LAPORAN TUGAS KECIL 3

IF2211 STRATEGI ALGORITMA

Implementasi 15 Puzzle Solver Menggunakan Algoritma Branch & Bound

Disusun oleh:

Ilham Bintang Nurmansyah 13520102

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

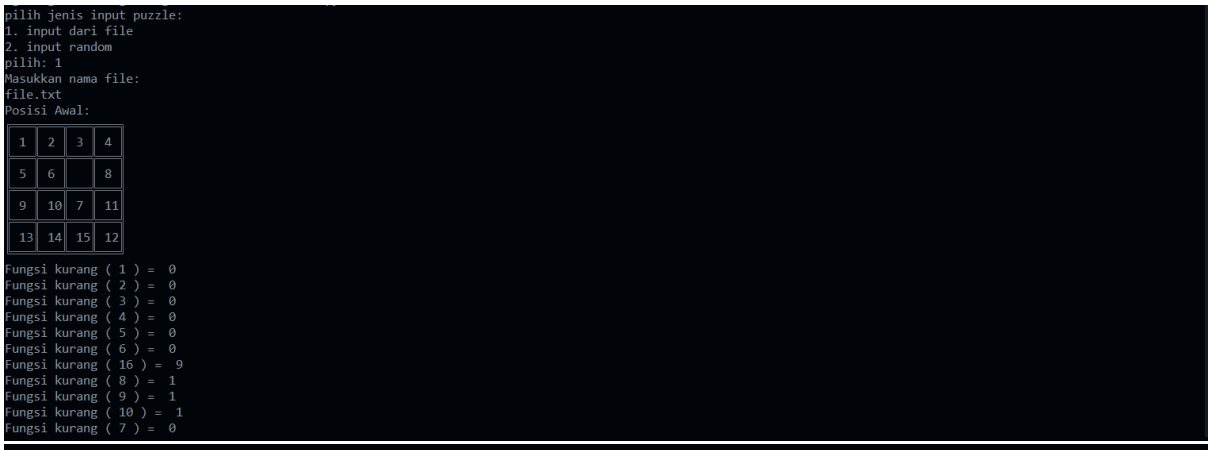
Institut Teknologi Bandung

2022

1. Penjelasan Algoritma *Branch and Bound*

Algoritma yang saya lakukan adalah pertama mencari dulu fungsi Kurang(i), apabila fungsi Kurang(i) + posisi ubin kosong = ganjil, maka puzzle *unsolvable*, sedangkan bila genap mana puzzle *solvable*. Bila solvable. Maka maka akan dibuat sebuah *queue*, dan menginisiasi nilai kedalaman pohon. Lalu memasukkan simpul akar kedalam queue. Melakukan iterasi dari semua kemungkinan gerakan (atas, bawah, kiri, kanan) dari simpul dan hasil gerakannya akan dimasukkan kedalam queue bila pergerakan yang sekarang berbeda dengan pergerakkan sebelumnya. Lalu mengambil simpul *queue* yang memiliki cost paling kecil, dan bila simpul tersebut merupakan jawaban akhir maka akan stop, jika tidak, iterasikan lagi semua kemungkinan pergerakkan.

2. Screen-shot Input-output Program



```
pilih jenis input puzzle:
1. input dari file
2. input random
pilih: 1
Masukkan nama file:
file2.txt
Posisi Awal:
```

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

```
Fungsi kurang ( 1 ) =  0
Fungsi kurang ( 6 ) =  4
Fungsi kurang ( 2 ) =  0
Fungsi kurang ( 3 ) =  0
Fungsi kurang ( 5 ) =  1
Fungsi kurang ( 7 ) =  1
Fungsi kurang ( 4 ) =  0
Fungsi kurang ( 16 ) =  8
Fungsi kurang ( 9 ) =  1
Fungsi kurang ( 10 ) =  1
Fungsi kurang ( 11 ) =  1
Fungsi kurang ( 8 ) =  0
Fungsi kurang ( 13 ) =  1
Fungsi kurang ( 14 ) =  1
Fungsi kurang ( 15 ) =  1
Fungsi kurang ( 12 ) =  0
Total Fungsi Kurangi  + posisi ubin =  20
Solvable Puzzle
```

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

Langkah 6 :

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

Langkah 7 :

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

Langkah 8 :

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

```
Waktu yang diperlukan =  0.0010001659393310547 detik
Jumlah simpul yang dihasilkan =  23
```

```
pilih jenis input puzzle:
1. input dari file
2. input random
pilih: 2
```

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

```
Fungsi kurang ( 12 ) =  11
Fungsi kurang ( 9 ) =  8
Fungsi kurang ( 8 ) =  7
Fungsi kurang ( 10 ) =  7
Fungsi kurang ( 7 ) =  6
Fungsi kurang ( 14 ) =  8
Fungsi kurang ( 16 ) =  9
Fungsi kurang ( 15 ) =  8
Fungsi kurang ( 3 ) =  2
Fungsi kurang ( 5 ) =  3
Fungsi kurang ( 13 ) =  5
Fungsi kurang ( 6 ) =  3
Fungsi kurang ( 11 ) =  3
Fungsi kurang ( 4 ) =  2
Fungsi kurang ( 1 ) =  0
Fungsi kurang ( 2 ) =  0
Total Fungsi Kurangi  + posisi ubin =  83
Unsolvable Puzzle
```

```
pilih jenis input puzzle:
1. input dari file
2. input random
pilih: 1
Masukkan nama file:
file2.txt
Posisi Awal:
```

