

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2024/2025
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute
Force



Disusun oleh :
13523116 - Fityatul Haq Rosyidi

INSTITUT TEKNOLOGI BANDUNG
2025

DAFTAR ISI

DAFTAR ISI	2
BAB I DESKRIPSI TUGAS	3
1.1 IQ Puzzler Pro	3
1.2 Algoritma Brute Force	3
BAB II STRATEGI ALGORITMA	4
2.1 Rancangan Algoritma	4
BAB III IMPLEMENTASI DALAM BAHASA JAVA	5
3.1 Source Code	5
BAB IV TESTING	13
BAB V LAMPIRAN	17
REFERENSI	18

BAB I DESKRIPSI TUGAS

1.1 IQ Puzzler Pro

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan piece (blok puzzle) yang telah tersedia.

Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

1. **Board (Papan)** – Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.
2. **Blok/Piece** – Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh block puzzle berhasil diletakkan.

Tugas anda adalah menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan *algoritma Brute Force*, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

1.2 Algoritma Brute Force

Algoritma Brute Force adalah algoritma penyelesaian suatu masalah berbasis komputasi dengan pendekatan yang lempang(straightforward) dan sederhana. Algoritma Brute Force adalah algoritma natural yang biasanya pertama kali terpikirkan oleh problem solver karena implementasinya yang mudah. Algoritma ini bekerja dengan mencoba semua kemungkinan jalan penyelesaian yang mungkin terjadi tanpa pikir panjang.

Hampir semua persoalan dapat diselesaikan dengan algoritma brute force, namun algoritma ini kurang efisien dan umumnya lambat untuk masukan yang besar.

BAB II STRATEGI ALGORITMA

2.1 Rancangan Algoritma

Tugas ini mengharuskan kami menggunakan Algoritma Brute Force. Jadi saya melakukan penyelesaian dengan algoritma brute force ditambah backtracking pada beberapa bagian.

Berikut adalah alur penyelesaian Puzzle:

1. File input dibaca. jika isi input benar, setiap informasi akan disimpan
2. Akan ada p banyak piece, piece-piece akan di pasang ke board dimulai dari piece pertama.
3. Jika piece pertama dapat masuk, lanjut masukkan piece kedua.
4. Seterusnya begitu sehingga terdapat piece yang tidak bisa masuk ke board.
5. Piece yang tidak bisa masuk ke board akan digeser untuk disesuaikan koordinat posisinya. Misal ukuran piece adalah 2 x 3 satuan, piece akan digeser horizontal 3 kali dan vertikal 2 kali. Untuk seterusnya, ini saya sebut dengan **pergeseran posisi**. Jika ada piece yang berhasil masuk, lanjut ke piece berikutnya.
6. Jika pergeseran posisi belum berhasil, selanjutnya dilakukan **perubahan bentuk**. Piece dapat dirotasikan sebesar 90 derajat berlawanan arah jarum jam dan dapat dicerminkan. Total semua kemungkinan perubahan bentuk ini ada 8. Pada masing2 perubahan bentuk, dilakukan juga pergeseran posisi. Jika ada piece yang berhasil masuk, lanjut ke piece berikutnya.
7. Jika perubahan bentuk masih belum berhasil, akan dilakukan **backtracking**. Mundur ke piece sebelumnya, lalu coba pergeseran posisi dan perubahan bentuk yang lain. Jika ada piece yang berhasil masuk, kembali ke piece selanjutnya. Jika tidak ada, mundur lagi ke piece sebelumnya, dan begitu seterusnya sampai kembali ke piece awal.
8. Pada setiap iterasi, selalu dicek apakah ada sel yang kosong di board. Jika sudah tidak ada sel kosong artinya puzzle terselesaikan, program berhenti dan board ditampilkan.

9. Jika semua kemungkinan sudah dilakukan dan board masih belum terisi penuh, program berhenti dan pesan gagal akan ditampilkan.

