Laporan Tugas Kecil 3

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Yoseph Alexander Siregar
K03 / 13520141

Teknik Informatika

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

I. Algoritma Branch and Bound - Permainan 15-Puzzle

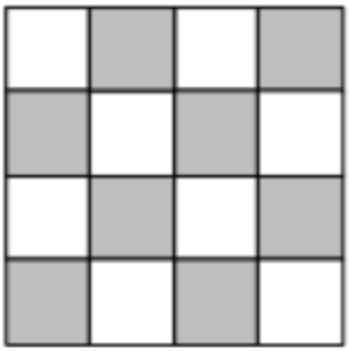
Algoritma Branch and Bound adalah algoritma yang biasa digunakan untuk persoalan optimisasi. Optimisasi yang dimaksud di sini adalah meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (constraints) persoalan tersebut.

Pada penerapannya, Algortima Branch and Bound memiliki konsep yang mirip dengan Algoritma BFS (*Breadth First Search)*, dimana penerepannya dilakukan dengan konsep pohon dengan akar dan penyelesaian persoalan dilakukan dengan menulusuri daun atau simpulnya. Berbeda dengan algortima BFS yang ekspansi simpul berikutnya didasarkan pada urutan pembangkitannya, algoritma Branch and Bound melakukan ekspansi simpul terhadap simpul dengan biaya atau *cost* terkecil. Penghitungan biaya pada simpul ini berbeda-beda menyesuaikan dengan persoalan yang ingin diselesaikan.

Salah satu persoalan yang dapat diselesaikan dengan menggunakan algoritma Branch and Bound ini adalah persoalan 15-Puzzle. 15-Puzzle merupakan persoalan puzzle (pada implementasinya menggunakan konsep matriks) berisi angka 1 hingga 15 pada tiap sel (*cell)* -nya dan sebuah sel kosong (untuk mempermudah diimplementasikan dengan angka 16) dengan posisi yang acak. Persoalan yang ingin diselesaikan adalah untuk mengubah posisi sel kosong dengan pergerakan 4 arah (atas, kanan, bawah, kiri) hingga seluruh angka pada Puzzle tersusun dari 1 hingga 15 dan diakhiri dengan sel kosong tersebut.

Pada tugas kali ini, diminta untuk membuat sebuah program yang dapat menyelesaikan persoalan 15-Puzzle dengan memanfaatkan Algoritma Branch and Bound.

Terdapat 16! atau kira -kira sama dengan 20,9 x 1012 susunan sel yang berbeda, dan hanya setengah yang dapat dicapai dari state awal sembarang. Untuk mempermudah mengetahui apakah susunan Puzzle awal dapat diselesaikan dan mencapai target yang diinginkan, digunakan jumlah fungsi Kurang(i) dengan i = 1 hingga 16 ditambah dengan suatu nilai X yang ditentukan berdasarkan posisi sel kosong pada puzzle (dapat berupa 0 atau 1).

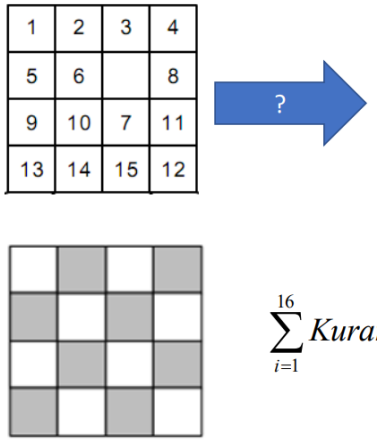


Gambar 1.1. Nilai X = 1 bila terletak pada sel yang diarsir

Sumber :

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

Fungsi Kurang(i) dapat dihitung dengan menghitung banyaknya sel yang memiliki nilai kurang dari i tetapi posisinya lebih besar dari sel i tersebut. Apabila hasil penjumlahan dari seluruh Kurang(i) ditambah dengan nilai variabel X berupa bilangan genap maka puzzle tersebut dapat diselesaikan, apabila berupa bilangan ganjil maka puzzle tersebut tidak dapat diselesaikan.



$$\sum_{i=1}^{16} Kurang(i) + X = 15 + 1 = 16$$

i	Kurang (i)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	1
10	1
11	0
12	0
13	1
14	1
15	1
16	9

Gambar 1.2. Contoh perhitungan

Sumber :

