LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh:

Kristo Anugrah 13522024

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

DAFTAR ISI

BAGIAN 1: ALGORITMA BRUTE FORCE 1
BAGIAN 2: SOURCE CODE 2
BAGIAN 3: TES INPUT DAN OUTPUT 7
BAGIAN 4: PRANALA REPOSITORY 16
BAGIAN 5: LAMPIRAN 17

BAGIAN 1: ALGORITMA BRUTE FORCE

Algoritma *brute force* adalah algoritma yang bersifat *straightforward*, yang seringkali membutuhkan waktu komputasi yang lama dan kurang efisien. Algoritma *brute force* berdasar dari pencarian seluruh kemungkinan yang ada.

Dalam penyelesaian *puzzle Cyberpunk 2077 Breach Protocol,* algoritma *brute force* digunakan untuk mencari sekuens token yang menghasilkan jumlah *reward* terbesar. Untuk suatu matriks G, panjang buffer B, dan kumpulan sekuens reward Q, langkah-langkah algoritma *brute force* yang digunakan adalah sebagai berikut:

1. Buat prosedur rekursif bruteForce, yang menerima input posisi sel matriks saat ini (cur), himpunan sel matriks yang sudah diambil (chosen), dan boolean yang menandakan arah gerak (isVer).
2. Jika banyak token yang dipilih sama dengan B, untuk setiap sekuens reward cek apakah sekuens tersebut ada pada token yang sudah dipilih. Jika ada, tambahkan *reward* ke suatu variabel totalReward. Misalkan sekuens *reward* memiliki panjang k dan muncul pada posisi i. Ubah variabel curLengOpt menjadi nilai maksimum dari curLengOpt dan k + i
3. Jika totalReward > maxReward, ubah nilai maxReward menjadi totalReward serta salin urutan token yang dipilih ke variabel global answer. Jadikan variabel global bufLeng sama dengan curLengOpt
4. Jika totalReward sama dengan maxReward namun curLengOpt < bufLeng, lakukan langkah 3
5. Jika sel matriks saat ini dalam keadaan “stuck”, dimana seluruh sel pada baris/kolom yang dapat dipilih sudah terambil, lakukan langkah 2
6. Untuk setiap sel S pada baris/kolom sesuai dengan isVer saat ini, masukkan posisi sel saat ini ke chosen dan panggil prosedur bruteForce dengan cur sama dengan S dan argumen isVer sama dengan komplemen dari isVer saat ini

Mudah dilihat bahwa algoritma *brute force* tersebut akan mendapatkan seluruh kemungkinan pengambilan sekuens token yang mungkin. Misalkan $F(G,B) = N^{\lceil B/2 \rceil} M^{\lfloor B/2 \rfloor}$, dengan N banyak baris dari matriks G dan M banyak kolom dari matriks G. Untuk suatu matriks G, banyak sekuens Q, panjang maksimum sekuens A, dan banyak buffer B, definisikan $T(G,Q,A,B)$ sebagai banyak operasi yang dilakukan algoritma di atas. Mudah dilihat bahwa $T(G,Q,A,B) < F(G,B)QBA \leq (\max(N,M))^B QBA$. Hal ini dikarenakan banyak kemungkinan sekuens dengan panjang B kurang dari sama dengan $F(G,B)$. Maka, dapat ditulis:

$$T(G,Q,A,B) = O\left(N^{\lceil B/2 \rceil} M^{\lfloor B/2 \rfloor} QBA\right) = O((\max(N,M))^B QBA)$$

Perhatikan bahwa algoritma tersebut berorde eksponensial. Untuk panjang buffer yang bertambah 1, algoritma membutuhkan B kali operasi. Hal ini merupakan salah satu karakteristik algoritma *brute force*, yang menjadikan algoritma ini kurang efisien dibandingkan dengan algoritma lain. Untuk mempercepat algoritma, seluruh token unik akan di-*map* ke sebuah angka unik. Proses ini akan mempercepat proses pengecekan kesamaan dua token.

