

Laporan Tugas Kecil 3 - Strategi Algoritma (IF2211)
“Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*”



Oleh:

Muhammad Hasan – 13518012 – K03

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER 2 TAHUN 2019/2020

Daftar Isi

Daftar Isi.....	2
BAB I.....	3
Cara Kerja Program <i>Branch and Bound</i> dalam Menyelesaikan 15-Puzzle	3
1.1 Definisi Algoritma Branch and Bound	3
1.2 Fungsi Pembatas pada Algoritma <i>Branch and Bound</i>	3
1.3 Algoritma <i>Branch and Bound</i> dalam menyelesaikan 15-Puzzle	3
BAB 2 – <i>Source Code</i> Program	5
2.1 <i>Checklist</i> Tabel Program	5
2.2 <i>Source Code</i> Program	5
BAB 3 – Screenshot Input-Output Program	10
3.1 Spesifikasi Personal Computer.....	10
3.2 Screenshot Input-Output Program	10
3.2.1 Test Case 1	11
3.2.2 Test Case 2.....	12
3.2.3 Test Case 3.....	13
3.2.4 Test Case 4.....	15
3.2.5 Test Case 5.....	18

BAB I

Cara Kerja Program *Branch and Bound* dalam Menyelesaikan 15-Puzzle

1.1 Definisi Algoritma Branch and Bound

Algoritma *Branch and Bound* (B&B) merupakan Algoritma yang digunakan untuk persoalan optimasi, yakni meminimalkan/memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (*constraint*) persoalan. Cara kerja Algoritma B&B ini biasanya menggunakan ilustrasi sebuah *tree*, dengan setiap *node* memiliki sebuah nilai cost:

$\hat{c}(i)$ = nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i

Kemudian, *node* berikutnya yang akan di-*expand* **tidak lagi** berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* terkecil (*least cost search*) – pada kasus minimasi.

1.2 Fungsi Pembatas pada Algoritma *Branch and Bound*

Algoritma B&B ini juga menerapkan “pemangkasan” pada jalur yang dianggap tidak lagi mengarah pada solusi, kriteria pemangkasan secara umumnya adalah sebagai berikut:

- Nilai simpul tidak lebih baik dari nilai simpul terbaik sejauh ini
- Simpul tidak merepresentasikan solusi yang *feasible* karena ada batasan yang dilanggar
- Solusi yang *feasible* pada *node* tersebut hanya terdiri atas satu titik artinya tidak ada pilihan lain

1.3 Algoritma *Branch and Bound* dalam menyelesaikan 15-Puzzle

Algoritma *Branch and Bound* ini mempunyai Algoritma global yang dapat digunakan untuk menyelesaikan permasalahan 15-Puzzle, caranya adalah sebagai berikut:

1. Masukkan *node root* ke dalam antrian *PrioQueue* (*Queue* dengan prioritas *cost node* terkecil). Jika *root node* adalah *goal node*. Maka solusi telah ditemukan. Stop.
2. Jika *PrioQueue* kosong, tidak ada solusi. Stop.
3. Jika *Prioqueue* tidak kosong, ambil *top* dari *PrioQueue* dan lakukan *pop*. Katakan saja *top* dari *PrioQueue* ini kita katakan sebagai *node u*
4. Jika *node u* ini merupakan *goal node* artinya solusi sudah ditemukan dan program dapat langsung dihentikan. Jika *node u* bukan merupakan *goal node*, bangkitkan semua anak-anak dari *node u* yang belum pernah dibangkitkan sebelumnya.

5. Untuk setiap anak yang dibangkitkan, katakan *node* v , hitunglah $\hat{c}(v)$. Pada kasus 15-Puzzle ini, $\hat{c}(v)$ didefinisikan sebagai berikut:

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

Keterangan:

$\hat{c}(v)$ = ongkos untuk *node* v

$\hat{f}(v)$ = panjang lintasan *node* v ke *root node*

$\hat{g}(v)$ = taksiran panjang lintasan terpendek dari v ke *goal node*

Taksiran $\hat{g}(v)$ dilakukan dengan menghitung jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. Setelah menghitung semua $\hat{c}(v)$, masukkan semua anak-anak tersebut ke dalam *PrioQueue*.

