

**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 JetBrains 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
<ul style="list-style-type: none"><li>- Divide : Membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama).</li><li>- Conquer : Memecahkan (menyelesaikan) masing-masing upa-masalah (secara rekursif).</li><li>- Combine : Menggabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula</li></ul>	<ol style="list-style-type: none"><li>1. Pertama-tama lakukan pengurutan terhadap titik-titik dari himpunan S yang diberikan berdasarkan koordinat absis-X, dengan kompleksitas waktu <math>O(n \log n)</math>.</li><li>2. Jika <math> S  \leq 3</math>, maka lakukan pencarian convex hull secara brute-force dengan kompleksitas waktu <math>O(1)</math>. (Basis).</li><li>3. Jika tidak, partisi himpunan titik-titik pada S menjadi 2 buah himpunan A dan B, dimana A terdiri dari setengah jumlah dari <math> S </math> dan titik dengan koordinat absis-X yang terendah dan B terdiri dari setengah dari jumlah <math> S </math> dan titik dengan koordinat absis-X terbesar.</li><li>4. Secara rekursif lakukan penghitungan terhadap <math>H_A = \text{conv}(A)</math> dan <math>H_B = \text{conv}(B)</math>.</li><li>5. Lakukan penggabungan (merge) terhadap kedua hull tersebut menjadi convex hull, H, dengan menghitung dan mencari upper dan lower tangents untuk <math>H_A</math> dan <math>H_B</math> dengan mengabaikan semua titik yang berada diantara dua buah tangen ini.</li></ol>

## B. Hasil Percobaan

### 1. Code Program

- Node.java

1	<code>package com.frogobox.branchbound;</code>
2	
3	<code>/**</code>
4	<code> * Created by Faisal Amir</code>
5	<code> * FrogoBox Inc License</code>
6	<code> * =====</code>
7	<code> * divide-conquer-branch-bound</code>
8	<code> * Copyright (C) 07/05/2020.</code>
9	<code> * ALL rights reserved</code>
10	<code> * -----</code>
11	<code> * Name      : Muhammad Faisal Amir</code>
12	<code> * E-mail    : faisalamircs@gmail.com</code>
13	<code> * Github    : github.com/amirisback</code>
14	<code> * LinkedIn  : linkedin.com/in/faisalamircs</code>
15	<code> * -----</code>
16	<code> * FrogoBox Software Industries</code>
17	<code> * com.frogobox.branchbound</code>
18	<code> */</code>
19	
20	<code>public class Node {</code>
21	
22	<code>    public Node parent;</code>
23	<code>    public int[][] matrix;</code>
24	
25	<code>    // Blank tile cordinates</code>
26	<code>    public int x, y;</code>
27	
28	<code>    // Number of misplaced tiles</code>
29	<code>    public int cost;</code>
30	
31	<code>    // The number of moves so far</code>
32	<code>    public int level;</code>
33	
34	<code>    public Node(int[][] matrix, int x, int y, int newX, int newY, int level, Node parent) {</code>
35	<code>        this.parent = parent;</code>
36	<code>        this.matrix = new int[matrix.length][];</code>
37	<code>        for (int i = 0; i &lt; matrix.length; i++) {</code>
38	<code>            this.matrix[i] = matrix[i].clone();</code>
39	<code>        }</code>
40	
41	<code>        // Swap value</code>

42	<code>        this.matrix[x][y]            = this.matrix[x][y] +</code> <code>this.matrix[newX][newY];</code>
43	<code>        this.matrix[newX][newY] = this.matrix[x][y] -</code> <code>this.matrix[newX][newY];</code>
44	<code>        this.matrix[x][y]            = this.matrix[x][y] -</code> <code>this.matrix[newX][newY];</code>
45	
46	<code>        this.cost = Integer.MAX_VALUE;</code>
47	<code>        this.level = level;</code>
48	<code>        this.x = newX;</code>
49	<code>        this.y = newY;</code>
50	<code>    }</code>
51	
52	<code>}</code>

