

Laporan Tugas Kecil 1 Strategi Algoritma

IF-2211 2024/2025 IQ Puzzler Pro

Noumisyifa Nabila Nareswari – 13523058 – K01

A. Algoritma

Sebagai konteks dan sedikit latar belakang, secara keseluruhan program dibangun menggunakan paradigma berorientasi objek menggunakan bahasa pemrograman Java. Terdapat tiga kelas yang dibentuk, yaitu *block* (balok), *board* (papan), dan *utils* (utilitas/perkakas). Data bentuk dari balok disimpan menggunakan *array of bitmask integers* dengan setiap baris direpresentasikan dengan nilai *binary*, bit 1 menyatakan bentuk balok sedangkan bit 0 menyatakan ruang kosong di representasi balok. Data bentuk dari papan disimpan dengan dua cara, yaitu *array of bitmask integers* untuk mempermudah algoritma penempatan balok dan juga *2D array of characters* untuk menyimpan bentuk visual papan yang akan ditampilkan di CLI. Pemilihan penggunaan *array of bitmask integers* didasarkan oleh efisiensi waktu dan memori yang ditawarkan oleh operasi *bitwise*. Algoritma pengecekan apakah sebuah balok valid untuk dimasukkan ke posisi tertentu di papan, penempatan balok ke papan, dan rotasi balok menggunakan operasi *bitwise* pada representasi *bitmask* papan dan balok.

Kemudian, inti dari algoritma penyelesaian permasalahan dari permainan IQ Puzzler Pro ini menggunakan strategi algoritma *brute force* yang menggabungkan rekursi permutasi yang memaksa program mengeksplorasi semua kemungkinan balok dengan berbagai kombinasi rotasi balok tersebut (rotasi hanya kelipatan 90 derajat), *backtracking*, dan teknik *pruning*. *Backtracking* digunakan mengeksplorasi kemungkinan lain menggunakan cuplikan kombinasi sebelumnya sedangkan teknik *pruning* digunakan untuk memangkas kombinasi susunan balok yang sudah pasti tidak memenuhi syarat guna meningkatkan efisiensi waktu.

Algoritma *brute force* ini pada intinya terbagi ke tiga fungsi, yaitu fungsi *initialCheck()*, *solve()*, dan fungsi *backtrack()*. Fungsi yang pertama dijalankan adalah fungsi *initialCheck()*. Fungsi ini berguna untuk mengecek apakah memungkinkan sekumpulan balok untuk memenuhi papan sesuai dengan aturan dari segi ukuran balok dan papan. Secara logika, jika jumlah *space* pada balok tidak sama dengan jumlah *space* di papan, maka tidak akan mungkin kombinasi balok ini dapat menyelesaikan bentuk papan. Berikut adalah notasi algoritmik dari fungsi *initialCheck()*

```
function initialCheck(input board: Board, input blocks: array of Block) -> boolean
```

```
{Mengembalikan nilai True jika jumlah ukuran blok sama dengan ukuran papan dan mengembalikan False jika tidak}
```

KAMUS LOKAL

boardSize, sumBlockSizes : integer

ALGORITMA

```
boardSize <- board.getSize()
sumBlockSizes <- 0
for block in blocks do
    sumBlockSizes = sumBlockSizes + block.getSize()
if (sumBlockSizes != boardSize) then
    -> false
else
    -> true
```

Selanjutnya ada fungsi Fungsi solve() yang berfungsi untuk menginisialisasi berjalannya fungsi backtrack().

procedure solve(board: Board, blocks: array of Block)

KAMUS LOKAL

possibilitiesExplored: integer

solved: boolean

ALGORITMA

```
possibilitiesExplored <- 0
solved <- false
if (backtrack(board, blocks)) then
    solve <- True
    output("Solusi ditemukan")
else
    output("Tidak ada solusi")
```

Terakhir adalah inti dari algoritma penyelesaian, yaitu fungsi backtrack() yang menjalankan algoritma rekursi dengan *backtracking* dan *pruning*. Logika dari berjalannya fungsi ini adalah sebagai berikut