6. Kembali ke langkah 2.

BAB 2 – Source Code Program

2.1 Checklist Tabel Program

Poin	Ya	Tidak
Program berhasil dikompilasi	✓	
Program berhasil <i>running</i>	✓	
Program dapat menerima input dan menuliskan output	✓	
Luaran sudah benar untuk semua data uji	✓	

2.2 Source Code Program

Program dibuat dengan bahasa pemrograman *Python*. Berikut adalah *source code* *puzzle15Solver.py* yang digunakan untuk menyelesaikan persoalan 15-Puzzle:

```
# Muhammad Hasan - 13518012
# Program : Puzzle15 - Solver
from heapq import heappop, heappush
import time

def print_matrix(matrix):
    for i in range(len(matrix)):
        print(matrix[i], end = ' ');
        if (i % 4 == 3):
            print("")
        print("")

def nilai_kurang(matrix):
    kurang = [0 for i in range(16)]
    for i in range(len(matrix)):
        hitungInversi = 0
        for j in range(i + 1, len(matrix)):
            if (matrix[j] < matrix[i]):
                hitungInversi += 1
        kurang[matrix[i] - 1] = hitungInversi

    return kurang

def index_ubin_kosong(matrix):
    for i in range(len(matrix)):
        if (matrix[i] == 16):
```

```

        return i

def ubah_matrix_to_alphabet(matrix):
    ret = ""
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    for element in matrix:
        ret += alphabet[element]
    return ret

def nilai_ongkos_g(matrix):
    ret = 0
    for i in range(len(matrix)):
        if (i + 1 != matrix[i]):
            ret += 1
    return ret

def up_matrix(matrix, idx):
    ret_matrix = matrix.copy()
    if ((idx // 4) > 0):
        ret_matrix[idx], ret_matrix[idx - 4] = ret_matrix[idx - 4], ret_matrix[
idx]
    return ret_matrix
    else:
        return -1

def down_matrix(matrix, idx):
    ret_matrix = matrix.copy()
    if ((idx // 4) < 3):
        ret_matrix[idx + 4], ret_matrix[idx] = ret_matrix[idx], ret_matrix[idx
+ 4]
    return ret_matrix
    else:
        return -1

def left_matrix(matrix, idx):
    ret_matrix = matrix.copy()
    if (idx % 4 > 0):
        ret_matrix[idx - 1], ret_matrix[idx] = ret_matrix[idx], ret_matrix[idx
- 1]
    return ret_matrix
    else:
        return -1

def right_matrix(matrix, idx):
    ret_matrix = matrix.copy()
    if (idx % 4 < 3):
        ret_matrix[idx], ret_matrix[idx + 1] = ret_matrix[idx + 1], ret_matrix[
idx]
    return ret_matrix

```

```

else:
    return -1

matrix_awal = []
for i in range(4):
    inp = list(map(int, input().strip().split(' ')));
    matrix_awal += inp

start_time = time.time()

print("matrix posisi awal:")
print_matrix(matrix_awal)

kurang = nilai_kurang(matrix_awal)
jumlah_kurang_x = 0

print("Didapat nilai kurang untuk setiap ubin-i adalah sebagai berikut:")
for i in range(16):
    print("Nilai kurang ke-" + str(i + 1) + " = " + str(kurang[i]))
    jumlah_kurang_x += kurang[i]
print("")

index_kosong = index_ubin_kosong(matrix_awal)
if (index_kosong % 8 < 4):
    if (index_kosong % 2 == 1):
        jumlah_kurang_x += 1
    elif (index_kosong % 2 == 0):
        jumlah_kurang_x += 1

print("Nilai dari sum_{i=1}^{16} kurang(i) + X adalah =", jumlah_kurang_x)
if (jumlah_kurang_x % 2 == 1):
    print("Karena nilai tersebut ganjil kita tidak akan dapat menyelesaikan p
ersoalan.")
    exit()

map_matrix = {}
map_node_to_matrix = {}
depth_node = {}
cost_node = {}
parent_node = {}

matrix_hash = ubah_matrix_to_alphabet(matrix_awal)
cur_node = 1

map_matrix[matrix_hash] = 1
map_node_to_matrix[1] = matrix_awal
depth_node[cur_node] = 0
cost_node[cur_node] = 0
parent_node[cur_node] = cur_node