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

Pada gambar di atas, dapat dilihat bahwa hasil penjumlahan berupa bilangan genap sehingga dapat ditentukan bahwa puzzle tersebut dapat diselesaikan. Untuk penyelesaiaan, biaya yang digunakan sebagai parameter ekspansi simpul didapat dengan membandingkan kondisi puzzle pada simpul dengan puzzle target (puzzle yang diinginkan) dan menghitung banyaknya sel tidak kosong yang tidak sesuai nilainya pada puzzle target. Pada pengerjaan tugas kali ini, simpul-simpul diimplementasikan dalam bentuk *node* yang berisi *root* yaitu *parent* dari node tersebut, matriks representasi puzzle, posisi baris dan kolom dari sel kosong, biaya dari simpul tersebut, *depth* atau kedalaman dari simpul, dan pergerakan sel kosong sebelumnya. Simpul-simpul yang ada disimpan pada sebuah *Priority Queue* yang prioritasnya didasarkan pada biaya simpul.

Langkah penyelesaian program saya untuk permainan 15-Puzzle ini adalah :

1. Meminta input dari pengguna untuk metode masukan puzzle yang akan digunakan, apakah menggunakan file text pada folder ‘test’ atau dengan melalui matriks random yang akan di-*generate* oleh program.
2. Program akan menampilkan matriks representasi puzzle yang akan diselesaikan.
3. Program lalu memberikan output dari fungsi Kurang(i) untuk tiap sel tidak kosong, nilai variabel X dan jumlah dari Sum(Kurang(i)) dan variabel X tersebut. Program juga akan memberikan pesan apakah puzzle tersebut dapat diselesaikan atau tidak (didasarkan dari hasil penjumlahan di atas).
4. Apabila puzzletidak dapat diselesaikan maka program sudah selesai dan berhenti. Apabila puzzle dapat diselesaikan, program akan menampilkan matriks posisi awal dari puzzle lalu menampilkan gerakan yang dialami sel kosong (Up/Down/Left/Right) beserta matriks puzzle setelah sel digerakkan, hingga matriks representasi puzzle sudah sesuai dengan puzzle target (puzzle sudah diselesaikan).

5. Program juga akan mengeluarkan output jumlah simpul yang dibangkitkan beserta waktu yang dibutuhkan program untuk menyelesaikan puzzle.

Algoritma Branch and Bound dalam program saya adalah sebagai berikut, program pertama-tama akan membuat sebuah *priority queue,* program lalu membuat sebuah node untuk matriks representasi puzzle, langkah pemecahan untuk puzzle nya adalah program akan membangkitkan simpul dari puzzle awal (atau simpul yang akan diekspansi) sejumlah pergerakan yang mungkin dari sel kosong (berdasarkan 4 arah gerak yang sudah disebutkan di atas) dalam bentuk *node* yang sudah dijelaskan di atas. Pada tiap simpul itu, disimpan juga biaya nya berdasarkan perbandingan dengan puzzle target. Tiap simpul tersebut lalu dimasukkan dalam *priority queue* untuk disimpan. Priority queue lalu akan di-*pop* dan mengeluarkan simpul dengan biaya terkecil, yang selanjutnya akan diekspansi. Proses ini dilakukan terus menerus hingga biaya dari salah satu simpul bernilai 0 ‘nol’ yang dimana menandakan posisi seluruh nilai puzzle sudah sesuai dengan puzzle target yang menandakan persoalan sudah berhasil diselesaikan.

II. Hasil Eksekusi (Input - Output)

1. Input test1.txt

```
////////////////////////////////////

Puzzle Solver for 15-Puzzle Game

Implemented with Branch and Bound Algorithm

Enter the initial state of the puzzle with the chosen method
1. Enter the puzzle from txt file
2. Enter the puzzle randomly
p.s. The second method may make a really complicated puzzle
p.s. If it does not work, try again or try the first method

Chosen Method : 1

Enter the name of txt file from the test folder (with extension .txt)
Filename : test1.txt
```

Gambar 2.1.1 - Input Program
Sumber : Arsip Pribadi

```
Puzzle Matrix :

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


Value of the function Kurang(i) on each non-empty cell:

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  1
Kurang( 4 ) =  1
Kurang( 5 ) =  0
Kurang( 6 ) =  0
Kurang( 7 ) =  1
Kurang( 8 ) =  0
Kurang( 9 ) =  0
Kurang( 10 ) =  0
Kurang( 11 ) =  3
Kurang( 12 ) =  6
Kurang( 13 ) =  0
Kurang( 14 ) =  4
Kurang( 15 ) =  11
Kurang( 16 ) =  10

Value of X =  0

Sum(Kurang(i)) + X =  37

Sorry... The puzzle is not solvable
////////////////////////////////////
```