1. Fungsi *increment* variabel possibilitiesExplored yang ada pada fungsi Solve() setiap fungsi backtrack() dijalankan. Setelah itu fungsi backtrack() mencari apakah masih ada ruang kosong di papan (board). Jika sudah tidak ada fungsi mengembalikan nilai *true* dan rekursif berhenti di titik ini. Jika masih ada daerah kosong, program akan menyimpan posisi kosong yang paling kiri dan atas di papan ke dalam nilai variabel row dan col kemudian lanjut ke baris kode selanjutnya.

```
function backtrack(input board: Board, input blocks: array of
Block) -> boolean
```

KAMUS LOKAL

```
emptyPos: array of pairs of integers ????????
row,col: integer
```

ALGORITMA

```
possibilitiesExplored <- possibilitiesExplored + 1
emptyPos <- findEmptySpace(board)
if (emptyPos = null) then
    -> true
row <- emptyPos[0]
col <- emptyPos[1]
```

2. Selanjutnya program mencoba untuk menaruh satu persatu balok yang belum dipasang di papan ke posisi yang disimpan dalam variabel row dan col tadi. Selain dari segi posisi, balok juga dicoba untuk diposisikan dalam berbagai rotasi kelipatan 90 derajat. Setelah itu cek kembali apakah penaruhan balok terakhir sudah berhasil memenuhi papan sesuai aturan atau belum, jika sudah fungsi mengembalikan *true* dan jika kombinasi gagal memenuhi papa, fungsi akan melakukan *backtrack* dengan melepas balok dari posisi papan dan mencoba kombinasi lain. Lalu jika semua kombinasi sudah dicoba dan tidak ditemukan kombinasi yang tepat, fungsi mengembalikan nilai *false*.

ALGORITMA PART 2

```
{Coba untuk memasang balok yang belum dipasang}
for(block in blocks) do
    if (not(block.getIsPlaced())) then
        {mencoba rotasi 0, 90, 180, dan 270 derajat}
        rotation traversal [0..3]

        {jika balok dapat dipasang di posisi (row, col)}
        if (isBlockFit(board, block, row, col)) then
            board.addBlock(block, row, col)
            block.setIsPlaced(true)

            {jika papan sudah penuh dan valid}
            if (backtrack(board, blocks)) then
                -> true

            {backtrack, menghapus balok dan coba
```

```

        kombinasi lain}

        board.removeBlock(block,row,col)
        block.setIsPlaced(false)

        block.rotate90() {rootasi balok 90 derajat}

        {jika semua kombinasi telah dicoba dan tidak menemukan jawaban
        yang valid}
        -> false

```

B. Source Code Program

Berikut *source code* Utils.java yang berisikan algoritma *brute force* yang diimplementasikan

```

package Functions;

import java.io.File;
import java.io.PrintWriter;
import java.util.Scanner;

public class Utils {

    private static int possibilitiesExplored = 0;
    private static long startTime;

    public static boolean initialCheck(Board board, Block[]
blocks) {
        int boardSize = board.getSize();
        int sumBlockSizes = 0;

        for (Block block : blocks) {
            sumBlockSizes += block.getSize();
        }

        if (sumBlockSizes != boardSize) {
            System.out.println("\nJumlah titik pada blok tidak
sesuai dengan jumlah space kosong di papan.\n");
            System.out.println("Waktu pencarian: 0 ms\n");
            System.out.println("Banyak kasus ditinjau: 0\n");
            return false;
        } else {
            return true;
        }
    }
}

