LAPORAN TUGAS KECIL
IF2211 STRATEGI ALGORITMA

Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*

Disusun oleh :

Vieri Mansyl 13520092

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

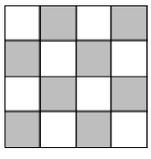
DAFTAR ISI

I. Cara Kerja Algoritma Branch and Bound dalam penyelesaian teka-teki *15-puzzle*

Algoritma Branch and Bound (BnB) merupakan algoritma yang pada umumnya digunakan untuk menyelesaikan permasalahan optimasi tanpa melanggar batasan (*constraint*) dari persoalan tersebut. Algoritma BnB merupakan gabungan dari algoritma *Breadth First Search* (BFS) dengan algoritma *Least Cost Search* (LCS), yang mana menunjukkan algoritma BnB mempertimbangkan beban (*cost*) dari setiap simpul pada pohon ruang status. Dengan begitu, simpul yang tidak memenuhi *constraint* melalui fungsi pembatas akan di-'potong' , yang mana menunjukkan simpul tersebut tidak 'mengarah' ke solusi.

Implementasi algoritma BnB pada program pemecahan persoalan teka-teki 15-puzzle diawali dengan pencatatan simpul akar (*root*) pada antrian berprioritas (Priority Queue; sebut saja *pq*). Selanjutnya, dilakukan *dequeue* pada *pq* agar mendapatkan nilai elemen pertama yang paling diprioritaskan. Apabila elemen pertama tersebut merupakan simpul solusi, maka solusi telah ditemukan. Apabila tidak, akan dilakukan pembangkitan anak simpul dari simpul elemen pertama tersebut. Simpul yang akan dibangkitkan wajib memenuhi 2 aturan, yaitu :

1. nilai $\sum_{i=0}^{16} Kurang(i) + X$ bernilai genap
   - *Kurang(i)* : banyaknya ubin bernomor j sedemikian sehingga j < i dengan posisi j > posisi i.
   - X : ubin kosong pada posisi ubin yg diarsir seperti pada gambar di bawah.



2. Tidak melakukan *illegal move*, dimana ubin kosong 'berpindah' balik pada posisi semula.

Setelah memenuhi kedua aturan di atas, setiap simpul akan dicatat nilai taksiran cost-nya, dinotasikan sebagai berikut.

$$c(p) = f(p) + g(p)$$

dimana

c(p) : taksiran cost dari simpul p

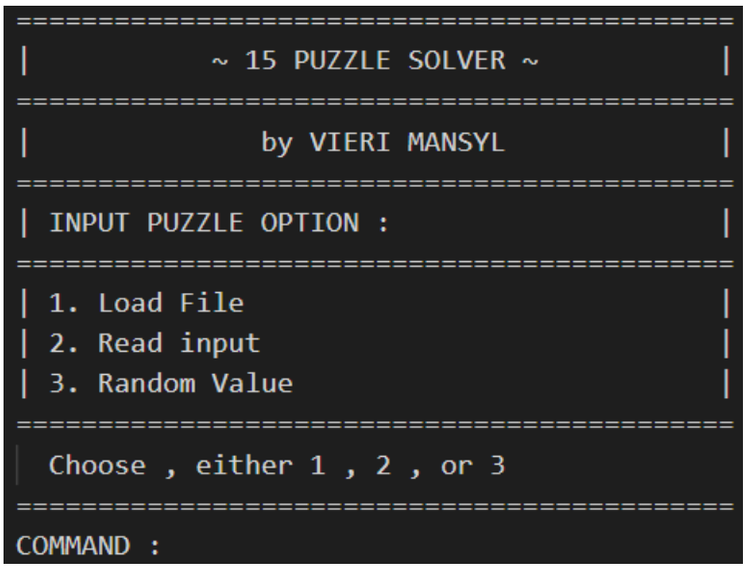
f(p) : panjang lintasan dari simpul akar ke simpul p

g(p) : taksiran panjang lintasan terpendek dari simpul p ke simpul solusi , atau dapat direpresentasikan sebagai jumlah ubin tidak kosong yang tidak sesuai dengan susunan akhir terhadap simpul solusi.

Setiap anak simpul yang telah dihitung cost-nya akan dicatat pada *pq* berdasarkan nilai cost-nya. Selanjutnya, dilakukan langkah yang sama seperti sebelumnya, yaitu dilakukan pengecekan pada elemen pertama dari *pq* sampai ditemukan simpul yang merupakan simpul solusi.