Gambar 2.1.2 - Output Program
Sumber : Arsip Pribadi

2. Matriks test2.txt

```
////////////////////////////////////

Puzzle Solver for 15-Puzzle Game

Implemented with Branch and Bound Algorithm

Enter the initial state of the puzzle with the chosen method
1. Enter the puzzle from txt file
2. Enter the puzzle randomly
p.s. The second method may make a really complicated puzzle
p.s. If it does not work, try again or try the first method

Chosen Method : 1

Enter the name of txt file from the test folder (with extension .txt)
Filename : test2.txt
```

Gambar 2.2.1 - Input Program
Sumber : Arsip Pribadi

```
Puzzle Matrix :

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


Value of the function Kurang(i) on each non-empty cell:

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  0
Kurang( 4 ) =  0
Kurang( 5 ) =  0
Kurang( 6 ) =  0
Kurang( 7 ) =  0
Kurang( 8 ) =  1
Kurang( 9 ) =  1
Kurang( 10 ) =  1
Kurang( 11 ) =  0
Kurang( 12 ) =  0
Kurang( 13 ) =  1
Kurang( 14 ) =  1
Kurang( 15 ) =  1
Kurang( 16 ) =  9

Value of X =  1

Sum(Kurang(i)) + X =  16

The puzzle is solvable

Steps to solve the puzzle :
```

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


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


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


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


Child Node Generated =  9

Execution Time =  0.010004758834838867 second
////////////////////////////////////
```

Gambar 2.2.2 - Output Program
Sumber : Arsip Pribadi

3. Matriks test3.txt

```
////////////////////////////////////

Puzzle Solver for 15-Puzzle Game

Implemented with Branch and Bound Algorithm

Enter the initial state of the puzzle with the chosen method
1. Enter the puzzle from txt file
2. Enter the puzzle randomly
p.s. The second method may make a really complicated puzzle
p.s. If it does not work, try again or try the first method

Chosen Method : 1

Enter the name of txt file from the test folder (with extension .txt)
Filename : test3.txt
```

Gambar 2.3.1 - Input Program
Sumber : Arsip Pribadi

```
Puzzle Matrix :

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



Value of the function Kurang(i) on each non-empty cell:

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  0
Kurang( 4 ) =  0
Kurang( 5 ) =  0
Kurang( 6 ) =  0
Kurang( 7 ) =  0
Kurang( 8 ) =  0
Kurang( 9 ) =  0
Kurang( 10 ) =  1
Kurang( 11 ) =  1
Kurang( 12 ) =  1
Kurang( 13 ) =  1
Kurang( 14 ) =  2
Kurang( 15 ) =  0
Kurang( 16 ) =  7

Value of X =  0

Sum(Kurang(i)) + X =  13

Sorry... The puzzle is not solvable
////////////////////////////////////
```

Gambar 2.3.2 - Output Program
Sumber : Arsip Pribadi

4. Matriks test4.txt

```
////////////////////////////////////

Puzzle Solver for 15-Puzzle Game

Implemented with Branch and Bound Algorithm

Enter the initial state of the puzzle with the chosen method
1. Enter the puzzle from txt file
2. Enter the puzzle randomly
p.s. The second method may make a really complicated puzzle
p.s. If it does not work, try again or try the first method

Chosen Method : 1

Enter the name of txt file from the test folder (with extension .txt)
Filename : test4.txt
```

Gambar 2.4.1 - Input Program
Sumber : Arsip Pribadi

```
Puzzle Matrix :

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



Value of the function Kurang(i) on each non-empty cell:

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  0
Kurang( 4 ) =  0
Kurang( 5 ) =  0
Kurang( 6 ) =  0
Kurang( 7 ) =  0
Kurang( 8 ) =  0
Kurang( 9 ) =  0
Kurang( 10 ) =  1
Kurang( 11 ) =  1
Kurang( 12 ) =  1
Kurang( 13 ) =  0
Kurang( 14 ) =  0
Kurang( 15 ) =  0
Kurang( 16 ) =  7

