

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



Gede Prasadha Bhawarnawa

K01

13520004

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2021

DAFTAR ISI

DAFTAR ISI.....	1
BAB 1	2
BAB 2	4
BAB 3	11
BAB 4	18
LAMPIRAN.....	19

BAB 1

Cara Kerja Algoritma *Branch and Bound* Pada Program

Algoritma *Branch and Bound* (B&B) adalah algoritma yang umum digunakan untuk persoalan optimasi, atau persoalan-persoalan yang perlu memaksimumkan atau meminimumkan suatu fungsi objektif tanpa melanggar fungsi batasan atau *constraints*. Algoritma ini menggabungkan konsep *Breadth-First Search* (BFS) dan *least-cost search*. Bila pada algoritma BFS hanya menggunakan aturan FIFO untuk pembangkitan simpul baru, algoritma B&B menggunakan nilai taksiran lintasan tercepat atau dengan *cost* menuju simpul tujuan atau *goal node* yang minimum untuk menentukan urutan ekspansi simpul, untuk kasus minimasi. Pada kasus maksimasi, maka yang diprioritaskan adalah simpul dengan nilai taksiran *cost* maksimum. Bila terdapat kasus dimana terdapat beberapa jalur untuk mencapai target solusi, maka dapat digunakan fungsi pembatas atau fungsi pemangkasan simpul dengan “membunuh” semua simpul yang estimasi *cost*-nya melebihi total *cost* realita pertama untuk mencapai simpul goal.

Pada program ini, penulis pertama-tama mendesain algoritma untuk menentukan apakah sebuah 15-puzzle dapat diselesaikan atau tidak. Penentuan ini menggunakan fungsi KURANG(i), atau akumulasi semua nilai KURANG dari setiap petak pada puzzle. Sebuah petak memiliki nilai KURANG setara dengan jumlah petak dengan nilai lebih kecil dari petak indeks dengan cara membaca matriks secara kontigu. Lalu, fungsi KURANG(i) ditambahkan dengan nilai X. Nilai X adalah fungsi boolean yang mengembalikan 1 (TRUE) bila posisi 16 atau tile kosong pada puzzle pada indeks ganjil dan 0 (FALSE) bila posisinya pada indeks genap, menggunakan asumsi puzzle adalah array *zero-based*. Bila hasil dari KURANG(i) adalah genap, maka puzzle memiliki solusi. Sebaliknya, jika hasilnya ganjil, maka puzzle tidak memiliki solusi.

Nilai KURANG(i) selain untuk menentukan ada tidaknya solusi pada sebuah puzzle, juga dapat menjadi estimasi banyaknya langkah dan jumlah simpul yang dibangkitkan untuk mencapai solusi. Semakin tinggi nilai KURANG(i), maka semakin jauh juga simpul solusi dari simpul posisi awal puzzle. Ini dikarenakan urutan target adalah 1 sampai 16 dan dengan nilai KURANG(i) yang besar, artinya terdapat banyak bilangan bernilai besar pada bagian awal/atas puzzle sehingga dibutuhkan lebih banyak langkah untuk mencari solusi karena perlu ditukar terlebih dahulu posisi bilangan bernilai besar dengan yang bernilai kecil.

Setelah menentukan ada tidaknya solusi untuk sebuah puzzle, dengan asumsi puzzle dapat dicari solusinya, maka selanjutnya adalah pembentukan simpul dan menentukan *cost* dari simpul. Pembentukan simpul menggunakan batasan (*constraint*) bahwa pada puzzle petak kosong hanya bisa ditukar dengan petak yang tersedia dan berada pada kanan, kiri, atas, atau bawah dari petak kosong, dengan asumsi petak yang ditukar ada (bukan batas ujung dari puzzle). Untuk setiap *move* atau pertukaran yang dilakukan terhadap simpul puzzle awal, maka dihasilkan satu buah simpul baru dengan *cost* yang dapat dihitung menggunakan penjelasan selanjutnya.

Pada program ini, dihitung nilai dari *cost* setiap simpul dengan menggunakan fungsi berikut:

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

Dengan nilai dari:

- a. $\hat{c}(P)$ adalah nilai *cost* dari suatu simpul X
- b. $\hat{f}(P)$ adalah panjang dari simpul awal hingga simpul X atau jumlah langkah yang dibutuhkan dari puzzle awal hingga mencapai puzzle X
- c. $\hat{g}(P)$ adalah estimasi *cost* untuk mencapai simpul solusi dari simpul X. Merupakan jumlah *tile* atau kotak yang tidak sesuai dengan target puzzle akhir pada simpul X.

Dari setiap simpul yang telah dibangkitkan beserta dengan nilai *cost* pada masing-masing simpul, informasi dari semua simpul yang baru dibangkitkan maupun yang telah dibangkitkan namun belum diperiksa akan ditambahkan ke dalam sebuah *priority queue*. *Priority queue* ini akan menganalisis simpul yang nilai *costnya* minimum pada *queue* dari semua simpul yang telah dibangkitkan sebelumnya sampai ditemukan simpul target.

Setelah ditemukan simpul target, maka akan di-*print* ke dalam sebuah text file (ekstensi .txt) informasi mengenai simpul awal, nilai KURANG(i) untuk setiap petak dan nilai akumulasinya, waktu yang dibutuhkan untuk menemukan hasil, jumlah simpul yang dibangkitkan, serta jumlah langkah yang dibutuhkan untuk mencapai tujuan. Selain itu, akan di-*print* juga langkah serta matriks puzzle dari awal hingga dicapai simpul tujuan.