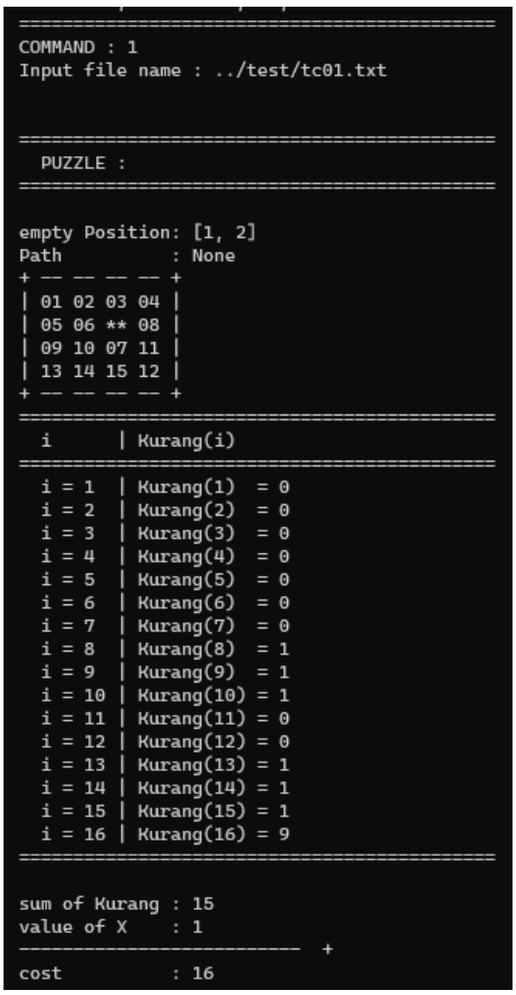
II. Hasil Input-Output Program

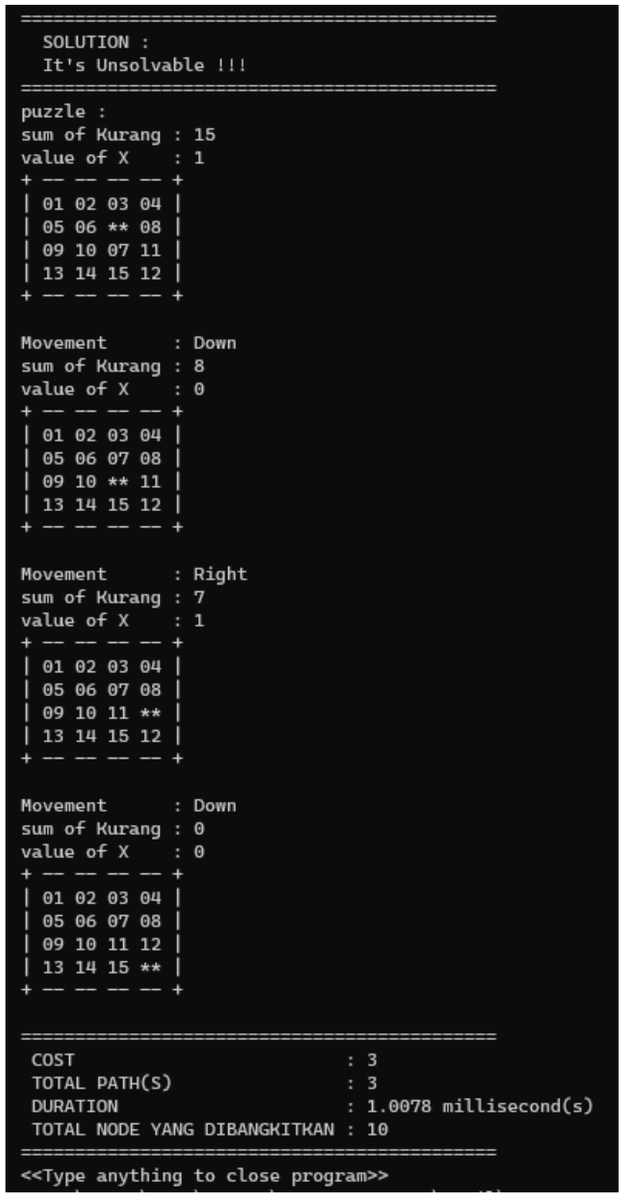
Berikut merupakan tampilan awal dari program.



Terdapat tiga cara pembacaan matriks 15-puzzle yang akan diselesaikan, yaitu melalui pembacaan Config file yang telah ada, pembacaan melalui masukan dari user, serta matriks yang disusun secara acak penyusunan ubinnya. Berikut *test case* yang diuji pada program.

