

Laporan Tugas Kecil 1 Strategi Algoritma 2024/2025

Muhammad Izzat Jundy - 13523092 - K-2

Deskripsi Masalah

IQ Puzzler Pro adalah sebuah permainan puzzle yang mengharuskan pemain untuk memasang seluruh piece yang ada ke dalam wadah yang disediakan, sedemikian sehingga wadah dipenuhi oleh piece-piece tersebut.

Algoritma *Brute Force*

Setiap *piece* yang diterima akan disimpan setiap skenario peletakannya. Untuk setiap kondisi *piece*, yaitu seberapa banyak terotasi dan tercerminkan atau tidak, serta untuk setiap posisi *piece* pada wadah. Akan diperoleh jumlah skenario yang disimpan sebanyak $n * m * 8$. Skenario-skenario tersebut kemudian dicocokkan antar-*piece*, hingga terdapat kumpulan skenario *piece* yang cocok satu sama lain. Pencocokan dilakukan dengan menggunakan algoritma kombinasi sederhana. Sebelum pencocokan dilakukan, *piece-piece* yang ada diperiksa apakah luas gabungannya sama dengan luas wadah. Jika iya, maka pencocokan dilakukan, sementara jika tidak, dianggap tidak ada solusi. Kompleksitas algoritma yang digunakan adalah $O((n * m)^p)$.

Source Code

Main.java

```
1  import java.io.File;
2  import java.util.Scanner;
3  import ADT.*;
4  import Function.*;
5
6  public class Main {
7
8      Run | Debug
9      public static void main(String[] args){
10
11          String cmd, s, temp, solutionString, fileName;
12          int n, m, p;
13          long timeStart, timeFinish, duration;
14          Scanner input = new Scanner(System.in), readFile;
15
16          System.out.println(x:"Selamat datang di Tugas Kecil 1 Strategi Algoritma oleh Muhammad Izzat Jundy");
17
18          while(true){
19              System.out.println();
20              System.out.println(x:"Program ini dapat membantumu menemukan solusi dari IQ Puzzler Pro. Ingin mencoba?");
21              System.out.println(x:"1. Input File");
22              System.out.println(x:"0. Keluar");
23              System.out.print(s:"> ");
24              cmd = input.nextLine();
25
26              if(cmd.equals(anObject:"1")){
27
28                  while(true){
29                      System.out.println();
30                      System.out.println(x:"Silakan masukkan nama file, diakhiri dengan \".txt\"");
31                      System.out.print(s:"> ");
32                      fileName = input.nextLine();
33
34                      if(fileName.endsWith(suffix:".txt")){
35
36                          try{
37                              temp = System.getProperty(key:"user.dir");
38
```

```

38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
    while(!temp.endsWith(suffix:"Tucill1_13523092")){
        temp = temp.substring(beginIndex:0, temp.length() - 1);
    }

    File file = new File(temp, "\\test\\input\\" + fileName).getAbsolutePath();
    readFile = new Scanner(file);

    timeStart = System.currentTimeMillis();

    n = readFile.nextInt();
    m = readFile.nextInt();
    p = readFile.nextInt();
    s = readFile.nextLine();
    s = readFile.nextLine();

    Piece solution = new Piece();
    solution = Solve.solve(n, m, p, s, temp + "\\test\\input\\" + fileName);

    timeFinish = System.currentTimeMillis();
    readFile.close();

    System.out.println();
    if(Solve.langkah != 0){
        if(solution.isEmpty()){
            System.out.println(x:"Tidak ditemukan solusi untuk mengisi board!");
        }else{
            System.out.println(x:"Diperoleh solusi sebagai berikut:");
            solution.printPiece();
        }

        duration = timeFinish - timeStart;
        System.out.println();

```

```

74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
        System.out.println("Waktu yang diperlukan: " + duration + " ms");

        System.out.println();
        System.out.println("Banyak kasus yang ditinjau: " + Solve.langkah);

    }else{
        System.out.println(x:"Piece yang ada tidak sesuai dengan board!");

        break;
    }

    while(true){
        System.out.println();
        System.out.println(x:"Apakah Anda ingin menyimpan solusi? (ya/tidak)");
        System.out.print(s:> ");
        cmd = input.nextLine();

        if(cmd.toLowerCase().equals(anObject:"ya")){
            solutionString = Piece.pieceToString(solution);
            Solve.writeStringToFile(solutionString, temp + "\\test\\output\\" + fileName);
            break;
        }else if(cmd.toLowerCase().equals(anObject:"tidak")){
            break;
        }else{
            System.out.println();
            System.out.println(x:"Input tidak valid!");
        }
    }

    break;
} catch (Exception e){
    System.out.println();
    System.out.println("Error message: " + e);
}