```

```

    }
}

public static void solve(Board board, Block[] blocks) {
    possibilitiesExplored = 0;
    startTime = System.currentTimeMillis();
    boolean solved = false;

    if (backtrack(board, blocks)) {
        System.out.println("\nSolusi ditemukan!\n");
        solved = true;
        board.printBoard();
    } else {
        System.out.println("\nTidak ada solusi valid.\n");
    }

    long endTime = System.currentTimeMillis(); // End
tracking time
    System.out.println("\nWaktu pencarian: " + (endTime -
startTime) + " ms\n");
    System.out.println("Banyak kasus ditinjau: " +
possibilitiesExplored + "\n");

    // Output handling
    if (solved) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Apakah anda ingin menyimpan
solusi? (ya/tidak): ");

            if (!scanner.hasNextLine()) {
                break;
            }

            String save =
scanner.nextLine().trim().toLowerCase().split(" ")[0];

            if (save.equals("ya")) {
                System.out.print("Masukkan nama file (tanpa
.txt): ");

                if (!scanner.hasNextLine()) {

```

```

        break;
    }

    String filename = scanner.nextLine().trim();

    File solutionFolder = new
File("test/Solutions/");
    if (!solutionFolder.exists()) {
        solutionFolder.mkdirs(); // Make the
folder if not exists yet
    }

    File solutionFile = new File(solutionFolder,
filename + ".txt");

    try (PrintWriter writer = new
PrintWriter(solutionFile)) {
        // Write the visualBoardd to the file
        writer.println("Solusi IQ Puzzler Pro:");
        for (char[] row : board.getVisualBoard())
        {
            writer.println(new String(row));
        }

        System.out.println("Solusi berhasil
disimpan di test/Solutions/" + filename + ".txt");
    } catch (Exception e) {
        System.out.println("Gagal menyimpan
solusi: " + e.getMessage());
    }

    break;
}
else if (save.equals("tidak")) {
    System.out.println("Solusi tidak disimpan.");
    break;
}
else {
    System.out.println("Input tidak valid. Input
hanya dapat 'ya' atau 'tidak', ulang.\n");
}
}

scanner.close();

```

```

    }
}

private static boolean backtrack(Board board, Block[] blocks)
{
    possibilitiesExplored++;

    // Find the first empty space that's most top and most
left
    int[] emptyPos = findEmptySpace(board);
    if (emptyPos == null) {
        return true; // All spaces filled = Solution found!
    }
    int row = emptyPos[0], col = emptyPos[1];

    // Try placing each unused block
    for (Block block : blocks) {
        if (!block.getIsPlaced()) { // If block is not yet
placed
            // Try placing in 4 different rotations (0, 90,
180, 270)
            for (int rotation = 0; rotation < 4; rotation++)
            {
                if (isBlockFit(board, block, row, col)) {
                    board.addBlock(block, row, col);
                    block.setIsPlaced(true);

                    if (backtrack(board, blocks)) {
                        return true; // Found a solution
                    }

                    // Backtrack: Remove the block and try
another
                    board.removeBlock(block, row, col);
                    block.setIsPlaced(false);
                }
                block.rotate90();
            }
        }
    }
    return false;
}

```

```

        // use visualBoard instead because the checking more
        computationally cheap than using bitmaskBoard
        private static int[] findEmptySpace(Board board) {
            char[][] visualBoard = board.getVisualBoard();
            for (int r = 0; r < board.getRow(); r++) {
                for (int c = 0; c < board.getColumn(); c++) {
                    if (visualBoard[r][c] == ' ') {
                        return new int[]{r, c};
                    }
                }
            }
            return null;
        }

        public static boolean isBlockFit(Board board, Block block,
            int rowCoord, int colCoord) {
            int[] bitmaskBoard = board.getBitmaskBoard();
            int[] bitmaskBlock = block.getBitmaskBlock();
            int boardCol = board.getColumn();
            int boardRow = board.getRow();
            int blockCol = block.getColumn();
            int blockRow = block.getRow();

            // Check if block goes out of bounds
            if (blockRow + rowCoord > boardRow || blockCol + colCoord
                > boardCol) {
                return false;
            } else {
                // Shift the block left by colCoord
                for (int i = 0; i < blockRow; i++) {
                    int shiftedBlockRow = bitmaskBlock[i] <<
                        (boardCol - blockCol - colCoord);

                    // Check if the shifted block row overlaps with
                    the existing block inside the board
                    if ((bitmaskBoard[rowCoord + i] &
                        shiftedBlockRow) != 0) {
                        return false; // Overlap detected, block
                        cannot be placed
                    }
                }
            }
        }