```

```

prioQueue = [(cost_node[cur_node], cur_node)]
found_solution = False

while (prioQueue) and (not found_solution):
    cost, node = heappop(prioQueue)

    matrix_node = map_node_to_matrix[node]
    index_kosong = index_ubin_kosong(matrix_node)

    matrix_ekspansi = []
    matrix_ekspansi.append(up_matrix(matrix_node, index_kosong))
    matrix_ekspansi.append(right_matrix(matrix_node, index_kosong))
    matrix_ekspansi.append(down_matrix(matrix_node, index_kosong))
    matrix_ekspansi.append(left_matrix(matrix_node, index_kosong))

    for matrix in matrix_ekspansi:
        if (matrix == -1):
            continue

        matrix_hash = ubah_matrix_to_alphabet(matrix)
        if (matrix_hash in map_matrix):
            continue

        cur_node += 1

        fungsi_g = nilai_ongkos_g(matrix)

        map_matrix[matrix_hash] = 1
        map_node_to_matrix[cur_node] = matrix
        depth_node[cur_node] = depth_node[node] + 1
        cost_node[cur_node] = depth_node[cur_node] + fungsi_g
        parent_node[cur_node] = node

        if (fungsi_g == 0):
            found_solution = True
            break

        heappush(prioQueue, (cost_node[cur_node], cur_node))

jumlah_node = cur_node
solusi_node = []

while True:
    solusi_node.append(cur_node)
    if (cur_node == 1):
        break
    cur_node = parent_node[cur_node]

```



```
solusi_node = solusi_node[::-1]

print("Karena nilai tersebut genap kita bisa memperoleh solusi!")
print("Solusinya adalah sebagai berikut:")
for i in range(len(solusi_node)):
    print("Langkah ke-" + str(i) + ":")
    print_matrix(map_node_to_matrix[solusi_node[i]])

print("Waktu eksekusi :", (time.time() - start_time) * 1000, "ms")
print("Jumlah node yang dibangkitkan adalah :", jumlah_node)
```

BAB 3 – Screenshot Input-Output Program

3.1 Spesifikasi Personal Computer

Spesifikasi *Personal Computer* (PC) yang digunakan dalam menjalankan program adalah sebagai berikut:

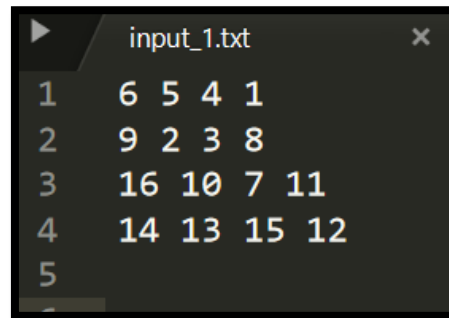
Processor	RAM	HardDrive
Intel® Core™ i7-8550U CPU @ 1.80 GHz – 1.99 GHz	DDR3 – 16GB	SSD SATA 3 – 512 GB

3.2 Screenshot Input-Output Program

Terdapat 5 *test case* yang telah diuji, dengan 2 *test case* tidak dapat diselesaikan dan 3 *test case* dapat diselesaikan. Catatan: untuk setiap kasus, ubin kosong dianggap sebagai angka 16.

3.2.1 Test Case 1

Test Case 1 ini berisi *input_1.txt* sebagai berikut:



Hasil *screenshot output*-nya adalah sebagai berikut:

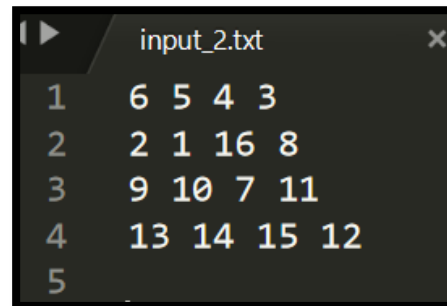
```
C:\Users\USER\github\IF2211_Strategi_Algoritma\Tucil3StrAlgo-13518012\src>python puzzle15Solver.py < input_1.txt
matrix posisi awal:
6 5 4 1
9 2 3 8
16 10 7 11
14 13 15 12

Didapat nilai kurang untuk setiap ubin-i adalah sebagai berikut:
Nilai kurang ke-1 = 0
Nilai kurang ke-2 = 0
Nilai kurang ke-3 = 0
Nilai kurang ke-4 = 3
Nilai kurang ke-5 = 4
Nilai kurang ke-6 = 5
Nilai kurang ke-7 = 0
Nilai kurang ke-8 = 1
Nilai kurang ke-9 = 4
Nilai kurang ke-10 = 1
Nilai kurang ke-11 = 0
Nilai kurang ke-12 = 0
Nilai kurang ke-13 = 1
Nilai kurang ke-14 = 2
Nilai kurang ke-15 = 1
Nilai kurang ke-16 = 7

Nilai dari  $\sum_{i=1}^{16} \text{kurang}(i) + X$  adalah = 29
Karena nilai tersebut ganjil kita tidak akan dapat menyelesaikan persoalan.
```