```

```

110         }else{
111
112             System.out.println();
113             System.out.println(x:"Input tidak valid!");
114         }
115     }
116 }
117
118 }else if(cmd.equals(anObject:"0")){
119     System.out.println();
120     System.out.println(x:"Terima kasih telah menjajal program ini!");
121     break;
122 }else{
123     System.out.println();
124     System.out.println(x:"Silakan masukkan input berupa angka yang sesuai!");
125 }
126 }
127
128 input.close();
129
130 }
131
132 }

```

ADT/Piece.java

```

1  package ADT;
2
3  import java.util.Arrays;
4
5  import Function.*;
6
7  public class Piece{
8
9      public char matrix[][];
10     public int row;
11     public int col;
12
13     public char MARK = '.';
14
15     public void toPiece(char a[], int row, int col){
16         this.matrix = a;
17         this.row = row;
18         this.col = col;
19     }
20
21     public boolean isEmpty(){
22         int i, j;
23
24         i = 0;
25         while(i<this.row){
26             j = 0;
27             while(j<this.col){
28                 if(this.matrix[i][j] != '.') return false;
29                 j+=1;
30             }
31             i+=1;
32         }
33
34         return true;
35     }
36
37     public static Piece copyOf(Piece p){
38
39         int i, j;

```

```

40     Piece res = new Piece();
41
42     res.row = p.row;
43     res.col = p.col;
44     res.matrix = new char[res.row][res.col];
45
46     i = 0;
47     while(i<res.row){
48         j = 0;
49         while(j<res.col){
50             res.matrix[i][j] = p.matrix[i][j];
51             j+=1;
52         }
53         i+=1;
54     }
55
56     return res;
57 }
58
59 public static Piece emptyBoard(int n, int m){
60
61     int i, j;
62     Piece res = new Piece();
63
64     res.row = n;
65     res.col = m;
66     res.matrix = new char[n][m];
67
68     i = 0;
69     while(i<res.row){
70
71         j = 0;
72         while(j<res.col){
73
74             res.matrix[i][j] = '.';
75

```

```

76             j+=1;
77         }
78
79         i+=1;
80     }
81
82     return res;
83 }
84
85 public void shiftRight(){
86     boolean isOnRight, isOnBottom;
87     int i, j, k, mostLeft;
88
89     isOnRight = false;
90     i = 0;
91     while(i<this.row){
92
93         if(this.matrix[i][this.col-1] != MARK){
94
95             isOnRight = true;
96
97             break;
98         }
99
100         i+=1;
101     }
102
103     isOnBottom = false;
104     i = 0;
105     while(i<this.col){
106
107         if(this.matrix[this.row-1][i] != MARK){
108
109             isOnBottom = true;
110
111             break;

```

```

112     }
113
114     i+=1;
115 }
116
117 if(isOnRight){
118
119     if(!isOnBottom){
120
121         // mendapatkan posisi part dari piece yang paling kiri
122         mostLeft = -1;
123         i = 0;
124         while(i<this.col){
125
126             j = 0;
127             while(j<this.row){
128
129                 if(this.matrix[j][i] != '.'){
130                     mostLeft = i;
131                     break;
132                 }
133
134                 j+=1;
135             }
136
137             if(mostLeft!=-1) break;
138
139             i+=1;
140         }
141
142         // mindahin piece ke paling kiri dan turun 1
143         i = this.row-1;
144         while(i>0){
145
146             j = mostLeft;
147             k = 0;

