

**TUGAS KECIL 3**  
**IF2211 STRATEGI ALGORITMA**



**Benedictus Nelson**  
**13523150**

Dosen Pengampu :  
Dr. Nur Ula Maulidevi, S.T, M.Sc.  
Dr. Ir. Rinaldi Munir, M.T.  
Monterico Adrrian, S.T, M.T.

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**JL. GANESA 10, BANDUNG 40132**  
**2025**

# DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB 1 ALGORITMA.....</b>	<b>3</b>
1.1 Main Idea.....	3
1. Representasi Papan Permainan.....	3
2. Membaca Konfigurasi Papan.....	3
3. Pencarian Solusi: Pathfinding.....	3
4. Heuristic.....	3
1.2 Langkah Terinci.....	3
1. Parse Input.....	3
2. Setup Data.....	4
3. Fungsi solve().....	4
4. Output.....	4
<b>BAB 2 PROGRAM.....</b>	<b>5</b>
1. Main.java.....	5
2. Board.java.....	7
3. BoardParser.java.....	12
4. Heuristics.java.....	15
5. Solver.java.....	17
6. State.java.....	18
7. Move.java.....	19
8. Piece.java.....	20
9. Printer.java.....	20
<b>BAB 3 EKSPERIMEN.....</b>	<b>23</b>
<b>LAMPIRAN.....</b>	<b>45</b>
Link Repository.....	45
Checklist.....	45

# BAB 1 ALGORITMA

## 1.1 Main Idea

### 1. Representasi Papan Permainan

- Setiap sel papan permainan dapat berisi:
  - Karakter untuk menyatakan sel kosong,
  - Huruf kapital untuk merepresentasikan kendaraan/piece,
  - P sebagai primary piece yang harus dikeluarkan,
  - K sebagai pintu keluar yang terletak di dinding papan.

### 2. Membaca Konfigurasi Papan

- Program membaca konfigurasi papan dari berkas .txt. Baris pertama menyatakan dimensi papan (rows cols), diikuti jumlah piece selain P, lalu representasi papan berupa rows baris. Setiap piece dikenali dari karakter uniknya.

### 3. Pencarian Solusi: Pathfinding

- Program menggunakan pendekatan pencarian jalur (pathfinding) untuk menggerakkan P dari posisinya saat ini hingga mencapai K. Tiga algoritma yang digunakan:
  - **Uniform Cost Search (UCS)**: menjelajah berdasarkan total cost minimum.
  - **Greedy Best-First Search** : menjelajah berdasarkan estimasi heuristic.
  - **A\* Search** : gabungan cost sejauh ini ( $g(n)$ ) + heuristic ( $h(n)$ ).

Primary piece hanya bisa bergerak sesuai orientasinya (horizontal = kiri-kanan, vertikal = atas-bawah). Setiap langkah menghasilkan state baru yang disimpan untuk dijelajahi.

### 4. Heuristic

- Untuk Greedy dan A\*, tersedia 3 jenis heuristic:
  - **Nol (ZERO)** : tidak menggunakan perkiraan.
  - **Blocking Count** : menghitung jumlah piece yang menghalangi jalan keluar.
  - **Manhattan Distance** : jarak P ke K berdasarkan grid.

## 1.2 Langkah Terinci

### 1. Parse Input

- Membaca ukuran papan rows, cols, dan jumlah piece.
- Menginisialisasi board sebagai array `char[rows][cols]`.
- Mengisi peta piece: posisi setiap kendaraan dicatat dan dicocokkan orientasinya (horizontal/vertikal).
- Memastikan semua piece segaris dan kontigu, tidak membentuk huruf L.

## 2. Setup Data

- State menyimpan snapshot board, posisi primary piece, langkah sejauh ini (g), dan pointer ke parent state.
- PriorityQueue digunakan untuk menyimpan state yang akan dijelajahi.
- visited adalah HashSet untuk menyimpan konfigurasi papan yang sudah dikunjungi agar tidak eksplorasi ulang.

## 3. Fungsi solve()

- Dipanggil dari Main.java setelah membaca input dan pilihan algoritma.
- Menyusuri state-space dari konfigurasi awal, menggunakan algoritma yang dipilih.
- Jika primary piece mencapai K, maka solusi ditemukan.
- Jika tidak ada lagi state yang bisa dijelajahi, program menyatakan "Tidak ada solusi."

## 4. Output

- Menampilkan urutan papan dari awal hingga akhir.
- Setiap langkah menyertakan informasi gerakan: [piece]-[arah].
- Mewarnai P, K, dan piece yang bergerak di terminal (ANSI color).
- Menyimpan output solusi ke file .txt.
- Statistik akhir: jumlah node yang dikunjungi dan waktu eksekusi program (dihitung dari System.nanoTime() sebelum dan sesudah pemrosesan).

---

### Catatan:

Tidak digunakan heuristic tambahan seperti penalti crossing atau look-ahead blocking. Pencarian dilakukan secara eksplisit, dan efisiensi sangat dipengaruhi pilihan heuristic dan urutan eksplorasi state. Kombinasi piece dapat sangat bervariasi dan mempengaruhi kedalaman pencarian solusi.

