TUGAS BESAR

DESAIN DAN ANALISIS ALGORITMA

(Tahap 2)

Dibuat Oleh:

Friskadini Ismayanti	(1301198496)
Muhammad Faisal Amir	(1301198497)
Ridho Maulana Cahyudi	(1301198515)

TELKOM UNIVERSITY

FAKULTAS INFORMATIKA

2020

Persiapan Alat

- Java Open JDK (SDK) - https://www.oracle.com/java/technologies/javase-jdk13-downloads.html
- IntelliJ IDEA JetBrain Community (IDE) - https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code=IIC
- Git (Windows) - https://git-scm.com/download/win

Algoritma Divide and Conquer

A. Percobaan

Divide And Conquer	
Rancangan	Skenario
- Divide : Membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama). - Conquer : Memecahkan (menyelesaikan) masing-masing upa-masalah (secara rekursif). - Combine : Menggabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula	1. Pertama-tama lakukan pengurutan terhadap titik-titik dari himpunan S yang diberika berdasarkan koordinat absis-X, dengan kompleksitas waktu O(n log n). 2. Jika |S| ≤ 3, maka lakukan pencarian convex hull secara brute-force dengan kompleksitas waktu O(1). (Basis). 3. Jika tidak, partisi himpunan titik-titik pada S menjadi 2 buah himpunan A dan B, dimana A terdiri dari setengah jumlah dari |S| dan titik dengan koordinat absix-X yang terendah dan B terdiri dari setengah dari jumlah |S| dan titik dengan koordinat absis-X terbesar. 4. Secara rekursif lakukan penghitungan terhadap HA = conv(A) dan HB = conv(B). 5. Lakukan penggabungan (merge) terhadap kedua hull tersebut menjadi convex hull, H, dengan menghitung dan mencari upper dan lower tangents untuk HA dan HB dengan mengabaikan semua titik yang berada diantara dua buah tangen ini.

B. Hasil Percobaan

1. Code Program

- Node.java

```java
package com.frogobox.branchbound;

/**
 * Created by Faisal Amir
 * FrogoBox Inc License
 * ========================================
 * divide-conquer-branch-bound
 * Copyright (C) 07/05/2020.
 * All rights reserved
 * ----------------------------------------
 * Name     : Muhammad Faisal Amir
 * E-mail   : faisalamircs@gmail.com
 * Github   : github.com/amirisback
 * LinkedIn : linkedin.com/in/faisalamircs
 * ----------------------------------------
 * FrogoBox Software Industries
 * com.frogobox.branchbound
 */

public class Node {

    public Node parent;
    public int[][] matrix;

    // Blank tile cordinates
    public int x, y;

    // Number of misplaced tiles
    public int cost;

    // The number of moves so far
    public int level;

    public Node(int[][] matrix, int x, int y, int newX, int newY, int level, Node parent) {
        this.parent = parent;
        this.matrix = new int[matrix.length][];
        for (int i = 0; i < matrix.length; i++) {
            this.matrix[i] = matrix[i].clone();
        }

        // Swap value
```

```java
        this.matrix[x][y]       = this.matrix[x][y] + this.matrix[newX][newY];
        this.matrix[newX][newY] = this.matrix[x][y] - this.matrix[newX][newY];
        this.matrix[x][y]       = this.matrix[x][y] - this.matrix[newX][newY];

        this.cost = Integer.MAX_VALUE;
        this.level = level;
        this.x = newX;
        this.y = newY;
    }

}
```