- Algorithm.java

1	<code>package com.frogobox.branchbound;</code>
2	
3	<code>import java.util.ArrayList;</code>
4	<code>import java.util.List;</code>
5	<code>import java.util.PriorityQueue;</code>
6	
7	<code>/**</code>
8	<code> * Created by Faisal Amir</code>
9	<code> * FrogoBox Inc License</code>
10	<code> * =====</code>
11	<code> * divide-conquer-branch-bound</code>
12	<code> * Copyright (C) 07/05/2020.</code>
13	<code> * All rights reserved</code>
14	<code> * -----</code>
15	<code> * Name      : Muhammad Faisal Amir</code>
16	<code> * E-mail    : faisalamircs@gmail.com</code>
17	<code> * Github    : github.com/amirisback</code>
18	<code> * LinkedIn  : linkedin.com/in/faisalamircs</code>
19	<code> * -----</code>
20	<code> * FrogoBox Software Industries</code>
21	<code> * com.frogobox.branchbound</code>
22	<code> */</code>
23	<code>public class Algorithm {</code>
24	
25	<code>    public int dimension = 3;</code>
26	
27	<code>    // Bottom, left, top, right</code>
28	<code>    int[] row = {1, 0, -1, 0};</code>
29	<code>    int[] col = {0, -1, 0, 1};</code>
30	
31	<code>    public int calculateCost(int[][] initial, int[][] goal) {</code>
32	<code>        int count = 0;</code>
33	<code>        int n = initial.length;</code>
34	<code>        for (int i = 0; i &lt; n; i++) {</code>

35	for (int j = 0; j < n; j++) {
36	if (initial[i][j] != 0 && initial[i][j] != goal[i][j]) {
37	count++;
38	}
39	}
40	}
41	return count;
42	}
43	
44	public void printMatrix(int[][] matrix) {
45	System.out.println("Resolve Puzzle");
46	for (int[] ints : matrix) {
47	for (int j = 0; j < matrix.length; j++) {
48	System.out.print(ints[j] + " ");
49	}
50	System.out.println();
51	}
52	}
53	
54	public boolean isSafe(int x, int y) {
55	return (x >= 0 && x < dimension && y >= 0 && y < dimension);
56	}
57	
58	public void printPath(Node root) {
59	if (root == null) {
60	return;
61	}
62	
63	printPath(root.parent);
64	printMatrix(root.matrix);
65	System.out.println();
66	
67	}
68	
69	public boolean isSolvable(int[][] matrix) {
70	int count = 0;
71	List<Integer> array = new ArrayList<Integer>();
72	
73	for (int[] ints : matrix) {
74	for (int j = 0; j < matrix.length; j++) {
75	array.add(ints[j]);
76	}
77	}
78	
79	Integer[] anotherArray = new Integer[array.size()];
80	array.toArray(anotherArray);
81	
82	for (int i = 0; i < anotherArray.length - 1; i++) {
83	for (int j = i + 1; j < anotherArray.length; j++) {