Value of X =  0

Sum(Kurang(i)) + X =  10

The puzzle is solvable

Steps to solve the puzzle :
```

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


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


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


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


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


Child Node Generated =  8

Execution Time =  0.011002063751220703 second
```

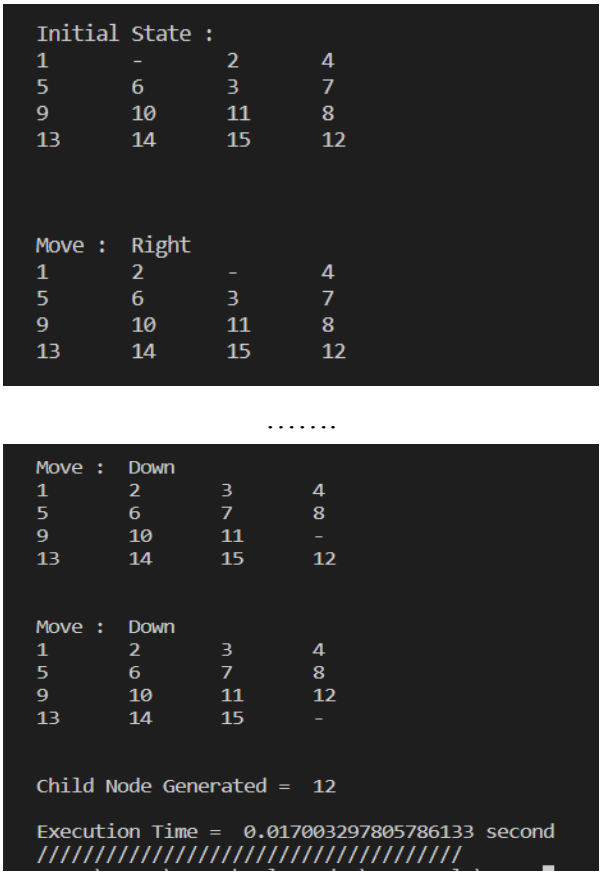
Gambar 2.4.2 - Output Program
Sumber : Arsip Pribadi

5. Matriks test5.txt

```
////////////////////////////////////

Puzzle Solver for 15-Puzzle Game

Implemented with Branch and Bound Algorithm

Enter the initial state of the puzzle with the chosen method
1. Enter the puzzle from txt file
2. Enter the puzzle randomly
p.s. The second method may make a really complicated puzzle
p.s. If it does not work, try again or try the first method

Chosen Method : 1

Enter the name of txt file from the test folder (with extension .txt)
Filename : test5.txt
```

Gambar 2.5.2 - Input Program
Sumber : Arsip Pribadi

```
Puzzle Matrix :

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



Value of the function Kurang(i) on each non-empty cell:

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  0
Kurang( 4 ) =  1
Kurang( 5 ) =  1
Kurang( 6 ) =  1
Kurang( 7 ) =  0
Kurang( 8 ) =  0
Kurang( 9 ) =  1
Kurang( 10 ) =  1
Kurang( 11 ) =  1
Kurang( 12 ) =  0
Kurang( 13 ) =  1
Kurang( 14 ) =  1
Kurang( 15 ) =  1
Kurang( 16 ) =  14

Value of X =  1

Sum(Kurang(i)) + X =  24

The puzzle is solvable

Steps to solve the puzzle :
```



Gambar 2.5.3 - Output Program
Sumber : Arsip Pribadi

6. Input Metode Random

```
////////////////////////////////////

Puzzle Solver for 15-Puzzle Game

Implemented with Branch and Bound Algorithm

Enter the initial state of the puzzle with the chosen method
1. Enter the puzzle from txt file
2. Enter the puzzle randomly
p.s. The second method may make a really complicated puzzle
p.s. If it does not work, try again or try the first method

Chosen Method : 2
```

Gambar 2.5.1 - Input Program
Sumber : Arsip Pribadi

```
Puzzle Matrix :

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



Value of the function Kurang(i) on each non-empty cell:

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  0
Kurang( 4 ) =  0
Kurang( 5 ) =  4
Kurang( 6 ) =  4
Kurang( 7 ) =  4
Kurang( 8 ) =  4
Kurang( 9 ) =  0
Kurang( 10 ) =  0
Kurang( 11 ) =  0
Kurang( 12 ) =  0
Kurang( 13 ) =  12
Kurang( 14 ) =  12
Kurang( 15 ) =  12
Kurang( 16 ) =  12

Value of X =  1

Sum(Kurang(i)) + X =  65

Sorry... The puzzle is not solvable
////////////////////////////////////
```