BAB III IMPLEMENTASI DALAM BAHASA JAVA

Disini penulis mengimplementasikan penyelesaian persoalan puzzle dengan bahasa pemrograman java. Penulis menerapkan paradigma Object Oriented Programming agar kode program lebih terstruktur dan meningkatkan readability.

3.1 Source Code

```
1  import java.util.*;
2  import java.io.*;
3
4  class Piece {
5      String letter;
6      int row, col;
7      List<List<Integer>> shape;
8
9      // constructor
10     public Piece(String letter, int row, int col, List<List<Integer>> shape) {
11         this.letter = letter;
12         this.row = row;
13         this.col = col;
14         this.shape = shapePadding(shape); // shape di padding dengan 0 pada bagian kosong agar berbentuk persegi /+ panjang
15     }
16
17     // method untuk padding shape
18     private List<List<Integer>> shapePadding(List<List<Integer>> shape) {
19         for (List<Integer> shapeRow : shape) {
20             int delta = this.col - shapeRow.size();
21             for (int i = 0; i < (delta); i++) {
22                 shapeRow.add(0);
23             }
24         }
25         return shape;
26     }
```

```

1 // method untuk menampilkan piece (debugging)
2 public void getPiece(){
3     System.out.println("=====");
4     System.out.println("Letter : " + this.letter);
5     System.out.println("row : " + this.row);
6     System.out.println("col : " + this.col);
7     System.out.println("shape : ");
8     for (int i = 0; i < this.shape.size(); i++){
9         for (int j = 0; j < this.shape.get(i).size(); j++) {
10             System.out.print(this.shape.get(i).get(j) + " ");
11         }
12         System.out.println();
13     }
14 }
15
16 // method untuk merotasi piece
17 private void rotate(){
18     List<List<Integer>> rotated = new ArrayList<>();
19     for (int j = this.col - 1; j > -1; j--){
20         List<Integer> rotatedRow = new ArrayList<>();
21         for (int i = 0; i < this.row; i++){
22             rotatedRow.add(this.shape.get(i).get(j));
23         }
24         rotated.add(rotatedRow);
25     }
26     this.shape = rotated;
27     this.row = rotated.size();
28     this.col = rotated.get(0).size();
29 }
30
31 // method untuk mirroring piece
32 private void mirror(){
33     List<List<Integer>> mirrored = new ArrayList<>();
34     for (int i = 0; i < this.row; i++){
35         List<Integer> mirroredRow = new ArrayList<>();
36         for (int j = this.col - 1; j > -1; j--){
37             mirroredRow.add(this.shape.get(i).get(j));
38         }
39         mirrored.add(mirroredRow);
40     }
41     this.shape = mirrored;
42     this.row = mirrored.size();
43     this.col = mirrored.get(0).size();
44 }

```

```

1
2 // method untuk menghasilkan 8 kemungkinan posisi piece setelah di rotasi dan mirroring
3 public List<List<List<Integer>>> getPositions(){
4     List<List<List<Integer>>> pos = new ArrayList<>();
5     pos.add(this.shape);
6     // rotasi
7     for (int i = 0; i < 3; i++){
8         this.rotate();
9         pos.add(this.shape);
10    }
11    // mirror
12    this.mirror();
13    pos.add(this.shape);
14    // rotasi mirror
15    for (int i = 0; i < 3; i++){
16        this.rotate();
17        pos.add(this.shape);
18    }
19    return pos;
20 }
21 }
22
23 class Board {
24     int row, col;
25     List<List<Integer>> flag; // board dalam bentuk 0 1
26     List<List<String>> shape; // board dalam bentuk huruf
27
28     // constructor
29     public Board(int row, int col){
30         this.row = row;
31         this.col = col;
32
33         List<List<Integer>> flag = new ArrayList<>();
34         List<List<String>> shape = new ArrayList<>();
35         for (int i = 0; i < row; i++){
36             List<Integer> flagRow = new ArrayList<>();
37             List<String> shapeRow = new ArrayList<>();
38             for (int j = 0; j < col; j++){
39                 flagRow.add(0);
40                 shapeRow.add(" ");
41             }
42             flag.add(flagRow);
43             shape.add(shapeRow);
44         }
45         this.flag = flagPadding(flag); // di padding biar aman kalo indexOutOfBounds
46         this.shape = shape;
47     }
48
49     // method untuk padding
50     private List<List<Integer>> flagPadding(List<List<Integer>> flag){
51         int max = Math.max(this.row, this.col);
52         for (int i = 0; i < this.row; i++){
53             for (int j = 0; j < max; j++){
54                 flag.get(i).add(1);
55             }
56         }
57
58         for (int i = 0; i < max; i++){
59             List<Integer> padRow = new ArrayList<>();
60             for (int j = 0; j < this.row + max; j++){
61                 padRow.add(1);
62             }
63             flag.add(padRow);
64         }
65         return flag;
66     }
67 }