1. tc01.txt – Solvable



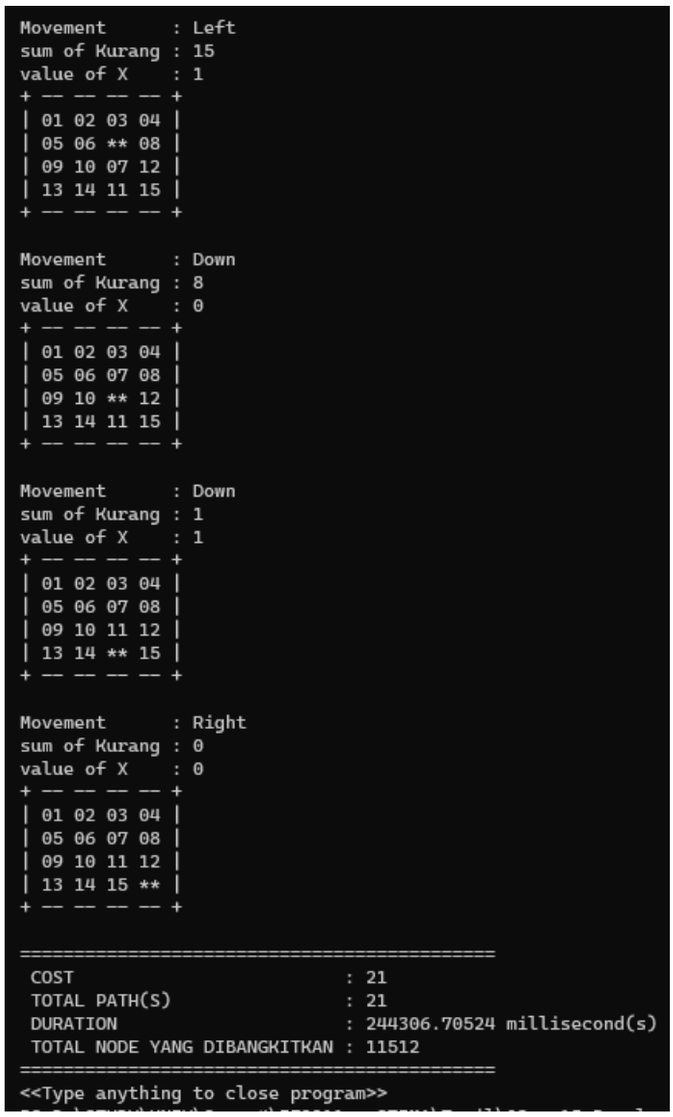

==========
SOLUTION :
It's Unsolvable !!!
==========
puzzle :
sum of Kurang : 15
value of X : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 ** 08 |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +
Movement : Down
sum of Kurang : 8
value of X : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 ** 11 |
| 13 14 15 12 |
+ -- -- -- -- +
Movement : Right
sum of Kurang : 7
value of X : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 11 ** |
| 13 14 15 12 |
+ -- -- -- -- +
Movement : Down
sum of Kurang : 0
value of X : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 11 12 |
| 13 14 15 ** |
+ -- -- -- -- +
==========
COST : 3
TOTAL PATH(S) : 3
DURATION : 1.0078 millisecond(s)
TOTAL NODE YANG DIBANGKITKAN : 10
==========
<<Type anything to close program>>

2. tc02.txt – Solvable

```
COMMAND : 1
Input file name : ../test/tc02.txt


==========================================
  PUZZLE :
==========================================

empty Position: [2, 1]
Path          : None
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| 06 ** 07 15 |
| 09 13 14 11 |
+ -- -- -- -- +
==========================================
  i      | Kurang(i)
==========================================
  i = 1  | Kurang(1)  = 0
  i = 2  | Kurang(2)  = 1
  i = 3  | Kurang(3)  = 1
  i = 4  | Kurang(4)  = 1
  i = 5  | Kurang(5)  = 3
  i = 6  | Kurang(6)  = 0
  i = 7  | Kurang(7)  = 0
  i = 8  | Kurang(8)  = 2
  i = 9  | Kurang(9)  = 0
  i = 10 | Kurang(10) = 3
  i = 11 | Kurang(11) = 0
  i = 12 | Kurang(12) = 4
  i = 13 | Kurang(13) = 1
  i = 14 | Kurang(14) = 1
  i = 15 | Kurang(15) = 4
  i = 16 | Kurang(16) = 6
==========================================

sum of Kurang : 27
value of X    : 1
--------------------------  +
cost          : 28
==========================================
  SOLUTION :
  It's Unsolvable !!!
==========================================
puzzle :
sum of Kurang : 27
value of X    : 1
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| 06 ** 07 15 |
| 09 13 14 11 |
+ -- -- -- -- +

Movement      : Left
sum of Kurang : 28
value of X    : 0
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| ** 06 07 15 |
| 09 13 14 11 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 25
value of X    : 1
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| 09 06 07 15 |
| ** 13 14 11 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 24
value of X    : 0
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| 09 06 07 15 |
| 13 ** 14 11 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 23
value of X    : 1
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| 09 06 07 15 |
| 13 14 ** 11 |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 30
value of X    : 0
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 10 12 |
| 09 06 ** 15 |
| 13 14 07 11 |
+ -- -- -- -- +
```

```
Movement      : Up
sum of Kurang : 33
value of X    : 1
+ -- -- -- -- +
| 02 05 03 04 |
| 01 08 ** 12 |
| 09 06 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Left
sum of Kurang : 34
value of X    : 0
+ -- -- -- -- +
| 02 05 03 04 |
| 01 ** 08 12 |
| 09 06 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 35
value of X    : 1
+ -- -- -- -- +
| 02 ** 03 04 |
| 01 05 08 12 |
| 09 06 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Left
sum of Kurang : 36
value of X    : 0
+ -- -- -- -- +
| ** 02 03 04 |
| 01 05 08 12 |
| 09 06 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 29
value of X    : 1
+ -- -- -- -- +
| 01 02 03 04 |
| ** 05 08 12 |
| 09 06 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 28
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 ** 08 12 |
| 09 06 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +
```

```
Movement      : Down
sum of Kurang : 21
value of X    : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 08 12 |
| 09 ** 10 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 20
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 08 12 |
| 09 10 ** 15 |
| 13 14 07 11 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 13
value of X    : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 08 12 |
| 09 10 07 15 |
| 13 14 ** 11 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 12
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 08 12 |
| 09 10 07 15 |
| 13 14 11 ** |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 13
value of X    : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 08 12 |
| 09 10 07 ** |
| 13 14 11 15 |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 14
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 08 ** |
| 09 10 07 12 |
| 13 14 11 15 |
+ -- -- -- -- +
```