## BAB 2 PROGRAM

### 1. Main.java

#### Main.java

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws IOException {
        if (args.length == 0) {
            System.out.println("Penggunaan : java Main
<input_file.txt>");
            return;
        }
        String inFile = args[0];
        BoardParser parser = new BoardParser();
        State start = parser.parse(inFile);

        Board board = new Board(start.board.length,
start.board[0].length);
        board.setExit(parser.getExitRow(), parser.getExitCol());
        Heuristics.setBoard(board);

        Scanner sc = new Scanner(System.in);
        Solver.Algorithm algo = askAlgorithm(sc);
        Heuristic heur = (algo == Solver.Algorithm.UCS) ?
Heuristics.ZERO : askHeuristic(sc);

        // *** output file name ***
        System.out.print("Nama file output (contoh: solution.txt): ");
        sc.nextLine(); // buang newline sisa
nextInt
        String outName = sc.nextLine().trim();
        if(outName.isEmpty()) outName = "solution.txt";

        long t0 = System.nanoTime();
        Solver.Result res = Solver.solve(start, board, algo, heur);
```

```

        long t1 = System.nanoTime();

        if (res == null) {
            System.out.println("Tidak ada solusi.");
            return;
        }
        // print stats
        System.out.println("\n== Statistik ==");
        System.out.printf("Algoritma          : %s\n", algo);
        System.out.printf("Heuristik          : %s\n", heur.name());
        System.out.printf("Node dikunjungi   : %d\n", res.visited);
        System.out.printf("Waktu eksekusi    : %.5f detik\n", (t1 - t0) /
1e9);

        // ----- cetak ke layar + simpan ke file -----
        Printer.printSolution(res);                // ke layar berwarna
        Printer.saveSolution(res, outName);         // ke file txt
        System.out.println("Solusi disimpan ke " + outName);
    }

    private static Solver.Algorithm askAlgorithm(Scanner sc) {
        System.out.println("Pilih algoritma :\n1. Uniform Cost Search
(UCS)\n2. Greedy Best First Search\n3. A* Search");
        int opt;
        do {
            System.out.print("> ");
            opt = sc.nextInt();
        } while (opt < 1 || opt > 3);

        Solver.Algorithm algo;
        switch (opt) {
            case 1:
                algo = Solver.Algorithm.UCS;
                break;
            case 2:
                algo = Solver.Algorithm.GREEDY;
                break;
            default:
                algo = Solver.Algorithm.ASTAR;

```

```

        break;
    }
    return algo;
}

private static Heuristic askHeuristic(Scanner sc) {
    System.out.println("Pilih heuristic :\n1. Blok penghalang
(admissible)\n2. Manhattan distance (approx)\n3. Nol (non-informatif)");
    int opt;
    do {
        System.out.print("> ");
        opt = sc.nextInt();
    } while (opt < 1 || opt > 3);

    Heuristic heur;
    switch (opt) {
        case 1:
            heur = Heuristics.BLOCKING;
            break;
        case 2:
            heur = Heuristics.MANHATTAN;
            break;
        default:
            heur = Heuristics.ZERO;
            break;
    }
    return heur;
}
}

```

## 2. Board.java

### Board.java

```

import java.util.*;

class Board {

```

```

final int R, C;
private int exitRow = -1, exitCol = -1;

int getExitRow() {
    return exitRow;
}

int getExitCol() {
    return exitCol;
}

Board(int R, int C) {
    this.R = R; this.C = C;
}

void setExit(int r, int c) {
    this.exitRow = r; this.exitCol = c;
}

boolean isGoal(State s) {
    // Handle the case where K is outside the board boundary
    if (exitCol >= C) {
        // For horizontal P pieces adjacent to the right edge
        Piece p = s.pieces.get('P');
        if (p.ori == Orientation.HORIZONTAL) {
            int pTailRow = s.primaryRow;
            int pTailCol = s.primaryCol + p.length - 1;
            return pTailRow == exitRow && pTailCol == C - 1;
        }
        return false;
    } else if (exitRow >= R) {
        // For vertical P pieces adjacent to the bottom edge
        Piece p = s.pieces.get('P');
        if (p.ori == Orientation.VERTICAL) {
            int pTailRow = s.primaryRow + p.length - 1;
            int pTailCol = s.primaryCol;
            return pTailCol == exitCol && pTailRow == R - 1;
        }
        return false;
    } else {

```



```

        // Normal case where K is inside the board
        char[][] board = s.board;
        return board[exitRow][exitCol] == 'P';
    }
}

List<State> expand(State s) {
    List<State> res = new ArrayList<>();
    char[][] board = s.board;
    for (Map.Entry<Character, Piece> entry : s.pieces.entrySet()) {
        char id = entry.getKey();
        Piece pc = entry.getValue();
        // cari posisi head piece (paling kiri/atas) di board
        int headR = -1, headC = -1;
        outer:
        for (int r = 0; r < R; r++) {
            for (int c = 0; c < C; c++) {
                if (board[r][c] == id) {
                    headR = r; headC = c; break outer;
                }
            }
        }
        // coba gerak -1 dan +1 pada orientasinya
        if (pc.ori == Orientation.HORIZONTAL) {
            // ke kiri
            int cLeft = headC - 1;
            if (cLeft >= 0 && board[headR][cLeft] == '.') {
                res.add(makeMove(s, id, headR, headC,
Direction.LEFT));
            }
            // ke kanan
            int tailC = headC + pc.length - 1;
            int cRight = tailC + 1;
            if (cRight < C && board[headR][cRight] == '.') {
                res.add(makeMove(s, id, headR, headC,
Direction.RIGHT));
            }
            // special: primary piece menuju exit
            if (id == 'P') {
                // Check if piece can move right to reach exit

```