```

```

148             while(j<this.col){
149
150                 this.matrix[i][k] = this.matrix[i-1][j];
151
152                 j+=1;
153                 k+=1;
154             }
155
156             while(k<this.col){
157
158                 this.matrix[i][k] = MARK;
159
160                 k+=1;
161             }
162
163             i-=1;
164         }
165
166         // ngosongin atasnya
167         i = 0;
168         while(i<this.col){
169             this.matrix[0][i] = '.';
170             i+=1;
171         }
172     }
173
174 }else{
175
176     i = this.row-1;
177     while(i>=0){
178
179         j = this.col-1;
180         while(j>0){
181
182             this.matrix[i][j] = this.matrix[i][j-1];
183

```

```

184         j-=1;
185     }
186
187     i-=1;
188 }
189
190 i = 0;
191 while(i<this.row){
192     this.matrix[i][0] = '.';
193     i+=1;
194 }
195
196 }
197
198 }
199
200 // NOTE: fungsi rotateRight hanya bisa dipakai saat piece berada di pojok kiri atas (awal)
201 public void rotateClockwise(){
202
203     int i, j, mostRight, mostBottom;
204     Piece temp = new Piece();
205     temp.row = this.row;
206     temp.col = this.col;
207     temp.matrix = new char[temp.row][temp.col];
208     i = 0;
209     while(i<this.row){
210         j = 0;
211         while(j<this.col){
212             temp.matrix[i][j] = MARK;
213             j+=1;
214         }
215         i+=1;
216     }
217
218     // mencari panjang x piece
219     mostRight = -1;

```

```

220     i = this.col-1;
221     while(i>=0){
222
223         j = 0;
224         while(j<this.row){
225
226             if(this.matrix[j][i] != MARK){
227                 mostRight = i;
228                 break;
229             }
230
231             j+=1;
232         }
233
234         if(mostRight!=-1) break;
235
236         i-=1;
237     }
238
239     // mencari panjang y piece
240     mostBottom = -1;
241     i = this.row-1;
242     while(i>=0){
243
244         j = 0;
245         while(j<this.col){
246
247             if(this.matrix[i][j] != MARK){
248                 mostBottom = i;
249                 break;
250             }
251
252             j+=1;
253         }
254
255         if(mostBottom!=-1) break;

```

```

256         i--;
257     }
258
259     if(mostRight < this.row && mostBottom < this.col){
260
261         i = 0;
262         while(i<=mostBottom){
263             j = 0;
264             while(j<=mostRight){
265                 temp.matrix[j][i] = this.matrix[mostBottom-i][j];
266                 j++;
267             }
268             i++;
269         }
270
271         i = 0;
272         while(i<this.row){
273             this.matrix[i] = Arrays.copyOf(temp.matrix[i], this.col);
274             i++;
275         }
276     }
277 }
278
279 }
280
281
282 // NOTE: fungsi mirrorX hanya bisa digunakan kalo piece-nya ada di pojok kiri atas (awal)
283 public void mirrorX(){
284
285     int i, j, mostRight;
286     char temp;
287
288     mostRight = -1;
289
290     i = this.col-1;
291     while(i>=0){
292

```

```

293         j = 0;
294         while(j<this.row){
295
296             if(this.matrix[j][i] != MARK){
297                 mostRight = i;
298                 break;
299             }
300
301             j++;
302         }
303
304         if(mostRight!=-1) break;
305
306         i--;
307     }
308
309     i = 0;
310     while(i<mostRight-i){
311
312         j = 0;
313         while(j<this.row){
314             temp = this.matrix[j][i];
315             this.matrix[j][i] = this.matrix[j][mostRight-i];
316             this.matrix[j][mostRight-i] = temp;
317             j++;
318         }
319
320         i++;
321     }
322 }
323
324
325 public void printPiece(){
326
327     int i, j;
328     char temp;
329
330     i = 0;
331     while(i<this.row){
332
333         j = 0;
334         while(j<this.col){
335
336             temp = this.matrix[i][j];