Movement : Left
sum of Kurang : 15
value of X : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 ** 08 |
| 09 10 07 12 |
| 13 14 11 15 |
+ -- -- -- -- +
Movement : Down
sum of Kurang : 8
value of X : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 ** 12 |
| 13 14 11 15 |
+ -- -- -- -- +
Movement : Down
sum of Kurang : 1
value of X : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 11 12 |
| 13 14 ** 15 |
+ -- -- -- -- +
Movement : Right
sum of Kurang : 0
value of X : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 11 12 |
| 13 14 15 ** |
+ -- -- -- -- +
==========
COST : 21
TOTAL PATH(S) : 21
DURATION : 244306.70524 millisecond(s)
TOTAL NODE YANG DIBANGKITKAN : 11512
==========
<<Type anything to close program>>

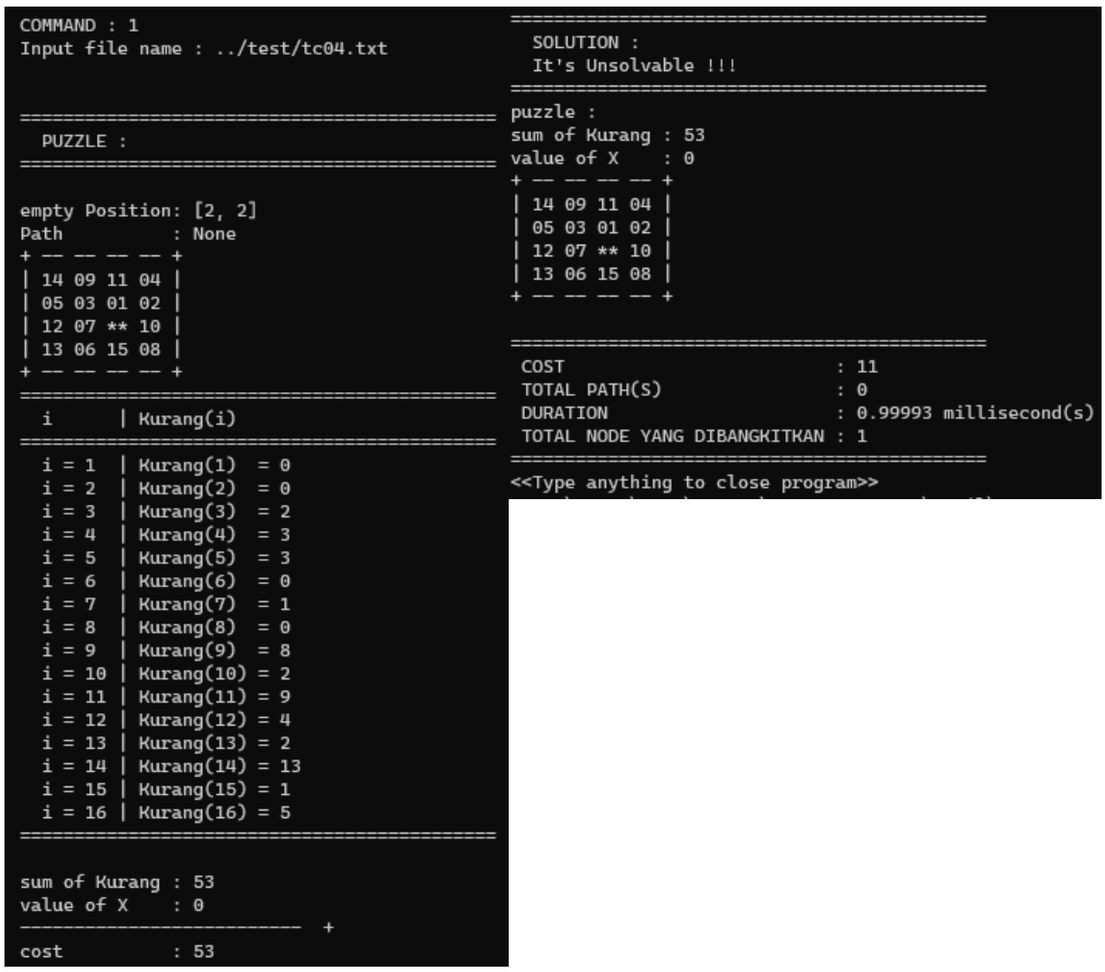
3. tc03.txt – Solvable

```
COMMAND : 1
Input file name : ../test/tc03.txt


==========================================
  PUZZLE :
==========================================

empty Position: [3, 0]
Path          : None
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 13 09 07 12 |
| ** 10 14 15 |
+ -- -- -- -- +
==========================================
  i      | Kurang(i)
==========================================
  i = 1  | Kurang(1)  = 0
  i = 2  | Kurang(2)  = 0
  i = 3  | Kurang(3)  = 1
  i = 4  | Kurang(4)  = 1
  i = 5  | Kurang(5)  = 1
  i = 6  | Kurang(6)  = 3
  i = 7  | Kurang(7)  = 0
  i = 8  | Kurang(8)  = 1
  i = 9  | Kurang(9)  = 1
  i = 10 | Kurang(10) = 0
  i = 11 | Kurang(11) = 3
  i = 12 | Kurang(12) = 1
  i = 13 | Kurang(13) = 4
  i = 14 | Kurang(14) = 0
  i = 15 | Kurang(15) = 0
  i = 16 | Kurang(16) = 3
==========================================

sum of Kurang : 19
value of X    : 1
---------------------------  +
cost          : 20
```

