

LAPORAN TUGAS KECIL 1
IF2211 - STRATEGI ALGORITMA
“Penyelesaian Cyberpunk 2077 Breach Protocol dengan
Algoritma Brute Force”



Dosen:

Ir. Rila Mandala, M.Eng., Ph.D.
Monterico Adrian, S.T., M.T.

K-03:

Muhammad Fatihul Irhab (13522143)

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II TAHUN 2023/2024

BAB I

LATAR BELAKANG MASALAH

Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video Cyberpunk 2077. Minigame ini merupakan simulasi peretasan jaringan local dari ICE (Intrusion Countermeasures Electronics) pada permainan Cyberpunk 2077.

Komponen pada permainan ini antara lain adalah:

1. Token – terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks – terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.
3. Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
4. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol antara lain:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
5. Setiap sekuens memiliki bobot hadiah atau reward yang variatif.
6. Sekuens memiliki panjang minimal berupa dua token.

BAB II

SPESIFIKASI TUGAS

Buatlah program sederhana dalam bahasa C/C#/C++/Java/Python/JavaScript/Go/PHP yang mengimplementasikan algoritma Brute Force untuk mencari solusi paling optimal permainan Breach Protocol.

Input:

Matriks permainan dapat dihasilkan dengan membaca sebuah berkas berekstensi .txt yang memiliki format sebagai berikut:

```
buffer_size  
matrix_widthmatrix_height  
matrix  
number_of_sequences  
sequences_1  
sequences_1_reward  
sequences_2  
sequences_2_reward  
...  
sequences_n  
sequences_n_reward
```

Contoh:

```
7  
66  
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  
3  
BD E9 1C  
15  
BD 7A BD  
20  
BD 1C BD 55  
30
```

Matriks & sekuens permainan (beserta dengan bobot rewardnya) dapat juga dihasilkan secara otomatis oleh program dengan masukan via CLI berupa *jumlah_token_unik*, *token*, *ukuran_buffer*, *ukuran_matriks*, *jumlah_sekuens*, dan *ukuran_maksimal_sekuens*

(tidak harus terurut). Output untuk fitur ini harus menampilkan Matriks dan sekuens permainan yang digunakan.

Contoh:

```
5  
BD 1C 7A 55 E9  
7  
6 6  
3  
4
```

Output:

1. Bobot reward atau reward maksimal yang dapat digunakan (diisi dengan nol bila tidak ada sekuens yang berhasil didapatkan)
2. Isi dari buffer (bila tidak ada sekuens yang memenuhi maka tidak usah).
3. Koordinat dari setiap token secara terurut (bila tidak ada sekuens yang memenuhi maka tidak usah).
4. Waktu eksekusi program dalam ms (tidak termasuk waktu membaca masukan dan menyimpan solusi).
5. *Prompt* untuk menyimpan solusi dalam sebuah berkas berekstensi .txt
6. Format luaran yang tercetak pada terminal dan berkas berekstensi .txt (untuk berkas berekstensi .txt tidak perlu ada *prompt*) harus sesuai dengan urutan format di bawah ini.

Contoh output:

```
50  
7A BD 7A BD 1C BD 55  
1, 1  
1, 4  
3, 4  
3, 5  
6, 5  
6, 4  
5, 4  
  
300 ms  
  
Apakah ingin menyimpan solusi? (y/n)
```

BAB III

ALGORITMA

Langkah-langkah algoritma dalam program ini :