```

        if (headR == exitRow && cRight == exitCol) {
            res.add(makeMove(s, id, headR, headC,
Direction.RIGHT));
        }
        // Check if piece can move left to reach exit
        if (headR == exitRow && cLeft == exitCol && cLeft >=
0) {
            res.add(makeMove(s, id, headR, headC,
Direction.LEFT));
        }
    }
} else { // VERTICAL
    // ke atas
    int rUp = headR - 1;
    if (rUp >= 0 && board[rUp][headC] == '.') {
        res.add(makeMove(s, id, headR, headC, Direction.UP));
    }
    // ke bawah
    int tailR = headR + pc.length - 1;
    int rDown = tailR + 1;
    if (rDown < R && board[rDown][headC] == '.') {
        res.add(makeMove(s, id, headR, headC,
Direction.DOWN));
    }
    if (id == 'P') {
        // Check if piece can move down to reach exit
        if (headC == exitCol && rDown == exitRow) {
            res.add(makeMove(s, id, headR, headC,
Direction.DOWN));
        }
        // Check if piece can move up to reach exit
        if (headC == exitCol && rUp == exitRow && rUp >= 0) {
            res.add(makeMove(s, id, headR, headC,
Direction.UP));
        }
    }
}
}
return res;
}

```

```

    private State makeMove(State s, char id, int headR, int headC,
Direction dir) {
    Piece pc = s.pieces.get(id);
    char[][] nb = deepCopy(s.board);
    // hapus piece dari board
    for (int i = 0; i < pc.length; i++) {
        int r = headR + (pc.ori == Orientation.VERTICAL ? i : 0);
        int c = headC + (pc.ori == Orientation.HORIZONTAL ? i : 0);
        nb[r][c] = '.';
    }
    // posisi baru head
    int newHeadR = headR + dir.dr;
    int newHeadC = headC + dir.dc;
    // tempatkan kembali piece
    for (int i = 0; i < pc.length; i++) {
        int r = newHeadR + (pc.ori == Orientation.VERTICAL ? i : 0);
        int c = newHeadC + (pc.ori == Orientation.HORIZONTAL ? i :
0);
        nb[r][c] = id;
    }
    // jika primary keluar, tandai K dengan P lalu isGoal akan
succeed pada next loop
    return new State(nb, s.pieces, id == 'P' ? newHeadR :
s.primaryRow,
                                id == 'P' ? newHeadC :
s.primaryCol,
                                s.g + 1, new Move(id, dir), s);
}

private char[][] deepCopy(char[][] src) {
    char[][] dst = new char[src.length][];
    for (int i = 0; i < src.length; i++) dst[i] =
Arrays.copyOf(src[i], src[i].length);
    return dst;
}
}

// Heuristic Interface

```

```

interface Heuristic {
    int estimate(State s);
    default String name() {
        return this.getClass().getSimpleName();
    }
}

```

### 3. BoardParser.java

#### BoardParser.java

```

import java.io.*;
import java.util.*;

class BoardParser {

    private int rows, cols;
    private int exitRow = -1, exitCol = -1;

    State parse(String filename) throws IOException {
        List<String> lines = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new
        FileReader(filename))) {
            StringTokenizer st = new StringTokenizer(br.readLine());
            rows = Integer.parseInt(st.nextToken());
            cols = Integer.parseInt(st.nextToken());

            // jumlah piece (tidak dipakai di parser)
            br.readLine();

            String line;
            while ((line = br.readLine()) != null &&
            !line.trim().isEmpty()) {
                lines.add(line.trim());
            }
        }
    }
}

```

```

        if (lines.size() != rows)
            throw new IllegalArgumentException("Jumlah baris tidak
sesuai. Ditemukan " + lines.size() + ", seharusnya " + rows);

        char[][] boardArr = new char[rows][cols];
        for (char[] row : boardArr) Arrays.fill(row, '.');

        Map<Character, List<int[]>> posMap = new HashMap<>();

        for (int r = 0; r < rows; r++) {
            String line = lines.get(r);

            if (line.length() != cols && !(line.length() == cols + 1 &&
line.charAt(cols) == 'K'))
                throw new IllegalArgumentException("Panjang baris ke-" +
(r + 1) + " = " + line.length() + " ≠ " + cols + " (kecuali K).");

            for (int c = 0; c < line.length(); c++) {
                char ch = line.charAt(c);

                if (ch == '.') continue;

                if (ch == 'K') {
                    // exit
                    exitRow = r;
                    exitCol = c;
                    // bisa = cols (di luar
papan)

                    continue;
                }

                if (c >= cols)
                    throw new IllegalArgumentException("Karakter '" + ch
+ "' di luar papan pada baris " + (r + 1));

                boardArr[r][c] = ch;
                posMap.computeIfAbsent(ch, k -> new
ArrayList<>()).add(new int[]{r, c});
            }
        }
    }

```