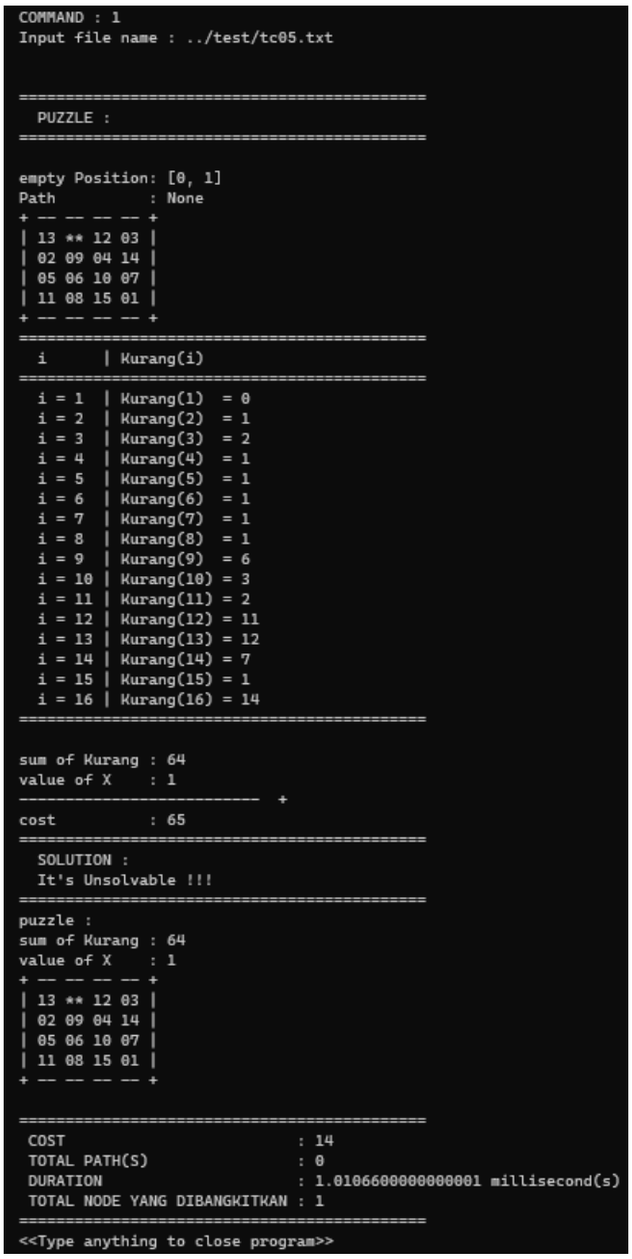
```
==========================================
  SOLUTION :
  It's Unsolvable !!!
==========================================
puzzle :
sum of Kurang : 19
value of X    : 1
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 13 09 07 12 |
| ** 10 14 15 |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 20
value of X    : 0
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| ** 09 07 12 |
| 13 10 14 15 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 19
value of X    : 1
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 09 ** 07 12 |
| 13 10 14 15 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 14
value of X    : 0
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 09 10 07 12 |
| 13 ** 14 15 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 13
value of X    : 1
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 09 10 07 12 |
| 13 14 ** 15 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 12
value of X    : 0
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 09 10 07 12 |
| 13 14 15 ** |
+ -- -- -- -- +
```

```
Movement      : Up
sum of Kurang : 19
value of X    : 1
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 11 |
| 09 10 07 ** |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 20
value of X    : 0
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 08 ** |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Left
sum of Kurang : 21
value of X    : 1
+ -- -- -- -- +
| 01 03 06 04 |
| 05 02 ** 08 |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Up
sum of Kurang : 22
value of X    : 0
+ -- -- -- -- +
| 01 03 ** 04 |
| 05 02 06 08 |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Left
sum of Kurang : 23
value of X    : 1
+ -- -- -- -- +
| 01 ** 03 04 |
| 05 02 06 08 |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 16
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 ** 06 08 |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 15
value of X    : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 ** 08 |
| 09 10 07 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 8
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 ** 11 |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Right
sum of Kurang : 7
value of X    : 1
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 11 ** |
| 13 14 15 12 |
+ -- -- -- -- +

Movement      : Down
sum of Kurang : 0
value of X    : 0
+ -- -- -- -- +
| 01 02 03 04 |
| 05 06 07 08 |
| 09 10 11 12 |
| 13 14 15 ** |
+ -- -- -- -- +

============================================
 COST                          : 15
 TOTAL PATH(S)                 : 15
 DURATION                      : 209.09119 millisecond(s)
 TOTAL NODE YANG DIBANGKITKAN : 315
============================================
<<Type anything to close program>>
```