Gambar 2.5.2 - Output Program
Sumber : Arsip Pribadi

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

III. Kode Program

fifteenPuzzleSolver.py

```
import numpy as np
from heapq import heappush, heappop

# the target matrix
target_matrix = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
# global variabel to count the number of child node generated
childNodeCount = 0

# function to read matrix from file
def read_matrix(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
    lines = [line.strip() for line in lines]
    lines = [line.split(' ') for line in lines]
    lines = [[int(x) for x in line] for line in lines]
    return np.array(lines)

# function to find the position of a certain value in a matrix
def find_value(matrix, value):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if matrix[i][j] == value: # if the value is found
                return i, j # return the row and column of the value

# function to determine the value of X based on the position of the blank cell
def x_value(matrix):
    null_row, null_col = find_value(matrix, 16) # find the position of the blank cell
    sum = null_row + null_col
    if sum % 2 == 1:
        return 1
    else:
        return 0

# function kurang(i) and the sum of it
def KURANG_I(matrix):
    sum = 0 # sum of kurang(i)
    for value in range(1,17): # for every value from 1 to 16
        row, col = find_value(matrix, value) # position of the value i
        count = 0 # the value of kurang(i)
        # function to find kurang(i) from 1 to 16
        for i in range(len(matrix)):
            for j in range(len(matrix[i])):
                if matrix[i][j] < value: # for every value smaller than i
                    # check whether the position of it is bigger than the position of value
i
                    if row == i: # if the row of it is the same as the row of value i
                        if j > col: # if the column of it is bigger than the column of value
i
```

```
                            count += 1 # increment the count
                    elif row < i: # if the row of it is smaller than the row of value i
                        count += 1 # increment the count
        sum += count # add the count to the sum
        print("Kurang(",value,") = ", count) # print the value of each kurang(i) from 1 to
16
    return sum

# function to determine whether the puzzle is solvable
def solvable(matrix):
    sum_kurang = KURANG_I(matrix) # sum of kurang(i)
    x = x_value(matrix) # value of x
    total = sum_kurang + x # sum of Sum(kurang(i)) + x
    print("\nValue of X = ", x) # print the value of x
    print("\nSum(Kurang(i)) + X = ", total) # print the sum of Kurang(i) and X
    # if the sum of Sum(Kurang(i)) and X is even, the puzzle is solvable
    if (total) % 2 == 0:
        return True
    else:
        return False

# function to find the cost of moving the blank cell (number 16)
def cost(matrix):
    cost = 0
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            # for every non-blank cell different than the supposed value in the target
matrix
            if matrix[i][j] != 16:
                if matrix[i][j] != target_matrix[i][j]:
                    cost += 1 # increment the cost
    return cost

class priority_queue: # priority queue class
    def __init__(self): # const
        self.heap = []

    def push(self, value): # push the value into the heap
        heappush(self.heap, value)

    def pop(self): # pop the value from the heap
        return heappop(self.heap)

    def isEmpty(self): # check whether the heap is empty
        if not self.heap:
            return True
        else:
            return False

class Node: # node class
    def __init__(self, root, mat, blank_row, blank_col, cost, depth, prev_move): # const
        self.root = root # node root
```

```
        self.mat = mat  # the matrix of the node
        self.blank_row = blank_row # row of the blank cell in the matrix
        self.blank_col = blank_col # column of the blank cell in the matrix
        self.cost = cost # cost of the node (the cost of moving the blank cell)
        self.depth = depth # depth of the node
        self.prev_move = prev_move # previous move to get to the node from the root

    def __lt__(self, other): # compare the cost of two nodes
        return self.cost < other.cost

def makeChildNode(node): # function to make child node
    childNode =[] # list of child node
    valid_move = [] # list of valid move
    valid_move = nextValidMove(node.blank_row,node.blank_col ,node.prev_move) # find the
valid move
    for move in valid_move: # for every valid move
        # find the new position for the blank cell in the matrix
        if move == "Up": # move the blank cell up
            new_row = node.blank_row - 1
            new_col = node.blank_col
        elif move == "Down": # move the blank cell down
            new_row = node.blank_row + 1
            new_col = node.blank_col
        elif move == "Left": # move the blank cell left
            new_row = node.blank_row
            new_col = node.blank_col - 1
        elif move == "Right": # move the blank cell right
            new_row = node.blank_row
            new_col = node.blank_col + 1
        global childNodeCount # global variable to count the number of child node generated
        mat = node.mat
        blank_row = node.blank_row
        blank_col = node.blank_col
        depth = node.depth

        new_mat = np.copy(mat) # copy the matrix
        save = new_mat[blank_row][blank_col] # save the value of the blank cell
        # swap the blank_cell based on the move
        new_mat[blank_row][blank_col] = new_mat[new_row][new_col]
        new_mat[new_row][new_col] = save

        # create a new node
        childNode.append(Node(node, new_mat, new_row, new_col, cost(new_mat) + depth + 1,
depth+1, move))
        childNodeCount += 1 # increment the number of child node
    return childNode

def nextValidMove(blank_row, blank_col, prev_move): # function to find the valid move
    validMove = ["Up", "Down", "Left", "Right"] # list of valid move
    if blank_row == 3 or prev_move == "Up": # if the blank cell is in the last row or the
previous move is up
        validMove.remove("Down") # remove down from the list of valid move
```

```
    if blank_col == 0 or prev_move == "Right": # if the blank cell is in the first column or
the previous move is right
        validMove.remove("Left") # remove left from the list of valid move
    if blank_row == 0 or prev_move == "Down": # if the blank cell is in the first row or the
previous move is down
        validMove.remove("Up") # remove up from the list of valid move
    if blank_col == 3 or prev_move == "Left": # if the blank cell is in the last column or
the previous move is left
        validMove.remove("Right") # remove right from the list of valid move
    return validMove

def print_steps(node): # function to print the steps from the original matrix until the
final matrix (same as target matrix)
    if node.root == None: # if the node is the root
        print("")
    else:
        print_steps(node.root) # print the steps from the root
        print("Move : ", node.prev_move) # print the move
        printMatrix(node.mat) # print the matrix

def printMatrix(mat): # function to print the matrix
    for i in range(len(mat)):
        for j in range(len(mat[i])):
            if mat[i][j] == 16: # if the value is 16, print blank cell
                print("-", end="\t")
            else: # print the value
                print(mat[i][j], end="\t")
        print("")
    print("\n")

def solvePuzzle(mat): # function to solve the puzzle
    pq = priority_queue() # priority queue
    pq.push(Node(None, mat, find_value(mat, 16)[0], find_value(mat,16)[1], cost(mat), 0,
"None")) # push the root node
    while not pq.isEmpty(): # while the priority queue is not empty
        node = pq.pop() # pop the node (the node with the lowest cost)
        if node.cost == 0 or node.cost == node.depth: # if the cost is 0 or the depth is the
same as the cost
            # the puzzle is solved
            print_steps(node) # print the steps
            print("Child Node Generated = ", childNodeCount) # print the number of child
node generated
            return
        else:
            childNode = makeChildNode(node) # make the child node
            for child in childNode: # for every child node
                pq.push(child) # push the child node into the priority queue
```