1. Membaca input dari user, input bisa dari file txt yang nanti akan diproses oleh program dengan fungsi fileToVector() lalu dipindahkan ke variable global. Input juga bisa melalui CLI yang nantinya user akan memberi input yang dibutuhkan setelah itu diproses oleh program melalui fungsi createRandomMatrix(), createRandomSequens(), dan createRandomReward() lalu dipindahkan ke variable global agar siap untuk dilanjutkan ke program utama.
2. Setelah semua data telah siap digunakan, maka dilanjutkan pencarian hasil yang diinginkan dengan menggunakan fungsi getAnswer(), di fungsi ini lah terjadi algoritma brute force, dimana fungsi ini akan melakukan looping berkali kali untuk mencari semua kemungkinan sequens, dengan algoritma loop pertama akan berada di baris pertama, lalu dilanjutkan loop dengan kolom yang sama, lalu dilanjutkan loop dengan baris yang sama, begitu seterusnya hingga maksimal buffer. Di setiap loop akan selalu di cek apakah isi dari buffer ini masih sesuai aturan atau tidak dengan menggunakan fungsi cekKordinat(), jika buffer tidak sesuai aturan maka loop diberhentikan, jika sesuai maka loop akan dilanjutkan, lalu buffer akan di cek dalam buffer tersebut bisa mendapatkan reward sebanyak apa dengan menggunakan fungsi getReward(). Setiap buffer yang telah di cek oleh fungsi getReward maka akan dibandingkan dengan buffer terbesar sebelumnya, nilai yang lebih besar akan menjadi keluaran dari fungsi getAnswer() sebagai jawaban.
3. Setelah hasil jawaban didapatkan, maka jawaban akan ditampilkan melalui CLI, serta terdapat pilihan apakah jawaban ingin disimpan ke file dengan format txt atau tidak, jika ingin disimpan maka akan menggunakan fungsi saveFile().

BAB IV

SOURCE CODE

```
1 //include <iostream>
2 #include <cstdio>
3 #include <filesystem>
4 #include <string>
5 #include <vector>
6 #include <iostream>
7 #include <chrono>
8 #include <fstream>
9 #include <cstdlib>
10 #include <ctime>
11 #include <random>
12
13 #ifdef _WIN32
14 #include <windows.h>
15 #include <direct.h>
16 #define GetCurrentDir _getcwd
17 #define ChangeDir _chdir
18 #define FileSeparator '\\'
19 #define clearCLI "cls"
20 #else
21 #include <unistd.h>
22 #define GetCurrentDir getcwd
23 #define ChangeDir chdir
24 #define FileSeparator '/'
25 #define clearCLI "clear"
26 #endif
27
28 using namespace std;
29
30 // struct
31 struct Answer
32 {
33     int reward, duration, *arrRewardSequences, amountSequences;
34     string buffer;
35     vector<string> arrKordinat, arrToken;
36     vector<vector<string>> matrix;
37 };
38
39 // Fungsi untuk memindahkan isi file ke dalam vektor yang bertipe string
40 vector<string> fileToVector(string fileName)
41 {
42     vector<string> lines;
43     char currentDir[FILENAME_MAX];
44     if (!GetCurrentDir(currentDir, sizeof(currentDir)))
45     {
46         cout << "Gagal mendapatkan path\n";
47     }
48
49     string pathToFile = "../test";
50
51     if (ChangeDir(pathToFile.c_str()) != 0)
52     {
53         cout << "Gagal mengubah path ke test: " << pathToFile << '\n';
54     }
55
56     filesystem::path fullPath = filesystem::absolute(filesystem::path(fileName));
57
58     FILE *inputFile = fopen(fullPath.string().c_str(), "rt");
59
60     if (!inputFile)
61     {
62         cout << "Gagal membuka file: " << fullPath << '\n';
63     }
64
65     char buffer[100];
66     while (fgets(buffer, sizeof(buffer), inputFile))
67     {
68         buffer[strcspn(buffer, "\n")] = '\0';
69         lines.push_back(buffer);
70     }
71     fclose(inputFile);
```

```

72     return lines;
73 }
75
76 void saveFile(Answer answer)
77 {
78     cout << "Apakah ingin menyimpan solusi? (y/n)" << endl;
79     string inputUser;
80     cin >> inputUser;
81
82     if (inputUser == "y" || inputUser == "Y")
83     {
84         cout << "Masukkan nama file yang ingin kamu simpan: ";
85         cin >> inputUser;
86         inputUser = inputUser + ".txt";
87         vector<string> lines;
88         char currentDir[FILENAME_MAX];
89         if (!GetCurrentDir(currentDir, sizeof(currentDir)))
90         {
91             cout << "Gagal mendapatkan path\n";
92         }
93
94         string pathToFile = "../test";
95
96         if (ChangeDir(pathToFile.c_str()) != 0)
97         {
98             cout << "Gagal mengubah path ke test: " << pathToFile << '\n';
99         }
100
101     filesystem::path fullPath = filesystem::absolute(filesystem::path(inputUser));
102
103     ofstream outputFile(fullPath);
104     if (outputFile.is_open())
105     {
106         outputFile << "Matriks yang digunakan" << endl;
107         int rows = answer.matrix.size();