4. tc04.txt – Unsolvable



5. tc05.txt – Unsolvable



```
COMMAND : 1
Input file name : ../test/tc05.txt


==========================================
  PUZZLE :
==========================================

empty Position: [0, 1]
Path          : None
+ -- -- -- -- +
| 13 ** 12 03 |
| 02 09 04 14 |
| 05 06 10 07 |
| 11 08 15 01 |
+ -- -- -- -- +
==========================================
  i      | Kurang(i)
==========================================
  i = 1  | Kurang(1)  = 0
  i = 2  | Kurang(2)  = 1
  i = 3  | Kurang(3)  = 2
  i = 4  | Kurang(4)  = 1
  i = 5  | Kurang(5)  = 1
  i = 6  | Kurang(6)  = 1
  i = 7  | Kurang(7)  = 1
  i = 8  | Kurang(8)  = 1
  i = 9  | Kurang(9)  = 6
  i = 10 | Kurang(10) = 3
  i = 11 | Kurang(11) = 2
  i = 12 | Kurang(12) = 11
  i = 13 | Kurang(13) = 12
  i = 14 | Kurang(14) = 7
  i = 15 | Kurang(15) = 1
  i = 16 | Kurang(16) = 14
==========================================

sum of Kurang : 64
value of X    : 1
-------------------------  +
cost          : 65
==========================================
  SOLUTION :
  It's Unsolvable !!!
==========================================
puzzle :
sum of Kurang : 64
value of X    : 1
+ -- -- -- -- +
| 13 ** 12 03 |
| 02 09 04 14 |
| 05 06 10 07 |
| 11 08 15 01 |
+ -- -- -- -- +

==========================================
 COST                          : 14
 TOTAL PATH(S)                 : 0
 DURATION                      : 1.0106600000000001 millisecond(s)
 TOTAL NODE YANG DIBANGKITKAN : 1
==========================================
<<Type anything to close program>>
```

III. Kode Program

Pemecahan 15-puzzle dengan meggunakan algoritma BnB diimplementasikan ke dalam 3 program, yaitu :