- Algorithm.java

```java
package com.frogobox.branchbound;

import java.util.ArrayList;
import java.util.List;
import java.util.PriorityQueue;

/**
 * Created by Faisal Amir
 * FrogoBox Inc License
 * ========================================
 * divide-conquer-branch-bound
 * Copyright (C) 07/05/2020.
 * All rights reserved
 * ----------------------------------------
 * Name      : Muhammad Faisal Amir
 * E-mail    : faisalamircs@gmail.com
 * Github    : github.com/amirisback
 * LinkedIn  : linkedin.com/in/faisalamircs
 * ----------------------------------------
 * FrogoBox Software Industries
 * com.frogobox.branchbound
 */
public class Algorithm {

    public int dimension = 3;

    // Bottom, left, top, right
    int[] row = {1, 0, -1, 0};
    int[] col = {0, -1, 0, 1};

    public int calculateCost(int[][] initial, int[][] goal) {
        int count = 0;
        int n = initial.length;
        for (int i = 0; i < n; i++) {
```

```java
            for (int j = 0; j < n; j++) {
                if (initial[i][j] != 0 && initial[i][j] != goal[i][j]) {
                    count++;
                }
            }
        }
        return count;
    }

    public void printMatrix(int[][] matrix) {
        System.out.println("Resolve Puzzle");
        for (int[] ints : matrix) {
            for (int j = 0; j < matrix.length; j++) {
                System.out.print(ints[j] + " ");
            }
            System.out.println();
        }
    }

    public boolean isSafe(int x, int y) {
        return (x >= 0 && x < dimension && y >= 0 && y < dimension);
    }

    public void printPath(Node root) {
        if (root == null) {
            return;
        }

        printPath(root.parent);
        printMatrix(root.matrix);
        System.out.println();

    }

    public boolean isSolvable(int[][] matrix) {
        int count = 0;
        List<Integer> array = new ArrayList<Integer>();

        for (int[] ints : matrix) {
            for (int j = 0; j < matrix.length; j++) {
                array.add(ints[j]);
            }
        }

        Integer[] anotherArray = new Integer[array.size()];
        array.toArray(anotherArray);

        for (int i = 0; i < anotherArray.length - 1; i++) {
            for (int j = i + 1; j < anotherArray.length; j++) {
```

```java
                if (anotherArray[i] != 0 && anotherArray[j] != 0 && anotherArray[i] > anotherArray[j]) {
                    count++;
                }
            }
        }

        return count % 2 == 0;
    }

    public void solve(int[][] initial, int[][] goal, int x, int y) {
        PriorityQueue<Node> pq = new PriorityQueue<Node>(1000, (a, b) -> (a.cost + a.level) - (b.cost + b.level));
        Node root = new Node(initial, x, y, x, y, 0, null);
        root.cost = calculateCost(initial, goal);
        pq.add(root);

        while (!pq.isEmpty()) {
            Node min = pq.poll();
            if (min.cost == 0) {
                printPath(min);
                return;
            }
            for (int i = 0; i < 4; i++) {
                if (isSafe(min.x + row[i], min.y + col[i])) {
                    Node child = new Node(min.matrix, min.x, min.y, min.x + row[i], min.y + col[i], min.level + 1, min);
                    child.cost = calculateCost(child.matrix, goal);
                    pq.add(child);
                }
            }
        }
    }


    public static void initGoal(int[][] goalState) {
        for (int[] ints : goalState) {
            for (int j = 0; j < goalState.length; j++) {
                System.out.print(ints[j] + " ");
            }
            System.out.println();
        }
    }

    public static void initInitial(int[][] intialState) {
        for (int[] ints : intialState) {
            for (int j = 0; j < intialState.length; j++) {
                System.out.print(ints[j] + " ");
            }
            System.out.println();
```

```
        }
    }

    public static void initState(int[][] intialState, int[][] goalState)
{
        System.out.println("Initial State");
        initInitial(intialState);
        System.out.println();
        System.out.println("Goal State");
        initGoal(goalState);
    }

}
```

