dengan beberapa test case yaitu sebagai berikut. Hasil Test juga akan disertakan secara langsung di bawah test case.

1. Test ke-1

```
Ukuran baris: 6
Ukuran kolom: 6
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

Sequences:
BD E9 1C  -> Score: 15
BD 7A BD  -> Score: 20
BD 1C BD 55  -> Score: 30

Hasil:
50
55 BD 7A BD 1C BD 55
1, 6
4, 6
4, 3
5, 3
5, 6
3, 6
3, 4

13 ms
```

2. Test ke-2

```
Ukuran baris: 2
Ukuran kolom: 2
Buffer Size: 3

Matrix:
AA 7B
1C 2E

Sequences:
AA  -> Score: 10
AA 1C  -> Score: 10
AA 1C 2E  -> Score: 10

Hasil:
30
AA 1C 2E
1, 1
2, 1
2, 2

0 ms
```

3. Test ke-3

```
Ukuran baris: 3
Ukuran kolom: 3
```

```
Buffer Size: 7

Matrix:
7A 2E 1C
E9 BD 7A
55 7A BD

Sequences:
7A E9 BD  -> Score: 15
BD 7A BD  -> Score: 20
7A BD 1C 2E  -> Score: 30

Hasil:
65
7A E9 BD 7A BD 1C 2E
1, 1
2, 1
2, 2
3, 2
3, 3
1, 3
1, 2

1 ms
```

4. Test ke-4

```
Ukuran baris: 6
Ukuran kolom: 6
Buffer Size: 6

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

Sequences:
EF EF 1C  -> Score: 15
BD EF BD  -> Score: 20
BD 1C BD AC  -> Score: 30

Tidak ada hasil yang ditemukan.
```

5. Test ke-5

```
Ukuran baris: 3
Ukuran kolom: 3
Buffer Size: 7

Matrix:
7A 2E 1C
```

```
E9 BD 7A
55 7A BD

Sequences:
7A E9 BD  -> Score: -15
BD 7A BD  -> Score: -20
7A BD 1C 2E  -> Score: -30

Tidak ada hasil yang ditemukan atau hasil terbaik adalah score 0 dengan
tidak bergerak.
```

6. Test ke-6 (Input menggunakan CLI)

```
Masukkan jumlah token: 4
Masukkan token: AA B9 1C E6
Masukkan ukuran buffer: 6
Masukkan ukuran matriks (width height): 6 7
Masukkan jumlah sequence: 3
Masukkan maksimal panjang sequence: 4

Ukuran baris: 7
Ukuran kolom: 6
Buffer Size: 6

Matrix:
AA AA 1C AA AA 1C
E6 1C 1C B9 AA B9
B9 B9 AA AA B9 1C
B9 E6 B9 B9 1C E6
AA B9 B9 E6 1C B9
B9 B9 E6 B9 AA E6
1C AA E6 B9 1C AA

Sequences:
AA B9 1C  -> Score: 50
B9 E6  -> Score: 90
1C B9  -> Score: 80

Hasil:
220
AA B9 1C B9 E6
1, 1
3, 1
3, 6
5, 6
4, 6

3 ms
```

D. Pranala Repository

Berikut adalah pranala ke repository github untuk code penyelesaian dari permasalahan yang diberikan https://github.com/maulvi-zm/Tucil1_13522122.

E. Lampiran

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		✓