BAB 2

Screenshot Input/Output Program dan Checklist Spesifikasi

Berikut adalah aliran *Input/Output* untuk beberapa testcase. File input harus diletakkan didalam folder “input” pada directory yang setingkat dengan folder “src”. Semua output akan disimpan di dalam sebuah text file pada folder “output”. Terdapat deskripsi dan pseudotext ‘U’, ‘D’, ‘R’, dan ‘L’ pada list semua langkah yang dilakukan. ‘U’ berarti langkah yang dilakukan adalah tukar ke atas, ‘D’ berarti langkah yang dilakukan adalah tukar ke bawah, ‘R’ berarti langkah yang dilakukan adalah tukar ke kanan, dan ‘L’ berarti langkah yang dilakukan adalah tukar ke kiri. Untuk pemilihan antara *random-generated* matriks atau *user-based text-file input*, dapat dilakukan pada terminal seperti pada contoh berikut:

1. Testcase “inputPossible1”

a. Input

```
C:\Users\Gede Prasadha\Documents\Kuliah Semester 4\15-Puzzle-Solver-with-Python\src>py main.py
Selamat datang di program 15-Puzzle Solver
Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): output1
Silahkan pilih metode input matriks puzzle
1. Random
2. Read From File
Silahkan masukkan metode pilihan anda: 2
Masukkan nama file (tanpa extension .txt): inputPossible1
Hasil perhitungan dan kalkulasi sudah ditulis di file C:/Users/Gede Prasadha/Documents/Kuliah Semester 4/15-Puzzle-Solver-with-Python/output/output1.txt
```

b. Output

```
output1.txt U X
output > output1.txt
1  Anda memakai input dari file inputPossible1.txt
2  1 2 3 4
3  5 6 16 8
4  9 10 7 11
5  13 14 15 12
6
7  Elemen 1 memiliki nilai KURANG(i) senilai 0
8  Elemen 2 memiliki nilai KURANG(i) senilai 0
9  Elemen 3 memiliki nilai KURANG(i) senilai 0
10 Elemen 4 memiliki nilai KURANG(i) senilai 0
11 Elemen 5 memiliki nilai KURANG(i) senilai 0
12 Elemen 6 memiliki nilai KURANG(i) senilai 0
13 Elemen X memiliki nilai KURANG(i) senilai 1
14 Elemen 16 memiliki nilai KURANG(i) senilai 9
15 Elemen 8 memiliki nilai KURANG(i) senilai 1
16 Elemen 9 memiliki nilai KURANG(i) senilai 1
17 Elemen 10 memiliki nilai KURANG(i) senilai 1
18 Elemen 7 memiliki nilai KURANG(i) senilai 0
19 Elemen 11 memiliki nilai KURANG(i) senilai 0
20 Elemen 13 memiliki nilai KURANG(i) senilai 1
21 Elemen 14 memiliki nilai KURANG(i) senilai 1
22 Elemen 15 memiliki nilai KURANG(i) senilai 1
23 Elemen 12 memiliki nilai KURANG(i) senilai 0
24 Posisi awal puzzle memiliki solusi dengan nilai KURANG(i) sebesar 16
25
26 Waktu yang dibutuhkan: 0.00102 detik
27 Jumlah node yang dibuat: 9
28 Jumlah langkah yang dilakukan: 3
29 Semua langkah yang dilakukan: ['D', 'R', 'D']
```

```

26 Waktu yang dibutuhkan: 0.00102 detik
27 Jumlah node yang dibuat: 9
28 Jumlah langkah yang dilakukan: 3
29 Semua langkah yang dilakukan: ['D', 'R', 'D']
30 Langkah ke-1 :
31 Move yang dilakukan: Bawah
32 1 2 3 4
33 5 6 7 8
34 9 10 16 11
35 13 14 15 12
36
37 Langkah ke-2 :
38 Move yang dilakukan: Kanan
39 1 2 3 4
40 5 6 7 8
41 9 10 11 16
42 13 14 15 12
43
44 Langkah ke-3 :
45 Move yang dilakukan: Bawah
46 1 2 3 4
47 5 6 7 8
48 9 10 11 12
49 13 14 15 16

```

2. Testcase “inputPossible2”

a. Input

```

C:\Users\Gede Prasidha\Documents\Kuliah Semester 4\15-Puzzle-Solver-with-Python\src>py main.py
Selamat datang di program 15-Puzzle Solver
Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): output2
Silahkan pilih metode input matriks puzzle
1. Random
2. Read From File
Silahkan masukkan metode pilihan anda: 2
Masukkan nama file (tanpa extension .txt): inputPossible2
Hasil perhitungan dan kalkulasi sudah ditulis di file C:/Users/Gede Prasidha/Documents/Kuliah Semester 4/15-Puzzle-Solver-with-Python/output/output2.txt