```

```

108         int cols = 0;
109         if (rows > 0)
110         {
111             cols = answer.matrix[0].size();
112         }
113         for (size_t i = 0; i < rows; ++i)
114         {
115             for (size_t j = 0; j < cols; ++j)
116             {
117                 outputFile << answer.matrix[i][j] << " ";
118             }
119             outputFile << endl;
120         }
121         outputFile << endl;
122
123         outputFile << "Sekuens yang digunakan" << endl;
124         for (int i = 0; i < answer.amountSequences; i++)
125         {
126             outputFile << answer.arrToken[i] << endl;
127             outputFile << answer.arrRewardSequences[i] << endl;
128         }
129         outputFile << endl;
130
131         outputFile << "Total bobot hadiah : " << answer.reward << endl;
132         outputFile << "Buffer : " << answer.buffer << endl;
133         int j = 0;
134         for (int i = 0; i < answer.arrKordinat.size(); i++)
135         {
136             int row = answer.arrKordinat[i][0] - '0' + 1;
137             int col = answer.arrKordinat[i][1] - '0' + 1;
138             outputFile << answer.buffer[j] << answer.buffer[j + 1] << " Terletak pada : " << col << ", " << row << endl;
139             j = j + 3;
140         }
141         outputFile << endl;
142         outputFile << "Duarsi vane dibutuhkan : " << answer.duration << " ms" << endl;

```

```

143     |     |     outputFile.close();
144     |     |     cout << "File berhasil disimpan di: " << fullPath << '\n';
145     |     }
146     |     else
147     |     {
148     |         cout << "Gagal membuka file untuk ditulis." << endl;
149     |     }
150     |
151 }
152
153 bool cekKordinat(const string &newKordinat, const vector<string> &listKordinat)
154 {
155     bool cek = false;
156     for (const string &kordinat : listKordinat)
157     {
158         if (kordinat == newKordinat)
159         {
160             cek = true;
161             break;
162         }
163     }
164     return cek;
165 }
166
167 int getReward(int *arrReward, vector<string> arrToken, string buffer)
168 {
169     int reward = -1;
170     for (int k = 0; k < arrToken.size(); k++)
171     {
172         int count = 0;
173         size_t lengthToken = arrToken[k].length();
174         size_t lengthBuffer = buffer.length();
175         if (lengthBuffer >= lengthToken)
176         {
177             for (size_t i = 0; i <= lengthBuffer - lengthToken; i++)
178                 {
179                     bool match = true;
180                     for (size_t j = 0; j < lengthToken; j++)
181                     {
182                         if (buffer[i + j] != arrToken[k][j])
183                         {
184                             match = false;
185                             break;
186                         }
187                     if (match)
188                     {
189                         count = 1;
190                     }
191                 }
192             reward = reward + arrReward[k] * count;
193         }
194     }
195     if (reward > 0)
196     {
197         return reward + 1;
198     }
199     return reward;
200 }
201
202 }
203
204 vector<vector<string>> createRandomMatrix(int col, int row, string token, int amountToken)
205 {
206     vector<string> arrToken;
207     vector<vector<string>> matrix;
208     for (int i = 0; i < amountToken * 3; i++)
209     {
210         string temp1(1, token[i]);
211         string temp2(1, token[i + 1]);
212         string temp = temp1 + temp2;
213         arrToken.push_back(temp);

