

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



Oleh:
Adzka Ahmadetya Zaidan 13520127

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II 2021/2022

Daftar Isi

Daftar Isi	1
1 Algoritma Branch and Bound Program dalam Menyelesaikan Persoalan	2
2 Screenshot Input-Output Program	3
2.1 Start of Program	3
2.2 Test Case tc1.txt Input-Output	4
2.3 Test Case tc2.txt Input-Output	5
2.4 Test Case tc3.txt Input-Output	7
2.5 Test Case tc4.txt Input-Output	10
2.6 Test Case tc5.txt Input-Output	10
3 Checklist Program	11
4 Kode Program	12
4.1 File main.py	12
4.2 File branchandbound.py	14
5 Berkas Teks Contoh Instansiasi Persoalan 15-puzzle	18
5.1 tc1.txt (Solvable)	18
5.2 tc2.txt (Solvable)	18
5.3 tc3.txt (Solvable)	18
5.4 tc4.txt (Unsolvable)	18
5.5 tc5.txt (Unsolvable)	18
6 Alamat Github yang Berisi Kode Program	19
Daftar Pustaka	19

1 Algoritma Branch and Bound Program dalam Menyelesaikan Persoalan

Algoritma branch and bound yang digunakan program untuk menyelesaikan persoalan 15-puzzle diawali dengan pengecekan nilai dari:

$$\sum_{i=1}^{16} KURANG(i) + X$$

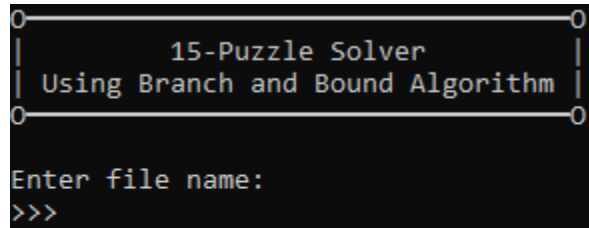
dan fungsi KURANG(i) mengembalikan jumlah tile di depan yang lebih kecil dari i, serta X merupakan nilai 0 atau 1 tergantung posisi nilai kosong. X didapatkan dengan penambahan posisi baris dengan kolom, dan dilakukan modulus dengan nilai 2. Jika nilai total tersebut genap, maka puzzle tersebut dapat diselesaikan dan program akan dilanjutkan.

Jika puzzle solvable, algoritma branch and bound akan dilakukan untuk pencarian simpul solusi dari simpul awal. Taksiran cost untuk simpul hidup merupakan jumlah step yang dibutuhkan dari simpul awal hingga suatu simpul tersebut, ditambah dengan jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir.

Langkah program dalam penyelesaian 15-puzzle menggunakan branch and bound diawali dengan membangkitkan simpul-simpul yang mungkin dibentuk dengan pertukaran tile kosong ke arah atas, kanan, bawah, dan kiri. Pertukaran dilakukan dengan constraint pertukaran tersebut dapat dilakukan (tidak out of bound) dan tidak kembali ke simpul sebelumnya (misal, tidak ekspansi ke arah kanan jika step simpul saat ini dibuat dari ekspansi ke arah kiri). Setelah itu, menghitung taksiran cost untuk masing-masing simpul yang telah dibangkitkan. Pembangkitan simpul selanjutnya dilakukan pada simpul dengan cost terkecil dan belum dibangkitkan. Program melakukan looping pencarian hingga ditemukan simpul yang sama dengan goal state.

2 *Screenshot Input-Output Program*

2.1 *Start of Program*



```
0-----0
|      15-Puzzle Solver      |
| Using Branch and Bound Algorithm |
0-----0

Enter file name:
>>>
```