```

b. Output

```

≡ output2.txt U X
output > ≡ output2.txt
1  Anda memakai input dari file inputPossible2.txt
2  5 1 3 4
3  9 2 7 8
4  16 6 15 11
5  13 10 14 12
6
7  Elemen 5 memiliki nilai KURANG(i) senilai 4
8  Elemen 1 memiliki nilai KURANG(i) senilai 0
9  Elemen 3 memiliki nilai KURANG(i) senilai 1
10 Elemen 4 memiliki nilai KURANG(i) senilai 1
11 Elemen 9 memiliki nilai KURANG(i) senilai 4
12 Elemen 2 memiliki nilai KURANG(i) senilai 0
13 Elemen 7 memiliki nilai KURANG(i) senilai 1
14 Elemen 8 memiliki nilai KURANG(i) senilai 1
15 Elemen X memiliki nilai KURANG(i) senilai 0
16 Elemen 16 memiliki nilai KURANG(i) senilai 7
17 Elemen 6 memiliki nilai KURANG(i) senilai 0
18 Elemen 15 memiliki nilai KURANG(i) senilai 5
19 Elemen 11 memiliki nilai KURANG(i) senilai 1
20 Elemen 13 memiliki nilai KURANG(i) senilai 2
21 Elemen 10 memiliki nilai KURANG(i) senilai 0
22 Elemen 14 memiliki nilai KURANG(i) senilai 1
23 Elemen 12 memiliki nilai KURANG(i) senilai 0
24 Posisi awal puzzle memiliki solusi dengan nilai KURANG(i) sebesar 28
25
26 Waktu yang dibutuhkan: 0.00251 detik
27 Jumlah node yang dibuat: 24
28 Jumlah langkah yang dilakukan: 10
29 Semua langkah yang dilakukan: ['U', 'U', 'R', 'D', 'D', 'D', 'R', 'U', 'R', 'D']

```

```

30 Langkah ke-1 :
31 Move yang dilakukan: Atas
32 5 1 3 4
33 16 2 7 8
34 9 6 15 11
35 13 10 14 12
36
37 Langkah ke-2 :
38 Move yang dilakukan: Atas
39 16 1 3 4
40 5 2 7 8
41 9 6 15 11
42 13 10 14 12
43
44 Langkah ke-3 :
45 Move yang dilakukan: Kanan
46 1 16 3 4
47 5 2 7 8
48 9 6 15 11
49 13 10 14 12
50
51 Langkah ke-4 :
52 Move yang dilakukan: Bawah
53 1 2 3 4
54 5 16 7 8
55 9 6 15 11
56 13 10 14 12
57
58 Langkah ke-5 :
59 Move yang dilakukan: Bawah
60 1 2 3 4
61 5 6 7 8
62 9 16 15 11
63 13 10 14 12
64
65 Langkah ke-6 :
66 Move yang dilakukan: Bawah
67 1 2 3 4
68 5 6 7 8
69 9 10 15 11
70 13 16 14 12
71
72 Langkah ke-7 :
73 Move yang dilakukan: Kanan
74 1 2 3 4
75 5 6 7 8
76 9 10 15 11
77 13 14 16 12
78
79 Langkah ke-8 :
80 Move yang dilakukan: Atas
81 1 2 3 4
82 5 6 7 8
83 9 10 16 11
84 13 14 15 12
85
86 Langkah ke-9 :
87 Move yang dilakukan: Kanan
88 1 2 3 4
89 5 6 7 8
90 9 10 11 16
91 13 14 15 12
92
93 Langkah ke-10 :
94 Move yang dilakukan: Bawah
95 1 2 3 4
96 5 6 7 8
97 9 10 11 12
98 13 14 15 16
99

```

3. Testcase “inputPossible3”

a. Input

```
C:\Users\Gede Prasidha\Documents\Kuliah Semester 4\15-Puzzle-Solver-with-Python\src>py main.py
Selamat datang di program 15-Puzzle Solver
Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): output3
Silahkan pilih metode input matriks puzzle
1. Random
2. Read From File
Silahkan masukkan metode pilihan anda: 2
Masukkan nama file (tanpa extension .txt): inputPossible3
Hasil perhitungan dan kalkulasi sudah ditulis di file C:/Users/Gede Prasidha/Documents/Kuliah Semester 4/15-Puzzle-Solver-with-Python/output/output3.txt
```

b. Output

```
output3.txt u x
output > output3.txt
1  Anda memakai input dari file inputPossible3.txt
2  6 5 2 4
3  9 1 3 8
4  10 16 7 15
5  13 14 12 11
6
7  Elemen 6 memiliki nilai KURANG(i) senilai 5
8  Elemen 5 memiliki nilai KURANG(i) senilai 4
9  Elemen 2 memiliki nilai KURANG(i) senilai 1
10 Elemen 4 memiliki nilai KURANG(i) senilai 2
11 Elemen 9 memiliki nilai KURANG(i) senilai 4
12 Elemen 1 memiliki nilai KURANG(i) senilai 0
13 Elemen 3 memiliki nilai KURANG(i) senilai 0
14 Elemen 8 memiliki nilai KURANG(i) senilai 1
15 Elemen 10 memiliki nilai KURANG(i) senilai 1
16 Elemen X memiliki nilai KURANG(i) senilai 1
17 Elemen 16 memiliki nilai KURANG(i) senilai 6
18 Elemen 7 memiliki nilai KURANG(i) senilai 0
19 Elemen 15 memiliki nilai KURANG(i) senilai 4
20 Elemen 13 memiliki nilai KURANG(i) senilai 2
21 Elemen 14 memiliki nilai KURANG(i) senilai 2
22 Elemen 12 memiliki nilai KURANG(i) senilai 1
23 Elemen 11 memiliki nilai KURANG(i) senilai 0
24 Posisi awal puzzle memiliki solusi dengan nilai KURANG(i) sebesar 34
25
26 Waktu yang dibutuhkan: 0.05870 detik
27 Jumlah node yang dibuat: 636
28 Jumlah langkah yang dilakukan: 17
29 Semua langkah yang dilakukan: ['L', 'U', 'U', 'R', 'D', 'L', 'U', 'R', 'R', 'D', 'D', 'R', 'D', 'L', 'U', 'R', 'D']
```