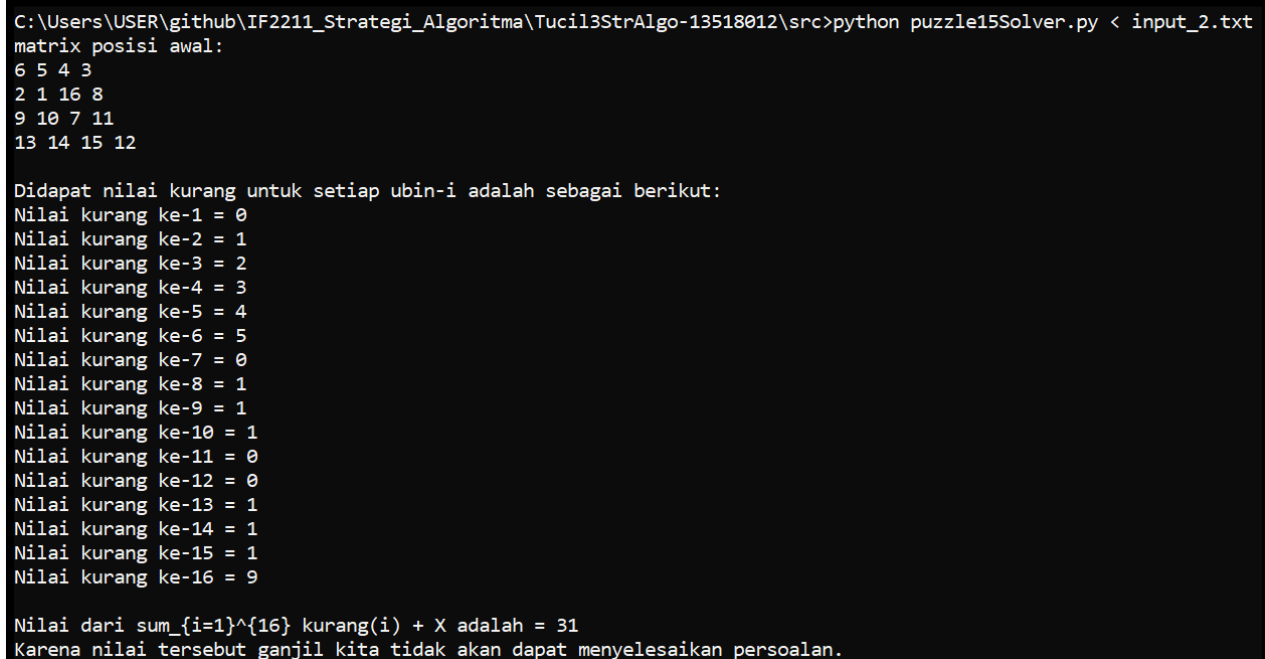
3.2.2 Test Case 2

Test Case 2 ini berisi *input_2.txt* sebagai berikut:



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

Hasil *screenshot output*-nya adalah sebagai berikut:



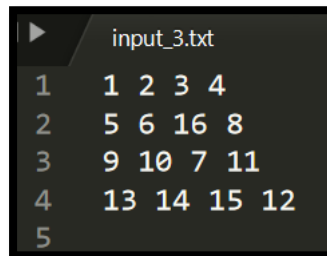
```
C:\Users\USER\github\IF2211_Strategi_Algoritma\Tucil3StrAlgo-13518012\src>python puzzle15Solver.py < input_2.txt
matrix posisi awal:
6 5 4 3
2 1 16 8
9 10 7 11
13 14 15 12

Didapat nilai kurang untuk setiap ubin-i adalah sebagai berikut:
Nilai kurang ke-1 = 0
Nilai kurang ke-2 = 1
Nilai kurang ke-3 = 2
Nilai kurang ke-4 = 3
Nilai kurang ke-5 = 4
Nilai kurang ke-6 = 5
Nilai kurang ke-7 = 0
Nilai kurang ke-8 = 1
Nilai kurang ke-9 = 1
Nilai kurang ke-10 = 1
Nilai kurang ke-11 = 0
Nilai kurang ke-12 = 0
Nilai kurang ke-13 = 1
Nilai kurang ke-14 = 1
Nilai kurang ke-15 = 1
Nilai kurang ke-16 = 9

Nilai dari  $\sum_{i=1}^{16} \text{kurang}(i) + X$  adalah = 31
Karena nilai tersebut ganjil kita tidak akan dapat menyelesaikan persoalan.
```