2.2 Test Case tc1.txt Input-Output

```
Enter file name:
>>> tc1

Searching...

———— Solution Found !!! ————

| 1  -  2  3 |
| 5  6  7  4 |
| 9  10 11 8 |
| 13 14 15 12|
|—————|

RIGHT

| 1  2  -  3 |
| 5  6  7  4 |
| 9  10 11 8 |
| 13 14 15 12|
|—————|

RIGHT

| 1  2  3  - |
| 5  6  7  4 |
| 9  10 11 8 |
| 13 14 15 12|
|—————|

DOWN

| 1  2  3  4 |
| 5  6  7  - |
| 9  10 11 8 |
| 13 14 15 12|
|—————|

DOWN

| 1  2  3  4 |
| 5  6  7  8 |
| 9  10 11 - |
| 13 14 15 12|
|—————|

DOWN

| 1  2  3  4 |
| 5  6  7  8 |
| 9  10 11 12|
| 13 14 15 - |
|—————|

Solution Sequence (5): RIGHT RIGHT DOWN DOWN DOWN
Time taken: 0.003 seconds
Simplices created: 11
```

2.3 Test Case tc2.txt Input-Output

Enter file name:

>>> tc2

Searching...

———— Solution Found !!! ————

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

DOWN

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

DOWN

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

RIGHT

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

DOWN

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

LEFT

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

UP

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

RIGHT

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

DOWN

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

RIGHT

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

RIGHT

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

Solution Sequence (10): DOWN DOWN RIGHT DOWN LEFT UP RIGHT DOWN RIGHT RIGHT
Time taken: 0.04 seconds
Simplices created: 55

2.4 Test Case tc3.txt Input-Output

Enter file name:

>>> tc3

Searching...

———— Solution Found !!! ————

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

RIGHT

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

RIGHT

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

UP

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

LEFT

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

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

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

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

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

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

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

RIGHT

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

DOWN

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

LEFT

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

LEFT

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

UP

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

RIGHT

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

```

RIGHT
| 1  2  3  - |
| 5  6  7  4 |
| 9 10 11  8 |
|13 14 15 12 |
|
DOWN
| 1  2  3  4 |
| 5  6  7  - |
| 9 10 11  8 |
|13 14 15 12 |
|
DOWN
| 1  2  3  4 |
| 5  6  7  8 |
| 9 10 11  - |
|13 14 15 12 |
|
DOWN
| 1  2  3  4 |
| 5  6  7  8 |
| 9 10 11 12 |
|13 14 15  - |
|

Solution Sequence (20): RIGHT RIGHT UP LEFT LEFT LEFT UP RIGHT UP RIGHT RIGHT DOWN LEFT LEFT UP RIGHT RIGHT DOWN DOWN DOWN
Time taken: 5.662 seconds
Simplices created: 5355

```

2.5 Test Case tc4.txt Input-Output

```

Enter file name:
>>> tc4

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

Puzzle is unsolvable.
SigmaKurang(i) + X = 107

```

2.6 Test Case tc5.txt Input-Output

```

Enter file name:
>>> tc5

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

Puzzle is unsolvable.
SigmaKurang(i) + X = 95