84	<b>if</b> (anotherArray[i] != 0 && anotherArray[j] != 0 &&
85	anotherArray[i] > anotherArray[j]) {
86	count++;
87	}
88	}
89	
90	<b>return</b> count % 2 == 0;
91	}
92	
93	<b>public void</b> solve( <b>int</b> [][] initial, <b>int</b> [][] goal, <b>int</b> x, <b>int</b> y) {
94	PriorityQueue<Node> pq = <b>new</b> PriorityQueue<Node>(1000, (a, b) ->
95	(a.cost + a.level) - (b.cost + b.level));
96	Node root = <b>new</b> Node(initial, x, y, x, y, 0, <b>null</b> );
97	root.cost = calculateCost(initial, goal);
98	pq.add(root);
99	<b>while</b> (!pq.isEmpty()) {
100	Node min = pq.poll();
101	<b>if</b> (min.cost == 0) {
102	printPath(min);
103	<b>return</b> ;
104	}
105	<b>for</b> ( <b>int</b> i = 0; i < 4; i++) {
106	<b>if</b> (isSafe(min.x + row[i], min.y + col[i])) {
107	Node child = <b>new</b> Node(min.matrix, min.x, min.y,
108	min.x + row[i], min.y + col[i], min.level + 1, min);
109	child.cost = calculateCost(child.matrix, goal);
110	pq.add(child);
111	}
112	}
113	}
114	
115	
116	<b>public static void</b> initGoal( <b>int</b> [][] goalState) {
117	<b>for</b> ( <b>int</b> [] ints : goalState) {
118	<b>for</b> ( <b>int</b> j = 0; j < goalState.length; j++) {
119	System.out.print(ints[j] + " ");
120	}
121	System.out.println();
122	}
123	}
124	
125	<b>public static void</b> initInitial( <b>int</b> [][] initialState) {
126	<b>for</b> ( <b>int</b> [] ints : initialState) {
127	<b>for</b> ( <b>int</b> j = 0; j < initialState.length; j++) {
128	System.out.print(ints[j] + " ");
129	}
130	System.out.println();

131	}
132	}
133	
134	<b>public static void</b> initState( <b>int</b> [][] initialState, <b>int</b> [][] goalState)
135	{
136	System.out.println("Initial State");
137	initInitial(initialState);
138	System.out.println();
139	System.out.println("Goal State");
140	initGoal(goalState);
141	}
142	}

- Main.java

1	<b>package</b> com.frogobox.branchbound;
2	
3	<b>import static</b> com.frogobox.branchbound.Algorithm.*;
4	
5	/**
6	* Created by Faisal Amir
7	* FrogoBox Inc License
8	* =====
9	* divide-conquer-branch-bound
10	* Copyright (C) 07/05/2020.
11	* All rights reserved
12	* -----
13	* Name : Muhammad Faisal Amir
14	* E-mail : faisalamircs@gmail.com
15	* Github : github.com/amirisback
16	* LinkedIn : linkedin.com/in/faisalamircs
17	* -----
18	* FrogoBox Software Industries
19	* com.frogobox.branchbound
20	*/
21	
22	
23	<b>public class</b> Main {
24	
25	<b>public static void</b> main(String[] args) {
26	
27	System.out.println("Java program to solve the 8 puzzle problem (using branch and bound algorithm)");
28	
29	// *** SAMPLE DATA ***
30	<b>int</b> [][] initial = {{1, 8, 2}, {0, 4, 3}, {7, 6, 5}};
31	<b>int</b> [][] goal = {{1, 2, 3}, {4, 5, 6}, {7, 8, 0}};
32	

33	");	System.out.println("-----
34		initState(initial, goal);
35	");	System.out.println("-----
36		
37		// White tile coordinate
38		int x = 1, y = 0;
39		
40		Algorithm branchBound = new Algorithm();
41		if (branchBound.isSolvable(initial)) {
42		branchBound.solve(initial, goal, x, y);
43		} else {
44	solve");	System.out.println("The given initial is impossible to
45		}
46		System.out.println("-----");
47		System.out.println("Finish Result : ");
48		initGoal(goal);
49		
50	}	
51		
52	}	

