LAPORAN TUGAS KECIL

IF2211/Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Dipersiapkan oleh:

Vionie Novencia Thanggestyo

13520006 K3

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

A. Cara Kerja program Branch and Bound

Puzzle yang ingin di solve diperlakukan sebagai matrix, mula-mula dihitung sigma kurang + X dari puzzle untuk mengecek apakah puzzle solveable, jika hasilnya negatif maka puzzle tidak solveable dan sebaliknya. Jika puzzle solveable, puzzle sebagai node parent, kemudian dibuat 4 child yang merupakan turunan dari parent. Setiap node dihitung costnya dengan cara menambahkan kedalaman simpul dari akar dan estimasi cost untuk ke final. simpul-simpul tersebut kemudian dimasukkan ke priority queue dengan sistem mengurut membesar. Simpul dengan cost terkecil dipilih sebagai parent node lokal, yang kemudian akan men-generate child node, begitu seterusnya sampai didapat puzzle final.

B. Screenshoot Input-Output Program

1. Puzzle1.txt

```
Masukkan nama file: puzzle1.txt
Kurang(1) = 0
Kurang(2) = 1
Kurang(3) = 1
Kurang(4) = 1
Kurang(5) = 0
Kurang(6) = 1
Kurang(7) = 2
Kurang(8) = 1
Kurang(9) = 3
Kurang(10) = 6
Kurang(11) = 2
Kurang(12) = 0
Kurang(13) = 3
Kurang(14) = 1
Kurang(15) = 4
Kurang(16) = 2
SigmaKurang(i): 28
2  3  4  10
1  7  9  6
11 13 15 8
5  16  14 12

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

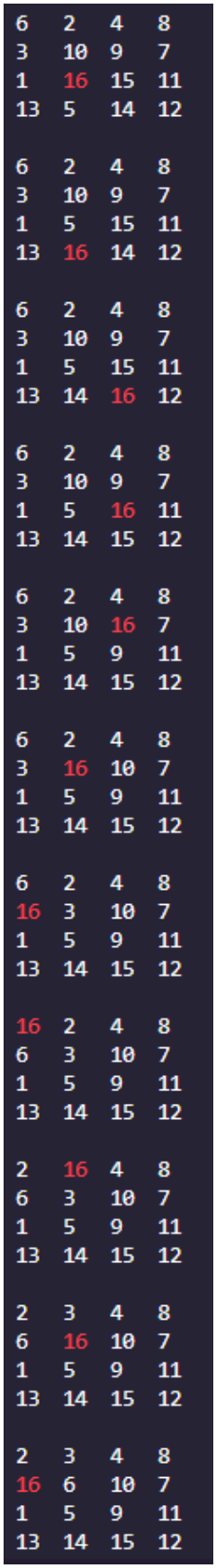
```
2  3  6  4
1  9  10 8
5  16  7  11
13 14 15 12

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

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

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

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

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

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

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

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

Banyak anak dibangkitkan: 1846802
Execution time:  77.24075746536255
```

2. Puzzle2.txt

```
Masukkan nama file: puzzle2.txt
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
SigmaKurang(i): 37
puzzle tidak solveable
```

3. Puzzle3.txt

```
Masukkan nama file: puzzle3.txt
Kurang(1) = 0
Kurang(2) = 1
Kurang(3) = 1
Kurang(4) = 2
Kurang(5) = 0
Kurang(6) = 5
Kurang(7) = 2
Kurang(8) = 4
Kurang(9) = 3
Kurang(10) = 4
Kurang(11) = 0
Kurang(12) = 1
Kurang(13) = 3
Kurang(14) = 4
Kurang(15) = 5
Kurang(16) = 4
SigmaKurang(i): 40
6   2   4   8
3   10  9   7
1   15  14  16
13  5   12  11