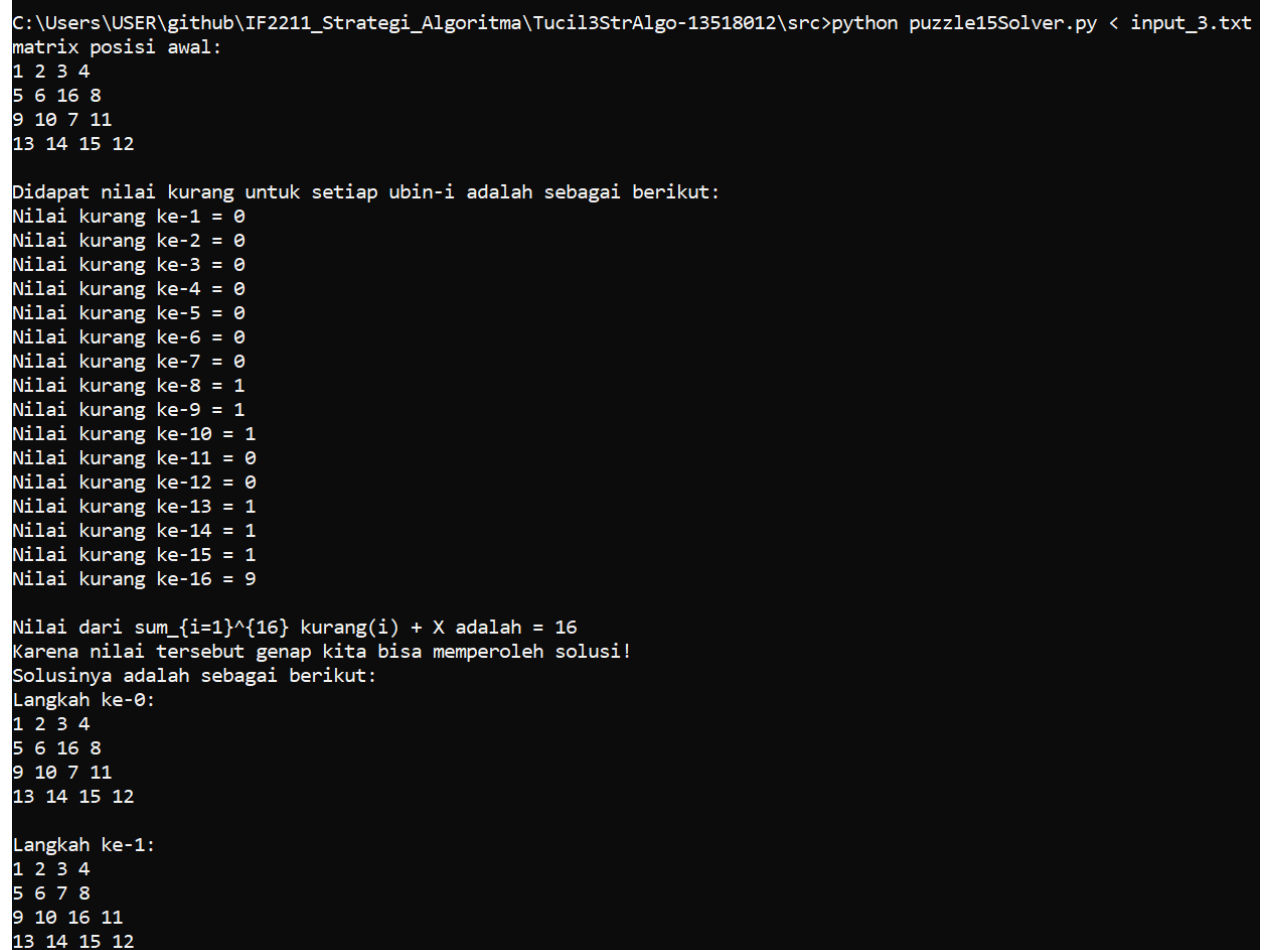
3.2.3 Test Case 3

Test Case 3 ini berisi *input_3.txt* sebagai berikut:



```
input_3.txt
1 1 2 3 4
2 5 6 16 8
3 9 10 7 11
4 13 14 15 12
5
```

Hasil *screenshot output*-nya adalah sebagai berikut:



```
C:\Users\USER\github\IF2211_Strategi_Algoritma\Tucil3StrAlgo-13518012\src>python puzzle15Solver.py < input_3.txt
matrix posisi awal:
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

Didapat nilai kurang untuk setiap ubin-i adalah sebagai berikut:
Nilai kurang ke-1 = 0
Nilai kurang ke-2 = 0
Nilai kurang ke-3 = 0
Nilai kurang ke-4 = 0
Nilai kurang ke-5 = 0
Nilai kurang ke-6 = 0
Nilai kurang ke-7 = 0
Nilai kurang ke-8 = 1
Nilai kurang ke-9 = 1
Nilai kurang ke-10 = 1
Nilai kurang ke-11 = 0
Nilai kurang ke-12 = 0
Nilai kurang ke-13 = 1
Nilai kurang ke-14 = 1
Nilai kurang ke-15 = 1
Nilai kurang ke-16 = 9

Nilai dari  $\sum_{i=1}^{16} \text{kurang}(i) + X$  adalah = 16
Karena nilai tersebut genap kita bisa memperoleh solusi!
Solusinya adalah sebagai berikut:
Langkah ke-0:
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

Langkah ke-1:
1 2 3 4
5 6 7 8
9 10 16 11
13 14 15 12
```