```



```

337         if(temp == 'A' || temp == 'H' || temp == 'O' || temp == 'V') System.err.print("\u001B[31m" + temp + "\u001B[0m");
338         else if(temp == 'B' || temp == 'I' || temp == 'P' || temp == 'W') System.err.print("\u001B[32m" + temp + "\u001B[0m");
339         else if(temp == 'C' || temp == 'J' || temp == 'Q' || temp == 'X') System.err.print("\u001B[33m" + temp + "\u001B[0m");
340         else if(temp == 'D' || temp == 'K' || temp == 'R' || temp == 'Y') System.err.print("\u001B[34m" + temp + "\u001B[0m");
341         else if(temp == 'E' || temp == 'L' || temp == 'S' || temp == 'Z') System.err.print("\u001B[35m" + temp + "\u001B[0m");
342         else if(temp == 'F' || temp == 'M' || temp == 'T') System.err.print("\u001B[36m" + temp + "\u001B[0m");
343         else if(temp == 'G' || temp == 'N' || temp == 'U') System.err.print("\u001B[35m" + temp + "\u001B[0m");
344         else System.out.print(temp);
345
346         j++;
347     }
348
349     System.out.println();
350
351     i++;
352 }
353
354 }
355
356 public void fillPieceWithMark(int startAt){
357     int i;
358     char[] temp = new char[0];
359
360     i = startAt;
361     while(i<this.row){
362         this.matrix[i] = Arrays.copyOf(Solve.fillRowWithMark(temp, this.col), this.col);
363         i++;
364     }
365 }
366
367 public static String pieceToString(Piece piece){
368     int i, j;
369     String res;
370
371     res = "";
372
373     i = 0;
374     while(i<piece.row){

```

```

375         j = 0;
376         while(j<piece.col){
377             res+=piece.matrix[i][j];
378
379             j++;
380         }
381
382         res+="\n";
383
384         i++;
385     }
386
387     return res;
388 }
389
390
391 public static boolean areEqual(Piece a, Piece b){
392
393     if(a.row != b.row || a.col != b.col) return false;
394
395     int i, j;
396
397     i = 0;
398     while(i<a.row){
399
400         j = 0;
401         while(j<a.col){
402             if(a.matrix[i][j] != b.matrix[i][j]) return false;
403
404             j++;
405         }
406
407         i++;
408     }
409
410     return true;
411 }
412
413
414 public int pieceSpace(){
415     int i, j, count;
416
417     count = 0;
418     i = 0;
419     while(i<this.row){

```

```

420         j = 0;
421         while(j<this.col){
422             if(this.matrix[i][j] != '.') count += 1;
423             j++;
424         }
425         i++;
426     }
427
428     return count;
429 }
430

```

Function/Solve.java

```

1  package Function;
2
3  import java.io.File;
4  import java.io.FileOutputStream;
5  import java.util.Arrays;
6  import java.util.Scanner;
7
8  import ADT.*;
9
10 public class Solve {
11
12     public static int langkah;
13
14     private static Piece[][] getPieces(int n, int m, int p, String s, String path){
15         Piece[][] res = new Piece[p][n*m*8];
16         Piece walkPiece = new Piece();
17         char[] signature, temp;
18         int i, j;
19         Scanner readFile;
20         String shifter;
21
22         File file = new File(path).getAbsolutePath();
23         try{
24             readFile = new Scanner(file);
25         }catch (Exception e){
26             System.out.println("Error: " + e);
27             return res;
28         }
29
30
31         i = 0;
32         while(i<p){
33
34             j = 0;
35             while(j<n*m*8){
36
37                 res[i][j] = new Piece();
38                 res[i][j].matrix = new char[n][m];
39                 res[i][j].row = n;
40                 res[i][j].col = m;
41
42                 j++;
43             }
44
45             i++;
46         }
47

```

```

48     shifter = readFile.nextLine();
49     shifter = readFile.nextLine();
50
51     signature = readFile.nextLine().toCharArray();
52     temp = signature;
53
54     temp = Solve.fillRowWithMark(temp, m);
55
56     while(p>0){
57
58         res[p-1][0].matrix[0] = Arrays.copyOf(temp, m);
59
60         i = 1;
61         if(readFile.hasNextLine()){
62
63             temp = readFile.nextLine().toCharArray();
64             temp = Solve.fillRowWithMark(temp, m);
65
66             while(temp[0] == getSignature(signature)){
67
68                 res[p-1][0].matrix[i] = Arrays.copyOf(temp, m);
69
70                 if(readFile.hasNextLine()){
71                     temp = readFile.nextLine().toCharArray();
72                     temp = Solve.fillRowWithMark(temp, m);
73                 }else{
74                     i+=1;
75                     break;
76                 }
77
78                 i+=1;
79             }
80
81         }
82
83         res[p-1][0].fillPieceWithMark(i);
84
85         // piece ditaruh di berbagai posisi
86
87         i = 0;
88         // normal - awal
89         walkPiece = Piece.copyOf(res[p-1][0]);
90         while(i<n*m*1){
91
92             res[p-1][i] = Piece.copyOf(walkPiece);