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

```
Fungsi kurang ( 1 ) =  0
Fungsi kurang ( 2 ) =  0
Fungsi kurang ( 3 ) =  0
Fungsi kurang ( 4 ) =  0
Fungsi kurang ( 6 ) =  1
Fungsi kurang ( 10 ) =  4
Fungsi kurang ( 7 ) =  1
Fungsi kurang ( 8 ) =  1
Fungsi kurang ( 5 ) =  0
Fungsi kurang ( 13 ) =  3
Fungsi kurang ( 15 ) =  4
Fungsi kurang ( 11 ) =  1
Fungsi kurang ( 9 ) =  0
Fungsi kurang ( 16 ) =  2
Fungsi kurang ( 14 ) =  1
Fungsi kurang ( 12 ) =  0
Total Fungsi Kurangi  + posisi ubin =  18
Solvable Puzzle
```

Langkah 8 :

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

Langkah 9 :

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

Langkah 10 :

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

```
Waktu yang diperlukan =  0.0019986629486083984 detik
Jumlah simpul yang dihasilkan =  45
PS D:\KULIAH\SEMESTER 4\Strategi Algoritma\Tugas\Tugas Kecil\Tucil 3>
```

3. Kode Program (Python)

```python
def generatePuzzle():
    listRandom = random.sample(range(1, 17), 16)
    puzzle = listToMatrix(listRandom)
    return listRandom, puzzle


def readfile(fileName):
    array = []
    with open(fileName, 'r') as file:
        for line in file:
            line.split(" ")
            strings = parseString(line.split())
            array += strings
    return array


def parseString(strings):
    for i in range(len(strings)):
        strings[i] = int(strings[i])
    return strings


def listToMatrix(array):
    puzzle = [[0 for a in range(4)] for b in range(4)]
    for i in range(4):
        for j in range(4):
            puzzle[i][j] = array[i*4+j]
    return puzzle

# mencari nilai kurang(i) dari satu elemen
def totalKurang(list, a, i, j):
    sum = 0
    for b in range((i*4+j), len(list)):
        if(a > list[b] and list[b] != 0):
            sum += 1
        else:
            sum += 0
    return sum
```

```python
import utility

print("pilih jenis input puzzle:")
print("1. input dari file")
print("2. input random")
pilihan = int(input("pilih: "))

if pilihan == 1:
    print("Masukkan nama file: ")
    array = utility.readfile(input())
    puzzle = utility.listToMatrix(array)
    print("Posisi Awal:")
    utility.printPuzzle(puzzle)
    utility.branchBound(puzzle, array)
else:
    array, puzzle = utility.generatePuzzle()
    utility.print_matrix(puzzle)
    utility.branchBound(puzzle, array)
```

```python
You, 5 minutes ago | 1 author (You)
import random
import time
target = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
moveset = ("right", "down", "left", "up")


You, 2 days ago | 1 author (You)
class PriorityQueue(object):
    def __init__(self):
        self.queue = []

    def __str__(self):
        return '\n'.join([str(i) for i in self.queue])

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        index = 0
        for i in range(len(self.queue)):
            if(self.queue[i][0] < self.queue[index][0]):
                index = i
        item = self.queue[index]
        del self.queue[index]
        return item


You, 2 days ago | 1 author (You)
class Node:
    def __init__(self, data=None):
        self.matrix = data
        self.parent = None
        self.depth = 0
```

```python
# mencari posisi ubin kosong pada puzzle
def posisiKosong(puzzle):
    for i in range(4):
        for j in range(4):
            if(puzzle[i][j] == 16):
                if((i+j) % 2 == 0):
                    return 0, i, j
                else:
                    return 1, i, j

# nilai formula kurang(i) + posisi ubin kosong
def kurangi(puzzle, array):
    sum = 0
    for i in range(4):
        for j in range(4):
            sum += totalKurang(array, puzzle[i][j], i, j)
    posisikosong, _, _ = posisiKosong(puzzle)
    return (sum + posisikosong)

# mencari nilai cost dari bentuk puzzle terkini
def cost(depth, matrix):
    sum = 0       You, now • Uncommitted changes
    for i in range(4):
        for j in range(4):
            if((matrix[i][j] != target[i][j]) and matrix[i][j] !=16):
                sum += 1       You, now • Uncommitted changes
    return (depth+sum)       You, now • Uncommitted changes
```

```python
def moveTiles(puzzle, movement):
    matrixCopy = copyMatrix(puzzle)
    _, x, y = posisiKosong(matrixCopy)
    if(movement == "left" and y != 0):
            y -= 1
    elif(movement == "right" and y != 3):
            y += 1
    elif(movement == "up" and x != 0):
            x -= 1
    elif(movement == "down" and x != 3):
            x += 1
    matrixCopy = swap(matrixCopy, x, y)
    return matrixCopy


def swap(matrix, row, column):
    _, x, y = posisiKosong(matrix)
    matrix[x][y] = matrix[row][column]
    matrix[row][column] = 16
    return matrix


def lawanMove(prevMove):
    if(prevMove == "up"):
        return "down"
    elif(prevMove == "down"):
        return "up"
    elif(prevMove == "left"):
        return "right"
    elif(prevMove == "right"):
        return "left"
```