30 Langkah ke-1 :	65 Langkah ke-6 :
31 Move yang dilakukan: Kiri	66 Move yang dilakukan: Kiri
32 6 5 2 4	67 5 1 2 4
33 9 1 3 8	68 16 6 3 8
34 16 10 7 15	69 9 10 7 15
35 13 14 12 11	70 13 14 12 11
36	71
37 Langkah ke-2 :	72 Langkah ke-7 :
38 Move yang dilakukan: Atas	73 Move yang dilakukan: Atas
39 6 5 2 4	74 16 1 2 4
40 16 1 3 8	75 5 6 3 8
41 9 10 7 15	76 9 10 7 15
42 13 14 12 11	77 13 14 12 11
43	78
44 Langkah ke-3 :	79 Langkah ke-8 :
45 Move yang dilakukan: Atas	80 Move yang dilakukan: Kanan
46 16 5 2 4	81 1 16 2 4
47 6 1 3 8	82 5 6 3 8
48 9 10 7 15	83 9 10 7 15
49 13 14 12 11	84 13 14 12 11
50	85
51 Langkah ke-4 :	86 Langkah ke-9 :
52 Move yang dilakukan: Kanan	87 Move yang dilakukan: Kanan
53 5 16 2 4	88 1 2 16 4
54 6 1 3 8	89 5 6 3 8
55 9 10 7 15	90 9 10 7 15
56 13 14 12 11	91 13 14 12 11
57	92
58 Langkah ke-5 :	93 Langkah ke-10 :
59 Move yang dilakukan: Bawah	94 Move yang dilakukan: Bawah
60 5 1 2 4	95 1 2 3 4
61 6 16 3 8	96 5 6 16 8
62 9 10 7 15	97 9 10 7 15
63 13 14 12 11	98 13 14 12 11

4. Testcase “inputImpossible1”

a. Input

```
C:\Users\Gede Prasidha\Documents\Kuliah Semester 4\15-Puzzle-Solver-with-Python\src>py main.py
Selamat datang di program 15-Puzzle Solver
Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): output4
Silahkan pilih metode input matriks puzzle
1. Random
2. Read From File
Silahkan masukkan metode pilihan anda: 2
Masukkan nama file (tanpa extension .txt): inputImpossible1
Hasil perhitungan dan kalkulasi sudah ditulis di file C:/Users/Gede Prasidha/Documents/Kuliah Semester 4/15-Puzzle-Solver-with-Python/output/output4.txt
```

b. Output

```
output4.txt U X
output > output4.txt
1  Anda memakai input dari file inputImpossible1.txt
2  11 3 15 9
3  13 6 7 8
4  4 1 14 12
5  10 16 2 5
6
7  Elemen 11 memiliki nilai KURANG(i) senilai 10
8  Elemen 3 memiliki nilai KURANG(i) senilai 2
9  Elemen 15 memiliki nilai KURANG(i) senilai 12
10 Elemen 9 memiliki nilai KURANG(i) senilai 7
11 Elemen 13 memiliki nilai KURANG(i) senilai 9
12 Elemen 6 memiliki nilai KURANG(i) senilai 4
13 Elemen 7 memiliki nilai KURANG(i) senilai 4
14 Elemen 8 memiliki nilai KURANG(i) senilai 4
15 Elemen 4 memiliki nilai KURANG(i) senilai 2
16 Elemen 1 memiliki nilai KURANG(i) senilai 0
17 Elemen 14 memiliki nilai KURANG(i) senilai 4
18 Elemen 12 memiliki nilai KURANG(i) senilai 3
19 Elemen 10 memiliki nilai KURANG(i) senilai 2
20 Elemen X memiliki nilai KURANG(i) senilai 0
21 Elemen 16 memiliki nilai KURANG(i) senilai 2
22 Elemen 2 memiliki nilai KURANG(i) senilai 0
23 Elemen 5 memiliki nilai KURANG(i) senilai 0
24 Posisi awal puzzle tidak memiliki solusi dengan nilai KURANG(i) sebesar 65
25
26 Puzzle is not solvable!
```

5. Testcase “inputImpossible2”

a. Input

```
C:\Users\Gede Prasidha\Documents\Kuliah Semester 4\15-Puzzle-Solver-with-Python\src>py main.py
Selamat datang di program 15-Puzzle Solver
Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): output5
Silahkan pilih metode input matriks puzzle
1. Random
2. Read From File
Silahkan masukkan metode pilihan anda: 2
Masukkan nama file (tanpa extension .txt): inputImpossible2
Hasil perhitungan dan kalkulasi sudah ditulis di file C:/Users/Gede Prasidha/Documents/Kuliah Semester 4/15-Puzzle-Solver-with-Python/output/output5.txt
```