- Main.java

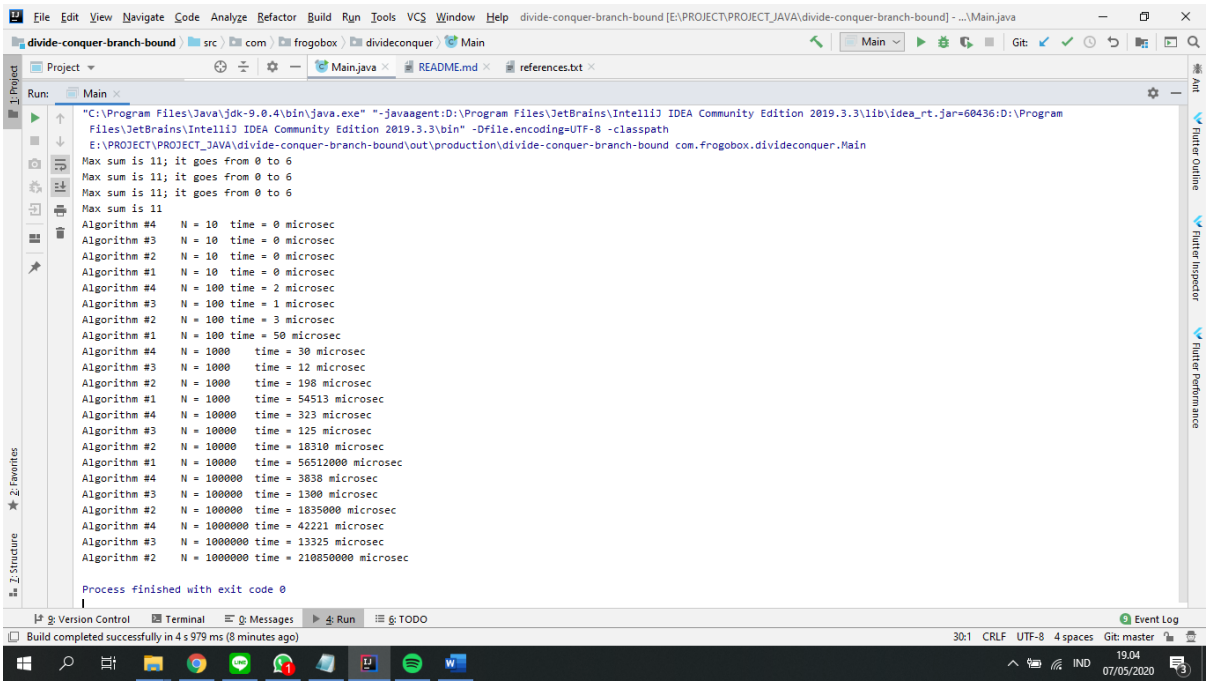
```
package com.frogobox.branchbound;

import static com.frogobox.branchbound.Algorithm.*;

/**
 * Created by Faisal Amir
 * FrogoBox Inc License
 * ========================================
 * divide-conquer-branch-bound
 * Copyright (C) 07/05/2020.
 * All rights reserved
 * ----------------------------------------
 * Name     : Muhammad Faisal Amir
 * E-mail   : faisalamircs@gmail.com
 * Github   : github.com/amirisback
 * LinkedIn : linkedin.com/in/faisalamircs
 * ----------------------------------------
 * FrogoBox Software Industries
 * com.frogobox.branchbound
 */


public class Main {

    public static void main(String[] args) {

        System.out.println("Java program to solve the 8 puzzle problem
(using branch and bound algorithm)");

        // *** SAMPLE DATA ***
        int[][] initial = {{1, 8, 2}, {0, 4, 3}, {7, 6, 5}};
        int[][] goal = {{1, 2, 3}, {4, 5, 6}, {7, 8, 0}};

```

```java
        System.out.println("----------------------------------------");
        initState(initial, goal);
        System.out.println("----------------------------------------");

        // White tile coordinate
        int x = 1, y = 0;

        Algorithm branchBound = new Algorithm();
        if (branchBound.isSolvable(initial)) {
            branchBound.solve(initial, goal, x, y);
        } else {
            System.out.println("The given initial is impossible to solve");
        }
        System.out.println("----------------------------------------");
        System.out.println("Finish Result : ");
        initGoal(goal);

    }

}
```

2. Data (Sample Data)

```java
// *** SAMPLE DATA ***
int[] sampleData = {4, -3, 5, -2, -1, 2, 6, -2};
```

3. Screen Shot Hasil Program



Algoritma Branch And Bound

A. Percobaan

Branch And Bound	
Rancangan	Skenario
- Buat penyelesaian masalah awal sebagai penetapan masalah.Solusi yang ditetapkan merupakan suatu perjalanan lengkap,buat batas tertinggi (*Upper bound*) pada nilai minimum fungsi objektif dengan mencari berbagai kemungkinan perjalanan. Batas ini ditunjukkan dengan f_U, dan lanjutkan ke langkah 2.	1. Masukkan simpul akar ke dalam antrian S. Jika simpul akar adalah simpul solusi yang ingin dicapai, maka solusi telah ditemukan. Pencarian selesai. 2. Jika antrian S kosong, maka solusi tidak ditemukan. Pencarian selesai. 3. Jika S tidak kosong, mata pilih dari antrian simpul yang memiliki *cost* paling kecil. Jika terdapat beberapa simpul dengan nilai *cost* yang minimal, maka pilih satu secara sembarang 4. Jika simpul yang dipilih adalah simpul solusi, maka solusi telah ditemukan. Pencarian selesai. Jika simpul yang dipilih bukan simpul solusi, maka bangkitkan anak-anak dari simpul tersebut. Jika simpul tidak memiliki anak, maka kembali ke langkah 2 5. Untuk setiap anak dari simpul yang dipilih, hitung *cost* dan masukkan anak-anak simpul tersebut ke dalam antrian S 6. Ulangi langkah 2