```

        if (exitRow == -1)
            throw new IllegalArgumentException("Input tidak valid:
            karakter 'K' (pintu keluar) hilang.");
        if (!posMap.containsKey('P'))
            throw new IllegalArgumentException("Input tidak valid:
            karakter 'P' (primary piece) hilang.");

        /* ==== buat objek Piece ==== */
        Map<Character, Piece> pieces = new HashMap<>();
        int pRow = -1, pCol = -1;

        for (Map.Entry<Character, List<int[]>> e : posMap.entrySet()) {
            char id = e.getKey();
            List<int[]> pts = e.getValue();

            /* -- pastikan kotak-kotak piece kontigu dan segaris -- */
            int len = pts.size();

            // Cek semua row sama (horizontal) atau semua col sama
            (vertikal)
            boolean sameRow = pts.stream().allMatch(p -> p[0] ==
            pts.get(0)[0]);
            boolean sameCol = pts.stream().allMatch(p -> p[1] ==
            pts.get(0)[1]);

            if (!(sameRow || sameCol)) {
                throw new IllegalArgumentException("Piece '" + id + "'
            tidak segaris");
            }

            // Sort posisi agar bisa dicek kontiguitas dengan benar
            pts.sort(Comparator.comparingInt((int[] p) ->
            p[0]).thenComparingInt(p -> p[1]));

            // Setelah sort, cek urutan kontigu
            for (int i = 1; i < pts.size(); i++) {
                int[] prev = pts.get(i - 1);
                int[] curr = pts.get(i);
            }
        }

```

```

        if (sameRow && curr[1] != prev[1] + 1)
            throw new IllegalArgumentException("Piece '" + id +
            "' tidak kontigu");

        if (sameCol && curr[0] != prev[0] + 1)
            throw new IllegalArgumentException("Piece '" + id +
            "' tidak kontigu");
    }

    Orientation ori = sameRow ? Orientation.HORIZONTAL :
    Orientation.VERTICAL;

    pieces.put(id, new Piece(id, ori, len));

    if (id == 'P') {
        pRow = pts.get(0)[0];
        pCol = pts.get(0)[1];
    }
}

return new State(boardArr, pieces, pRow, pCol, 0, null, null);
}

int getExitRow() {
    return exitRow;
}

int getExitCol() {
    return exitCol;
}
}

```

#### 4. Heuristics.java

### Heuristics.java

```
class Heuristics {
```

```

// Static reference to the Board instance
private static Board boardInstance;

// Method to set the board reference
public static void setBoard(Board board) {
    boardInstance = board;
}

static final Heuristic ZERO = new Heuristic() {
    public int estimate(State s) { return 0; }
    public String name() { return "Zero (tidak ada heuristik)"; }
};

static final Heuristic BLOCKING = new Heuristic() {
    public int estimate(State s) {
        Piece p = s.pieces.get('P');
        int row = s.primaryRow, colTail = s.primaryCol + p.length -
1;

        int dist = 0, blockers = 0;
        for (int c = colTail + 1; c < s.board[0].length; c++) {
            if (s.board[row][c] == '.') dist++;
            else if (s.board[row][c] == 'K') break;
            else { blockers++; }
        }
        return blockers * 2 + dist; // sederhana & admissible
    }
};

static final Heuristic MANHATTAN = s -> {
    // Check if board is set
    if (boardInstance == null) {
        throw new IllegalStateException("Papan belum diatur di kelas
Heuristik. Panggil aturPapan() terlebih dahulu.");
    }

    Piece p = s.pieces.get('P');
    int relevantRow = (p.ori == Orientation.VERTICAL)
        ? s.primaryRow + p.length - 1 // tail row
        : s.primaryRow;
    int relevantCol = (p.ori == Orientation.HORIZONTAL)

```



```

        ? s.primaryCol + p.length - 1 // tail col
        : s.primaryCol;

    return Math.abs(relevantRow - boardInstance.getExitRow())
        + Math.abs(relevantCol - boardInstance.getExitCol());
};
}

```

## 5. Solver.java

### Solver.java

```

import java.util.*;

class Solver {
    enum Algorithm { UCS, GREEDY, ASTAR }

    static class Result {
        final State goal;
        final long visited;
        Result(State goal, long visited) { this.goal = goal; this.visited
= visited; }
    }

    static Result solve(State start, Board board, Algorithm algo,
Heuristic h) {
        Set<State> visited = new HashSet<>();
        Comparator<State> comp;
        switch (algo) {
            case UCS:
                comp = Comparator.comparingInt(s -> s.g);
                break;
            case GREEDY:
                comp = Comparator.comparingInt(s -> h.estimate(s));
                break;
            case ASTAR:
                comp = Comparator.comparingInt(s -> s.g + h.estimate(s));
                break;
        }
    }
}

```

```

        default:
            throw new IllegalArgumentException("Algoritma tidak
dikenal");
        }

        PriorityQueue<State> pq = new PriorityQueue<>(comp);
        pq.add(start);
        long nodes = 0;
        while (!pq.isEmpty()) {
            State cur = pq.poll();
            nodes++;
            if (visited.contains(cur)) continue;
            visited.add(cur);
            if (board.isGoal(cur)) {
                return new Result(cur, nodes);
            }
            for (State nxt : board.expand(cur)) {
                if (!visited.contains(nxt)) {
                    pq.add(nxt);
                }
            }
        }
        return null;
    }
}

```

## 6. State.java

### State.java