## 2. Data (Sample Data)

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

## 3. Screen Shot Hasil Program

```

C:\Program Files\Java\jdk-9.0.4\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.3\lib\idea_rt.jar=60436:D:\Program
Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.3\bin" -Dfile.encoding=UTF-8 -classpath
E:\PROJECT\PROJECT_JAVA\divide-conquer-branch-bound\out\production\divide-conquer-branch-bound com.frogobox.divideconquer.Main
Max sum is 11; it goes from 0 to 6
Max sum is 11; it goes from 0 to 6
Max sum is 11; it goes from 0 to 6
Max sum is 11
Algorithm #4 N = 10 time = 0 microsec
Algorithm #3 N = 10 time = 0 microsec
Algorithm #2 N = 10 time = 0 microsec
Algorithm #1 N = 10 time = 0 microsec
Algorithm #4 N = 100 time = 2 microsec
Algorithm #3 N = 100 time = 1 microsec
Algorithm #2 N = 100 time = 3 microsec
Algorithm #1 N = 100 time = 50 microsec
Algorithm #4 N = 1000 time = 30 microsec
Algorithm #3 N = 1000 time = 12 microsec
Algorithm #2 N = 1000 time = 198 microsec
Algorithm #1 N = 1000 time = 54513 microsec
Algorithm #4 N = 10000 time = 323 microsec
Algorithm #3 N = 10000 time = 125 microsec
Algorithm #2 N = 10000 time = 18310 microsec
Algorithm #1 N = 10000 time = 56512000 microsec
Algorithm #4 N = 100000 time = 3838 microsec
Algorithm #3 N = 100000 time = 1300 microsec
Algorithm #2 N = 100000 time = 1835000 microsec
Algorithm #4 N = 1000000 time = 42221 microsec
Algorithm #3 N = 1000000 time = 13325 microsec
Algorithm #2 N = 1000000 time = 210850000 microsec
Process finished with exit code 0

```



# Algoritma Branch And Bound

## A. Percobaan

Branch And Bound	
Rancangan	Skenario
<p>- Buat penyelesaian masalah awal sebagai penetapan masalah. Solusi yang ditetapkan merupakan suatu perjalanan lengkap, buat batas tertinggi (<i>Upper bound</i>) pada nilai minimum fungsi objektif dengan mencari berbagai kemungkinan perjalanan. Batas ini ditunjukkan dengan <math>f_u</math>, dan lanjutkan ke langkah 2.</p>	<ol style="list-style-type: none"> <li>1. Masukkan simpul akar ke dalam antrian S. Jika simpul akar adalah simpul solusi yang ingin dicapai, maka solusi telah ditemukan. Pencarian selesai.</li> <li>2. Jika antrian S kosong, maka solusi tidak ditemukan. Pencarian selesai.</li> <li>3. Jika S tidak kosong, maka pilih dari antrian simpul yang memiliki <i>cost</i> paling kecil. Jika terdapat beberapa simpul dengan nilai <i>cost</i> yang minimal, maka pilih satu secara sembarang.</li> <li>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.</li> <li>5. Untuk setiap anak dari simpul yang dipilih, hitung <i>cost</i> dan masukkan anak-anak simpul tersebut ke dalam antrian S.</li> <li>6. Ulangi langkah 2.</li> </ol>

## B. Hasil Percobaan

### 1. Code Program

- Algorithm.java

1	<code>package com.frogobox.divideconquer;</code>
2	
3	<code>import java.util.Random;</code>
4	
5	<code>/**</code>
6	<code> * Created by Faisal Amir</code>
7	<code> * FrogoBox Inc License</code>
8	<code> * =====</code>
9	<code> * divide-conquer-branch-bound</code>
10	<code> * Copyright (C) 07/05/2020.</code>
11	<code> * All rights reserved</code>
12	<code> * -----</code>
13	<code> * Name : Muhammad Faisal Amir</code>
14	<code> * E-mail : faisalamircs@gmail.com</code>
15	<code> * Github : github.com/amirisback</code>
16	<code> * LinkedIn : linkedin.com/in/faisalamircs</code>
17	<code> * -----</code>