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

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

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

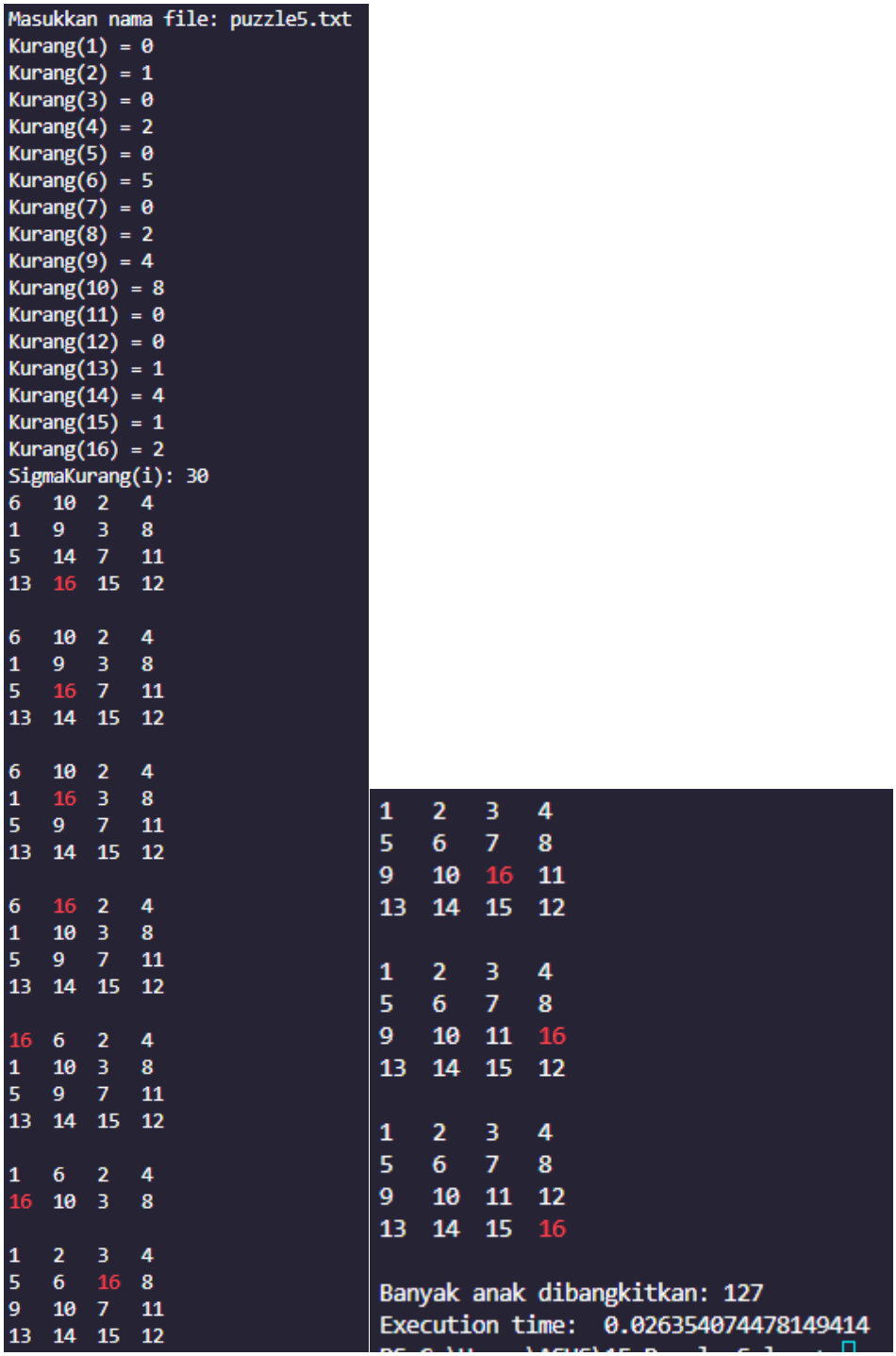

6 2 4 8
3 10 9 7
1 16 15 11
13 5 14 12
6 2 4 8
3 10 9 7
1 5 15 11
13 16 14 12
6 2 4 8
3 10 9 7
1 5 15 11
13 14 16 12
6 2 4 8
3 10 9 7
1 5 16 11
13 14 15 12
6 2 4 8
3 10 16 7
1 5 9 11
13 14 15 12
6 2 4 8
3 16 10 7
1 5 9 11
13 14 15 12
6 2 4 8
16 3 10 7
1 5 9 11
13 14 15 12
16 2 4 8
6 3 10 7
1 5 9 11
13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