```



```

        return true; // No overlap, block fits
    }
}
}

```

C. Tangkapan Layar Hasil Pengetesan

1. Test Case 1

Pada test case ini, input normal (tidak ada kesalahan penulisan dan format), tipe papan DEFAULT, dan memiliki jawaban.

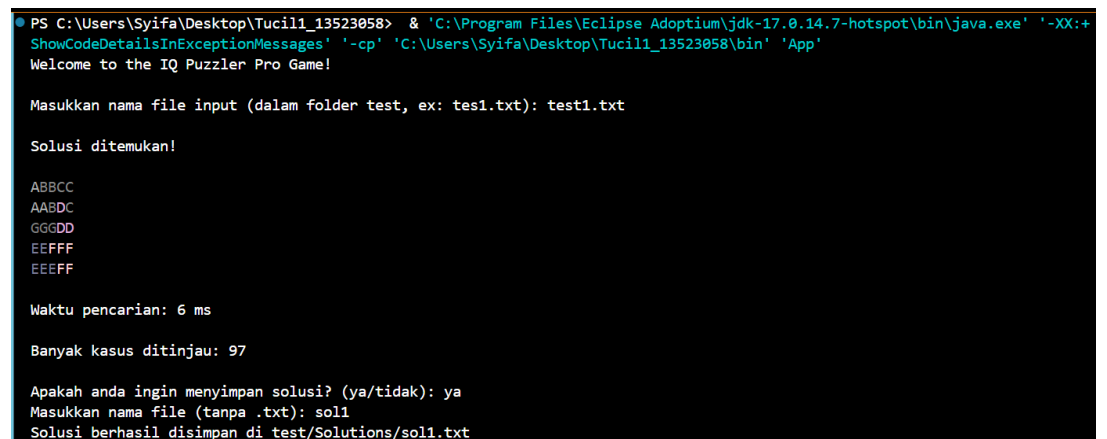
test1.txt

```

5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG

```

Screenshot program



```

PS C:\Users\Syifa\Desktop\Tucil1_13523058> & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+
ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_13523058\bin' 'App'
Welcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test1.txt

Solusi ditemukan!

ABBCC
AABDC
GGGDD
EEEFF
EEEFF

Waktu pencarian: 6 ms

Banyak kasus ditinjau: 97

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file (tanpa .txt): sol1
Solusi berhasil disimpan di test/Solutions/sol1.txt

```

soll.txt

Solusi IQ Puzzler Pro:

ABBCC

AABDC

GGGDD

EEFFF

EEEFF

2. Test Case 2

Pada test case ini, input normal (tidak ada kesalahan penulisan dan format), tipe papan DEFAULT, dan tidak memiliki jawaban.

test2.txt

5 4 7

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

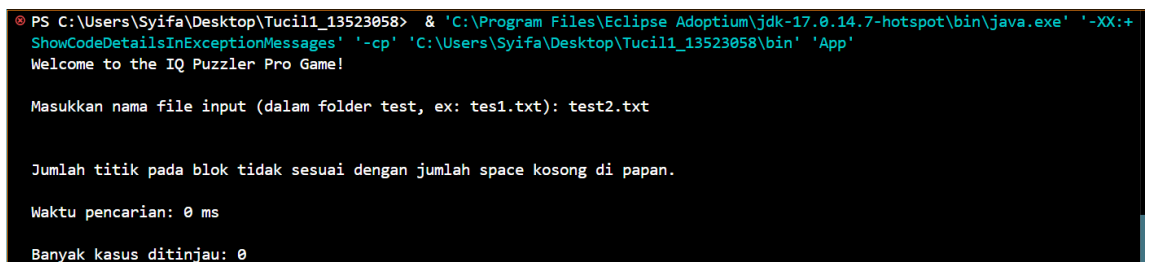
FF

FF

F

GGG

Screenshot program



```
PS C:\Users\Syifa\Desktop\Tucil1_13523058> & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_13523058\bin' 'App'
Welcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test2.txt

Jumlah titik pada blok tidak sesuai dengan jumlah space kosong di papan.

Waktu pencarian: 0 ms

Banyak kasus ditinjau: 0
```