Solusinya adalah sebagai berikut:

Langkah ke-0:

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

Langkah ke-1:

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

Langkah ke-2:

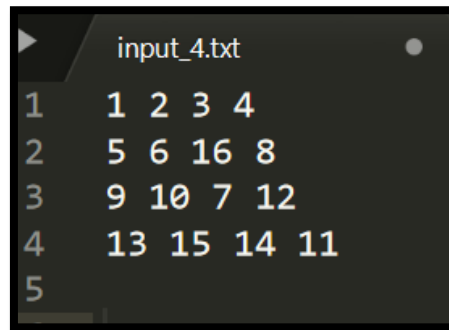
```
1 2 3 4
5 6 7 8
9 10 11 16
13 14 15 12
```

Langkah ke-3:

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

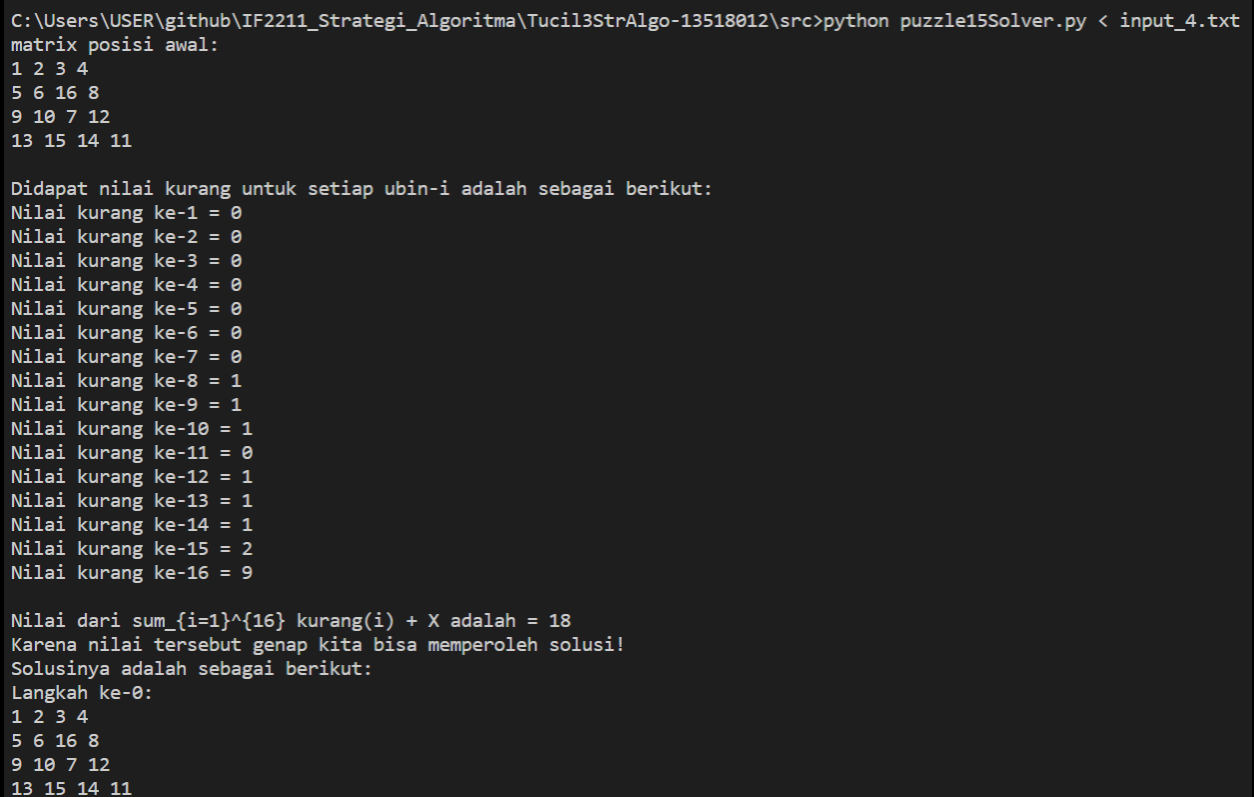
3.2.4 Test Case 4

Test Case 4 ini berisi *input_4.txt* sebagai berikut:



```
input_4.txt
1 1 2 3 4
2 5 6 16 8
3 9 10 7 12
4 13 15 14 11
5
```