b. Output

```
output5.txt U X
output > output5.txt
1  Anda memakai input dari file inputImpossible2.txt
2  4 10 9 14
3  8 13 7 11
4  1 3 12 5
5  6 15 16 2
6
7  Elemen 4 memiliki nilai KURANG(i) senilai 3
8  Elemen 10 memiliki nilai KURANG(i) senilai 8
9  Elemen 9 memiliki nilai KURANG(i) senilai 7
10 Elemen 14 memiliki nilai KURANG(i) senilai 10
11 Elemen 8 memiliki nilai KURANG(i) senilai 6
12 Elemen 13 memiliki nilai KURANG(i) senilai 8
13 Elemen 7 memiliki nilai KURANG(i) senilai 5
14 Elemen 11 memiliki nilai KURANG(i) senilai 5
15 Elemen 1 memiliki nilai KURANG(i) senilai 0
16 Elemen 3 memiliki nilai KURANG(i) senilai 1
17 Elemen 12 memiliki nilai KURANG(i) senilai 3
18 Elemen 5 memiliki nilai KURANG(i) senilai 1
19 Elemen 6 memiliki nilai KURANG(i) senilai 1
20 Elemen 15 memiliki nilai KURANG(i) senilai 1
21 Elemen X memiliki nilai KURANG(i) senilai 1
22 Elemen 16 memiliki nilai KURANG(i) senilai 1
23 Elemen 2 memiliki nilai KURANG(i) senilai 0
24 Posisi awal puzzle tidak memiliki solusi dengan nilai KURANG(i) sebesar 61
25
26 Puzzle is not solvable!
```

6. Testcase random

a. Input

```
C:\Users\Gede Prasidha\Documents\Kuliah Semester 4\15-Puzzle-Solver-with-Python\src>py main.py
Selamat datang di program 15-Puzzle Solver
Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): output6
Silahkan pilih metode input matriks puzzle
1. Random
2. Read From File
Silahkan masukkan metode pilihan anda: 1
Hasil perhitungan dan kalkulasi sudah ditulis di file C:/Users/Gede Prasidha/Documents/Kuliah Semester 4/15-Puzzle-Solver-with-Python/output/output6.txt
```

b. Output

```
output6.txt U X
output > output6.txt
1  Anda memakai random input
2  13 15 16 12
3  4 6 8 9
4  10 7 11 3
5  5 14 1 2
6
7  Elemen 13 memiliki nilai KURANG(i) senilai 12
8  Elemen 15 memiliki nilai KURANG(i) senilai 13
9  Elemen X memiliki nilai KURANG(i) senilai 0
10 Elemen 16 memiliki nilai KURANG(i) senilai 13
11 Elemen 12 memiliki nilai KURANG(i) senilai 11
12 Elemen 4 memiliki nilai KURANG(i) senilai 3
13 Elemen 6 memiliki nilai KURANG(i) senilai 4
14 Elemen 8 memiliki nilai KURANG(i) senilai 5
15 Elemen 9 memiliki nilai KURANG(i) senilai 5
16 Elemen 10 memiliki nilai KURANG(i) senilai 5
17 Elemen 7 memiliki nilai KURANG(i) senilai 4
18 Elemen 11 memiliki nilai KURANG(i) senilai 4
19 Elemen 3 memiliki nilai KURANG(i) senilai 2
20 Elemen 5 memiliki nilai KURANG(i) senilai 2
21 Elemen 14 memiliki nilai KURANG(i) senilai 2
22 Elemen 1 memiliki nilai KURANG(i) senilai 0
23 Elemen 2 memiliki nilai KURANG(i) senilai 0
24 Posisi awal puzzle tidak memiliki solusi dengan nilai KURANG(i) sebesar 85
25
26 Puzzle is not solvable!
```

Checklist Spesifikasi

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil running	✓	
3. Program dapat menerima input dan menuliskan output	✓	
4. Luaran sudah benar untuk semua data uji	✓	
5. Bonus dibuat		✓

BAB 3

Kode Program dalam Java/Python

Untuk memudahkan proses debugging dan pembacaan *source code*, kode dibagi menjadi tiga bagian: `main.py`, `matrixCreator.py`, `bNb.py`. `Main.py` adalah sebuah file yang berfungsi untuk menangani I/O utama dan memanggil fungsi-fungsi lain yang dibutuhkan untuk menyelesaikan puzzle. `matrixCreator.py` adalah sebuah file untuk menangani pembacaan matriks puzzle, baik itu dari dibuat secara *random* oleh program ataupun membaca file dari pengguna. `bNb.py`, atau singkatan dari *Branch and Bound*, adalah file yang mengimplementasikan algoritma *Branch and Bound* yang telah dijelaskan pada Bab 1.

Selain adanya tiga file terpisah untuk membantu memodularkan program, penulis juga mengimplementasikan prinsip *Object and Classes*. Penulis menggunakan *class PuzzleTuple* untuk membuat simpul-simpul yang kemudian akan diperiksa satu-persatu dan dimasukkan ke dalam *priority queue*. *Class PuzzleTuple* ini dapat memiliki tiga atribut: `puzzle` (puzzle yang disimpan *instance class* itu sendiri), `cost` (estimasi *cost* pada simpul tersebut), dan `moveMade` (langkah yang perlu dilakukan untuk mencapai simpul tersebut).