```

```

93
94             walkPiece.shiftRight();
95             i+=1;
96         }
97
98         // normal - rotate 1
99         walkPiece = Piece.copyOf(res[p-1][0]);
100         walkPiece.rotateClockwise();
101         while(i<n*m*2){
102
103             res[p-1][i] = Piece.copyOf(walkPiece);
104
105             walkPiece.shiftRight();
106             i+=1;
107         }
108
109         // normal - rotate 2
110         walkPiece = Piece.copyOf(res[p-1][0]);
111         walkPiece.rotateClockwise();
112         walkPiece.rotateClockwise();
113         while(i<n*m*3){
114
115             res[p-1][i] = Piece.copyOf(walkPiece);
116
117             walkPiece.shiftRight();
118             i+=1;
119         }
120
121         // normal - rotate 3
122         walkPiece = Piece.copyOf(res[p-1][0]);
123         walkPiece.rotateClockwise();
124         walkPiece.rotateClockwise();
125         walkPiece.rotateClockwise();
126         while(i<n*m*4){
127
128             res[p-1][i] = Piece.copyOf(walkPiece);
129
130             walkPiece.shiftRight();
131             i+=1;
132         }
133
134         // mirror - awal
135         walkPiece = Piece.copyOf(res[p-1][0]);
136         walkPiece.mirrorX();
137         while(i<n*m*5){

```

```

138
139         res[p-1][i] = Piece.copyOf(walkPiece);
140
141         walkPiece.shiftRight();
142         i+=1;
143     }
144
145     // mirror - rotate 1
146     walkPiece = Piece.copyOf(res[p-1][0]);
147     walkPiece.mirrorX();
148     walkPiece.rotateClockwise();
149     while(i<n*m*6){
150
151         res[p-1][i] = Piece.copyOf(walkPiece);
152
153         walkPiece.shiftRight();
154         i+=1;
155     }
156
157     // mirror - rotate 2
158     walkPiece = Piece.copyOf(res[p-1][0]);
159     walkPiece.mirrorX();
160     walkPiece.rotateClockwise();
161     walkPiece.rotateClockwise();
162     while(i<n*m*7){
163
164         res[p-1][i] = Piece.copyOf(walkPiece);
165
166         walkPiece.shiftRight();
167         i+=1;
168     }
169
170     // mirror - rotate 3
171     walkPiece = Piece.copyOf(res[p-1][0]);
172     walkPiece.mirrorX();
173     walkPiece.rotateClockwise();
174     walkPiece.rotateClockwise();
175     walkPiece.rotateClockwise();
176     while(i<n*m*8){
177
178         res[p-1][i] = Piece.copyOf(walkPiece);
179
180         walkPiece.shiftRight();
181         i+=1;
182     }

```

```

183
184         signature = Arrays.copyOf(temp, temp.length);
185         p-=1;
186     }
187
188     readFile.close();
189
190     return res;
191 }
192
193 private static boolean isFit(Piece a, Piece b){
194     int i, j;
195
196     i = 0;
197     while(i<a.row){
198
199         j = 0;
200         while(j<a.col){
201             if(a.matrix[i][j] != '.' && b.matrix[i][j] != '.'){
202                 return false;
203             }
204             j+=1;
205         }
206
207         i+=1;
208     }
209
210     return true;
211 }
212
213 private static boolean isFull(Piece piece){
214     int i, j;
215
216     i = 0;
217     while(i<piece.row){
218         j = 0;
219         while(j<piece.col){
220             if(piece.matrix[i][j] == '.') return false;
221             j+=1;
222         }
223         i+=1;
224     }
225
226     return true;
227 }