B. Hasil Percobaan

1. Code Program

- Algorithm.java

```java
package com.frogobox.divideconquer;

import java.util.Random;

/**
 * Created by Faisal Amir
 * FrogoBox Inc License
 * ========================================
 * divide-conquer-branch-bound
 * Copyright (C) 07/05/2020.
 * All rights reserved
 * ----------------------------------------
 * Name     : Muhammad Faisal Amir
 * E-mail   : faisalamircs@gmail.com
 * Github   : github.com/amirisback
 * LinkedIn : linkedin.com/in/faisalamircs
 * ----------------------------------------
```

```
 * FrogoBox Software Industries
 * com.frogobox.divideconquer
 */
public class Algorithm {

    public static int seqStart = 0;
    public static int seqEnd = -1;
    private static Random rand = new Random();
    /**
     * Cubic maximum contiguous subsequence sum algorithm.
     * seqStart and seqEnd represent the actual best sequence.
     */

    /**
     * Cubic maximum contiguous subsequence sum algorithm.
     * seqStart and seqEnd represent the actual best sequence.
     */
    public static int maxSubSum1(int[] a) {
        int maxSum = 0;

        for (int i = 0; i < a.length; i++)
            for (int j = i; j < a.length; j++) {
                int thisSum = 0;

                for (int k = i; k <= j; k++)
                    thisSum += a[k];

                if (thisSum > maxSum) {
                    maxSum = thisSum;
                    seqStart = i;
                    seqEnd = j;
                }
            }

        return maxSum;
    }

    /**
     * Quadratic maximum contiguous subsequence sum algorithm.
     * seqStart and seqEnd represent the actual best sequence.
     */
    public static int maxSubSum2(int[] a) {
        int maxSum = 0;

        for (int i = 0; i < a.length; i++) {
            int thisSum = 0;
            for (int j = i; j < a.length; j++) {
                thisSum += a[j];

                if (thisSum > maxSum) {
```

```
                    maxSum = thisSum;
                    seqStart = i;
                    seqEnd = j;
                }
            }
        }

        return maxSum;
    }

    /**
     * Linear-time maximum contiguous subsequence sum algorithm.
     * seqStart and seqEnd represent the actual best sequence.
     */
    public static int maxSubSum3(int[] a) {
        int maxSum = 0;
        int thisSum = 0;

        for (int i = 0, j = 0; j < a.length; j++) {
            thisSum += a[j];

            if (thisSum > maxSum) {
                maxSum = thisSum;
                seqStart = i;
                seqEnd = j;
            } else if (thisSum < 0) {
                i = j + 1;
                thisSum = 0;
            }
        }

        return maxSum;
    }

    /**
     * Recursive maximum contiguous subsequence sum algorithm.
     * Finds maximum sum in subarray spanning a[left..right].
     * Does not attempt to maintain actual best sequence.
     */
    private static int maxSumRec(int[] a, int left, int right) {
        int maxLeftBorderSum = 0, maxRightBorderSum = 0;
        int leftBorderSum = 0, rightBorderSum = 0;
        int center = (left + right) / 2;

        if (left == right)  // Base case
            return a[left] > 0 ? a[left] : 0;

        int maxLeftSum = maxSumRec(a, left, center);
        int maxRightSum = maxSumRec(a, center + 1, right);

```

```java
        for (int i = center; i >= left; i--) {
            leftBorderSum += a[i];
            if (leftBorderSum > maxLeftBorderSum)
                maxLeftBorderSum = leftBorderSum;
        }

        for (int i = center + 1; i <= right; i++) {
            rightBorderSum += a[i];
            if (rightBorderSum > maxRightBorderSum)
                maxRightBorderSum = rightBorderSum;
        }

        return max3(maxLeftSum, maxRightSum,
                maxLeftBorderSum + maxRightBorderSum);
    }

    /**
     * Return maximum of three integers.
     */
    private static int max3(int a, int b, int c) {
        return a > b ? a > c ? a : c : b > c ? b : c;
    }

    /**
     * Driver for divide-and-conquer maximum contiguous
     * subsequence sum algorithm.
     */
    public static int maxSubSum4(int[] a) {
        return a.length > 0 ? maxSumRec(a, 0, a.length - 1) : 0;
    }

    public static void getTimingInfo(int n, int alg) {
        int[] test = new int[n];

        long startTime = System.currentTimeMillis();
        ;
        long totalTime = 0;

        int i;
        for (i = 0; totalTime < 4000; i++) {
            for (int j = 0; j < test.length; j++)
                // *** SAMPLE DATA ***
                test[j] = rand.nextInt(100) - 50;

            switch (alg) {
                case 1:
                    maxSubSum1(test);
                    break;
                case 2:
                    maxSubSum2(test);
```

```
                    break;
                case 3:
                    maxSubSum3(test);
                    break;
                case 4:
                    maxSubSum4(test);
                    break;
            }

            totalTime = System.currentTimeMillis() - startTime;
        }

        System.out.println("Algorithm #" + alg + "\t"
                + "N = " + test.length
                + "\ttime = " + (totalTime * 1000 / i) + " microsec");
    }

}
```