Implementasi *PuzzleTuple*, atribut dan *method* termasuk:

```
bNb.py X
src > bNb.py > checksSolvable
1  import numpy as np
2
3  class PuzzleTuple:
4
5      # INITIATOR FUNCTIONS
6
7      def __init__(self, puzzle, cost, moveMade):
8          self.puzzle = puzzle # Puzzle contains a 1D numpy array of 16 elements
9          self.cost = cost
10         self.moveMade = moveMade
11
12     # GETTER FUNCTIONS
13     def returnPuzzleBytes(self):
14         return self.puzzle.tobytes()
15
16     def returnPuzzle(self):
17         return self.puzzle
18
19     def returnCost(self):
20         return self.cost
21
22     def returnMoveMade(self):
23         return self.moveMade
24
25     # SETTER FUNCTIONS
26
27     def setPuzzle(self, puzzle):
28         self.puzzle = puzzle
29
30     def setCost(self, cost):
31         self.cost = cost
32
33     def setMoveMade(self, moveMade):
34         self.moveMade = moveMade
35
```

```

36     # COMPARISON FUNCTIONS
37
38     def __lt__(self, other):
39         return self.returnCost() < other.returnCost()
40
41     def isGoal(self):
42         return self.returnPuzzleBytes() == np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]).tobytes()
43
44     # VOID FUNCTIONS
45
46     def printPuzzle(self, f):
47         for i in range(16):
48             if i % 4 == 0 and i != 0:
49                 print("\n", end="", file=f)
50             print(self.puzzle[i], end=" ", file=f)
51         print("\n\n", end="", file=f)
52
53     def printPuzzleTerminal(self):
54         for i in range(16):
55             if i % 4 == 0 and i != 0:
56                 print("\n", end="")
57             print(self.puzzle[i], end=" ")
58         print("\n\n", end="")
59
60     # COST COUNTER
61
62     def countCost(self):
63         cost = 0
64         for i in range(16):
65             if self.puzzle[i] != i + 1:
66                 cost += 1
67         return cost
68
69     # COPY CONSTRUCTOR
70
71     def copy(self):
72         return PuzzleTuple(self.returnPuzzle(), self.returnCost(), self.returnMoveMade())

```

Implementasi main.py

```

main.py
src > main.py > ...
1  from queue import PriorityQueue
2  import matrixCreator
3  import bNb
4  import time
5  import numpy as np
6  import os
7
8  def mainProgram():
9      print("Selamat datang di program 15-Puzzle Solver")
10     pathDirectory = os.path.abspath(os.path.join(os.path.dirname( __file__ ), '..', 'output'))
11     targetFile = input("Silahkan input nama file txt untuk menyimpan puzzle yang ingin dicari solusinya (tanpa extension .txt): ")
12     targetFilePath = pathDirectory + "/" + targetFile + ".txt"
13     targetFilePath = targetFilePath.replace("\\", "/")
14     f = open(targetFilePath, "w")
15     print("Silahkan pilih metode input matriks puzzle")
16     print("1. Random")
17     print("2. Read From File")
18     choice = input("Silahkan masukkan metode pilihan anda: ")
19     while not choice.isdigit() and choice != "1" and choice != "2":
20         choice = input("Error. Please input a choice in range of 1 to 2: ")
21     choice = int(choice)
22     if choice == 1:
23         print("Anda memakai random input", file=f)
24         initPTuple = matrixCreator.random_matrix()
25         initPTuple.printPuzzle(f)
26         isSolvable = bNb.checkIsSolvable(initPTuple, f)
27     elif choice == 2:
28         initPTuple, fileNameWithExtension = matrixCreator.readFromFile()
29         print("Anda memakai input dari file {}".format(fileNameWithExtension), file=f)
30         initPTuple.printPuzzle(f)
31         isSolvable = bNb.checkIsSolvable(initPTuple, f)

```

```

32 if (isSolvable):
33     # Inisiasi jumlah simpul awal, waktu awal, dan dictionary untuk menyimpan puzzle yang pernah ditemukan
34     nodeCreated = 0
35     startTime = time.time()
36     visitedStates = dict()
37     # Isi dari priorityQueue : (cost, kedalaman (negatif), [puzzle, dictionary])
38     # Empat blok dibawah ini adalah inisiasi awal dari informasi simpul root
39     prioQueue = PriorityQueue()
40     currentMoveMade = []
41     prioQueue.put((0, -1, [initPTuple, currentMoveMade]))
42     visitedStates[initPTuple.returnPuzzleBytes()] = True
43     # Fungsi loop yang akan break/keluar ketika menemukan solusi
44     hasFound = False
45     while not hasFound:
46         # Akan mengambil elemen pertama dari priority queue dan mengecek apakah puzzle yang ditampung merupakan solusi atau bukan
47         currentNode = prioQueue.get()
48         currentTuple = currentNode[2][0]
49         currentMoveMade = currentNode[2][1]
50         if currentTuple.isGoal():
51             hasFound = True
52             timeTaken = time.time() - startTime
53             printResult(initPTuple, currentMoveMade, timeTaken, nodeCreated, f)
54         else:
55             # Bila bukan solusi, maka akan dilakukan perulangan berdasarkan semua langkah yang mungkin dilakukan dari posisi petak kosong pada puzzle simpul tersebut
56             currentEmptyBlockIdx = bNb.whereEmptyBlock(currentTuple)
57             currentMoveList = bNb.whereToMove(currentEmptyBlockIdx)
58             for move in currentMoveList:
59                 newPuzzle = bNb.moveTile(currentTuple.copy(), currentEmptyBlockIdx, move)
60                 newPuzzle.setCost = newPuzzle.countCost()
61                 newPuzzle.setMoveMade(len(currentMoveMade) + 1)
62                 # Simpul baru hanya ditambahkan jika belum pernah ditemukan sebelumnya state puzzlenya
63                 if newPuzzle.returnPuzzleBytes() not in visitedStates.keys():
64                     nodeCreated += 1
65                     visitedStates[newPuzzle.returnPuzzleBytes()] = True
66                     # Kedalaman suatu simpul disimpan dalam bentuk currentNode[1]
67                     # Semakin negatif nilai currentNode[1], maka posisi simpul semakin dalam
68                     # Untuk dua simpul dengan cost yang sama, maka prioritas akan jatuh kepada simpul yang lebih dalam
69                     prioQueue.put((newPuzzle.countCost() - currentNode[1], currentNode[1] - 1, [newPuzzle, currentMoveMade + [move]]))
70     else:
71         print("Puzzle is not solvable!", file=f)
72     print("Hasil perhitungan dan kalkulasi sudah ditulis di file {}".format(targetFilePath))
73     f.close()