```

import java.util.*;

class State {
    final char[][] board;           // immutable snapshot
    final Map<Character, Piece> pieces; // metadata orientation &
length
}

```

```

    final int primaryRow, primaryCol;    // head posisi P (baris, kolom)
    final int g;                        // cost so far (langkah)
    final Move move;                    // move that produced this
state (null untuk start)
    final State prev;                  // parent link

    State(char[][] board, Map<Character, Piece> pieces, int pRow, int
pCol, int g, Move move, State prev) {
        this.board = board;
        this.pieces = pieces;
        this.primaryRow = pRow;
        this.primaryCol = pCol;
        this.g = g;
        this.move = move;
        this.prev = prev;
    }

    // hash representation - serialize board ke String
    @Override public int hashCode() {
        return Arrays.deepHashCode(board);
    }
    @Override
    public boolean equals(Object o) {
        if (!(o instanceof State)) return false;
        State s = (State) o;
        return Arrays.deepEquals(board, s.board);
    }
}

```

## 7. Move.java

### Move.java

```

class Move {
    final char id;
    final Direction dir;
    Move(char id, Direction dir) { this.id = id; this.dir = dir; }
}

```

```

    @Override public String toString() { return id + "-" + dir.text; }
}

enum Direction {
    UP(-1, 0, "atas"), DOWN(1, 0, "bawah"), LEFT(0, -1, "kiri"), RIGHT(0,
1, "kanan");
    final int dr, dc; final String text;
    Direction(int dr, int dc, String t) { this.dr = dr; this.dc = dc;
this.text = t; }
}

```

## 8. Piece.java

### Piece.java

```

class Piece {
    final char id;           // character symbol (A,B,C, ..., P)
    final Orientation ori;   // horizontal / vertical
    final int length;        // 2 atau 3 (default)

    Piece(char id, Orientation ori, int length) {
        this.id = id;
        this.ori = ori;
        this.length = length;
    }
}

enum Orientation { HORIZONTAL, VERTICAL }

```

## 9. Printer.java

### Printer.java

```

import java.util.*;

```

```

import java.io.*;

class Printer {
    private static final String RESET = "\u001B[0m";
    private static final String RED    = "\u001B[31m";
    private static final String GREEN  = "\u001B[32m";
    private static final String BLUE   = "\u001B[34m";

    static void printSolution(Solver.Result res) {
        List<State> path = new ArrayList<>();
        for (State cur = res.goal; cur != null; cur = cur.prev)
path.add(cur);
        Collections.reverse(path);
        render(path, System.out, true);
    }

    static void saveSolution(Solver.Result res, String fileName) throws
IOException {
        try (PrintWriter pw = new PrintWriter(fileName)) {
            // Convert result to path list, similar to printSolution
            List<State> path = new ArrayList<>();
            for (State cur = res.goal; cur != null; cur = cur.prev)
path.add(cur);
            Collections.reverse(path);
            render(path, pw, false);           // false = tanpa warna
        }
    }

    private static void render(List<State> path, Appendable out, boolean
colored) {
        append(out, "\n== Urutan Gerakan ==\nPapan Awal\n");
        printBoard(out, path.get(0).board, null, colored);
        for (int i = 1; i < path.size(); i++) {
            State s = path.get(i);
            append(out, String.format("\nGerakan %d: %s\n", i, s.move));
            printBoard(out, s.board, s.move, colored);
        }
    }
}

```

```

private static void printBoard(Appendable out, char[][] board,
                               Move highlight, boolean colored) {
    for (char[] row : board) {
        for (char ch : row) {
            if (colored) {
                if (ch == 'P')
                    append(out, RED + ch + RESET);
                else if (ch == 'K')
                    append(out, GREEN + ch + RESET);
                else if (highlight != null && ch == highlight.id)
                    append(out, BLUE + ch + RESET);
                else
                    append(out, ch);
            } else {
                append(out, ch);
            }
        }
        append(out, '\n');
    }
}