BAGIAN 2: SOURCE CODE

Source code ditulis dalam bahasa Java. Algoritma terdapat dalam file working.java dan terdiri dari 2 prosedur utama, yaitu prosedur rekursif bruteForce dan prosedur utama solvemain. Prosedur utama solvemain digunakan untuk melakukan *mapping* token unik ke sebuah angka unik.

```
public class working {

    static int cnt2 = 0;
    static int N, M, B, Q;
    static int maxReward = Integer.MIN_VALUE;
    static List<List<Byte>> mat = new ArrayList<List<Byte>>();
    static List< List<Byte> > seqs = new ArrayList<List<Byte>>();
    static List<Integer> rewards = new ArrayList<Integer>();
    static HashMap<String, Byte> masterMap = new HashMap<String, Byte>();
    static List<Byte> bs = new ArrayList<Byte>();
    static Stack<Byte> chosen = new Stack<Byte>();
    static List<Byte> answer = new ArrayList<Byte>();
    static List<List<String>> matrix = new ArrayList<List<String>>();
    static List<String[]> allline = new ArrayList<String[]>();
    static long timeinMs = 150;
    static int bufLeng = 0;

    working(List<String[]> allline, List<List<String>> matrix, List<Integer>
rewards, int N, int M, int B, int Q){
        working.allline = allline;
        working.matrix = matrix;
        working.rewards = rewards;
        working.N = N;
        working.M = M;
        working.B = B;
        working.Q = Q;
        working.maxReward = Integer.MIN_VALUE;
        working.cnt2 = 0;
        working.mat.clear();
        working.seqs.clear();
        working.masterMap.clear();
        working.bs.clear();
        working.chosen.clear();
        working.answer.clear();
        working.timeinMs = 150;
        working.bufLeng = 0;
    }

    public static void bruteForce(int cur, boolean ver) {
        if(chosen.size() == B) {
            int totalReward = 0;
            boolean noExist = true;
```

```
            int curLengOpt = 0;
            for(int i = 0; i < Q; ++i) {
                boolean allok = false;
                for(int j = 0; j < B - seqs.get(i).size() + 1 && !allok; ++j)
{
                    boolean ok = true;
                    for(int k = 0; k < seqs.get(i).size() && ok; ++k) {
                        byte curRow = (byte)(chosen.get(k + j) / M);
                        byte curCol = (byte)(chosen.get(k + j) % M);
                        if(seqs.get(i).get(k) != mat.get(curRow).get(curCol))
{
                            ok = false;
                        }

                    }
                    if(ok) {
                        curLengOpt = Math.max(curLengOpt, j +
seqs.get(i).size());
                        allok = true;
                        noExist = false;
                    }
                }
                if(allok) {
                    totalReward += rewards.get(i);
                }
            }
            if(noExist) {
                return;
            }
            if(maxReward < totalReward || (maxReward == totalReward &&
curLengOpt < bufLeng)) {
                maxReward = totalReward;
                bufLeng = curLengOpt;
                for(int i = 0; i < chosen.size(); ++i) {
                    answer.set(i, chosen.get(i));
                }
            }
            return;
        }
        int k = M;
        if(ver) {
            k = N;
        }
        boolean stuck = true;
        int curRow = (cur / M);
        int curCol = (cur % M);
        for(int i = 0; i < k; ++i) {
            int cell;
```

```
            if(ver) {
                cell = i * M + curCol;
            }else {