```

```

75 def translateLangkah(langkah):
76     # Langkah disimpul dalam bentuk char untuk membantu mempermudah penulisan dan debugging
77     # Char akan ditranslasi menjadi string untuk memudahkan pengguna mengerti output file
78     if langkah == 'U':
79         return "Atas"
80     elif langkah == 'D':
81         return "Bawah"
82     elif langkah == 'L':
83         return "Kiri"
84     elif langkah == 'R':
85         return "Kanan"
86
87 def printResult(initPTuple, currentMoveMade, timeTaken, nodeCreated, f):
88     print("Waktu yang dibutuhkan: %.5f detik" %timeTaken, file=f)
89     print("Jumlah node yang dibuat:", nodeCreated, file=f)
90     print("Jumlah langkah yang dilakukan:", len(currentMoveMade), file=f)
91     print("Semua langkah yang dilakukan:", currentMoveMade , file=f)
92     stepCount = 1
93     # Output puzzle dan langkah yang dilakukan dari awal hingga solusi
94     while len(currentMoveMade) != 0:
95         langkah = currentMoveMade.pop(0)
96         textLangkah = translateLangkah(langkah)
97         print("Langkah ke-%d" %stepCount, ":", file=f)
98         print("Move yang dilakukan:", textLangkah, file=f)
99         initPTuple = bNb.moveTile(initPTuple, bNb.whereEmptyBlock(initPTuple), langkah)
100         initPTuple.printPuzzle(f)
101         stepCount += 1
102
103
104 mainProgram()

```

Implementasi dari matrixCreator.py

```
matrixCreator.py M X
src > matrixCreator.py > random_matrix
1  import numpy as np
2  import bNb
3  import os
4
5  def random_matrix():
6      # Buat untuk main.py pake fungsi random ukuran 1 sampai 16
7      # Pembuatan puzzle random menggunakan fungsi default_rng dari numpy.random
8      # Penggunaan parameter replace = False agar tidak ada angka yang sama
9      rng = np.random.default_rng()
10     puzzle = rng.choice(16, size=(16), replace=False)
11     for i in range(16):
12         puzzle[i] += 1
13     puzzle = np.array(puzzle)
14     resultPTuple = bNb.PuzzleTuple(puzzle, 0, 0)
15     return resultPTuple
16
17 def readFromFile():
18     # Buat untuk main.py pake fungsi readFromFile
19     # Pengambilan path directory menggunakan absolute path komputer pengguna
20     pathDirectory = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'input'))
21     filename = input("Masukkan nama file (tanpa extension .txt): ")
22     filePath = pathDirectory + "/" + filename + ".txt"
23     filePath = filePath.replace("\\", "/")
24     f = open(filePath, "r")
25     puzzle = [int(i) for i in f.read().split()]
26     puzzle = np.array(puzzle)
27     resultPTuple = bNb.PuzzleTuple(puzzle, 0, 0)
28     filenameExtension = filename + ".txt"
29     return resultPTuple, filenameExtension
```