private static void append(Appendable a, Object s) {
    try {
        a.append(String.valueOf(s));
    } catch (IOException e) {
        /* ignore */
    }
}
}

```

## BAB 3 EKSPERIMEN

No.	Input	Output
1.	<pre> 6 6 11 AAB..F ..BCDF GPPCDFK GH.III GHJ... LLJMM.</pre>	<pre> == Urutan Gerakan == Papan Awal AAB..F ..BCDF GPPCDF GH.III GHJ... LLJMM.  Gerakan 1: C-atas AABC.F ..BCDF GPP.DF GH.III GHJ... LLJMM.  Gerakan 2: P-kanan AABC.F ..BCDF G.PPDF GH.III GHJ... LLJMM.  Gerakan 3: D-atas AABCDF ..BCDF G.PP.F GH.III GHJ... LLJMM.</pre>

		<p>Gerakan 4: P-kanan</p> <p>AABCDF  ..BCDF  G..PPF  GH.III  GHJ...  LLJMM.</p> <p>Gerakan 5: M-kanan</p> <p>AABCDF  ..BCDF  G..PPF  GH.III  GHJ...  LLJ.MM</p> <p>Gerakan 6: H-atas</p> <p>AABCDF  ..BCDF  GH.PPF  GH.III  G.J...  LLJ.MM</p> <p>Gerakan 7: M-kiri</p> <p>AABCDF  ..BCDF  GH.PPF  GH.III  G.J...  LLJMM.</p> <p>Gerakan 8: J-atas</p> <p>AABCDF  ..BCDF  GH.PPF  GHJIII  G.J...</p>
--	--	--



		<pre> LL.MM.  Gerakan 9: M-kanan AABCDF ..BCDF GH.PPF GHJIII G.J... LL..MM  Gerakan 10: L-kanan AABCDF ..BCDF GH.PPF GHJIII G.J... ..LL.MM  Gerakan 11: M-kiri AABCDF ..BCDF GH.PPF GHJIII G.J... ..LLMM.  Gerakan 12: J-atas AABCDF ..BCDF GHJPPF GHJIII G..... ..LLMM.  Gerakan 13: H-bawah AABCDF ..BCDF G.JPPF GHJIII </pre>
--	--	--

		<pre> GH.... ..LLMM.  Gerakan 14: M-kanan AABCDF ..BCDF G.JPPF GHJIII GH.... ..LL.MM  Gerakan 15: L-kanan AABCDF ..BCDF G.JPPF GHJIII GH.... ..LLMM  Gerakan 16: J-bawah AABCDF ..BCDF G..PPF GHJIII GHJ... ..LLMM  Gerakan 17: H-bawah AABCDF ..BCDF G..PPF G.JIII GHJ... ..HLLMM  Gerakan 18: G-atas AABCDF G.BCDF G..PPF </pre>
--	--	--

		<pre> G.JIII .HJ... .HLLMM  Gerakan 19: J-atas AABCDF G.BCDF G.JPPF G.JIII .H.... .HLLMM  Gerakan 20: G-bawah AABCDF ..BCDF G.JPPF G.JIII GH.... .HLLMM  Gerakan 21: G-bawah AABCDF ..BCDF ..JPPF G.JIII GH.... GHLLMM  Gerakan 22: J-bawah AABCDF ..BCDF ...PPF G.JIII GHJ... GHLLMM  Gerakan 23: B-bawah AA.CDF ..BCDF </pre>
--	--	---

		<pre> ..BPPF G.JIII GHJ... GHLLMM  Gerakan 24: H-atas AA.CDF ..BCDF ..BPPF GHJIII GHJ... G.LLMM  Gerakan 25: L-kiri AA.CDF ..BCDF ..BPPF GHJIII GHJ... GLL.MM  Gerakan 26: M-kiri AA.CDF ..BCDF ..BPPF GHJIII GHJ... GLLMM.  Gerakan 27: G-atas AA.CDF ..BCDF G.BPPF GHJIII GHJ... .LLMM.  Gerakan 28: M-kanan AA.CDF </pre>
--	--	--

		<pre> ..BCDF G.BPPF GHJIII GHJ... ..LL.MM  Gerakan 29: L-kanan AA.CDF ..BCDF G.BPPF GHJIII GHJ... ..LLMM  Gerakan 30: H-atas AA.CDF ..BCDF GHBPPF GHJIII G.J... ..LLMM  Gerakan 31: L-kiri AA.CDF ..BCDF GHBPPF GHJIII G.J... ..LL.MM  Gerakan 32: M-kiri AA.CDF ..BCDF GHBPPF GHJIII G.J... ..LLMM.  Gerakan 33: L-kiri </pre>
--	--	---

		AA.CDF ..BCDF GHBPPF GHJIII G.J... LL.MM.  Gerakan 34: M-kanan AA.CDF ..BCDF GHBPPF GHJIII G.J... LL..MM  Gerakan 35: J-bawah AA.CDF ..BCDF GHBPPF GH.III G.J... LLJ.MM  Gerakan 36: M-kiri AA.CDF ..BCDF GHBPPF GH.III G.J... LLJMM.  Gerakan 37: I-kiri AA.CDF ..BCDF GHBPPF GHIII. G.J... LLJMM.
--	--	--

		<p>Gerakan 38: M-kanan</p> <p>AA.CDF</p> <p>..BCDF</p> <p>GHBPPF</p> <p>GHIII.</p> <p>G.J...</p> <p>LLJ.MM</p> <p>Gerakan 39: H-bawah</p> <p>AA.CDF</p> <p>..BCDF</p> <p>G.BPPF</p> <p>GHIII.</p> <p>GHJ...</p> <p>LLJ.MM</p> <p>Gerakan 40: M-kiri</p> <p>AA.CDF</p> <p>..BCDF</p> <p>G.BPPF</p> <p>GHIII.</p> <p>GHJ...</p> <p>LLJMM.</p> <p>Gerakan 41: G-atas</p> <p>AA.CDF</p> <p>G.BCDF</p> <p>G.BPPF</p> <p>GHIII.</p> <p>.HJ...</p> <p>LLJMM.</p> <p>Gerakan 42: M-kanan</p> <p>AA.CDF</p> <p>G.BCDF</p> <p>G.BPPF</p> <p>GHIII.</p> <p>.HJ...</p> <p>LLJ.MM</p>