```

3 *Checklist Program*

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		√

4 Kode Program

4.1 File main.py

```
# 13520127 Adzka Ahmadya Zaidan
import numpy as np
import branchandbound as f
import time as t
import os

cwd = os.path.dirname(os.getcwd())
os.chdir(cwd+"\\test")

print('O—————O'
)
print('|      15-Puzzle Solver      |')
print('| Using Branch and Bound Algorithm |')
print('O—————O\
n')

while True:
    filename = str(input('Enter file name:\n>>> '))
    if (len(filename) < 4):
        filename += '.txt'
    elif (filename[len(filename)-1] != 't' or filename[len(filename)-2] != 'x' or
filename[len(filename)-3] != 't' or filename[len(filename)-4] != '.'):
        filename += '.txt'
    simplice = np.loadtxt(filename, dtype='str', delimiter=' ')

    # Inisialisasi
    startS = simplice
    path = [[]]
    costs = [f.EstimationG(startS)+len(path[0])]
    directions = ['UP', 'RIGHT', 'DOWN', 'LEFT']

    print()
    # BRANCH AND BOUND
    if not f.Reachable(startS): # Jika puzzle unsolvable
        f.DisplaySimplices(startS, path[0])
        print('Puzzle is unsolvable.')
        print('SigmaKurang(i) + X = '+str(f.SigmaKurang(startS) + f.X(startS))+'\n')
    else: # Jika simpul awal memiliki suatu solusi
        print('Searching...')
        start = t.time()
        idx = 0
        # Melakukan loop jika belum ditemukan solusi
```

```

# Jika jumlah simpul yang dibangkitkan sudah terlalu banyak, pencarian berhenti
while not f.IsGoalState(f.SimpliceFromPath(startS, path[idx])):
    currS = f.SimpliceFromPath(startS, path[idx])
    if len(path[idx]) != 0:
        prevMove = path[idx][len(path[idx])-1]
    else:
        prevMove = ""
    # Melakukan branching dari suatu simpul hidup
    for dir in directions:
        if f.ValidMove(currS, dir, prevMove):
            path.append([])
            for i in range(len(path[idx])):
                path[len(path)-1].append(path[idx][i])
            path[len(path)-1].append(dir)
            costs.append(f.Cost(f.Move(currS, dir), len(path[idx])+1))
    costs[idx] = 999 # Berikan nilai cost yang sangat tinggi agar tidak lagi di-expand
    idx = costs.index(min(costs)) # Memilih cost terendah untuk di-expand

# Jika ditemukan suatu solusi
if f.IsGoalState(f.SimpliceFromPath(startS, path[idx])):
    end = t.time()
    timeTaken = round((end-start), 4)
    print('\n————— Solution Found !!! —————')
    f.DisplaySimplices(startS, path[idx])
    print('Solution Sequence'+(' '+str(len(path[idx]))+'): ', end="")
    for i in range(len(path[idx])):
        print(path[idx][i], end=' ')
    print('\nTime taken:', str(timeTaken), 'seconds')
    print('Simplices created:', len(path), '\n')

flag = True
while flag:
    exit = str(input('Continue program? (Y/N)\n>>> '))
    if (exit == 'Y' or exit == 'y'):
        flag = False
    elif (exit == 'N' or exit == 'n'):
        break
    else:
        print('Invalid input.\n')
    if (exit == 'N' or exit == 'n'):
        break
print('\nProgram successfully closed.\n')

```

4.2 File branchandbound.py

```
# 13520127 Adzka Ahmadetya Zaidan
import numpy as np

# Mengembalikan nilai dari Kurang(i)
def Kurang(flatM, i):
    idx = np.where(flatM == i)[0][0]
    count = 0
    for j in range(idx+1, 16):
        if (int(flatM[j]) > int(flatM[idx])):
            count += 1
    return count

# Mengembalikan nilai SigmaKurangi dari M
def SigmaKurang(M):
    flatM = M.flatten()
    emptyIdx = np.where(flatM == '-')
    flatM[emptyIdx] = '16'
    arrOfKurangi = [Kurang(flatM, i) for i in flatM]
    return sum(arrOfKurangi)

# Mengembalikan array of row, col dari nilai '-' simpul M
def IdxofEmpty(M):
    emptyIndexes = np.where(M == '-')
    return [emptyIndexes[0][0], emptyIndexes[1][0]]

# Mengembalikan X
def X(M):
    X = sum(IdxofEmpty(M)) % 2
    return X

# Mengembalikan boolean apakah simpul M reachable atau tidak
def Reachable(M):
    return (SigmaKurang(M) + X(M)) % 2 == 0

# Mengembalikan suatu simpul hasil dari simpul sebelumnya yang sudah digerakkan
def Move(M, direction):
    coords = IdxofEmpty(M)
    tempM = []
    for i in range(len(M)):
        cols = []
        for j in range(len(M[0])):
            cols.append(M[i][j])
        tempM.append(cols)
    if direction == 'UP':
```

```

    tempM[coords[0]][coords[1]] = tempM[coords[0]-1][coords[1]]
    tempM[coords[0]-1][coords[1]] = '-'
elif direction == 'RIGHT':
    tempM[coords[0]][coords[1]] = tempM[coords[0]][coords[1]+1]
    tempM[coords[0]][coords[1]+1] = '-'
elif direction == 'DOWN':
    tempM[coords[0]][coords[1]] = tempM[coords[0]+1][coords[1]]
    tempM[coords[0]+1][coords[1]] = '-'
elif direction == 'LEFT':
    tempM[coords[0]][coords[1]] = tempM[coords[0]][coords[1]-1]
    tempM[coords[0]][coords[1]-1] = '-'
return np.array([tempM[0], tempM[1], tempM[2], tempM[3]])