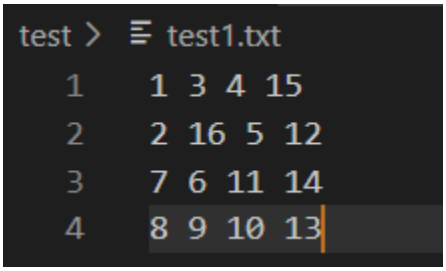
main.py

```
from fifteenPuzzleSolver import *
import time

print("////////////////////////////////////")
print("\nPuzzle Solver for 15-Puzzle Game\n")
print("Implemented with Branch and Bound Algorithm\n")
print("Enter the initial state of the puzzle with the chosen method")
print("1. Enter the puzzle from txt file")
print("2. Enter the puzzle randomly")
print("p.s. The second method may make a really complicated puzzle")
print("p.s. If it does not work, try again or try the first method")
method = int(input("\nChosen Method : "))

if method == 1:
    print("\nEnter the name of txt file from the test folder (with extension .txt)")
    try:
        filename = input("Filename : ")
        location = "../test/" + filename
        matrix = read_matrix(location)
        print("\nPuzzle Matrix : \n")
        printMatrix(matrix)
        print("\nValue of the function Kurang(i) on each non-empty cell:\n")
        solvable = solvable(matrix)
        if solvable:
            print("\nThe puzzle is solvable")
            print("\nSteps to solve the puzzle : ")
            print("\nInitial State : ")
            printMatrix(matrix)
            start = time.time()
            solvePuzzle(matrix)
            stop = time.time()
            print("\nExecution Time = ", stop - start, "second")
            print("////////////////////////////////////")
        else:
            print("\nSorry... The puzzle is not solvable")
            print("////////////////////////////////////")
    except:
        print("\nSorry... The file is not found")
        print("////////////////////////////////////")
elif method == 2:
    mat = np.arange(1,17).reshape(4,4)
    mat = np.random.permutation(mat)
    print("\nPuzzle Matrix : \n")
    printMatrix(mat)
    print("\nValue of the function Kurang(i) on each non-empty cell:\n")
    solvable = solvable(mat)
    if solvable:
        print("\nThe puzzle is solvable")
        print("\nSteps to solve the puzzle : ")
        print("\nInitial State : ")
        printMatrix(mat)
```

```
        start = time.time()
        solvePuzzle(mat)
        stop = time.time()
        print("\nExecution Time = ", stop - start, "second")
        print("////////////////////////////////////")

    else:
        print("\nSorry... The puzzle is not solvable")
        print("////////////////////////////////////")
else:
    print("Sorry... Wrong Input")
    print("////////////////////////////////////")
```