--	--	--

		<p>Gerakan 43: I-kanan</p> <p>AA.CDF G.BCDF G.BPPF GH.III .HJ... LLJ.MM</p> <p>Gerakan 44: M-kiri</p> <p>AA.CDF G.BCDF G.BPPF GH.III .HJ... LLJMM.</p> <p>Gerakan 45: J-atas</p> <p>AA.CDF G.BCDF G.BPPF GHJIII .HJ... LL.MM.</p> <p>Gerakan 46: M-kanan</p> <p>AA.CDF G.BCDF G.BPPF GHJIII .HJ... LL..MM</p> <p>Gerakan 47: H-atas</p> <p>AA.CDF G.BCDF GHBPPF GHJIII ..J...</p>
--	--	---



		<pre> LL..MM  Gerakan 48: M-kiri AA.CDF G.BCDF GHBPPF GHJIII ..J... LL.MM.  Gerakan 49: M-kiri AA.CDF G.BCDF GHBPPF GHJIII ..J... LLMM..  Gerakan 50: H-bawah AA.CDF G.BCDF G.BPPF GHJIII .HJ... LLMM..  Gerakan 51: G-bawah AA.CDF ..BCDF G.BPPF GHJIII GHJ... LLMM..  Gerakan 52: H-atas AA.CDF ..BCDF GHBPPF GHJIII </pre>
--	--	---

		<p>G.J... LLMM..</p> <p>Gerakan 53: H-atas</p> <p>AA.CDF .HBCDF GHBPPF G.JIII G.J... LLMM..</p> <p>Gerakan 54: A-kanan</p> <p>.AACDF .HBCDF GHBPPF G.JIII G.J... LLMM..</p> <p>Gerakan 55: H-bawah</p> <p>.AACDF ..BCDF GHBPPF GHJIII G.J... LLMM..</p> <p>Gerakan 56: M-kanan</p> <p>.AACDF ..BCDF GHBPPF GHJIII G.J... LL.MM.</p> <p>Gerakan 57: M-kanan</p> <p>.AACDF ..BCDF GHBPPF</p>
--	--	--

		<p>GHJIII</p> <p>G.J...</p> <p>LL..MM</p> <p>Gerakan 58: L-kanan</p> <p>.AACDF</p> <p>..BCDF</p> <p>GHBPPF</p> <p>GHJIII</p> <p>G.J...</p> <p>.LL.MM</p> <p>Gerakan 59: H-bawah</p> <p>.AACDF</p> <p>..BCDF</p> <p>G.BPPF</p> <p>GHJIII</p> <p>GHJ...</p> <p>.LL.MM</p> <p>Gerakan 60: L-kiri</p> <p>.AACDF</p> <p>..BCDF</p> <p>G.BPPF</p> <p>GHJIII</p> <p>GHJ...</p> <p>LL..MM</p> <p>Gerakan 61: G-atas</p> <p>.AACDF</p> <p>G.BCDF</p> <p>G.BPPF</p> <p>GHJIII</p> <p>.HJ...</p> <p>LL..MM</p> <p>Gerakan 62: M-kiri</p> <p>.AACDF</p> <p>G.BCDF</p>
--	--	---

		<p>G.BPPF  GHJIII  .HJ...  LL.MM.</p> <p>Gerakan 63: M-kiri</p> <p>.AACDF  G.BCDF  G.BPPF  GHJIII  .HJ...  LLMM..</p> <p>Gerakan 64: H-atas</p> <p>.AACDF  G.BCDF  GHBPPF  GHJIII  ..J...  LLMM..</p> <p>Gerakan 65: H-atas</p> <p>.AACDF  GHBCDF  GHBPPF  G.JIII  ..J...  LLMM..</p> <p>Gerakan 66: M-kanan</p> <p>.AACDF  GHBCDF  GHBPPF  G.JIII  ..J...  LL.MM.</p> <p>Gerakan 67: M-kanan</p> <p>.AACDF</p>
--	--	---

		<p>           GHBCDF            GHBPPF            G.JIII            ..J...            LL..MM         </p> <p>           Gerakan 68: J-bawah            .AACDF            GHBCDF            GHBPPF            G..III            ..J...            LLJ.MM         </p> <p>           Gerakan 69: M-kiri            .AACDF            GHBCDF            GHBPPF            G..III            ..J...            LLJMM.         </p> <p>           Gerakan 70: I-kiri            .AACDF            GHBCDF            GHBPPF            G.III.            ..J...            LLJMM.         </p> <p>           Gerakan 71: M-kanan            .AACDF            GHBCDF            GHBPPF            G.III.            ..J...            LLJ.MM         </p> <p>           Gerakan 72: I-kiri         </p>
--	--	---

		<pre> .AACDF GHBCDF GHBPPF GIII.. ..J... LLJ.MM  Gerakan 73: M-kiri .AACDF GHBCDF GHBPPF GIII.. ..J... LLJMM.  Gerakan 74: G-bawah .AACDF .HBCDF GHBPPF GIII.. G.J... LLJMM.  Gerakan 75: M-kanan .AACDF .HBCDF GHBPPF GIII.. G.J... LLJ.MM  Gerakan 76: F-bawah .AACD. .HBCDF GHBPPF GIII.F G.J... LLJ.MM </pre>
--	--	---