- Main.java

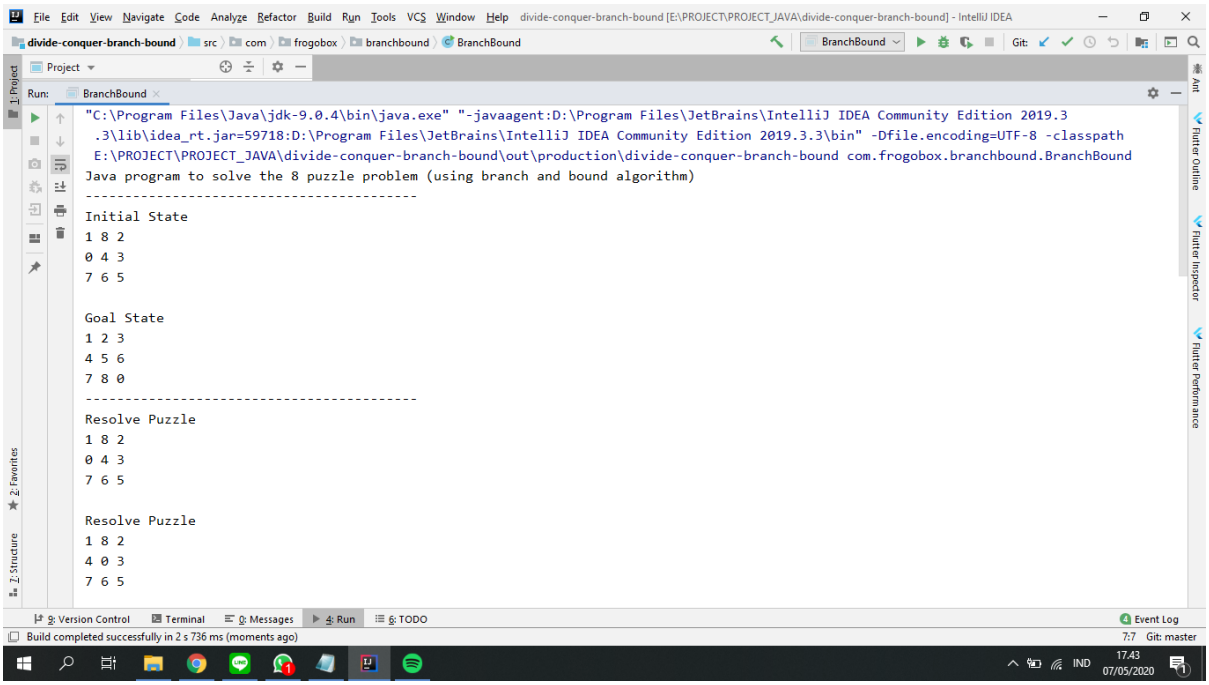
```
package com.frogobox.divideconquer;

import static com.frogobox.divideconquer.Algorithm.*;

/**
 * Created by Faisal Amir
 * FrogoBox Inc License
 * ========================================
 * divide-conquer-branch-bound
 * Copyright (C) 07/05/2020.
 * All rights reserved
 * ----------------------------------------
 * Name     : Muhammad Faisal Amir
 * E-mail   : faisalamircs@gmail.com
 * Github   : github.com/amirisback
 * LinkedIn : linkedin.com/in/faisalamircs
 * ----------------------------------------
 * FrogoBox Software Industries
 * com.frogobox.divideconquer
 */

public final class Main {

    public static void main(String[] args) {
        // *** SAMPLE DATA ***
        int[] sampleData = {4, -3, 5, -2, -1, 2, 6, -2};
        int maxSum;

        maxSum = maxSubSum1(sampleData);
```

```java
        System.out.println("Max sum is " + maxSum + "; it goes from " + seqStart + " to " + seqEnd);
        maxSum = maxSubSum2(sampleData);
        System.out.println("Max sum is " + maxSum + "; it goes from " + seqStart + " to " + seqEnd);
        maxSum = maxSubSum3(sampleData);
        System.out.println("Max sum is " + maxSum + "; it goes from " + seqStart + " to " + seqEnd);
        maxSum = maxSubSum4(sampleData);
        System.out.println("Max sum is " + maxSum);

        // Get some timing info
        for (int n = 10; n <= 1000000; n *= 10) {
            for (int alg = 4; alg >= 1; alg--) {
                if (alg == 1 && n > 50000)
                    continue;
                getTimingInfo(n, alg);
            }
        }

    }
}
```