```

```

178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213

```

```

214     i++;
215     i++;
216 }
217
218 random_device rd;
219 mt19937 gen(rd());
220 uniform_int_distribution<> distribution(0, arrToken.size()-1);
221 matrix.resize(row, std::vector<std::string>(col));
222 for (int i = 0; i < row; i++)
223 {
224     for (int j = 0; j < col; j++)
225     {
226         int random = distribution(gen);
227         matrix[i][j] = arrToken[random];
228     }
229 }
230
231 return matrix;
232 }
233
234 vector<string> createRandomSequens(int maxSequences, int amountSequences, string token, int amountToken)
235 {
236     vector<string> arrToken;
237     vector<vector<string>> matrix;
238     for (int i = 0; i < amountToken * 3; i++)
239     {
240         string temp1(i, token[i]);
241         string temp2(i, token[i + 1]);
242         string temp = temp1 + temp2;
243         arrToken.push_back(temp);
244         i++;
245         i++;
246     }
247
248     random_device rd;
249     mt19937 gen(rd());

```

```

250     uniform_int_distribution<> distribution(2, maxSequences);
251     uniform_int_distribution<> distribution2(0, arrToken.size()-1);
252     vector<string> arrSequens;
253     for (int i = 0; i < amountSequences; i++)
254     {
255         string temp = "";
256         int random = distribution(gen);
257         for (int j = 0; j < random; j++)
258         {
259             int random = distribution2(gen);
260             temp = temp + arrToken[random];
261             temp = temp + " ";
262         }
263         arrSequens.push_back(temp);
264     }
265
266     return arrSequens;
267 }
268
269 void createRandomReward(int amountSequences, int arrRewardSequences[])
270 {
271     random_device rd;
272     mt19937 gen(rd());
273     uniform_int_distribution<> distribution(10, 50);
274     for (int i = 0; i < amountSequences; i++)
275     {
276         int random = distribution(gen);
277         arrRewardSequences[i] = random;
278     }
279 }
280
281 Answer getAnswer(int col, int row, int bufferSize, const vector<vector<string>> &matriks, int *arrReward, vector<string> arrToken)
282 {
283     Answer answer;
284     answer.reward = -1;
285     string tempBuffer;

```



```

358     answer.arrKordinat = tempKordinat;
359 }
360 if (buffSize > 5)
361 {
362     for (int n = 0; n < row; n++)
363     {
364         string tempBuffer5 = tempBuffer4 + " " + matiks[n][m];
365         tempString = to_string(n) + to_string(m);
366         cek = cekKordinat(tempString, tempKordinat);
367         if (!cek)
368         {
369             int reward = getReward(arrReward, arrToken, tempBuffer5);
370             tempKordinat.push_back(tempString);
371             if (reward > answer.reward)
372             {
373                 answer.reward = reward;
374                 answer.buffer = tempBuffer5;
375                 answer.arrKordinat = tempKordinat;
376             }
377         }
378     }
379     if (buffSize > 6)
380     {
381         for (int o = 0; o < col; o++)
382         {
383             string tempBuffer6 = tempBuffer5 + " " + matiks[n][o];
384             tempString = to_string(n) + to_string(o);
385             cek = cekKordinat(tempString, tempKordinat);
386             if (!cek)
387             {
388                 int reward = getReward(arrReward, arrToken, tempBuffer6);
389                 tempKordinat.push_back(tempString);
390                 if (reward > answer.reward)
391                 {
392                     answer.reward = reward;
393                     answer.buffer = tempBuffer6;
394                     answer.arrKordinat = tempKordinat;
395                 }
396             }
397         }
398     }
399     if (buffSize > 7)
400     {
401         for (int p = 0; p < row; p++)
402         {
403             string tempBuffer7 = tempBuffer6 + " " + matiks[p][o];
404             tempString = to_string(p) + to_string(o);
405             cek = cekKordinat(tempString, tempKordinat);
406             if (!cek)
407             {
408                 int reward = getReward(arrReward, arrToken, tempBuffer7);
409                 tempKordinat.push_back(tempString);
410                 if (reward > answer.reward)
411                 {
412                     answer.reward = reward;
413                     answer.buffer = tempBuffer7;
414                     answer.arrKordinat = tempKordinat;
415                 }
416             }
417             if (buffSize > 8)
418             {
419                 for (int q = 0; q < col; q++)
420                 {
421                     string tempBuffer8 = tempBuffer7 + " " + matiks[p][q];
422                     tempString = to_string(p) + to_string(q);
423                     cek = cekKordinat(tempString, tempKordinat);
424                     if (!cek)
425                     {
426                         int reward = getReward(arrReward, arrToken, tempBuffer8);
427                         tempKordinat.push_back(tempString);
428                         if (reward > answer.reward)
429                         {
430                             answer.reward = reward;
431                             answer.buffer = tempBuffer8;
432                             answer.arrKordinat = tempKordinat;
433                         }
434                     }
435                 }
436             }
437         }
438     }
439 }