```

```

1
2 // method untuk nampilin shape
3 public void showBoardShape(){
4     for (List<String> row : this.shape){
5         for (String c : row) {
6             System.out.print(c + " ");
7         }
8         System.out.println();
9     }
10 }
11
12 // method untuk nampilin flag
13 public void showBoardFlag(){
14     for (List<Integer> row : this.flag){
15         for (Integer c : row) {
16             System.out.print(c + " ");
17         }
18         System.out.println();
19     }
20 }
21
22 // method untuk masukin piece ke board
23 public void install(List<List<Integer>> shape, int rowOffset, int colOffset, String letter){
24     for (int i = 0; i < shape.size(); i++){
25         for (int j = 0; j < shape.get(0).size(); j++){
26             if (shape.get(i).get(j) == 1){
27                 this.flag.get(i + rowOffset).set(j + colOffset, 1);
28                 this.shape.get(i + rowOffset).set(j + colOffset, letter);
29             }
30         }
31     }
32 }
33
34 // method untuk ngeluarin piece dari board
35 public void uninstall(List<List<Integer>> shape, int rowOffset, int colOffset){
36     for (int i = 0; i < shape.size(); i++){
37         for (int j = 0; j < shape.get(0).size(); j++){
38             if (shape.get(i).get(j) == 1){
39                 this.flag.get(i + rowOffset).set(j + colOffset, 0);
40                 this.shape.get(i + rowOffset).set(j + colOffset, " ");
41             }
42         }
43     }
44 }
45 }
46