2. Data (Sample Data)

```java
// *** SAMPLE DATA ***
int[][] initial = {{1, 8, 2}, {0, 4, 3}, {7, 6, 5}};
int[][] goal = {{1, 2, 3}, {4, 5, 6}, {7, 8, 0}};
```

3. Screen Shot Hasil Program



```
Resolve Puzzle
1 0 2
4 8 3
7 6 5

Resolve Puzzle
1 2 0
4 8 3
7 6 5

Resolve Puzzle
1 2 3
4 8 0
7 6 5

Resolve Puzzle
1 2 3
4 8 5
7 6 0

Resolve Puzzle
1 2 3
4 8 5
7 0 6
```

```
7 0 6

Resolve Puzzle
1 2 3
4 0 5
7 8 6

Resolve Puzzle
1 2 3
4 5 0
7 8 6

Resolve Puzzle
1 2 3
4 5 6
7 8 0

----------------------------------------
Finish Result :
1 2 3
4 5 6
7 8 0

Process finished with exit code 0
```

Petunjuk Menjalankan Program

Menggunakan IntelliJ IDEA Jetbrains Community

1. Ikuti tutorial resmi dari IntelliJ - https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html
2. Buka project ini
3. Algorithm Divide and Conquer
   - [root_project]/divide-conquer-branch-bound/src/com/frogobox/divideconquer
   - run - Main.java
4. Algorithm Branch and Bound
   - [root_project]/divide-conquer-branch-bound/src/com/frogobox/branchbound
   - run - Main.java

Menggunakan Command Prompt

1. Buka CMD
2. Pergi ke folder tujuan
   - Algorithm Divide and Conquer
     [root_project]/divide-conquer-branch-bound/src/com/frogobox/divideconquer

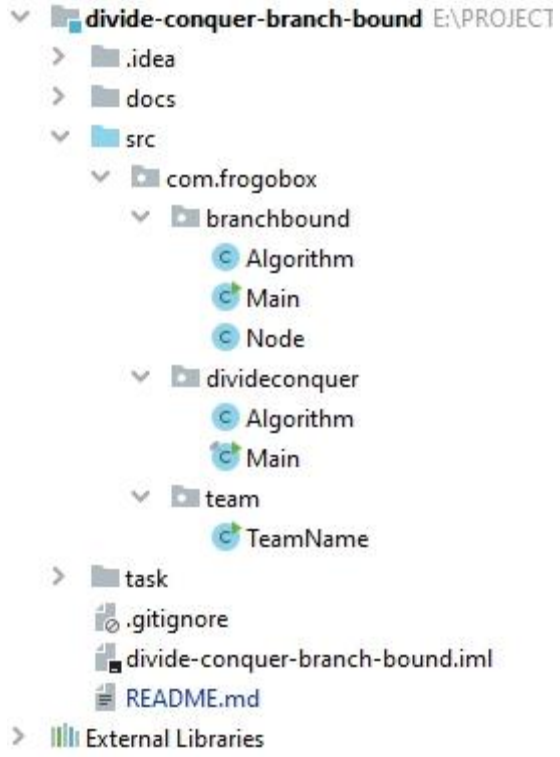
   - Algorithm Branch and Bound
     [root_project]/divide-conquer-branch-bound/src/com/frogobox/branchbound

3. Run code dengan Javac dan Java
   - Algorithm Divide and Conquer
     ```
     $ javac *.java
     $ java com.frogobox.divideconquer.Main
     ```

   - Algorithm Branch and Bound
     ```
     $ javac *.java
     $ java com./frogobox.branchbound.Main
     ```

Strukture Project

Link Project : https://github.com/amirisback/divide-conquer-branch-bound


divide-conquer-branch-bound E:\PROJECT
.idea
docs
src
com.frogobox
branchbound
Algorithm
Main
Node
divideconquer
Algorithm
Main
team
TeamName
task
.gitignore
divide-conquer-branch-bound.iml
README.md
External Libraries