```

```
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
```

```

502     cin >> inputUser;
503
504     bool inputUserCheck = false;
505     while (!inputUserCheck)
506     {
507         if (inputUser == 1)
508         {
509             inputUserCheck = true;
510
511             vector<string> dataFromFile = fileToVector("tes.txt");
512
513             istringstream issBufferSize(dataFromFile[0]);
514             issBufferSize >> bufferSize;
515
516             istringstream issMatrix(dataFromFile[1]);
517             issMatrix >> matCol >> matRow;
518
519             matrix.resize(matRow, std::vector<std::string>(matCol));
520             for (int i = 0; i < matRow; i++)
521             {
522                 int k = 0;
523                 for (int j = 0; j < matCol * 3 - 1; j++)
524                 {
525                     char temp1 = dataFromFile[2 + i][j];
526                     char temp2 = dataFromFile[2 + i][j + 1];
527                     string temp3(i, temp1);
528                     string temp4(1, temp2);
529                     string temp = temp3 + temp4;
530                     matrix[i][k] = temp;
531                     j++;
532                     j++;
533                     k++;
534                 }
535             }
536
537             istringstream issAmSeq(dataFromFile[matRow + 2]);
538
539             issAmSeq >> amountSequences;
540
541             for (int i = 0; i < amountSequences; i++)
542             {
543                 arrToken.push_back(dataFromFile[matRow + 3 + i * 2]);
544             }
545
546             for (int i = 1; i <= amountSequences; i++)
547             {
548                 istringstream issReward(dataFromFile[matRow + 2 + i * 2]);
549                 issReward >> arrRewardSequences[i - 1];
550             }
551             system("clearCLI");
552         }
553         else if (inputUser == 2)
554         {
555             inputUserCheck = true;
556
557             cout << "Masukkan jumlah token unik : ";
558             cin >> amountToken;
559             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
560             cout << "Masukkan token : ";
561             getline(cin, token);
562             cout << "Masukkan ukuran buffer : ";
563             cin >> bufferSize;
564             cout << "Masukkan ukuran matriks : \n";
565             cout << "Ukuran kolom : ";
566             cin >> matCol;
567             cout << "Ukuran baris : ";
568             cin >> matRow;
569             cout << "Masukkan jumlah sekuens : ";
570             cin >> amountSequences;
571             cout << "Masukkan ukuran maksimal sekuens : ";
572             cin >> maxSequences;
573
574             matrix.resize(matRow, std::vector<std::string>(matCol));