                cell = curRow * M + i;
            }
            if(bs.get(cell) == (byte)1) {
                continue;
            }
            stuck = false;
            bs.set(cell, (byte)1);
            chosen.push((byte)cell);
            bruteForce(cell, !ver);
            bs.set(cell, (byte)0);
            chosen.pop();
        }
        if(!stuck) {
            return;
        }
        int curLengOpt = 0;
        int totalReward = 0;
        boolean noExist = true;
        for(int i = 0; i < Q; ++i) {
            boolean allok = false;
            for(int j = 0; j < chosen.size() - seqs.get(i).size() + 1
&& !allok; ++j) {
                boolean ok = true;
                for(int k1 = 0; k1 < seqs.get(i).size() && ok; ++k1) {
                    byte curRow1 = (byte)(chosen.get(k1 + j) / M);
                    byte curCol1 = (byte)(chosen.get(k1 + j) % M);
                    if(seqs.get(i).get(k1) != mat.get(curRow1).get(curCol1)) {
                        ok = false;
                    }

                }
                if(ok) {
                    curLengOpt = Math.max(curLengOpt, j + seqs.get(i).size());
                    allok = true;
                    noExist = false;
                }
            }
            if(allok) {
                totalReward += rewards.get(i);
            }
        }
        if(noExist) {
            return;
        }
```

```
        if(maxReward < totalReward || (maxReward == totalReward && curLengOpt
< bufLeng)) {
            maxReward = totalReward;
            bufLeng = curLengOpt;
            for(int i = 0; i < chosen.size(); ++i) {
                answer.set(i, chosen.get(i));
            }
        }
    }

    public static void solvemain() {
        for(int i = 0; i < B; ++i) {
            answer.add((byte)0);
        }
        int cnt = 0;
        for(int i = 0; i < N; ++i) {
            mat.add(new ArrayList<Byte>());
        }
        for(int i = 0; i < N; ++i) {
            for(int j = 0; j < M; ++j) {
                mat.get(i).add((byte)0);
            }
        }
        for(int i = 0; i < N; ++i) {
            for(int j = 0; j < M; ++j) {
                String s = matrix.get(i).get(j);
                if(!masterMap.containsKey(s)) {
                    masterMap.put(s, (byte)cnt);
                    cnt++;
                }
                mat.get(i).set(j, masterMap.get(s));
                bs.add((byte)0);
            }
        }
        for(int i = 0; i < Q; ++i) {
            String[] splitted = allline.get(i);
            List<Byte> temp = new ArrayList<Byte>();
            for(int j = 0; j < splitted.length; ++j) {
                if(!masterMap.containsKey(splitted[j])) {
                    masterMap.put(splitted[j], (byte)cnt);
                    cnt++;
                }
                temp.add((byte)masterMap.get(splitted[j]));
            }
            seqs.add(temp);
        }
        long startTime = System.nanoTime();
        for(int i = 0; i < M; ++i) {
```

```
            bs.set(i, (byte)1);
            chosen.push((byte)i);
            bruteForce(i, true);
            chosen.pop();
            bs.set(i, (byte)0);
        }
        if(working.maxReward <= 0){
            working.maxReward = Integer.MIN_VALUE;
            working.bufLeng = 0;
        }
        long estimatedTime = System.nanoTime() - startTime;
        long timeinMillis = TimeUnit.NANOSECONDS.toMillis(estimatedTime);
        timeinMs = timeinMillis;
    }

    public static void main(String[] args) {
        solvemain();
    }
}
```