```

```

228
229 private static Piece placePiece(Piece board, Piece piece){
230
231     // Prekondisi: board dan piece sudah pasti fit
232
233     int i, j;
234
235     i = 0;
236     while(i < board.row){
237
238         j = 0;
239         while(j < board.col){
240
241             if(piece.matrix[i][j] != '.') board.matrix[i][j] = piece.matrix[i][j];
242
243             j+=1;
244         }
245
246         i+=1;
247     }
248
249     return board;
250 }
251
252 public static char[] fillRowWithMark(char[] s, int m){
253
254     char[] hasil = new char[m];
255     int i;
256
257     i = 0;
258     while(i < s.length){
259         if(s[i] == ' '){
260             hasil[i] = '.';
261         }else{
262             hasil[i] = s[i];
263         }
264         i+=1;
265     }
266
267     while(i < m){
268         hasil[i] = '.';
269         i+=1;
270     }
271
272     return hasil;
273 }

```

```

274
275 private static boolean isProbablySolvable(Piece[][] pieces, int p){
276     // menentukan apakah luas piece-piece-nya sama atau engga dengan wadah
277
278     int i, count;
279
280     count = 0;
281     i = 0;
282     while(i<p){
283         count += pieces[i][0].pieceSpace();
284         i++;
285     }
286
287     if(count == pieces[0][0].row * pieces[0][0].col) return true;
288     else return false;
289 }
290
291 private static Piece permutation(Piece board, Piece[][] pieces, int pieceIndex){
292
293     if(isFull(board)){
294         if(pieceIndex < 0){
295             return board;
296         }else{
297             return Piece.emptyBoard(board.row, board.col);
298         }
299     }else if(pieceIndex < 0){
300         return Piece.emptyBoard(board.row, board.col);
301     }else{
302
303         Piece tempBoard, res;
304         int i;
305
306         i = 0;
307         while(i<board.row*board.col*8){
308             langkah++;
309             if(i>0){
310                 if(i % (board.row * board.col) == 0){
311                     if(Piece.areEqual(pieces[pieceIndex][i], pieces[pieceIndex][0])){
312                         break;
313                     }
314                 }else if(Piece.areEqual(pieces[pieceIndex][i], pieces[pieceIndex][i-1])){
315                     i = ((i / (board.row * board.col)) + 1) * board.row * board.col;
316                     if(i==board.row*board.col*8) break;
317                 }
318             }
319

```

```

320                 if(isFit(board, pieces[pieceIndex][i])){
321                     tempBoard = placePiece(Piece.copyOf(board), pieces[pieceIndex][i]);
322
323                     res = permutation(tempBoard, pieces, pieceIndex-1);
324
325                     if(isFull(res)) return res;
326                 }
327                 i++;
328             }
329
330             return Piece.emptyBoard(board.row, board.col);
331         }
332     }
333 }
334
335 public static Piece solve(int n, int m, int p, String s, String path){
336
337     Piece res = new Piece();
338     Piece[][] pieces = getPieces(n, m, p, s, path);
339
340     langkah = 0;
341     if(!isProbablySolvable(pieces, p)) return Piece.emptyBoard(n, m);
342
343     langkah++;
344     res = permutation(Piece.emptyBoard(n, m), pieces, p-1);
345
346     return res;
347 }
348
349 public static void writeStringToFile(String s, String path){
350
351     try{
352         FileOutputStream output = new FileOutputStream(path);
353
354         output.write(s.getBytes());
355
356         output.close();
357
358         System.out.println();
359         System.out.println("Solusi berhasil tersimpan di " + path);
360     }catch (Exception e){
361         System.out.println();
362         System.out.println("Error: " + e);
363     }
364 }

```

```

365
366     private static char getSignature(char[] signature){
367         int i;
368
369         i = 0;
370         while(i<signature.length){
371             if(signature[i] >= 65 && signature[i] <= 90) return signature[i];
372             i++;
373         }
374
375         return ' ';
376     }
377 }
378

```

Contoh *Input* dan *Output*

Test Case 0 (dari Spesifikasi)

Input

```

1   5 5 7
2   DEFAULT
3   A
4   AA
5   B
6   BB
7   C
8   CC
9   D
10  DD
11  EE
12  EE
13  E
14  FF
15  FF
16  F
17  GGG
18

```

Output

```

Silakan masukkan nama file, diakhiri dengan ".txt"
> 0.txt

Diperoleh solusi sebagai berikut:
AGGGD
AABDD
CCBBE
CFFEE
FFFEF

Waktu yang diperlukan: 148 ms

Banyak kasus yang ditinjau: 2731226

Apakah Anda ingin menyimpan solusi? (ya/tidak)
>

