

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

Disusun oleh
Grace Claudia - 13520078



Teknik Informatika

Institut Teknologi Bandung

Bandung

2022

DAFTAR ISI

BAB 1: DESKRIPSI PERSOALAN	3
BAB 2: ALGORITMA BRANCH AND BOUND	5
BAB 3: SOURCE PROGRAM	7
BAB 4: SCREENSHOT INPUT OUTPUT	17
LAMPIRAN	30

BAB I

Deskripsi Persoalan

Buatlah program dalam Java/Python untuk menyelesaikan persoalan 15-Puzzle dengan menggunakan Algoritma Branch and Bound seperti pada materi kuliah. Nilai bound tiap simpul adalah penjumlahan cost yang diperlukan untuk sampai suatu simpul x dari akar, dengan taksiran cost simpul x untuk sampai ke goal. Taksiran cost yang digunakan adalah jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir (goal state). Untuk semua instansiasi persoalan 15-puzzle, susunan akhir yang diinginkan sesuai dengan Gambar 1.

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

Gambar 1. Susunan Akhir persoalan 15-puzzle

Masukan: matriks yang merepresentasikan posisi awal suatu instansiasi persoalan 15-puzzle. Posisi awal 15-puzzle dibangkitkan secara acak oleh program dan/atau dimasukkan dari file teks.

Program harus dapat menentukan apakah posisi awal suatu masukan dapat diselesaikan hingga mencapai susunan akhir, dengan mengimplementasikan fungsi Kurang(i) dan posisi ubin kosong di kondisi awal (X), seperti pada materi kuliah. Jika posisi awal tidak bisa mencapai susunan akhir, program akan menampilkan pesan tidak bisa diselesaikan,. Jika dapat diselesaikan, program dapat menampilkan urutan matriks rute (path) aksi yang dilakukan dari posisi awal ke susunan akhir. Sebagai contoh pada Gambar 2, matriks yang ditampilkan ke layar adalah matriks pada simpul 1, simpul 4, simpul 10 dan simpul 23.

Luaran:

1. Matriks posisi awal 15-puzzle.
2. Nilai dari fungsi Kurang (i) untuk setiap ubin tidak kosong pada posisi awal (nilai ini tetap dikeluarkan, baik persoalan bisa diselesaikan atau tidak bisa diselesaikan).

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

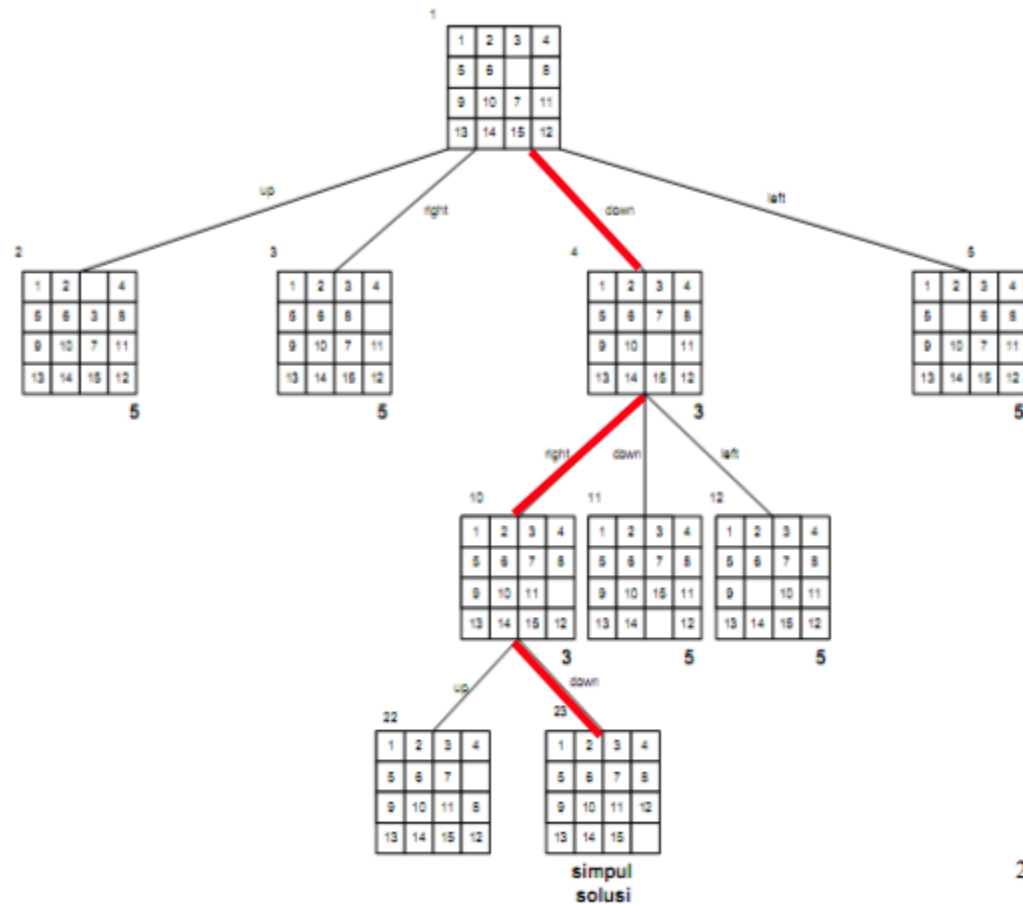
3. Nilai dari

4. Jika persoalan tidak dapat diselesaikan (berdasarkan hasil butir 2) keluar pesan.

5. Jika persoalan dapat diselesaikan (berdasarkan hasil butir 2), menampilkan urutan matriks dari posisi awal ke posisi akhir seperti pada penjelasan sebelumnya.