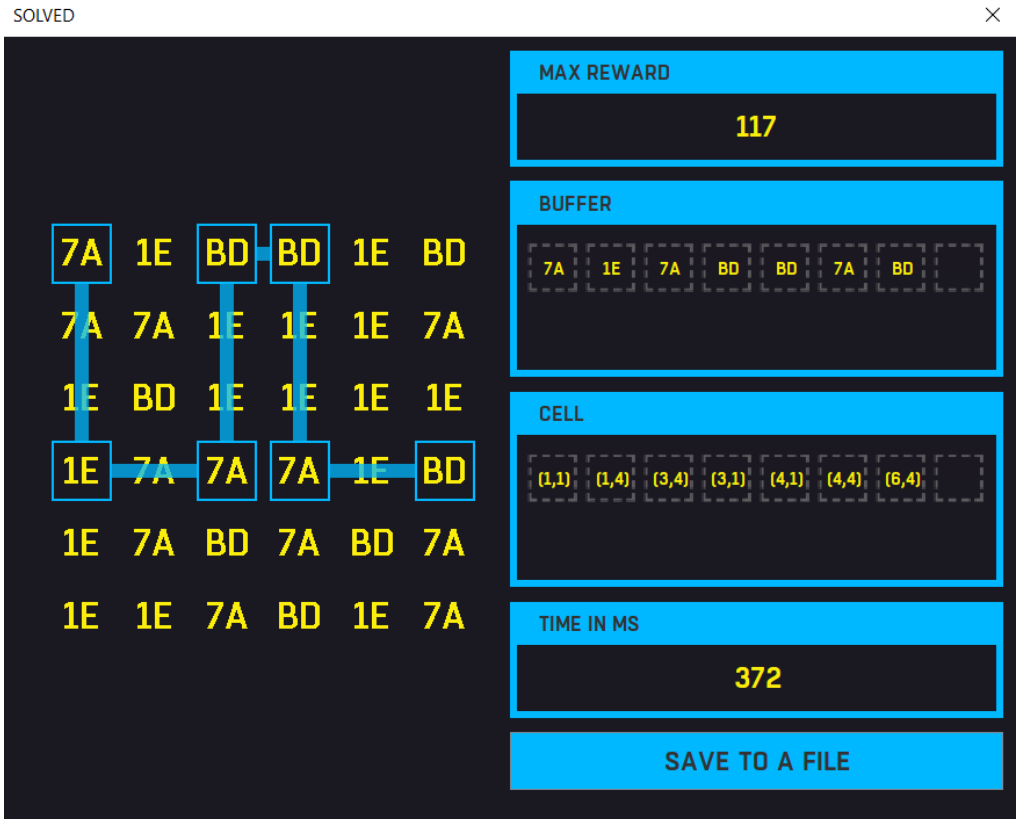
BAGIAN 3: TES INPUT DAN OUTPUT

1. Input

Output

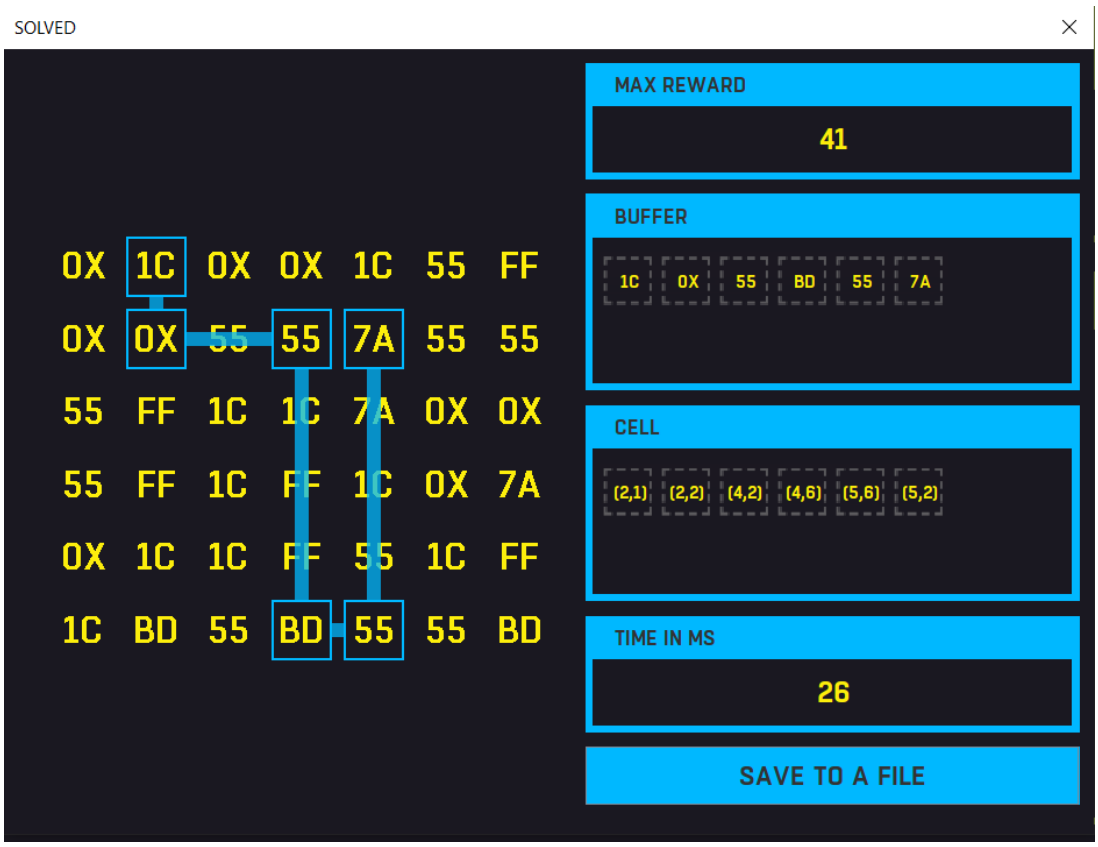
2. Input

Output

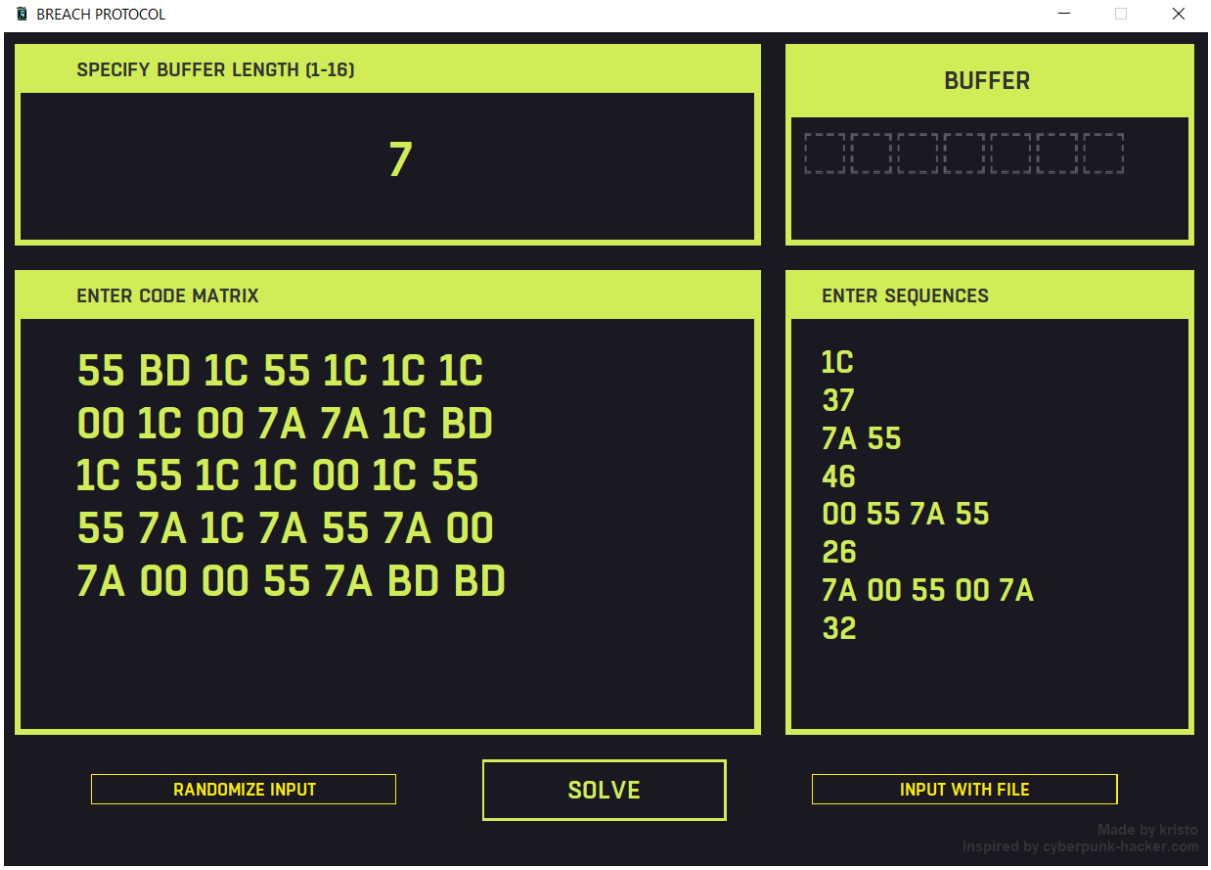


3. Input

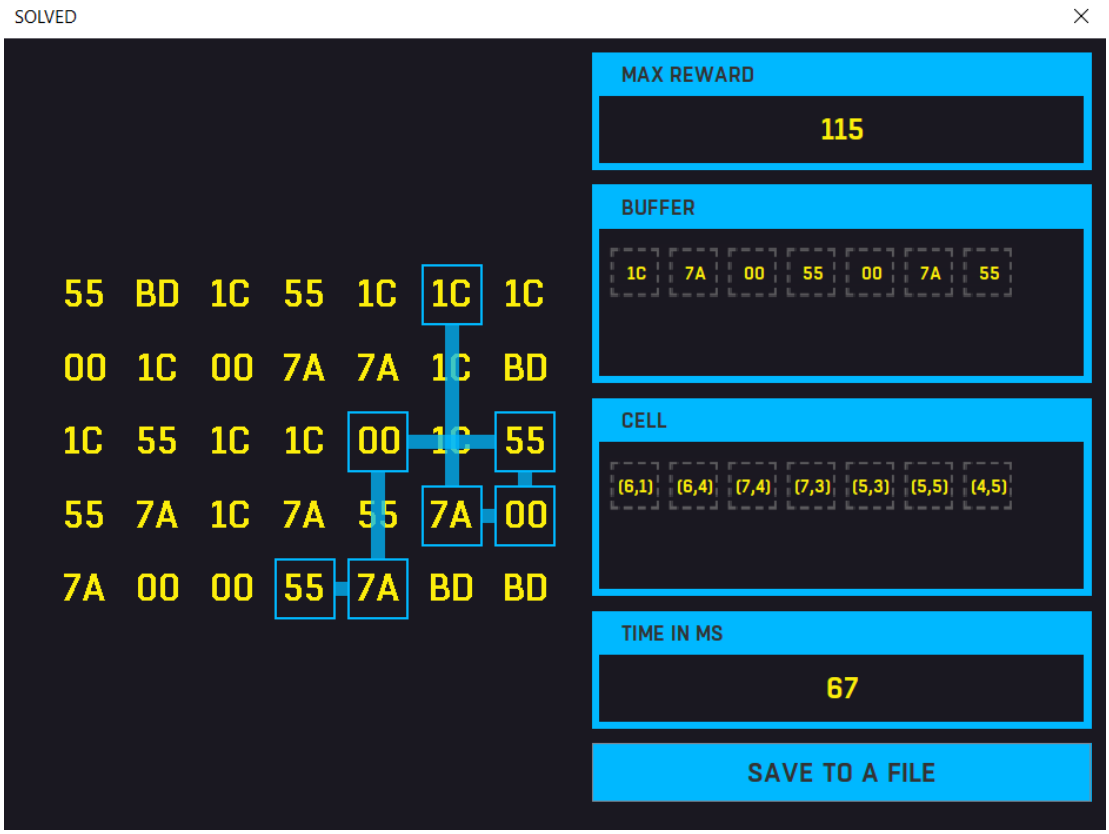
Output



4. Input

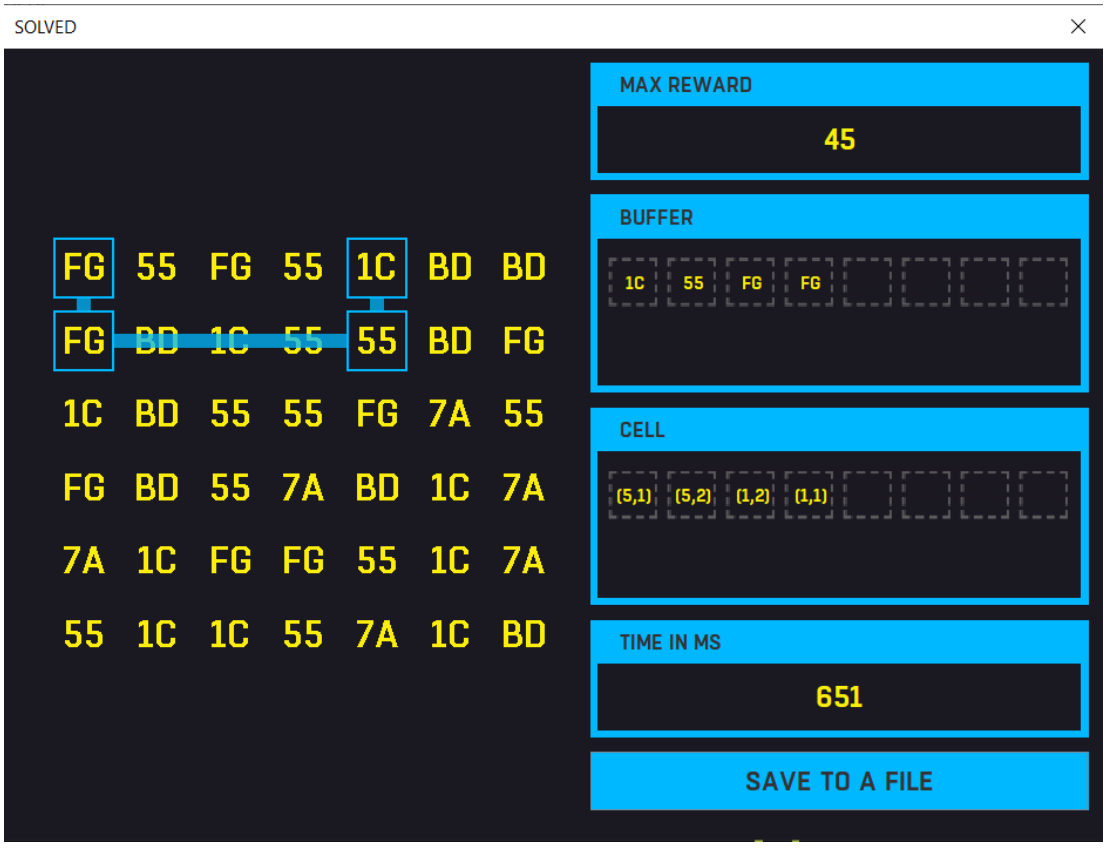


Output



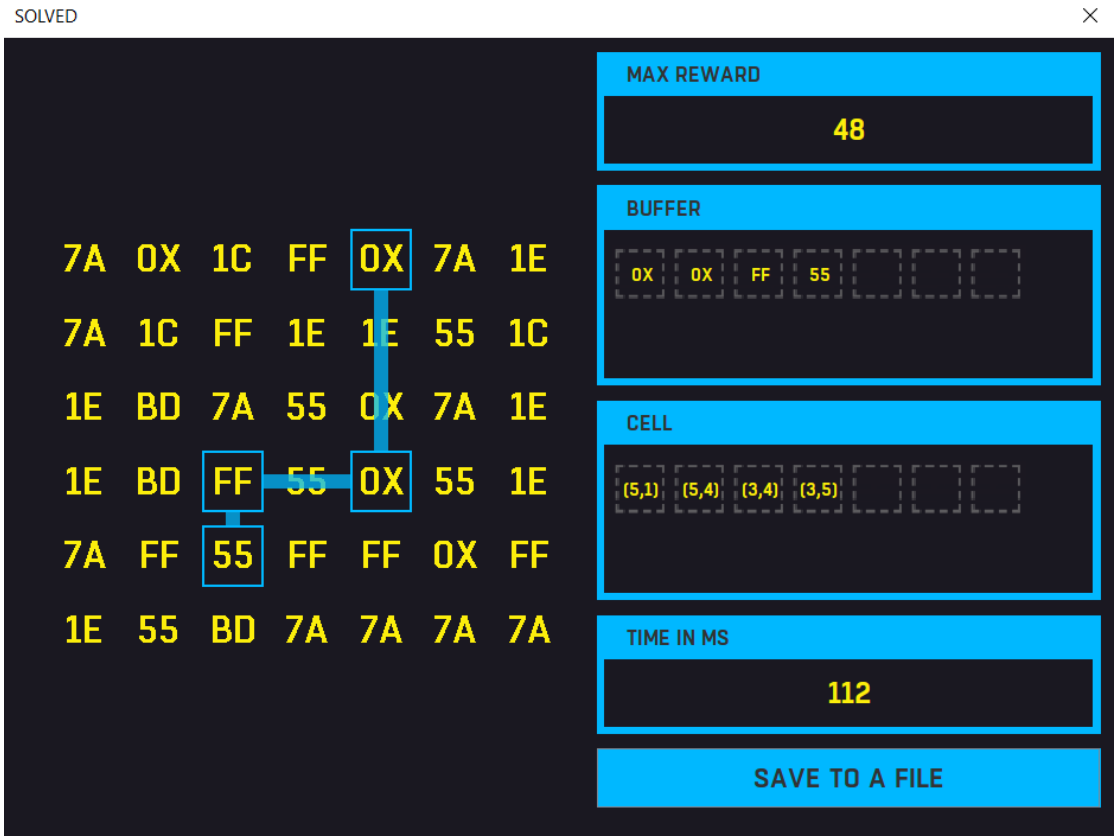