```

Mengembalikan estimasi g dari simpul M

```

def EstimationG(M):
    count = 0
    flatM = M.flatten()
    idx = IdxofEmpty(M)[0]*4 + IdxofEmpty(M)[1]
    flatM[idx] = '16'
    for i in range(16):
        if int(flatM[i]) != i+1:
            if i != idx:
                count += 1
    return count

```

Mengembalikan estimasi cost dari simpul M

```

def Cost(M, steps):
    return steps + EstimationG(M)

```

Mengembalikan boolean apakah suatu direction adalah valid untuk simpul M atau tidak

```

def ValidMove(M, direction, prevMove):
    coords = IdxofEmpty(M)
    if (direction == 'UP'):
        return prevMove != 'DOWN' and coords[0] != 0
    elif (direction == 'RIGHT'):
        return prevMove != 'LEFT' and coords[1] != 3
    elif (direction == 'DOWN'):
        return prevMove != 'UP' and coords[0] != 3
    elif (direction == 'LEFT'):
        return prevMove != 'RIGHT' and coords[1] != 0

```

Melakukan checking apakah M merupakan goalstate atau bukan

```

def IsGoalState(M):
    tempM = M.flatten()
    for i in range(15):
        if tempM[i] == '-':

```



```

    return False
    if int(tempM[i]) != i+1:
        return False
    return True

# Mengembalikan Simplicex yang terbentuk dengan sequence of directions dari startS
def SimplicexFromPath(startS, path):
    # path = array of direction
    tempM = []
    for i in range(len(startS)):
        cols = []
        for j in range(len(startS[0])):
            cols.append(startS[i][j])
        tempM.append(cols)
    tempM = np.array([tempM[0], tempM[1], tempM[2], tempM[3]])
    for i in range(len(path)):
        tempM = Move(tempM, path[i])
    return np.array([tempM[0], tempM[1], tempM[2], tempM[3]])

# Display sequence simpul hingga menuju simpul startS
def DisplaySimplices(startS, path):
    tempM = []
    for i in range(len(startS)):
        cols = []
        for j in range(len(startS[0])):
            cols.append(startS[i][j])
        tempM.append(cols)
    tempM = np.array([tempM[0], tempM[1], tempM[2], tempM[3]])
    PrintSimplicex(tempM) # PRINTING MATRIX
    for i in range(len(path)):
        tempM = Move(tempM, path[i])
        print(path[i])
        PrintSimplicex(np.array([tempM[0], tempM[1], tempM[2], tempM[3]]))

# Print suatu simpul M
def PrintSimplicex(M):
    print('_____')
    for i in range(4):
        for j in range(4):
            if j == 0:
                print('| ', end="")
            if (len(M[i][j]) == 2):
                print(M[i][j]+' ', end="")
            else:
                print(M[i][j]+' ', end="")
            if j == 3:

```

```
print('')  
print('_____')
```

5 Berkas Teks Contoh Instansiasi Persoalan 15-puzzle

5.1 tc1.txt (Solvable)

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

5.2 tc2.txt (Solvable)

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

5.3 tc3.txt (Solvable)

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

5.4 tc4.txt (Unsolvable)

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

5.5 tc5.txt (Unsolvable)

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

6 Alamat Github yang Berisi Kode Program

<https://github.com/Voguelish/branch-and-bound-tucil-3-stima>

Daftar Pustaka

- 1) “Algoritma Branch & Bound Bagian 1”. informatika.stei.itb.ac.id. 2022. Diakses pada tanggal 1 April 2022.
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf>
- 2) “Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound”. informatika.stei.itb.ac.id. 2022. Diakses pada tanggal 2 April 2022.
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Tugas-Kecil-2-\(2022\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Tugas-Kecil-2-(2022).pdf)