18	<i>* FrogoBox Software Industries</i>
19	<i>* com.frogobox.divideconquer</i>
20	<i>*/</i>
21	<b>public class</b> Algorithm {
22	
23	<b>public static int</b> seqStart = 0;
24	<b>public static int</b> seqEnd = -1;
25	<b>private static</b> Random rand = new Random();
26	<i>/**</i>
27	<i>* Cubic maximum contiguous subsequence sum algorithm.</i>
28	<i>* seqStart and seqEnd represent the actual best sequence.</i>
29	<i>*/</i>
30	
31	<i>/**</i>
32	<i>* Cubic maximum contiguous subsequence sum algorithm.</i>
33	<i>* seqStart and seqEnd represent the actual best sequence.</i>
34	<i>*/</i>
35	<b>public static int</b> maxSubSum1( <b>int</b> [] a) {
36	<b>int</b> maxSum = 0;
37	
38	<b>for</b> ( <b>int</b> i = 0; i < a.length; i++)
39	<b>for</b> ( <b>int</b> j = i; j < a.length; j++) {
40	<b>int</b> thisSum = 0;
41	
42	<b>for</b> ( <b>int</b> k = i; k <= j; k++)
43	thisSum += a[k];
44	
45	<b>if</b> (thisSum > maxSum) {
46	maxSum = thisSum;
47	seqStart = i;
48	seqEnd = j;
49	}
50	}
51	
52	<b>return</b> maxSum;
53	}
54	
55	<i>/**</i>
56	<i>* Quadratic maximum contiguous subsequence sum algorithm.</i>
57	<i>* seqStart and seqEnd represent the actual best sequence.</i>
58	<i>*/</i>
59	<b>public static int</b> maxSubSum2( <b>int</b> [] a) {
60	<b>int</b> maxSum = 0;
61	
62	<b>for</b> ( <b>int</b> i = 0; i < a.length; i++) {
63	<b>int</b> thisSum = 0;
64	<b>for</b> ( <b>int</b> j = i; j < a.length; j++) {
65	thisSum += a[j];
66	
67	<b>if</b> (thisSum > maxSum) {

68	maxSum = thisSum;
69	seqStart = i;
70	seqEnd = j;
71	}
72	}
73	}
74	
75	return maxSum;
76	}
77	
78	/**
79	* Linear-time maximum contiguous subsequence sum algorithm.
80	* seqStart and seqEnd represent the actual best sequence.
81	*/
82	public static int maxSubSum3(int[] a) {
83	int maxSum = 0;
84	int thisSum = 0;
85	
86	for (int i = 0, j = 0; j < a.length; j++) {
87	thisSum += a[j];
88	
89	if (thisSum > maxSum) {
90	maxSum = thisSum;
91	seqStart = i;
92	seqEnd = j;
93	} else if (thisSum < 0) {
94	i = j + 1;
95	thisSum = 0;
96	}
97	}
98	
99	return maxSum;
100	}
101	
102	/**
103	* Recursive maximum contiguous subsequence sum algorithm.
104	* Finds maximum sum in subarray spanning a[left..right].
105	* Does not attempt to maintain actual best sequence.
106	*/
107	private static int maxSumRec(int[] a, int left, int right) {
108	int maxLeftBorderSum = 0, maxRightBorderSum = 0;
109	int leftBorderSum = 0, rightBorderSum = 0;
110	int center = (left + right) / 2;
111	
112	if (left == right) // Base case
113	return a[left] > 0 ? a[left] : 0;
114	
115	int maxLeftSum = maxSumRec(a, left, center);
116	int maxRightSum = maxSumRec(a, center + 1, right);
117	