```




```
1
2 public class Main {
3     // fungsi untuk baca input
4     public static Object[] readFromFile(String filePath) {
5         BufferedReader br = null;
6         try {
7             br = new BufferedReader(new FileReader(filePath));
8
9             // Baris pertama
10            String[] firstLine = br.readLine().split(" ");
11            int m = Integer.parseInt(firstLine[0]);
12            int n = Integer.parseInt(firstLine[1]);
13            int p = Integer.parseInt(firstLine[2]);
14
15            // Baris kedua
16            String mode = br.readLine();
17
18            // List warna
19            String[] colors = {
20                "\u001b[31m", // Merah
21                "\u001b[32m", // Hijau
22                "\u001b[33m", // Kuning
23                "\u001b[34m", // Biru
24                "\u001b[35m", // Magenta
25                "\u001b[36m", // Cyan
26                "\u001b[37m", // Putih
27                "\u001b[91m", // Merah Terang
28                "\u001b[92m", // Hijau Terang
29                "\u001b[93m", // Kuning Terang
30                "\u001b[94m", // Biru Terang
31                "\u001b[95m", // Magenta Terang
32                "\u001b[96m", // Cyan Terang
33                "\u001b[97m", // Putih Terang
34                "\u001b[90m", // Hitam Terang
35                "\u001b[30m" // Hitam
36            };
```

```

1
2 // Baris ketiga
3 List<Piece> pieces = new ArrayList<>();
4 List<List<Integer>> currShape = new ArrayList<>();
5 char currLetter = ' ';
6 int row = 0;
7 int col = 0;
8 int colorIndex = 0;
9 String line;
10 while((line = br.readLine()) != null) {
11     line = line.replaceAll("\\s+", "");
12     if (line.isEmpty()) continue;
13
14     char firstChar = line.replaceAll("^\\s+", "").charAt(0);
15
16     if (currLetter == ' ') {
17         currLetter = firstChar;
18     }
19
20     if (firstChar != currLetter) {
21         // buat piece baru
22         pieces.add(new Piece(colors[colorIndex] + currLetter + "\u001b[0m", row, col, currShape));
23         currLetter = firstChar;
24         currShape = new ArrayList<>();
25         row = 0;
26         col = 0;
27         colorIndex++;
28         colorIndex %= 16;
29     }
30
31     List<Integer> currShapeRow = new ArrayList<>();
32     int currCol = 0;
33     for (char c : line.toCharArray()) {
34         if (c == ' ') {
35             currShapeRow.add(0);
36         } else {
37             currShapeRow.add(1);
38         }
39         currCol++;
40     }
41
42     currShape.add(currShapeRow);
43     col = Math.max(col, currCol);
44     row++;
45 }
46
47 // menambahkan piece terakhir
48 if (!currShape.isEmpty()) {
49     pieces.add(new Piece(colors[colorIndex] + currLetter + "\u001b[0m", row, col, currShape));
50 }
51
52 br.close();
53
54
55
56 return new Object[]{pieces, m, n, p};
57 } catch (IOException | ArrayIndexOutOfBoundsException e) {
58     return null;
59 }
60 }

```

```

1
2 // fungsi untuk menulis output ke file
3 public static void writeToFile(Board board, String filePath){
4     try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))){
5         for (List<String> row : board.shape){
6             for (String s : row){
7                 writer.write(s.replaceAll("\\u001b\\[[0-9;]*m", "") + " ");
8             }
9             writer.newLine();
10        }
11        System.out.println("Output berhasil ditulis ke file: " + filePath);
12    } catch (IOException e) {
13
14        System.out.println("Output Gagal ditulis ke file: " + filePath);
15    }
16 }
17
18 // fungsi untuk mengecek apakah piece bisa masuk ke board
19 public static boolean isMatch(Board board, List<List<Integer>> shape, int rowOffset, int colOffset){
20     try {
21         if ((rowOffset < 0) || (colOffset < 0)) {return false;}
22
23         int cell;
24         for (int i = 0; i < shape.size(); i++) {
25             for (int j = 0; j < shape.get(0).size(); j++){
26                 cell = board.flag.get(rowOffset + i).get(colOffset + j) + shape.get(i).get(j);
27                 if (cell > 1) {return false;}
28             }
29         }
30         return true;
31     } catch (ArrayIndexOutOfBoundsException e) {
32
33         return false;
34     }
35 }
36
37 // fungsi utama, melakukan bruteforcing dan backtracking untuk menemukan solusi
38 public static Object[] bruteForce(Board board, List<Piece> pieces, int count) {
39     for (int rowOffset = 0; rowOffset < board.row; rowOffset++){
40         for (int colOffset = 0; colOffset < board.col; colOffset++){
41             if (board.flag.get(rowOffset).get(colOffset) == 0) { // jika ditemukan space kosong pada board
42                 for (Piece currPiece : pieces){
43                     String letter = currPiece.letter;
44                     for (List<List<Integer>> shape : currPiece.getPositions()){
45                         for (int rowPointer = 0; rowPointer < shape.size(); rowPointer++){
46                             for (int colPointer = 0; colPointer < shape.get(0).size(); colPointer++){
47                                 count++;
48                                 if (isMatch(board, shape, rowOffset + rowPointer, colOffset + colPointer)){ // jika piece bisa masuk ke board
49                                     board.install(shape, rowOffset + rowPointer, colOffset + colPointer, letter);
50
51                                     List<Piece> nextPieces = new ArrayList<>();
52                                     for (Piece pc : pieces){
53                                         if (!pc.equals(currPiece)){
54                                             nextPieces.add(pc); // nextPieces berisi semua piece kecuali piece saat ini
55                                         }
56                                     }
57                                     Object[] result = bruteForce(board, nextPieces, count); // rekurens
58                                     boolean isComplete = (boolean)result[0];
59                                     count = (int)result[1];
60                                     if (isComplete){
61                                         return new Object[]{true, count}; // basis
62                                     }
63                                     board.uninstall(shape, rowOffset + rowPointer, colOffset + colPointer);
64                                 }
65                             }
66                         }
67                     }
68                 }
69                 return new Object[]{false, count};
70             }
71         }
72     }
73     return new Object[]{true, count}; // basis
74 }

