

**Tugas Kecil 3 IF2211 Strategi Algoritma**  
**Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch & Bound***



**Adelline Kania Setiyawan**

**13520084 / K-03**

**Program Studi Sarjana Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2022**

## A. Cara Kerja Program dengan Algoritma *Branch & Bound*

1. Pertama kali, program akan menerima input berupa pilihan 15 Puzzle yang diinginkan oleh *user*, yaitu berdasarkan file input atau secara random. Apabila *user* memilih file input, program akan meminta input file dalam format txt yang berisi 15 Puzzle yang akan diselesaikan, sedangkan apabila *user* memilih secara random, program akan *men-generate* 15 Puzzle secara random.
2. Setelah data untuk 15 Puzzle sudah tersedia, program akan mengecek terlebih dahulu apakah 15 Puzzle tersebut dapat diselesaikan atau tidak dengan memanggil fungsi `findKurang()` yang akan mengembalikan sebuah integer. Fungsi `findKurang()` akan menghitung banyaknya ubin bernomor  $j$  sedemikian sehingga  $j < i$  dan  $\text{Posisi}(j) > \text{Posisi}(i)$ . Apabila nilainya berupa bilangan genap, maka 15 Puzzle tersebut dapat diselesaikan, sedangkan apabila nilai berupa bilangan ganjil, 15 Puzzle tersebut tidak dapat diselesaikan.
3. Apabila 15 Puzzle tidak dapat diselesaikan, program akan menampilkan output berupa pesan bahwa 15 Puzzle tidak bisa diselesaikan.
4. Apabila 15 Puzzle dapat diselesaikan, 15 Puzzle yang awalnya disimpan dalam struktur data array dua dimensi akan disimpan dalam suatu kelas *Tree* yang memiliki atribut berupa *parent*, *child*, *data*, *countC*, *count*, *direction*, *isSolution*. Data atau matriks dari 15 Puzzle akan disimpan juga pada sebuah *priority queue* yang terurut berdasarkan *cost* puzzle tersebut. Setelah itu, 15 Puzzle akan diselesaikan dengan memanggil prosedur `solve15Puzzle()` dengan *queue* sebagai parameternya.
5. Pada prosedur `solve15Puzzle()`, program akan *men-dequeue* elemen pertama pada *queue* untuk mendapatkan simpul dengan matriks atau data dengan *cost* yang minimum. Kemudian, data dari simpul tersebut akan dicek apakah data tersebut sudah pernah ditelusuri sebelumnya dengan melihat *dictionary visitedData*. Apabila puzzle tersebut sudah pernah ditelusuri sebelumnya, program akan terus mencari puzzle yang belum pernah ditelusuri untuk dijadikan *minTree*.
6. Setelah *minTree* ditemukan, akan dicek apakah *minTree* sudah merupakan solusi dengan melihat nilai  $\hat{g}(P)$  bernilai 0. Jika tidak, akan dicari semua kemungkinan *child* dari *minTree*. Kemudian, akan ditentukan juga *cost* untuk setiap *childnya* dengan menggunakan rumus

$$\hat{c}(P) = f(P) + \hat{g}(P)$$

dengan

$\hat{c}(P)$  = cost simpul  $p$

$f(P)$  = panjang lintasan dari simpul akar ke  $P$

$\hat{g}(P)$  = taksiran panjang lintasan terpendek dari  $P$  ke simpul solusi pada upapohon yang akarnya  $P$

7. Semua *child* dari minTree akan dimasukkan ke dalam *priority queue* yang secara otomatis akan mengurutkan berdasarkan *cost* yang terendah hingga tertinggi. Setelah itu, akan dilakukan pengulangan pada tahap ke-5 hingga ditemukan sebuah minTree dengan nilai  $\hat{g}(P)$  adalah nol. Apabila sudah berhasil ditemukan, akan ditampilkan semua urutan puzzle dari posisi awal hingga akhir

## B. Input dan Output Program

### *Input & Output 1*

```
Jenis Input Data 15 Puzzle yang Diberikan:
1. File
2. Random
(Pilih 1 atau 2)
>>> 1

Masukkan path file...
(ex: ./tesisolved.txt)
>>> solved1.txt

15 Puzzle yang Dihasilkan:
[1, 2, 3, 4]
[5, 6, 16, 8]
[9, 10, 7, 11]
[13, 14, 15, 12]
Jumlah Nilai Kurang: 16

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 16, 11]
[13, 14, 15, 12]
```

```
Pergeseran tile kosong: right
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 16]
[13, 14, 15, 12]

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, 16]
```

### *Input & Output 2*

```
Jenis Input Data 15 Puzzle yang Diberikan:
1. File
2. Random
(Pilih 1 atau 2)
>>> 1

Masukkan path file...
(ex: ./tesisolved.txt)
>>> notsolved1.txt

15 Puzzle yang Dihasilkan:
[13, 10, 3, 11]
[12, 7, 6, 5]
[4, 9, 2, 1]
[14, 15, 8, 16]
Jumlah Nilai Kurang: 59

15 Puzzle Tidak Bisa Diselesaikan....
```