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

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

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

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

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

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

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

Banyak anak dibangkitkan: 1592846
Execution time:  53.68953609466553
```

4. Puzzle4.txt

```
Masukkan nama file: puzzle4.txt
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 0
Kurang(4) = 2
Kurang(5) = 4
Kurang(6) = 2
Kurang(7) = 5
Kurang(8) = 2
Kurang(9) = 0
Kurang(10) = 4
Kurang(11) = 3
Kurang(12) = 2
Kurang(13) = 2
Kurang(14) = 3
Kurang(15) = 5
Kurang(16) = 1
SigmaKurang(i): 37
puzzle tidak solveable
```

5. Puzzle5.txt



C. Source Code

1. preprogress.py

```
#mengisi matrix puzzle dengan inputan user
def getPuzzle():
    puzzle = [list(map(int,input().split())) for i in range(4)]
    return puzzle

#16 = ubin kosong

#menghitung nilai fungsi kurang(i) dari ubin i
def kurang(puzzle,i,j ):
    count = 0
    ubin = i*4 + j+1
    for u in range(4):
        for v in range(4):
            ubin2 = u*4 + v+1
            if(puzzle[u][v] < puzzle[i][j] and ubin2>ubin):
                count+=1
    return count

def printKurang(puzzle):
    array = [0 for i in range(17)]
    for p in range(4):
        for q in range(4):
            array[puzzle[p][q]] = kurang(puzzle,p,q)
    for i in range(1,17):
        print("Kurang("+ str(i)+ ") =", array[i])

#hitung kurang(i)+X
def isReachableGoal(puzzle):
    sum = 0
    X = -1
    for i in range(4):
        for j in range(4):
            if (puzzle[i][j] == 16):
                X = (i+j)%2
            sum += kurang(puzzle,i,j)
    if (X != -1):
        sum += X
        return sum
    else:
        return -1

#mendapatkan tile yang kosong
def emptyTile(puzzle):
```

```
    for i in range(4):
        for j in range(4):
            if(puzzle[i][j] == 16):
                return [i,j]

```

2. Solver.py

```
from asyncio.windows_events import NULL
import copy
from heapq import heappush, heappop

#Class priorityqueue
class priorityQueue:
    def __init__(self):
        self.heap = []

    def push(self, k):
        heappush(self.heap, k)

    def pop(self):
        return heappop(self.heap)

    def isEmpty(self):
        if not self.heap:
            return True
        else:
            return False
#class node
class node:

    def __init__(self, parent,puz, empty_tile_pos,
                 cost, level,direction):

        self.parent = parent

        self.puz = puz

        self.empty_tile_pos = empty_tile_pos

        self.cost = cost

        self.level = level

        self.direction = direction
```

```
    def __lt__(self, nxt):
        return self.cost < nxt.cost

#menghitung cost simpul
def calculateCost(puz, final,level) -> int:

    count = level
    for i in range(4):
        for j in range(4):
            if (puz[i][j] and puz[i][j] != final[i][j]):
                count += 1

    return count

#membuat node baru
def newNode(mat, empty_tile_pos, new_empty_tile_pos,
            level, parent, final,direction) -> node:

    new_mat = copy.deepcopy(mat)

    x1 = empty_tile_pos[0]
    y1 = empty_tile_pos[1]
    x2 = new_empty_tile_pos[0]
    y2 = new_empty_tile_pos[1]
    new_mat[x1][y1], new_mat[x2][y2] = new_mat[x2][y2], new_mat[x1][y1]

    cost = calculateCost(new_mat, final,level)

    new_node = node(parent, new_mat, new_empty_tile_pos,
                    cost, level,direction)
    return new_node

#mencetak jawaban dari akar ke daun
def printTree(root):

    if root == None:
        return

    printTree(root.parent)
    printMatrix(root.puz)
    print()