118	<b>for</b> ( <b>int</b> i = center; i >= left; i--) {
119	leftBorderSum += a[i];
120	<b>if</b> (leftBorderSum > maxLeftBorderSum)
121	maxLeftBorderSum = leftBorderSum;
122	}
123	
124	<b>for</b> ( <b>int</b> i = center + 1; i <= right; i++) {
125	rightBorderSum += a[i];
126	<b>if</b> (rightBorderSum > maxRightBorderSum)
127	maxRightBorderSum = rightBorderSum;
128	}
129	
130	<b>return</b> max3(maxLeftSum, maxRightSum,
131	maxLeftBorderSum + maxRightBorderSum);
132	}
133	
134	/**
135	* Return maximum of three integers.
136	*/
137	<b>private static int</b> max3( <b>int</b> a, <b>int</b> b, <b>int</b> c) {
138	<b>return</b> a > b ? a > c ? a : c : b > c ? b : c;
139	}
140	
141	/**
142	* Driver for divide-and-conquer maximum contiguous
143	* subsequence sum algorithm.
144	*/
145	<b>public static int</b> maxSubSum4( <b>int</b> [] a) {
146	<b>return</b> a.length > 0 ? maxSumRec(a, 0, a.length - 1) : 0;
147	}
148	
149	<b>public static void</b> getTimingInfo( <b>int</b> n, <b>int</b> alg) {
150	<b>int</b> [] test = <b>new int</b> [n];
151	
152	<b>long</b> startTime = System.currentTimeMillis();
153	;
154	<b>long</b> totalTime = 0;
155	
156	<b>int</b> i;
157	<b>for</b> (i = 0; totalTime < 4000; i++) {
158	<b>for</b> ( <b>int</b> j = 0; j < test.length; j++)
159	// *** SAMPLE DATA ***
160	test[j] = <b>rand</b> .nextInt(100) - 50;
161	
162	<b>switch</b> (alg) {
163	<b>case</b> 1:
164	maxSubSum1(test);
165	<b>break</b> ;
166	<b>case</b> 2:
167	maxSubSum2(test);

168	<code>break;</code>
169	<code>case 3:</code>
170	<code>maxSubSum3(test);</code>
171	<code>break;</code>
172	<code>case 4:</code>
173	<code>maxSubSum4(test);</code>
174	<code>break;</code>
175	<code>}</code>
176	
177	<code>totalTime = System.currentTimeMillis() - startTime;</code>
178	<code>}</code>
179	
180	<code>System.out.println("Algorithm #" + alg + "\t"</code>
181	<code>+ "N = " + test.length</code>
182	<code>+ "\ttime = " + (totalTime * 1000 / i) + " microsec");</code>
183	<code>}</code>
184	
185	<code>}</code>