```

```

1
2 @SuppressWarnings("unchecked") // tipe list dipastikan sesuai, ga perlu di cek
3 public static void main(String[] args) {
4
5     try {
6
7         Scanner scanner = new Scanner(System.in);
8         System.out.println("Masukkan nama file input : ");
9         String filename = scanner.nextLine().strip();
10
11         Object[] result = readFromFile("test/" + filename);
12
13         List<Piece> pieces = (List<Piece>) result[0];
14         int m = (int) result[1];
15         int n = (int) result[2];
16         int p = (int) result[3];
17
18         // membuat object board
19         Board board = new Board(m, n);
20
21
22         int count = 0;
23
24         long startTime = System.currentTimeMillis();
25         Object[] values = bruteForce(board, pieces, count);
26         long endTime = System.currentTimeMillis();
27
28         boolean isComplete = (boolean) values[0];
29         count = (int) values[1];
30
31         if (isComplete){
32             System.out.println("\nBerhasil mengisi board\n");
33             board.showBoardShape();
34         } else {
35             System.out.println("\nGagal mengisi board\n");
36         }
37
38         System.out.println("\nJumlah percobaan : " + count);
39         System.out.println("Waktu eksekusi : " + (endTime - startTime) + " ms");
40
41         if (isComplete){
42             while (true){
43                 System.out.println("Apakah anda ingin menyimpan solusi?(y/n)");
44                 String ans = scanner.nextLine().strip();
45
46                 if (ans.equals("y") | ans.equals("Y")) {
47                     System.out.print("nama file: ");
48                     String filePath = scanner.nextLine().strip();
49                     writeToFile(board, "output/" + filePath);
50                     break;
51                 } else if (ans.equals("n") | ans.equals("N")){
52                     break;
53                 }
54             }
55
56             scanner.close();
57         }
58
59         } catch ( ArrayIndexOutOfBoundsException | NullPointerException | NumberFormatException e) {
60             System.out.println("\nPastikan nama dan isi file input sudah benar!");
61         }
62     }
63 }
64
65
66
67
68
69
70

```

BAB IV TESTING

No	Test case		Penjelasan
	input	output	
1	<pre>test > input1.txt 1 3 3 2 2 DEFAULT 3 AAA 4 BBB 5 BBB</pre>	<pre>Berhasil mengisi board A A A B B B B B B Jumlah percobaan : 2 Waktu eksekusi : 0 ms</pre>	Kasus sederhana
2	<pre>test > input2.txt 1 5 5 6 2 DEFAULT 3 AA 4 A 5 BBB 6 CC 7 C 8 DDDD 9 D 10 EEE 11 EE 12 FFF 13 FFF</pre>	<pre>Berhasil mengisi board A A B B B A C C F F E E C F F E E D F F E D D D D Jumlah percobaan : 9345 Waktu eksekusi : 6 ms</pre>	Kasus 5 X 5
3	<pre>test > input3.txt 1 5 5 6 2 DEFAULT 3 AAA 4 A 5 BBB 6 CC 7 C 8 DDDD 9 D 10 EEE 11 EE 12 FFF 13 FFF</pre>	<pre>Gagal mengisi board Jumlah percobaan : 13764040 Waktu eksekusi : 933 ms</pre>	Kasus gagal, board tidak berhasil diisi