```

Test Case 1 - Semua Huruf, Berhasil

Input

```
1 3 9 26
2 DEFAULT
3 A
4 B
5 C
6 D
7 E
8 F
9 G
10 H
11 I
12 J
13 K
14 L
15 M
16 N
17 O
18 P
19 Q
20 R
21 S
22 T
23 U
24 V
25 W
26 X
27 Y
28 ZZ
29 |
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 1.txt

Diperoleh solusi sebagai berikut:
ABCEFGHI
JKLMNOPQR
STUVWXYZ

Waktu yang diperlukan: 9 ms

Banyak kasus yang ditinjau: 350

Apakah Anda ingin menyimpan solusi? (ya/tidak)
> tidak
```


Test Case 2 - Semua Huruf, Piece Kurang

Input

```
1 3 9 26
2 DEFAULT
3 A
4 B
5 C
6 D
7 E
8 F
9 G
10 H
11 I
12 J
13 K
14 L
15 M
16 N
17 O
18 P
19 Q
20 R
21 S
22 T
23 U
24 V
25 W
26 X
27 Y
28 Z
29
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 2.txt

Piece yang ada tidak sesuai dengan board!

Program ini dapat membantumu menemukan solusi dari IQ Puzzler Pro. Ingin mencoba?
1. Input File
0. Keluar
> 
```

Test Case 3 - Piece dengan Rongga, Berhasil

Input

```
1 3 3 3
2 DEFAULT
3 | A
4 AAA
5 A
6 BB
7 CC
8 |
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 3.txt

Diperoleh solusi sebagai berikut:
BBA
AAA
ACC

Waktu yang diperlukan: 43 ms

Banyak kasus yang ditinjau: 9

Apakah Anda ingin menyimpan solusi? (ya/tidak)
> 
```

Test Case 4 - Piece dengan Rongga, Tidak Ada Solusi

Input

```
1 3 3 3
2 DEFAULT
3 | A
4 AAA
5 A
6 B
7 BB
8 C
9 |
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 4.txt

Tidak ditemukan solusi untuk mengisi board!

Waktu yang diperlukan: 4 ms

Banyak kasus yang ditinjau: 274

Apakah Anda ingin menyimpan solusi? (ya/tidak)
> 
```

Test Case 5 - Perlu Rotasi

Input

```
1 3 3 2
2 DEFAULT
3 A
4 AA
5 A
6 BB
7 B
8 BB
9 
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 5.txt

Diperoleh solusi sebagai berikut:
ABB
AAB
ABB

Waktu yang diperlukan: 4 ms

Banyak kasus yang ditinjau: 8

Apakah Anda ingin menyimpan solusi? (ya/tidak)
> 
```

Test Case 6 - Perlu Mirror

Input

```
1 5 5 6
2 DEFAULT
3 AAA
4 B
5 BBBB
6 CCCCC
7 DDDD
8 F
9 G
10 GGGGG
11
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 6.txt
```

```
Diperoleh solusi sebagai berikut:
```

```
AAABF
BBBBB
CCCCC
DDDDG
GGGGG
```

```
Waktu yang diperlukan: 48 ms
```

```
Banyak kasus yang ditinjau: 111531
```

```
Apakah Anda ingin menyimpan solusi? (ya/tidak)
>
```

Pranala Repository

https://github.com/izzatjundy/Tucil_13523092

| No. | Poin | Ya | Tidak |
|-----|--|----|-------|
| 1 | Program berhasil dikompilasi tanpa kesalahan | ✓ | |
| 2 | Program berhasil dijalankan | ✓ | |
| 3 | Solusi yang diberikan program benar dan mematuhi aturan permainan | ✓ | |
| 4 | Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt | ✓ | |
| 5 | Program memiliki Graphical User Interface (GUI) | | ✓ |
| 6 | Program dapat menyimpan solusi dalam bentuk file gambar | | ✓ |
| 7 | Program dapat menyelesaikan kasus konfigurasi custom | | ✓ |
| 8 | Program dapat menyelesaikan kasus konfigurasi Piramida (3D) | | ✓ |
| 9 | Program dibuat oleh saya sendiri | ✓ | |