- Main.java

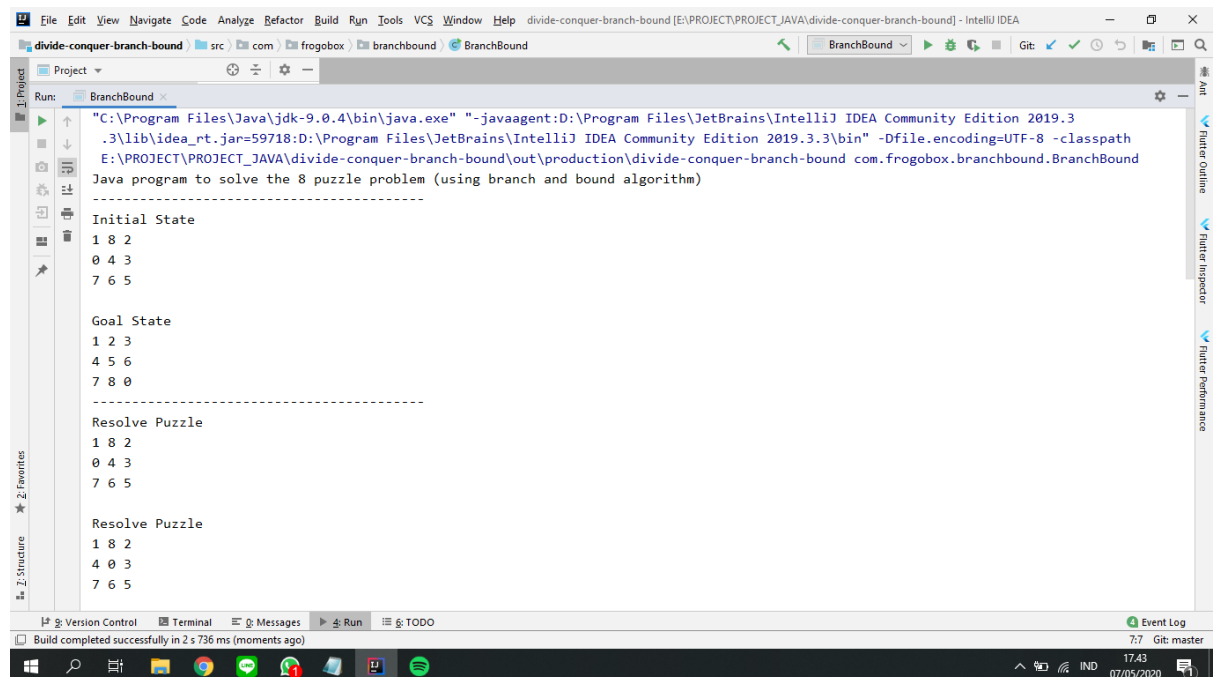
1	<code>package com.frogobox.divideconquer;</code>
2	
3	<code>import static com.frogobox.divideconquer.Algorithm.*;</code>
4	
5	<code>/**</code>
6	<code> * Created by Faisal Amir</code>
7	<code> * FrogoBox Inc License</code>
8	<code> * =====</code>
9	<code> * divide-conquer-branch-bound</code>
10	<code> * Copyright (C) 07/05/2020.</code>
11	<code> * All rights reserved</code>
12	<code> * -----</code>
13	<code> * Name : Muhammad Faisal Amir</code>
14	<code> * E-mail : faisalamircs@gmail.com</code>
15	<code> * Github : github.com/amirisback</code>
16	<code> * LinkedIn : linkedin.com/in/faisalamircs</code>
17	<code> * -----</code>
18	<code> * FrogoBox Software Industries</code>
19	<code> * com.frogobox.divideconquer</code>
20	<code> */</code>
21	
22	<code>public final class Main {</code>
23	
24	<code>public static void main(String[] args) {</code>
25	<code>    // *** SAMPLE DATA ***</code>
26	<code>int[] sampleData = {4, -3, 5, -2, -1, 2, 6, -2};</code>
27	<code>int maxSum;</code>
28	
29	<code>maxSum = maxSubSum1(sampleData);</code>

30	System.out.println("Max sum is " + maxSum + "; it goes from " + seqStart + " to " + seqEnd);
31	maxSum = maxSubSum2(sampleData);
32	System.out.println("Max sum is " + maxSum + "; it goes from " + seqStart + " to " + seqEnd);
33	maxSum = maxSubSum3(sampleData);
34	System.out.println("Max sum is " + maxSum + "; it goes from " + seqStart + " to " + seqEnd);
35	maxSum = maxSubSum4(sampleData);
36	System.out.println("Max sum is " + maxSum);
37	
38	// Get some timing info
39	for (int n = 10; n <= 1000000; n *= 10) {
40	for (int alg = 4; alg >= 1; alg--) {
41	if (alg == 1 && n > 50000)
42	continue;
43	getTimingInfo(n, alg);
44	}
45	}
46	
47	}
48	}

## 2. Data (Sample Data)

```
// *** 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



IntelliJ IDEA interface showing the first run of the BranchBound program. The Run console displays the following output:

```
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
```

The status bar indicates the build completed successfully in 2 s 736 ms (moments ago). The system tray shows the time as 17:43 on 07/05/2020.

IntelliJ IDEA interface showing the second run of the BranchBound program. The Run console displays the following output:

```
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
```

The status bar indicates the build completed successfully in 2 s 736 ms (moments ago). The system tray shows the time as 17:43 on 07/05/2020.

# 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