4	<pre> test > Input4.txt 1 8 8 4 2 DEFAULT 3 AAAAAAAAA 4 AAAAAA 5 AAAAAA 6 AAAAAA 7 A AAAA 8 AAAAAA 9 A AAAA 10 BB B 11 BBBB BBBB 12 CC 13 C 14 DDD </pre>	<pre> Berhasil mengisi board A A A A A A A A A A A A A A C C A A A A A A C C A A A A A A A A A D D D A A A A A A A A A A A A A B A A A A B B B B B B B B B B Jumlah percobaan : 385 Waktu eksekusi : 4 ms </pre>	Kasus bentuk piece aneh
5	<pre> test > Input5.txt 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 </pre>	<pre> Berhasil mengisi board A B B C C A A B D C E E E D D E E F F F G G G F F Jumlah percobaan : 897 Waktu eksekusi : 2 ms </pre>	Kasus berhasil

6	<pre> test > input6.txt 1 8 8 15 2 DEFAULT 3 AAA 4 AA 5 ✓ BBB 6 B 7 CCC 8 ✓ CC 9 D 10 ✓ DDD 11 D 12 E 13 EEEE 14 FFF 15 F 16 F 17 GGGGG 18 HHH 19 II 20 I 21 JJ 22 JJJ 23 K 24 KKK 25 LLL 26 LL 27 ✓ MMM 28 M 29 N 30 NN 31 OOO </pre>	<p>Berhasil mengisi board</p> <pre> A A A B B B C C A A E E B C C C F F F E G H H H F I I E G J J J F I D E G K J J L D D D G K N N L L D M G K K N L L M M M O O O </pre> <p>Jumlah percobaan : 29063 Waktu eksekusi : 21 ms</p>	Kasus dengan banyak piece
7	<pre> test > input7.txt 1 AKU SIGMA </pre>	<p>Masukkan nama file input : input7.txt</p> <p>Pastikan nama dan isi file input sudah benar!</p>	Kasus file yang tidak sesuai

8	<pre> test > input8.txt 1 10 12 8 2 DEFAULT 3 AAAAAAAAAA 4 AAAAAAAAAA 5 AAAA 6 AAAAAA 7 AAAAAA 8 AAAAAAAAAA 9 AAAAAAAAAA 10 AAAAAAAAAA 11 AAA AAA 12 AAA AAA 13 HHH 14 H 15 BBB 16 BBBBB 17 BBBBBB 18 RR 19 KKKKK 20 VW 21 VW 22 III 23 III 24 NNN 25 NNN </pre>	<pre> Berhasil mengisi board R A A A A A A A A A V V R A A A A A A A A A V V H H H B B B A A A A V V H B B B B B A A A A A A B B B B B B A A A A A A K A A A A A A A A A A A K A A A A A A A A A A A K A A A A A A A A I I K A A A N N N A A A I I K A A A N N N A A A I I Jumlah percobaan : 5623548 Waktu eksekusi : 470 ms </pre>	<p>Kasus amogus</p>
---	--	--	---------------------

BAB V LAMPIRAN

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	✓	

Github repository: [FityatulhaqRosyidi/Tucil1_13523116](https://github.com/FityatulhaqRosyidi/Tucil1_13523116)

REFERENSI

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf)

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf)

<https://youtube.com/shorts/MWiPAS3wfGM?feature=shared>

[Java Tutorial](#)