Implementasi bnB.py (selain inisiasi class beserta atribut dan *methodnya*)

```
bNb.py M X
src > bNb.py > ...
/4
75 def checkIsSolvable(PuzzleTuple, f):
76     totalKurang = 0
77     # Fungsi KURANG(i) adalah jumlah kurang dari puzzle[j] dari puzzle[i] ke-i
78     # Akumulasi fungsi KURANG(i) ditambah X disimpan dalam totalKurang
79     # Bila nilai totalKurang ganjil, maka tidak ada solusi
80     # Sebaliknya, bila genap, ada solusi yang bisa dicapai dari puzzle awal
81     for i in range(len(PuzzleTuple.returnPuzzle())):
82         countKurang = 0
83         testPuzzle = PuzzleTuple.returnPuzzle()[i]
84         if testPuzzle == 16 and i in [1,3,4,6,9,11,12,14]:
85             totalKurang += 1
86             print("Elemen X memiliki nilai KURANG(i) senilai 1", file=f)
87         elif testPuzzle == 16 and i not in [1,3,4,6,9,11,12,14]:
88             print("Elemen X memiliki nilai KURANG(i) senilai 0", file=f)
89         for j in range(i + 1, len(PuzzleTuple.puzzle)):
90             if PuzzleTuple.puzzle[i] > PuzzleTuple.puzzle[j]:
91                 countKurang += 1
92         print(f'Elemen {testPuzzle} memiliki nilai KURANG(i) senilai {countKurang}', file=f)
93         totalKurang += countKurang
94     if totalKurang % 2 == 0:
95         print("Posisi awal puzzle memiliki solusi dengan nilai KURANG(i) sebesar {}".format(totalKurang), file=f)
96         return True
97     else:
98         print("Posisi awal puzzle tidak memiliki solusi dengan nilai KURANG(i) sebesar {}".format(totalKurang), file=f)
99         return False
100
101 # Fungsi untuk mencari posisi petak kosong dari suatu puzzle
102 def whereEmptyBlock(PuzzleTuple):
103     for i in range(len(PuzzleTuple.returnPuzzle())):
104         if PuzzleTuple.returnPuzzle()[i] == 16:
105             return i
106
```

```
107 # Fungsi untuk mencari semua gerakan legal dari suatu puzzle
108 def whereToMove(emptyBlockIdx):
109     # Urutan prioritas : 'R', 'L', 'U', 'D'
110     if (emptyBlockIdx == 0):
111         return ['R', 'D']
112     elif (emptyBlockIdx == 3):
113         return ['L', 'D']
114     elif (emptyBlockIdx == 12):
115         return ['R', 'U']
116     elif (emptyBlockIdx == 15):
117         return ['L', 'U']
118     elif (emptyBlockIdx == 1 or emptyBlockIdx == 2):
119         return ['R', 'L', 'D']
120     elif (emptyBlockIdx == 4 or emptyBlockIdx == 8):
121         return ['R', 'U', 'D']
122     elif (emptyBlockIdx == 7 or emptyBlockIdx == 11):
123         return ['L', 'U', 'D']
124     elif (emptyBlockIdx == 13 or emptyBlockIdx == 14):
125         return ['R', 'L', 'U']
126     else:
127         return ['R', 'L', 'U', 'D']
128
```

```

129 | # Fungsi gerakan dari puzzle, tergantung pada posisi petak kosong dan command yang diberikan
130 | def switchLeft(PuzzleTuple, emptyBlockIdx):
131 |     PuzzleTuple.setPuzzle(np.insert(PuzzleTuple.returnPuzzle(), emptyBlockIdx - 1, 16))
132 |     PuzzleTuple.setPuzzle(np.delete(PuzzleTuple.returnPuzzle(), emptyBlockIdx + 1))
133 |     return PuzzleTuple
134 |
135 | def switchRight(PuzzleTuple, emptyBlockIdx):
136 |     PuzzleTuple.setPuzzle(np.insert(PuzzleTuple.returnPuzzle(), emptyBlockIdx + 2, 16))
137 |     PuzzleTuple.setPuzzle(np.delete(PuzzleTuple.returnPuzzle(), emptyBlockIdx))
138 |     return PuzzleTuple
139 |
140 | def switchUp(PuzzleTuple, emptyBlockIdx):
141 |     temp = PuzzleTuple.returnPuzzle()[emptyBlockIdx - 4]
142 |     PuzzleTuple.setPuzzle(np.insert(PuzzleTuple.returnPuzzle(), emptyBlockIdx - 4, 16))
143 |     PuzzleTuple.setPuzzle(np.delete(PuzzleTuple.returnPuzzle(), emptyBlockIdx - 3))
144 |     PuzzleTuple.setPuzzle(np.insert(PuzzleTuple.returnPuzzle(), emptyBlockIdx, temp))
145 |     PuzzleTuple.setPuzzle(np.delete(PuzzleTuple.returnPuzzle(), emptyBlockIdx + 1))
146 |     return PuzzleTuple
147 |
148 | def switchDown(PuzzleTuple, emptyBlockIdx):
149 |     temp = PuzzleTuple.returnPuzzle()[emptyBlockIdx + 4]
150 |     PuzzleTuple.setPuzzle(np.insert(PuzzleTuple.returnPuzzle(), emptyBlockIdx + 4, 16))
151 |     PuzzleTuple.setPuzzle(np.delete(PuzzleTuple.returnPuzzle(), emptyBlockIdx + 5))
152 |     PuzzleTuple.setPuzzle(np.insert(PuzzleTuple.returnPuzzle(), emptyBlockIdx, temp))
153 |     PuzzleTuple.setPuzzle(np.delete(PuzzleTuple.returnPuzzle(), emptyBlockIdx + 1))
154 |     return PuzzleTuple
155 |
156 | def moveTile(PuzzleTuple, emptyBlockIdx, move):
157 |     if move == 'R':
158 |         PuzzleTuple = switchRight(PuzzleTuple, emptyBlockIdx)
159 |     elif move == 'L':
160 |         PuzzleTuple = switchLeft(PuzzleTuple, emptyBlockIdx)
161 |     elif move == 'U':
162 |         PuzzleTuple = switchUp(PuzzleTuple, emptyBlockIdx)
163 |     elif move == 'D':
164 |         PuzzleTuple = switchDown(PuzzleTuple, emptyBlockIdx)
165 |     return PuzzleTuple

```

BAB 4

Instansiasi Test Case dalam Bentuk File Ekstensi txt

1. inputPossible1

1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

2. inputPossible2

5 1 3 4
9 2 7 8
16 6 15 11
13 10 14 12

3. inputPossible3

6 5 2 4
9 1 3 8
10 16 7 15
13 14 12 11

4. inputImpossible1

11 3 15 9
13 6 7 8
4 1 14 12
10 16 2 5

5. inputImpossible2

4 10 9 14
8 13 7 11
1 3 12 5
6 15 16 2

Pada program ini, petak kosong diisi dengan angka 16.

LAMPIRAN

Link Github : <https://github.com/LordGedelicious/15-Puzzle-Solver-with-Python>