Hasil *screenshot output*-nya adalah sebagai berikut:



```
C:\Users\USER\github\IF2211_Strategi_Algoritma\Tucil3StrAlgo-13518012\src>python puzzle15Solver.py < input_4.txt
matrix posisi awal:
1 2 3 4
5 6 16 8
9 10 7 12
13 15 14 11

Didapat nilai kurang untuk setiap ubin-i adalah sebagai berikut:
Nilai kurang ke-1 = 0
Nilai kurang ke-2 = 0
Nilai kurang ke-3 = 0
Nilai kurang ke-4 = 0
Nilai kurang ke-5 = 0
Nilai kurang ke-6 = 0
Nilai kurang ke-7 = 0
Nilai kurang ke-8 = 1
Nilai kurang ke-9 = 1
Nilai kurang ke-10 = 1
Nilai kurang ke-11 = 0
Nilai kurang ke-12 = 1
Nilai kurang ke-13 = 1
Nilai kurang ke-14 = 1
Nilai kurang ke-15 = 2
Nilai kurang ke-16 = 9

Nilai dari  $\sum_{i=1}^{16} \text{kurang}(i) + X$  adalah = 18
Karena nilai tersebut genap kita bisa memperoleh solusi!
Solusinya adalah sebagai berikut:
Langkah ke-0:
1 2 3 4
5 6 16 8
9 10 7 12
13 15 14 11
```

Langkah ke-0:

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

Langkah ke-1:

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

Langkah ke-2:

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

Langkah ke-3:

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

Langkah ke-4:

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

Langkah ke-5:

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

Langkah ke-6:

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

Langkah ke-7:

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

Langkah ke-7:

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

Langkah ke-8:

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

Langkah ke-9:

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

Langkah ke-10:

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

Langkah ke-11:

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

Langkah ke-12:

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

Langkah ke-13:

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

Langkah ke-14:

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

Langkah ke-15:

1 2 3 4

5 6 7 8

9 10 11 12

13 16 14 15

Langkah ke-16:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 16 15

Langkah ke-17:

1 2 3 4

5 6 7 8

9 10 11 12

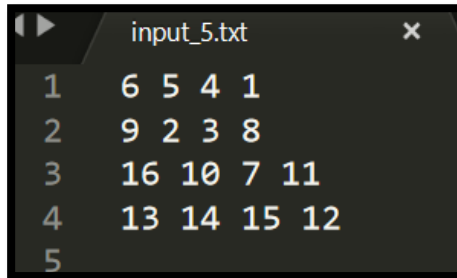
13 14 15 16

Waktu eksekusi : 410.9818935394287 ms

Jumlah node yang dibangkitkan adalah : 7221

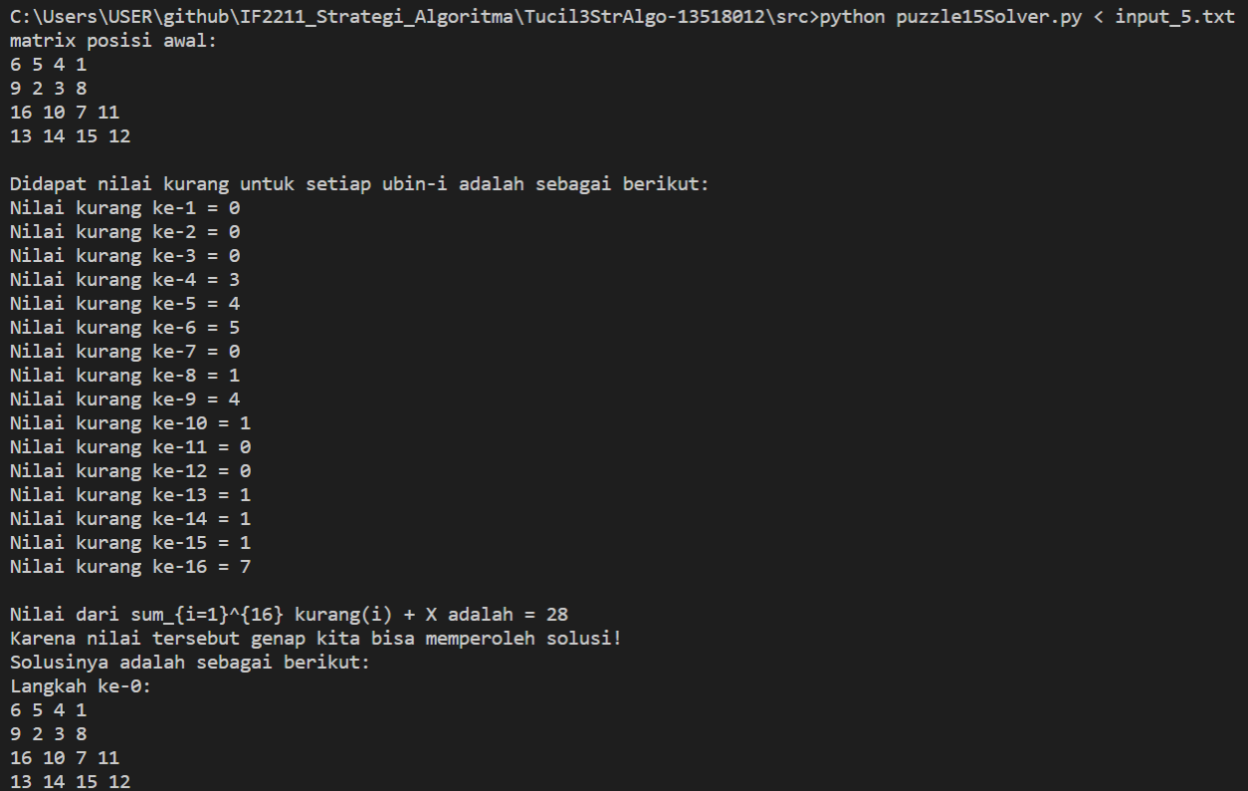
3.2.5 Test Case 5

Test Case 5 ini berisi *input_5.txt* sebagai berikut:



```
input_5.txt
1 6 5 4 1
2 9 2 3 8
3 16 10 7 11
4 13 14 15 12
5
```

Hasil *screenshot output*-nya adalah sebagai berikut:



```
C:\Users\USER\github\IF2211_Strategi_Algoritma\Tucil3StrAlgo-13518012\src>python puzzle15Solver.py < input_5.txt
matrix posisi awal:
6 5 4 1
9 2 3 8
16 10 7 11
13 14 15 12

Didapat nilai kurang untuk setiap ubin-i adalah sebagai berikut:
Nilai kurang ke-1 = 0
Nilai kurang ke-2 = 0
Nilai kurang ke-3 = 0
Nilai kurang ke-4 = 3
Nilai kurang ke-5 = 4
Nilai kurang ke-6 = 5
Nilai kurang ke-7 = 0
Nilai kurang ke-8 = 1
Nilai kurang ke-9 = 4
Nilai kurang ke-10 = 1
Nilai kurang ke-11 = 0
Nilai kurang ke-12 = 0
Nilai kurang ke-13 = 1
Nilai kurang ke-14 = 1
Nilai kurang ke-15 = 1
Nilai kurang ke-16 = 7

Nilai dari sum_{i=1}^{16} kurang(i) + X adalah = 28
Karena nilai tersebut genap kita bisa memperoleh solusi!
Solusinya adalah sebagai berikut:
Langkah ke-0:
6 5 4 1
9 2 3 8
16 10 7 11
13 14 15 12
```

Solusinya adalah sebagai berikut:

Langkah ke-0:

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

Langkah ke-1:

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

Langkah ke-2:

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

Langkah ke-3:

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

Langkah ke-4:

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

Langkah ke-5:

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

Langkah ke-6:

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

Langkah ke-7:

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

Langkah ke-7:

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

Langkah ke-8:

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

Langkah ke-9:

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

Langkah ke-10:

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

Langkah ke-11:

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

Langkah ke-12:

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

Langkah ke-13:

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

Langkah ke-14:

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

Langkah ke-14:

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

Langkah ke-15:

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

Langkah ke-16:

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

Langkah ke-17:

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

Langkah ke-18:

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

Langkah ke-19:

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

Langkah ke-20:

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

Langkah ke-21:

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

Langkah ke-21:

1 2 3 4

5 6 16 8

9 10 7 11

13 14 15 12

Langkah ke-22:

1 2 3 4

5 6 7 8

9 10 16 11

13 14 15 12

Langkah ke-23:

1 2 3 4

5 6 7 8

9 10 11 16

13 14 15 12

Langkah ke-24:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Waktu eksekusi : 1938.6367797851562 ms

Jumlah node yang dibangkitkan adalah : 72907