3. Test Case 3

Pada test case ini, input valid namun terdapat baris kosong dan alfabet *lowercase*, tipe papan DEFAULT, dan tidak memiliki jawaban.

test3.txt

5 5 7

DEFAULT

A

AA

b

BB

C

CC

D

DD

EE

ee

E

FF

FF

F

GGG

screenshot program

```
PS C:\Users\Syifa\Desktop\Tucil1_13523058> & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_13523058\bin' 'App'
Welcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test3.txt

Solusi ditemukan!

ABBCC
AABDC
GGGDD
EEFFF
EEFFF

Waktu pencarian: 4 ms

Banyak kasus ditinjau: 97

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file (tanpa .txt): sol3
Solusi berhasil disimpan di test/Solutions/sol3.txt
```

sol3.txt

Solusi IQ Puzzler Pro:

ABBCC
AABDC
GGGDD
EEFFF
EEEEF

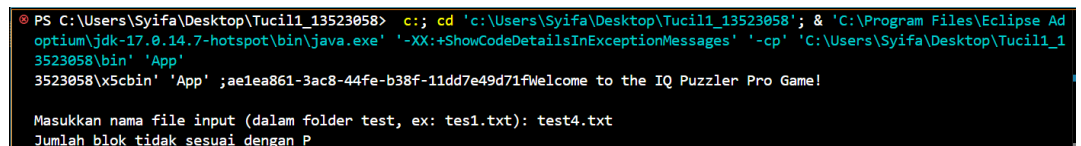
4. Test Case 4

Pada test case ini, nilai P tidak sesuai dengan jumlah blok yang ada pada input.

test4.txt

5 5 6
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG

screenshot program



```
PS C:\Users\Syifa\Desktop\Tucil1_13523058> c:; cd 'c:\Users\Syifa\Desktop\Tucil1_13523058'; & 'C:\Program Files\Eclipse Ad
optium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_1
3523058\bin' 'App'
3523058\x5cbin' 'App' ;ae1ea861-3ac8-44fe-b38f-11dd7e49d71fWelcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test4.txt
Jumlah blok tidak sesuai dengan P
```

5. Test Case 5

Pada test case ini, input valid, tipe papan CUSTOM, dan memiliki jawaban.

test5.txt

```

5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXXX
.XXXXX.
...X...
  A
AAAAA
BB
BBB
CC
C
D
DD
D
E

```

screenshot program

```

PS C:\Users\Syifa\Desktop\Tucil1_13523058> c::; cd 'c:\Users\Syifa\Desktop\Tucil1_13523058'; & 'C:\Program Files\Eclipse Ad
optium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_1
3523058\bin' 'App'
3523058\x5cbin' 'App' ;ae1ea861-3ac8-44fe-b38f-11dd7e49d71fWelcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test5.txt

Bentuk papan CUSTOM:
... ..
. . .
. . .

. . .
... ..

Solusi ditemukan!

...E...
.AAAAA.
BBBADDD
.BBCCD.
...C...

Waktu pencarian: 1 ms

Banyak kasus ditinjau: 186

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file (tanpa .txt): sol5
Solusi berhasil disimpan di test/Solutions/sol5.txt

```

sol5.txt

Solusi IQ Puzzler Pro:

```

...E...
.AAAAA.
BBBADDD
.BBCCD.
...C...