```

```

574     system(clearCLI);
575     matrix = createRandomMatrix(matCol, matRow, token, amountToken);
576     arrToken = createRandomSequens(maxSequences, amountSequences, token, amountToken);
577     createRandomReward(amountSequences, arrRewardSequences);
578   }
579   else
580   {
581     system(clearCLI);
582     cout << "Masukkan kamu salah, silahkan masukkan lagi\n";
583     cout << "1. Masukkan lewat file.txt\n";
584     cout << "2. Masukkan lewat CLI\n";
585     cout << ">> ";
586     cin >> inputUser;
587   }
588 }
589
Answer answer;
auto start_time = chrono::high_resolution_clock::now();
answer = getAnswer(matCol, matRow, bufferSize, matrix, arrRewardSequences, arrToken);
auto end_time = chrono::high_resolution_clock::now();
auto duration = chrono::duration_cast<chrono::milliseconds>(end_time - start_time);
answer.duration = static_cast<int>(duration.count());
answer.matrix = matrix;
answer.amountSequences = amountSequences;
answer.arrToken = arrToken;
answer.arrRewardSequences = arrRewardSequences;
600
cout << "Matriks yang digunakan" << endl;
int rows = answer.matrix.size();
int cols = 0;
if (rows > 0)
{
  cols = answer.matrix[0].size();
}
for (size_t i = 0; i < rows; ++i)
{
  for (size_t j = 0; j < cols; ++j)
  {
    cout << answer.matrix[i][j] << " ";
  }
  cout << endl;
}
cout << endl;

cout << "Sekuens yang digunakan" << endl;
for (int i = 0; i < answer.amountSequences; i++)
{
  cout << answer.arrToken[i] << endl;
  cout << answer.arrRewardSequences[i] << endl;
}
cout << endl;

if (answer.reward == -1)
{
  cout << "Tidak ditemukan sekuen yang memenuhi" << endl;
}
else
{
  cout << "Total bobot hadiah : " << answer.reward << endl;
  cout << "Buffer : " << answer.buffer << endl;
  int j = 0;
  for (int i = 0; i < answer.arrKordinat.size(); i++)
  {
    int row = answer.arrKordinat[i][0] - '0' + 1;
    int col = answer.arrKordinat[i][1] - '0' + 1;
    cout << answer.buffer[j] << answer.buffer[j + 1] << " Terletak pada : " << col << ", " << row << endl;
    j = j + 3;
  }
  cout << endl;
  cout << "Duarsi yang dibutuhkan : " << duration.count() << " ms" << endl;
}

645 saveFile(answer);
646 return 0;
647 }
648
649

```

BAB V

TESTING PROGRAM

Input	Output
7 6 6 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 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	Matriks yang digunakan 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 Sekuens yang digunakan BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30 Total bobot hadiah : 50 Buffer : 7A BD 7A BD 1C BD 55 7A Terletak pada : 1, 1 BD Terletak pada : 1, 4 7A Terletak pada : 3, 4 BD Terletak pada : 3, 5 1C Terletak pada : 6, 5 BD Terletak pada : 6, 3 55 Terletak pada : 1, 3 Duarsi yang dibutuhkan : 79 ms

8
7 7
DD LL CC DD BB BB BB
CC DD LL LL CC CC LL
BB DD CC LL DD BB LL
CC LL DD CC BB DD DD
BB LL CC CC LL DD DD
DD BB DD LL LL BB BB
DD LL LL CC CC DD CC
5
BB BB DD CC
28
LL CC BB BB
5
CC BB CC BB CC
17
LL BB
6
BB BB CC
20

Matriks yang digunakan

DD LL CC DD BB BB BB
CC DD LL LL CC CC LL
BB DD CC LL DD BB LL
CC LL DD CC BB DD DD
BB LL CC CC LL DD DD
DD BB DD LL LL BB BB
DD LL LL CC CC DD CC

Sekuens yang digunakan

BB BB DD CC
28
LL CC BB BB
5
CC BB CC BB CC
17
LL BB
6
BB BB CC
20

Total bobot hadiah : 54

Buffer : LL BB BB DD CC BB BB CC
LL Terletak pada : 2, 1
BB Terletak pada : 2, 6
BB Terletak pada : 6, 6
DD Terletak pada : 6, 4
CC Terletak pada : 1, 4
BB Terletak pada : 1, 3
BB Terletak pada : 6, 3
CC Terletak pada : 6, 2