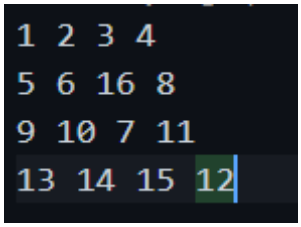
```python
def solved(matrix):
    for i in range(4):
        for j in range(4):
            if(matrix[i][j] != target[i][j]):
                return False
    return True


def printPuzzle(matrix):
    print("╔═══╦═══╦═══╦═══╗")
    for i in range(4):
        for j in range(4):
            print("║ ", end="")
            print(matrix[i][j], end="")
            if(matrix[i][j] < 10):
                print(" ", end="")
        print("║")
        if(i != 3):
            print("╠═══╬═══╬═══╬═══╣")
    print("╚═══╩═══╩═══╩═══╝")


def solved(matrix, target):
    for i in range(4):
        for j in range(4):
            if(matrix[i][j] != target[i][j]):
                return False
    return True


def copyMatrix(matrix):
    newMatrix = [[0 for a in range(4)] for b in range(4)]
    for i in range(4):
        for j in range(4):
            newMatrix[i][j] = matrix[i][j]
    return newMatrix
```

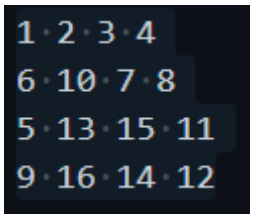
```python
def printStep(node):
    if(node.parent == None):
        return
    printStep(node.parent)
    print("\n")
    print("Langkah "+str(node.depth)+" : ")
    printPuzzle(node.matrix)

def printKurangi(puzzle, array):
    for i in range(4):
        for j in range(4):
            print("Fungsi kurang (" ,puzzle[i][j], ") = ", totalKurang(array, puzzle[i][j], i, j))
    print("Total Fungsi Kurangi = ", kurangi(puzzle, array))
```

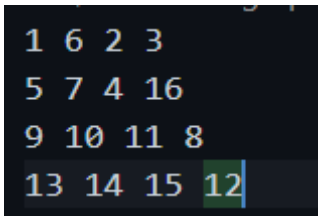
```python
def branchBound(puzzle, array):
    if(kurangi(puzzle, array) % 2 == 0):
        timeStart = time.time()
        printKurangi(puzzle, array)
        print("Solvable Puzzle\n")
        Queue = PriorityQueue()
        generatedNode = 0

        simpul = Node(puzzle)
        Queue.enqueue((cost(0, simpul.matrix), simpul, "", 0))

        tempQueue = Queue.dequeue()
        simpul = tempQueue[1]
        matrixMove = simpul.matrix
        prevMove = ""
        depth = tempQueue[3] + 1
        generatedNode += 1      You, 13 hours ago • fix bug, 99% …

        while(not solved(matrixMove, target)):
            for mov in moveset:
                if(mov != prevMove):
                    after_move = moveTiles(matrixMove, mov)
                    if(not solved(after_move, matrixMove)):
                        newNode = Node(after_move)
                        newNode.parent = simpul
                        newNode.depth = simpul.depth + 1
                        generatedNode += 1
                        Queue.enqueue(
                            (cost(depth, newNode.matrix), newNode, mov, depth))

            tempQueue = Queue.dequeue()
            simpul = tempQueue[1]
            matrixMove = simpul.matrix
            moveNew = tempQueue[2]
            prevMove = lawanMove(moveNew)
            depth = tempQueue[3] + 1
        timeEnd = time.time()
        timeEstimated = timeEnd - timeStart
        printStep(simpul)
        print("Waktu yang diperlukan = ", timeEstimated, "detik")
        print("Jumlah simpul yang dihasilkan = ", generatedNode)

    else:
        printKurangi(puzzle, array)
        print("Unsolvable Puzzle\n")
```

4. Contoh Instansiasi 5 Buah Persoalan

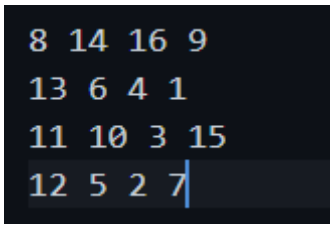
Solvable:

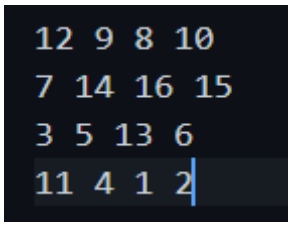






Unsolvable





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		✓

Link Github: https://github.com/Hambinn/Tucil-3-Stima