5. Input

Output



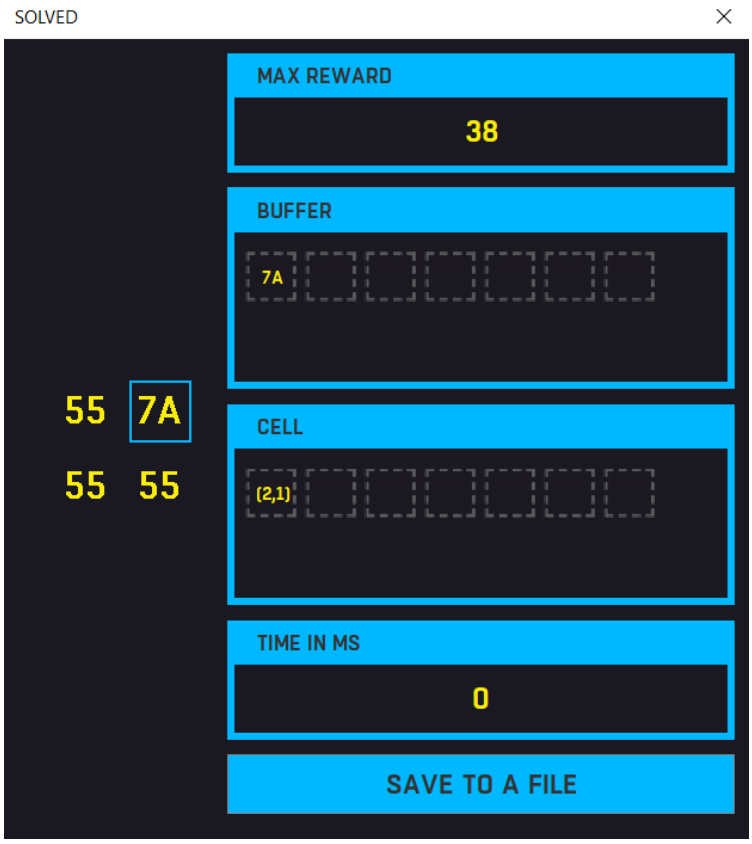
6. Input

Output



7. Input

Output

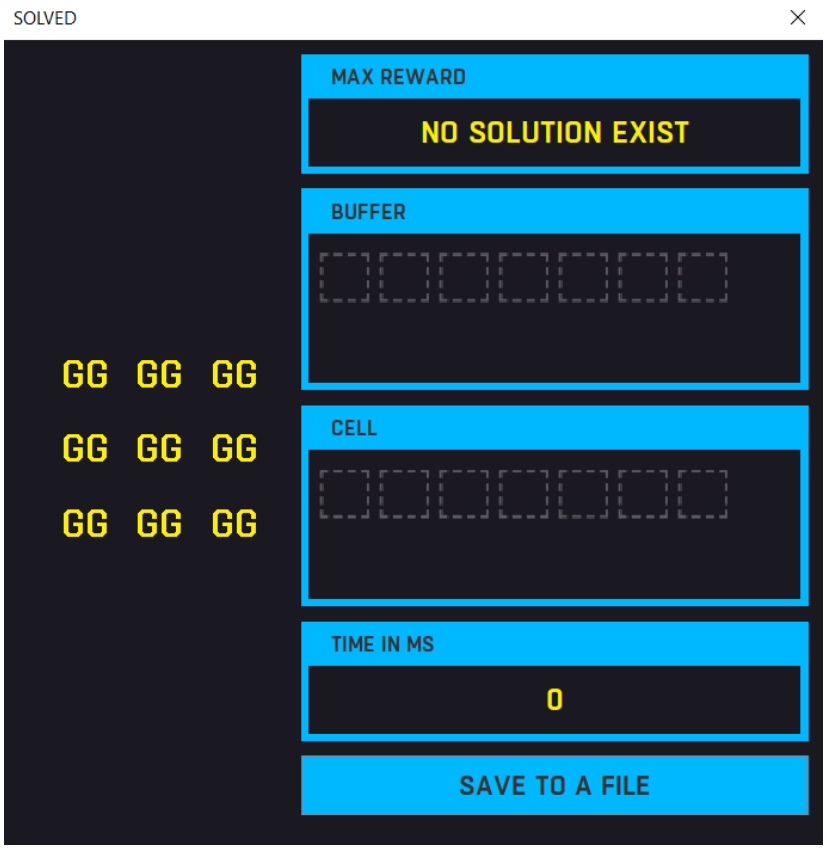


8. Input

Output

9. Input

Output



BAGIAN 4: PRANALA REPOSITORY

https://github.com/qrst0/Tucil1_13522024

BAGIAN 5: LAMPIRAN

https://docs.oracle.com/javase/jp/8/docs/api/javax/swing/JFrame.html

https://www.geeksforgeeks.org/java-swing-jdialog-examples/

https://stackoverflow.com/questions/21210204/java-load-custom-font-file-ttf

https://www.youtube.com/watch?v=Kmgo00avvEw&t=8490s&ab_channel=BroCode

https://www.javatpoint.com/java-gridbaglayout

https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html

https://docs.oracle.com/javase/8/docs/api/javax/swing/JOptionPane.html

https://stackoverflow.com/questions/11093326/restricting-jtextfield-input-to-integers

https://stackoverflow.com/questions/1727840/disable-horizontal-scroll-in-jscrollpane

https://stackoverflow.com/questions/51022662/having-the-windows-ui-display-when-using-jfilechooser

https://stackoverflow.com/questions/33029792/jfilechooser-and-copying-files

https://www.javatpoint.com/Graphics-in-swing

https://stackoverflow.com/questions/10488112/how-do-i-put-graphics-on-a-jpanel

https://stackoverflow.com/questions/12295343/how-to-change-appearance-of-pressed-clicked-selected-button-in-java

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	✓	