### Input & Output 3

```
Jenis Input Data 15 Puzzle yang Diberikan:
1. File
2. Random
(Pilih 1 atau 2)
>>> 1

Masukkan path file...
(ex: ./tes1solved.txt)
>>> solved2.txt

15 Puzzle yang Dihasilkan:
[1, 6, 2, 3]
[5, 16, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]
Jumlah Nilai Kurang: 22

Pergeseran tile kosong: up
[1, 16, 2, 3]
[5, 6, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: right
[1, 2, 16, 3]
[5, 6, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: right
[1, 2, 3, 16]
[5, 6, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]
```

```
Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 8, 16]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: left
[1, 2, 3, 4]
[5, 6, 16, 8]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 16, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: right
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 16]
[13, 14, 15, 12]

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, 16]
```

### Input & Output 4

```
Jenis Input Data 15 Puzzle yang Diberikan:
1. File
2. Random
(Pilih 1 atau 2)
>>> 1

Masukkan path file...
(ex: ./tes1solved.txt)
>>> notsolved2.txt

15 Puzzle yang Dihasilkan:
[10, 13, 1, 15]
[3, 7, 9, 12]
[4, 11, 2, 5]
[14, 8, 6, 16]
Jumlah Nilai Kurang: 55

15 Puzzle Tidak Bisa Diselesaikan....
```

### Input & Output 5

```
Jenis Input Data 15 Puzzle yang Diberikan:
1. File
2. Random
(Pilih 1 atau 2)
>>> 1

Masukkan path file...
(ex: ./tes1solved.txt)
>>> solved3.txt

15 Puzzle yang Dihasilkan:
[1, 6, 2, 3]
[16, 10, 8, 4]
[5, 14, 7, 11]
[9, 13, 15, 12]
Jumlah Nilai Kurang: 32

Pergeseran tile kosong: down
[1, 6, 2, 3]
[5, 10, 8, 4]
[16, 14, 7, 11]
[9, 13, 15, 12]

Pergeseran tile kosong: down
[1, 6, 2, 3]
[5, 10, 8, 4]
[9, 14, 7, 11]
[16, 13, 15, 12]

Pergeseran tile kosong: up
[1, 6, 2, 3]
[5, 10, 8, 4]
[9, 16, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: up
[1, 6, 2, 3]
[5, 16, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: up
[1, 16, 2, 3]
[5, 6, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: right
[1, 2, 16, 3]
[5, 6, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: right
[1, 2, 3, 16]
[5, 6, 8, 4]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 8, 16]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: left
[1, 2, 3, 4]
[5, 6, 16, 8]
[9, 10, 7, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 16, 11]
[13, 14, 15, 12]

Pergeseran tile kosong: right
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 16]
[13, 14, 15, 12]

Pergeseran tile kosong: down
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, 16]
```

### Input & Output 6

```
Jenis Input Data 15 Puzzle yang Diberikan:
1. File
2. Random
(Pilih 1 atau 2)
>>> 2

15 Puzzle yang Dihasilkan:
[8, 5, 7, 10]
[14, 6, 1, 15]
[11, 2, 13, 9]
[12, 4, 3, 16]
Jumlah Nilai Kurang: 55

15 Puzzle Tidak Bisa Diselesaikan....
```

### C. Checklist Point

Poin	Ya	Tidak
1. Program berhasil dikompilasi	V	
2. Program berhasil <i>running</i>	V	

3. Program dapat menerima input dan menuliskan output	V	
4. Luaran sudah benar untuk semua data uji	V	
5. Bonus dibuat		V

## D. Kode Program

## E. Test Case

<i>solved1.txt</i>	<i>solved2.txt</i>	<i>solved3.txt</i>
<pre> ≡ solved1.txt 1    1 2 3 4 2    5 6 16 8 3    9 10 7 11 4    13 14 15 12 </pre>	<pre> ≡ solved2.txt 1    1 6 2 3 2    5 16 8 4 3    9 10 7 11 4    13 14 15 12 </pre>	<pre> ≡ solved3.txt 1    1 6 2 3 2    16 10 8 4 3    5 14 7 11 4    9 13 15 12 </pre>

<i>notsolved1.txt</i>	<i>notsolved2.txt</i>
<pre> ≡ notsolved1.txt 1    13 10 3 11 2    12 7 6 5 3    4 9 2 1 4    14 15 8 16 </pre>	<pre> ≡ notsolved2.txt 1    10 13 1 15 2    3 7 9 12 3    4 11 2 5 4    14 8 6 16 </pre>

## F. Alamat Source Code

<https://github.com/adellinekania/Tucil3-15Puzzle>