```

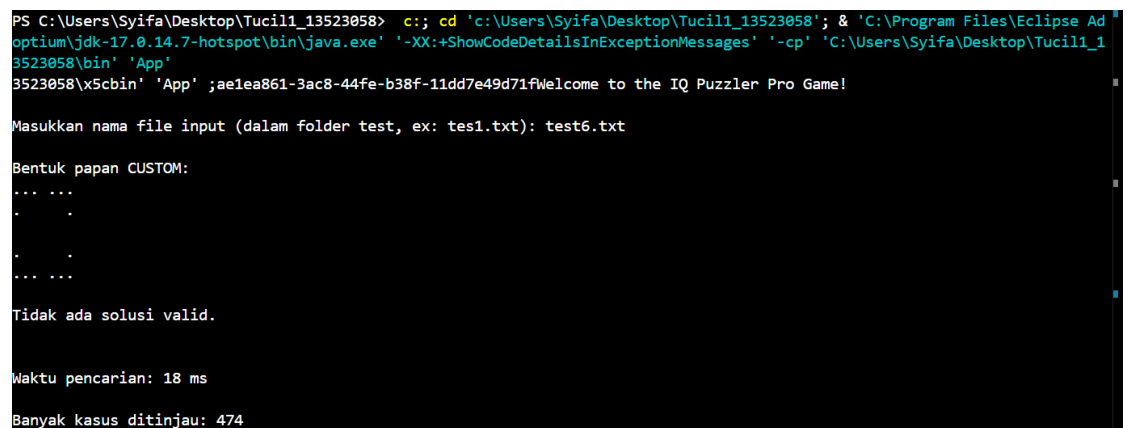
6. Test Case 6

Pada test case ini, input valid, tipe papan CUSTOM, dan tidak memiliki jawaban.

test6.txt

```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
C
D
EEE
E
```

screenshot program



```
PS C:\Users\Syifa\Desktop\Tucil1_13523058> c.; cd 'c:\Users\Syifa\Desktop\Tucil1_13523058'; & 'C:\Program Files\Eclipse Ad
optium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_1
3523058\bin' 'App'
3523058\x5cbin' 'App' ;ae1ea861-3ac8-44fe-b38f-11dd7e49d71fWelcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test6.txt

Bentuk papan CUSTOM:
... ..
. .
. .
... ..

Tidak ada solusi valid.

Waktu pencarian: 18 ms

Banyak kasus ditinjau: 474
```

7. Test Case 7

Pada test case ini, input normal (tidak ada kesalahan penulisan dan format), tipe papan DEFAULT, dan memiliki jawaban.

test7.txt

5 11 12

DEFAULT

A

AA

A

BB

B

B

B

C

CC

CC

D

DD

D

D

E

EE

EE

F

FF

F

GG

G

GG

HH

H

H

I

II

J

J

JJ

K

K

KK

K

L

LL

LL

Screenshot program

```

PS C:\Users\Syifa\Desktop\Tucil1_13523058> c.;; cd 'c:\Users\Syifa\Desktop\Tucil1_13523058'; & 'C:\Program Files\Eclipse Ad
optium\jdk-17.0.14.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Syifa\Desktop\Tucil1_1
3523058\bin' 'App'
Welcome to the IQ Puzzler Pro Game!

Masukkan nama file input (dalam folder test, ex: tes1.txt): test7.txt

Solusi ditemukan!

ALLCCFFFKBB
AALLCCFDKKB
IAJLCGGDDKB
IIJEEEGDHKB
JJJEEEGDHHH

Waktu pencarian: 973 ms

Banyak kasus ditinjau: 1592731

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file (tanpa .txt): sol7
Solusi berhasil disimpan di test/Solutions/sol7.txt

```

sol7.txt

Solusi IQ Puzzler Pro:

```

ALLCCFFFKBB
AALLCCFDKKB
IAJLCGGDDKB
IIJEEEGDHKB
JJJEEEGDHHH

```

D. Repository

Berikut terlampir link untuk repositori Github program:

https://github.com/numshv/Tucil1_13523058

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	✓	