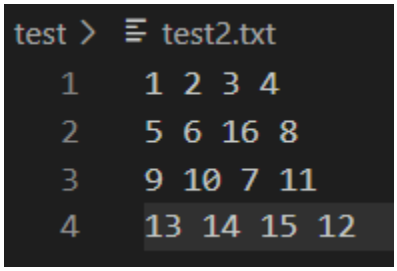
IV. Lampiran (Instansiasi 5 buah persoalan 15-puzzle)

1. Matriks test1.txt - status tujuan tidak dapat dicapai



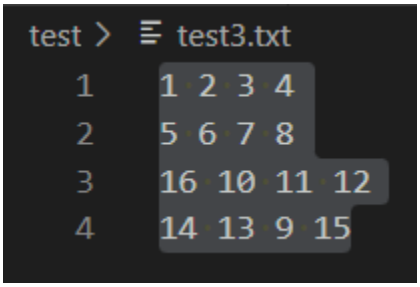
Gambar 4.1.1 - Matriks test1.txt
Sumber : Arsip Pribadi

2. Matriks test2.txt - status tujuan dapat dicapai



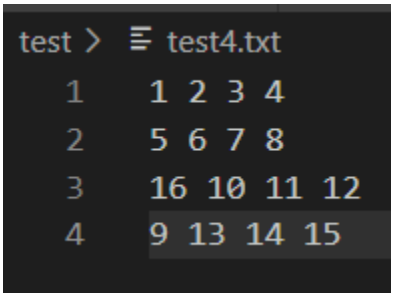
Gambar 4.2.1 - Matriks test2.txt
Sumber : Arsip Pribadi

3. Matriks test3.txt - status tujuan tidak dapat dicapai



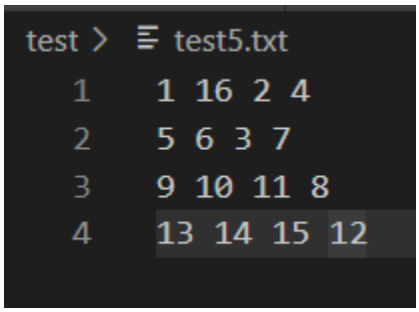
Gambar 4.3.1 - Matriks test3.txt
Sumber : Arsip Pribadi

4. Matriks test4.txt - status tujuan]dapat dicapai



Gambar 4.4.1 - Matriks test4.txt
Sumber : Arsip Pribadi

5. Matriks test5.txt - status tujuan dapat dicapai



Gambar 4.5.1 - Matriks test5.txt
Sumber : Arsip Pribadi

V. Link

https://github.com/yosalx/15-Puzzle

VI. Daftar Pustaka

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Tugas-Kecil-3-(2022).pdf

https://www.geeksforgeeks.org/branch-and-bound-algorithm/

https://www.geeksforgeeks.org/priority-queue-in-python/

https://www.codecademy.com/learn/learn-data-structures-and-algorithms-with-python/modules/nodes/cheatsheet