		<p>Gerakan 77: F-bawah</p> <p>.AACD. .HBCD. GHBPPF GIII.F G.J..F LLJ.MM</p> <p>Gerakan 78: M-kiri</p> <p>.AACD. .HBCD. GHBPPF GIII.F G.J..F LLJMM.</p> <p>Gerakan 79: I-kanan</p> <p>.AACD. .HBCD. GHBPPF G.IIIF G.J..F LLJMM.</p> <p>Gerakan 80: M-kanan</p> <p>.AACD. .HBCD. GHBPPF G.IIIF G.J..F LLJ.MM</p> <p>Gerakan 81: H-bawah</p> <p>.AACD. ..BCD. GHBPPF GHIIIF G.J..F LLJ.MM</p>
--	--	--

		<p>Gerakan 82: H-bawah</p> <p>.AACD.  ..BCD.  G.BPPF  GHIIIF  GHJ..F  LLJ.MM</p> <p>Gerakan 83: M-kiri</p> <p>.AACD.  ..BCD.  G.BPPF  GHIIIF  GHJ..F  LLJMM.</p> <p>Gerakan 84: H-atas</p> <p>.AACD.  ..BCD.  GHBPPF  GHIIIF  G.J..F  LLJMM.</p> <p>Gerakan 85: F-bawah</p> <p>.AACD.  ..BCD.  GHBPP.  GHIIIF  G.J..F  LLJMMF</p> <p>Gerakan 86: P-kanan</p> <p>.AACD.  ..BCD.  GHB.PP  GHIIIF  G.J..F</p>
--	--	--



		LLJMMF
2.	7 6 5 QQ.... .GGHH. .AABB. PPLMN.K ..LMN. ..DD.. RR....	<pre> == Urutan Gerakan == Papan Awal QQ.... .GGHH. .AABB. PPLMN. ..LMN. ..DD.. RR....  Gerakan 1: D-kiri QQ.... .GGHH. .AABB. PPLMN. ..LMN. .DD... RR....  Gerakan 2: M-bawah QQ.... .GGHH. .AABB. PPL.N. ..LMN. .DDM.. RR....  Gerakan 3: N-bawah QQ.... .GGHH. .AABB. PPL... ..LMN. </pre>

		<pre> .DDMN. RR....  Gerakan 4: D-kiri QQ.... .GGHH. .AABB. PPL... ..LMN. DD.MN. RR....  Gerakan 5: L-bawah QQ.... .GGHH. .AABB. PP.... ..LMN. DDL MN. RR....  Gerakan 6: P-kanan QQ.... .GGHH. .AABB. .PP... ..LMN. DDL MN. RR....  Gerakan 7: P-kanan QQ.... .GGHH. .AABB. ..PP.. ..LMN. DDL MN. RR.... </pre>
--	--	--

		<pre> Gerakan 8: P-kanan QQ.... .GGHH. .AABB. ...PP. ..LMN. DDL MN. RR....  Gerakan 9: P-kanan QQ.... .GGHH. .AABB. ....PP ..LMN. DDL MN. RR.... </pre>
3.	<pre> 6 6 4 PPA... ..A... BB.... ..... .....K ..... </pre>	<pre> Tidak ada solusi. </pre>
4.	<pre> 6 6 3 P..ABB ....BB ..... ..... ..... .....K </pre>	<pre> &gt; java -cp bin Main TestCase4.txt Exception in thread "main" java.lang.IllegalArgumentException: Piece 'B' tidak segaris     at BoardParser.parse(BoardParser.java:80)     at Main.main(Main.java:12) </pre>

5.	<pre> 6 6 2 P.P... ..... P..K.. ..... ..... ..... </pre>	<pre> &gt; java -cp bin Main TestCase5.txt Exception in thread "main" java.lang.IllegalArgumentException: Piece 'P' tidak segaris     at BoardParser.parse(BoardParser.java:80)     at Main.main(Main.java:12) </pre>
6.	<pre> 6 6 1 PP.K.. ..... </pre>	<pre> &gt; java -cp bin Main TestCase6.txt Exception in thread "main" java.lang.IllegalArgumentException: Jumlah baris tidak sesuai. Ditemukan 2, seharusnya 6     at BoardParser.parse(BoardParser.java:27)     at Main.main(Main.java:12) </pre>
7.	<pre> 6 6 1 PP..K. ..... ..... ..... ..... .....  .....  </pre>	<pre> &gt; java -cp bin Main TestCase7.txt Exception in thread "main" java.lang.IllegalArgumentException: Panjang baris ke-2 = 7 ? 6 (kecuali K).     at BoardParser.parse(BoardParser.java:38)     at Main.main(Main.java:12) </pre>

## LAMPIRAN

### Link Repository

[https://github.com/BenedictusNelson/Tucil3\\_13523150](https://github.com/BenedictusNelson/Tucil3_13523150)

### Checklist

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 dan menyimpan solusi berupa print board tahap per tahap dalam berkas .txt	✓	
5. [Bonus] Implementasi algoritma pathfinding alternatif		✓
6. [Bonus] Implementasi 2 atau lebih heuristik alternatif	✓	
7. [Bonus] Program memiliki GUI		✓
8. Program dan laporan dibuat (kelompok) sendiri	✓	