Duarsa yang dibutuhkan : 2255 ms

3
2 2
7A 55
BD E9
3
55 BD 55
25
7A E9
7
7A 55 7A
18

Matriks yang digunakan

7A 55
BD E9

Sekuens yang digunakan

55 BD 55
25
7A E9
7
7A 55 7A
18

Total bobot hadiah : -1

Buffer :

Duarsa yang dibutuhkan : 0 ms

4
5 5
7A 1C 1C 1C E9
7A BD E9 BD BD
E9 7A E9 1C 55
E9 BD 1C 1C 7A
E9 E9 55 BD 7A
5
7A BD BD
12
E9
12
BD 7A
26
55
5
E9 BD 1C 1C
17

Matriks yang digunakan

7A 1C 1C 1C E9
7A BD E9 BD BD
E9 7A E9 1C 55
E9 BD 1C 1C 7A
E9 E9 55 BD 7A

Sekuens yang digunakan

7A BD BD
12
E9
12
BD 7A
26
55
5
E9 BD 1C 1C
17

Total bobot hadiah : 38

Buffer : 7A E9 BD 7A
7A Terletak pada : 1, 1
E9 Terletak pada : 1, 4
BD Terletak pada : 2, 4
7A Terletak pada : 2, 3

Duarsa yang dibutuhkan : 0 ms

7

10 10

1C 55 55 1C 7A BD E9 BD 1C 55
E9 E9 1C E9 BD BD 7A 55 7A 7A
E9 55 55 1C 55 1C E9 1C 7A E9
1C 1C 1C E9 1C 1C E9 E9 7A
55 55 55 E9 7A 7A E9 55 55 BD
7A 1C 55 E9 55 1C BD 55 7A 55
BD 55 1C 1C E9 BD E9 55 BD 7A
BD 55 BD 1C 1C 7A 7A 1C 7A 55
E9 1C 7A E9 7A 7A E9 BD BD 55
1C BD E9 1C E9 E9 1C E9 55 55

3

BD 1C

1

E9 7A 1C BD

7

1C

9

Matriks yang digunakan

1C 55 55 1C 7A BD E9 BD 1C 55
E9 E9 1C E9 BD BD 7A 55 7A 7A
E9 55 55 1C 55 1C E9 1C 7A E9
1C 1C 1C E9 1C 1C E9 E9 E9 7A
55 55 55 E9 7A 7A E9 55 55 BD
7A 1C 55 E9 55 1C BD 55 7A 55
BD 55 1C 1C E9 BD E9 55 BD 7A
BD 55 BD 1C 1C 7A 7A 1C 7A 55
E9 1C 7A E9 7A 7A E9 BD BD 55
1C BD E9 1C E9 E9 1C E9 55 55

Sekuens yang digunakan

BD 1C

1

E9 7A 1C BD

7

1C

9

Total bobot hadiah : 17

Buffer : 1C E9 7A 1C BD 1C

1C Terletak pada : 1, 1

E9 Terletak pada : 1, 2

7A Terletak pada : 7, 2

1C Terletak pada : 7, 10

BD Terletak pada : 2, 10

1C Terletak pada : 2, 4

Duarsa yang dibutuhkan : 4838 ms

4
5 5
55 BD E9 BD 7A
E9 1C 55 BD E9
55 BD E9 55 BD
E9 55 E9 BD BD
BD 55 7A 55 1C
3
7A BD
42
E9 E9
33
BD BD
42

Matriks yang digunakan
55 BD E9 BD 7A
E9 1C 55 BD E9
55 BD E9 55 BD
E9 55 E9 BD BD
BD 55 7A 55 1C

Sekuens yang digunakan
7A BD
42
E9 E9
33
BD BD
42

Total bobot hadiah : 84
Buffer : 7A BD BD BD
7A Terletak pada : 5, 1
BD Terletak pada : 5, 3
BD Terletak pada : 2, 3
BD Terletak pada : 2, 1

Duarsa yang dibutuhkan : 0 ms

BAB VI

LAMPIRAN

1. https://github.com/muhhul/Tucill_13522143

2.

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		✓