#print matrix
def printMatrix(mat):
```

```
    for i in range(4):
        for j in range(4):
            if(mat[i][j] == 16):
                print("\033[91m{}\033[00m".format(str(mat[i][j])), end = "  ")
            elif(mat[i][j] < 10):
                print(mat[i][j], end = "  ")
            else:
                print(mat[i][j], end = " ")


        print()


def isValid(x, y):
    return x >= 0 and x < 4 and y >= 0 and y < 4

#empty slot baru
def new_empty(empty_tile,dir):
    new = [-1,-1]
    if (dir == "left"):
        new[0] = empty_tile[0]
        new[1] = empty_tile[1]-1
    elif(dir == "right"):
        new[0] = empty_tile[0]
        new[1] = empty_tile[1]+1
    elif(dir == "up"):
        new[0] = empty_tile[0]-1
        new[1] = empty_tile[1]
    elif(dir == "down"):
        new[0] = empty_tile[0]+1
        new[1] = empty_tile[1]
    return new

#memeriksa apakah puzzle = final
def isEqual(puzzle,final):
    for i in range(4):
        for j in range(4):
            if(puzzle[i][j] != final[i][j]):
                return False
    return True

def solve(initial, empty_tile_pos, final):
    count = 0

    pq = priorityQueue()
```

```
    cost = calculateCost(initial, final,0)
    root = node(None, initial,
                empty_tile_pos, cost, 0, NULL)
    pq.push(root)
    while not pq.isEmpty():

        minimum = pq.pop()
        #memeriksa apakah sudah sesuai dengan final
        if isEqual(minimum.puz,final):
            printTree(minimum)
            print("Banyak anak dibangkitkan:" , count)
            return

        #menentukan arah direction baru agar tidak looping
        if minimum.direction == NULL:
            directions = ["left","right","up","down"]
        elif minimum.direction == "left":
            directions = ["left","up","down"]
        elif minimum.direction == "right":
            directions = ["right","up","down"]
        elif minimum.direction == "up":
            directions = ["left","right","up"]
        elif minimum.direction == "down":
            directions = ["left","right","down"]

        for i in range(len(directions)):
            new_tile_pos = new_empty(minimum.empty_tile_pos,directions[i])

            if isValid(new_tile_pos[0], new_tile_pos[1]):

                #buat node baru
                new_node = newNode(minimum.puz,
                                   minimum.empty_tile_pos,
                                   new_tile_pos,
                                   minimum.level + 1,
                                   minimum, final, directions[i])
                #masukkan node baru ke prioqueue
                pq.push(new_node)
                count += 1
```

3. main.py

```
import sys
```

```
#change with ur path
sys.path.insert(0, 'C:/Users/ASUS/15-Puzzle-Solver/src')
import os
import time
import preprogress
import Solver

final = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]

os.chdir("./TestCase")
file = input("Masukkan nama file: ")

f = open(file,"r")


puzzle = []
for item in f:
    arr = (item.strip("\n").split(" ")[:4])
    arr2 = []
    for i in arr:
        idx = int(i)
        arr2.append(idx)
    puzzle.append(arr2)
f.close()

#print kurang(i)
preprogress.printKurang(puzzle)

#print sigmakurang
sigmakurang = preprogress.isReachableGoal(puzzle)
print("SigmaKurang(i):",sigmakurang)

#is reachable goal?
if(sigmakurang%2 == 1):
    print("puzzle tidak solveable")
else:
    start = time.time()
    Solver.solve(puzzle,preprogress.emptyTile(puzzle),final)
    end = time.time()
    print("Execution time: ",end-start)
```

D. Berkas Text Puzzle

1. Puzzle1.txt

   2 3 4 10
   1 7 9 6
   11 13 15 8

5 16 14 12

2. Puzzle2.txt
   1 3 4 15
   2 16 5 12
   7 6 11 14
   8 9 10 13

3. Puzzle3.txt
   6 2 4 8
   3 10 9 7
   1 15 14 16
   13 5 12 11

4. Puzzle4.txt
   5 7 2 4
   6 10 8 11
   1 15 12 14
   13 2 16 9

5. Puzzle5.txt
   6 10 2 4
   1 9 3 8
   5 14 7 11
   13 16 15 12

E. Checklist

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

Github : https://github.com/VionieNovencia/15-Puzzle-Solver.git