- **Puzzle.py** : berisi definisi kelas **puzzle** yang merupakan representasi dari bentuk matriks dari 15-puzzle.

```python
import copy

class fifteenpuzzle:

    # construct puzzle
    '''
    mat     : matrix of 15-puzzle
    x , y   : 'null' slot position
    path    : path from 'root' matrix up to current matrix
    '''
    def __init__(self , mat , x , y , path):
        self.matrix = copy.deepcopy(mat)
        self.emptyslot = [x,y]
        self.kurang = self.sumOfKurang()
        self.xValue = self.valueOfX()
        self.path = copy.deepcopy(path)

    def __lt__(self, next):
        return self.cost() < next.cost()

    # change to Array
    def toArray(self):
        return [elmt for elmts in self.matrix for elmt in elmts]

    # SOLVABILITY
    # sum of Kurang(i)
    def sumOfKurang(self):
        count = 0
        arr = self.toArray()
        for i in range(16):
            pointer = arr[i]
            for idx in range(i , 16):
                if (pointer > arr[idx]):
                    count += 1
        return count

    # value of X
    def valueOfX(self):
        row , col = self.emptyslot[0] , self.emptyslot[1]
        return 1 if ((row % 2 == 0 and col %2 == 1) or (row % 2 == 1 and col %2 == 0)) else 0

    def solvable(self):
        total = self.sumOfKurang() + self.valueOfX()
        return total % 2 == 0


    # MOVEMENT
    def illegalMove(self , movement):
        lastIdx = len(self.path)-1
        if(lastIdx >= 0):
            diff = abs(self.path[lastIdx] - movement)
            if (diff == 2):
                return True
        else :
            return False
```

```python
def move(self , newX , newY):
    x , y = self.emptyslot[0] , self.emptyslot[1]
    self.emptyslot[0] += newX
    self.emptyslot[1] += newY
    self.matrix[x][y] , self.matrix[self.emptyslot[0]][self.emptyslot[1]] = self.matrix[self.emptyslot[0]][self.emptyslot[1]] , self.matrix[x][y]
    self.kurang = self.sumOfKurang()
    self.xValue = self.valueOfX()

# DISPLAY INFO
def displayMat(self):
    print("+ -- -- -- -- +")
    for i in range(4):
        print("| ", end="")
        for j in range(4):
            if(self.matrix[i][j] == 16) : print("**",end="")
            else :
                if(self.matrix[i][j] < 10):
                    print("0" + str(self.matrix[i][j]),end="")
                else :
                    print(self.matrix[i][j],end="")
            if(j != 3) : print(" ", end= "")
        print(" |")
    print("+ -- -- -- -- +")

# Display all Info
def printInfo(self):
    print(f"empty Position: {self.emptyslot}")
    print(f"Path          : ", end="")
    if(len(self.path) == 0):
        print("None", end="")
    else:
        for i in range (len(self.path)):
            if(self.path[i] == 0):
                print("bottom", end="")
            elif(self.path[i] == 1):
                print("left", end="")
            elif(self.path[i] == 2):
                print("top", end="")
            elif(self.path[i] == 3):
                print("right", end="")

            if(i+1 != len(self.path)):
                print(" -> ",end="")
    print()
    self.displayMat()
    print("=========================================")
    print("  i      | Kurang(i)")
    print("=========================================")
    self.kurangI()
    print("=========================================")
    print()
    print(f"sum of Kurang : {self.kurang}")
    print(f"value of X    : {self.xValue}")
    print("-------------------------  +")
    print(f"cost          : {self.xValue + self.kurang}")
```

```python
# show Kurang(i)
def kurangI(self):
    arr = self.toArray()
    lessValue = [0 for i in range(16)]
    for i in range(16):
        count = 0
        for idx in range(i , 16):
            if (arr[i] > arr[idx]):
                count += 1
        lessValue[arr[i]-1] = count

    for i in range(len(lessValue)):
        if(i < 9):
            print(f"  i = {i+1}  | Kurang({i+1})  = {lessValue[i]}")
        else:
            print(f"  i = {i+1} | Kurang({i+1}) = {lessValue[i]}")

# COUNT COST
# puzzle's goal
goal = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]

# count g(p)
def calculateGoal(self):
    count = 0
    for i in range(4):
        for j in range(4):
            if(self.matrix[i][j] != 16):
                if self.matrix[i][j] != fifteenpuzzle.goal[i][j]:
                    count += 1
    return count

def cost(self):
    return len(self.path) + self.calculateGoal()
```

- **PrioQueue.py** : berisi definisi kelas **priority queue** yang merupakan representasi antrian yang mempertimbangkan nilai *cost* dari masing-masing puzzle yang tercatat sebagai elemen prioritasnya.

```
class prioQueue:
    # construct Priority Queue
    def __init__(self):
        self.heap = []
        self.length = 0
        self.countNode = 0

    # head of Priority Queue
    def head(self):
        return self.heap[0]

    def printCost(self):
        print("path with cost")
        for puzzle in self.heap:
            print(puzzle.cost() , end=" ")
        print()

    # enqueue
    def enqueue(self, val):
        if self.isEmpty():
            self.heap.append(val)
            self.length += 1
        else:
            i = 0
            while i < self.length:
                if(self.heap[i].cost() > val.cost()): break
                else                                : i += 1
            self.heap.insert(i , val)
            self.length += 1
        self.countNode += 1

    # dequeue
    def dequeue(self):
        head = self.heap.pop(0)
        self.length -= 1
        return head

    # if the Queue is empty
    def isEmpty(self):
        return self.length == 0
```

- **PuzzleGame.py**: berisi program utama yang mengimplementasikan algoritma BnB.

```
#15 Puzzle Game menggunakan algoritma Branch and Bound
from random import shuffle
from puzzle import fifteenpuzzle
from prioQueue import prioQueue
import copy
import time

# movement value : bottom, left, top, right
moveRow = [ 1, 0, -1, 0 ]
moveCol = [ 0, -1, 0, 1 ]

# membaca file .txt
# mengembalikan array berisikan nilai dari matriks yang terbaca

def readFile(filename):
    arr = []
    with open(filename,'r') as f:
        lines=f.read().splitlines()
        for line in lines:
            words = line.split(" ")
            for word in words:
                arr.append(word)

    for i in range(len(arr)):
        try:
            arr[i] = int(arr[i])
        except ValueError:
            arr[i] = 16
    return arr


def toMatrix(arr):
    return [arr[i:i+4] for i in range(0,len(arr), 4)]


def find16(mat):
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            if(mat[i][j] == 16):
                return i , j


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


def movement(i):
    print("Movement      : ", end="")
    if(i == 0):
        print("Down")
    elif(i == 1):
        print("Left")
    elif(i == 2):
        print("Up")
    elif(i == 3):
        print("Right")
```

```python
def childNode(parent , direction):
    prev = copy.deepcopy(parent)
    prevRow = prev.emptyslot[0]
    prevCol = prev.emptyslot[1]
    prevPath = prev.path

    # create child node
    row = prevRow + moveRow[direction]
    col = prevCol + moveCol[direction]
    path = prevPath + [direction]
    prev.matrix[prevRow][prevCol] , prev.matrix[row][col] = prev.matrix[row][col] , prev.matrix[prevRow][prevCol]

    child = fifteenpuzzle(prev.matrix , row , col , path)
    return child


def solve(puzzle) -> fifteenpuzzle:            # implement Branch and Bound Algorithm
    pq = prioQueue()
    pq.enqueue(puzzle)
    currentNode = puzzle
    while not pq.isEmpty():                    # there's child node left to be checked
        currentNode = pq.dequeue()
        if currentNode.calculateGoal() == 0:   # reach goal
            return currentNode , pq.countNode
        else:                                  # have not reach goal
            for i in range(4):
                newNullPositon = [currentNode.emptyslot[0] + moveRow[i] , currentNode.emptyslot[1] + moveCol[i]]
                check = currentNode.illegalMove(i)
                if isSafe(newNullPositon[0], newNullPositon[1]) and not check:
                    child = childNode(currentNode , i)
                    if(child.solvable()):      # if child node is solvable -> ∑KURANG(i) + X is even
                        pq.enqueue(child)
    return currentNode , pq.countNode
```

```python
# INTERFACE TERMINAL
def welcome():
    print("=============================================")
    print("|           ~ 15 PUZZLE SOLVER ~            |")
    print("=============================================")
    print("|              by VIERI MANSYL              |")
    print("=============================================")

def choose():
    print("| INPUT PUZZLE OPTION :                     |")
    print("=============================================")
    print("| 1. Load File                              |")
    print("| 2. Read input                             |")
    print("| 3. Random Value                           |")
    print("=============================================")
    print("  Choose , either 1 , 2 , or 3")
    print("=============================================")

def process(puz):
    print("\n")
    print("=============================================")
    print("  PUZZLE :")
    print("=============================================")
    print()
    puz.printInfo()
    print("=============================================")
    print("  SOLUTION :")

def failed():
    print("  It's Unsolvable !!!")
    print("=============================================")

def success():
    print("  We found the path !!!")
    print("=============================================")
```

```
#ALGORITMA UTAMA
# "Opening" Interface
welcome()
choose()

# INPUT
option = input("COMMAND : ")
if(option == "1"):                                          # LOAD FILE
    filepath = input("Input file name : ")
    arr = readFile(filepath)
    mat = toMatrix(arr)
elif(option == "2"):                                        # READ INPUT
    print(" P.S. : to input empty slot, use '16'")
    mat = [[0 for col in range(4)] for row in range(4)]
    for i in range(4):
        for j in range(4):
            puzzleVal = int(input(f"[{i+1}][{j+1}] value : "))
            mat[i][j] = puzzleVal
else :                                                      # RANDOM VALUE
    arr = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
    shuffle(arr)
    mat = toMatrix(arr)
```

```
# CREATE PUZZLE
i , j = find16(mat)
puzzle = fifteenpuzzle(mat, i , j , [])

#"Process" Interface
process(puzzle)

# SOLVE
start = time.time()
answer , nodes= solve(puzzle)
end = time.time()

answerPath = copy.deepcopy(answer.path)
if(answer.cost() > 0):          # FAILED
    failed()
else:                           # SUCCESS
    success()
```

```
# OUTPUT
# first puzzle
print("puzzle :")
print(f"sum of Kurang : {puzzle.kurang}")
print(f"value of X    : {puzzle.xValue}")
puzzle.displayMat()
print()

# each movement's path
for path in answerPath:
    movement(path)
    puzzle.move(moveRow[path] , moveCol[path])
    print(f"sum of Kurang : {puzzle.kurang}")
    print(f"value of X    : {puzzle.xValue}")
    puzzle.displayMat()
    print()

print("=========================================")
print(f" COST                          : {answer.cost()}")
print(f" TOTAL PATH(S)                 : {len(answerPath)}")
print(f" DURATION                      : {round(end - start , 8) * 1000} millisecond(s)")
print(f" TOTAL NODE YANG DIBANGKITKAN : {nodes}")
print("=========================================")
close = input("<<Type anything to close program>>")
```

IV. Penilaian

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		

V. Tautan Terkait

- Repository berisikan kode program.

https://github.com/VieriMansyl/Tucil3_13520092