6. Waktu eksekusi program (diluar baca input/tulis output)

7. Jumlah simpul yang dibangkitkan di dalam pohon ruang status pencarian.



20

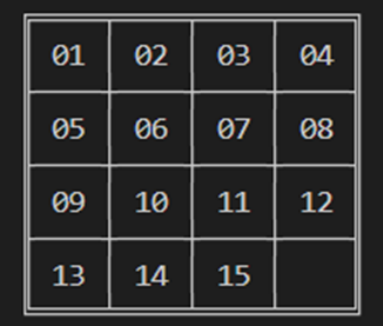
Gambar 2. Contoh Pohon Ruang Status Persoalan 15-puzzle

BAB II

Algoritma *Branch and Bound*

I. Algoritma Branch and Bound

Persoalan 15 Puzzle merupakan teka-teki populer berupa grid berukuran 4x4 yang berisikan angka 1-15 dengan 1 bagian kosong yang dapat dipindah dalam grid sehingga menghasilkan susunan solusi. Sebuah susunan dikatakan sebuah solusi apabila isinya berurutan 1-15 mengikuti urutan sebagai berikut



01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Gambar 1. Jawaban Persoalan 15 Puzzle

(sumber: dokumen pribadi penulis)

Persoalan ini dapat diselesaikan salah satunya dengan mengimplementasikan algoritma Branch and Bound. Secara singkat, cara kerja algoritma ini adalah memetakan persoalan menjadi pohon ruang status dengan menyimpan cost setiap simpul yang ada. Pada kasus 15 Puzzle Game, cost dihitung berdasarkan kedalaman node + jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. Penelusuran node selanjutnya dipilih dari node yang memiliki kost terkecil. Node dengan kost terkecil akan selalu dibangkitkan sampai menemukan solusi pada upa pohonnya.

II. Penjelasan Algoritma yang Dibuat

Secara garis besar, Langkah-langkah penyelesaiannya 15-Puzzle dengan memanfaatkan algoritma branch and bound adalah sebagai berikut:

1. Membaca input berupa txt file atau input manual matriks yang menjadi initial state position puzzle (root node)
2. Memperkirakan apakah puzzle memenuhi syarat agar dapat diselesaikan, syarat: fungsi Σ kurang(i) + x hasilnya genap, Fungsi kurang(i), fungsi ini diimplementasikan dengan menghitung banyak tile j dimana $j < i$ dan $POSISI(j) >$

POSISI(i). x didapatkan dari penempatan ubin kosong yang memiliki nilai 1 pada posisi yang diarsir sedangkan 0 untuk posisi yang tidak diarsir. Jika tidak reachable maka akan mengeluarkan pesan tidak dapat diselesaikan, jika bisa maka dilanjutkan dengan penyelesaian persoalan dengan algoritma branch and bound.

3. Proses BnB:

- 1) penginisialisasian **PQ** sebagai priority queue yang nantinya akan di append node-node calon solusi berdasarkan **priority cost terkecil di paling depan**. Cost pada priority queue dihitung sebagai cost masing-masing node (jumlah ubin yang tidak berada pada tempatnya) dan kedalaman node tersebut dari root.
 - 2) Untuk pengiterasian pertama, root akan menjadi parent node yang kemudian di push ke PQ. selama PQ tidak kosong akan dilakukan looping untuk mengiterasi pohon ruang status yang dapat terbentuk. Pengiterasian dilakukan dengan pop elemen terdepan prioqueue yaitu dengan cost terkecil, kemudian akan ada dua kemungkinan:
 - Parent node ini adalah **solusi**, ditandai dengan **cost = 0**
 - Parent node ini belum **menjadi solusi** dan melanjutkan pembangunan pohon ruang status
 - 3) Jika pembuatan ruang status dilanjutkan, maka akan generate child nodes dari parent node yang telah ditentukan. Dilakukan dengan pemanggilan fungsi **generateChildNode**
 - 4) Proses generate child nodes akan dilakukan untuk gerakan **DOWN, UP, RIGHT, LEFT** dengan memperhatikan jika **memungkinkan gerakan terjadi** (masih di dalam ukuran 4x4) dan **tidak** dalam **visitedNodes**
 - 5) Lalu dalam proses generate child node akan mereturn nodes yang telah tergenerate
 - 6) Dari nodes yang tergenerate, maka akan di push sesuai urutan cost ke dalam **PQ**, ulangi pengiterasian dengan menunjuk ke langkah 2) dengan root berupa elemen PQ paling depan
4. Setelah semua proses BnB selesai, akan dihitung jumlah nodes yang dibangkitkan, nodes yang dikunjungi, total step, serta step keseluruhan,
5. Untuk mendisplay seluruh step, maka kita akan mengiterasi **nodeSolution** yang merupakan hasil dari return algoritma BnB yang telah dilakukan. NodeSolution ini melakukan **pengiterasian dengan memanggil parentnya sampai root node** (parent node == None) sambil memasukkan ke dalam stack. Lalu terakhir, kita tinggal pop elemen dari stack yang sudah kita sediakan sesuai urutan LIFO dan menampilkan informasi yang dikehendaki seperti cost, matrix, depth, dan step ke berapa.

BAB III

SOURCE PROGRAM

Main.py: diluar fungsi algoritma

```
1  import time
2  import functions
3
4  # fungsi untuk input file berupa txt
5  def inputFile():
6      file = input("Enter file to read: ")
7      with open ('../test/'+file, 'r') as f:
8          m = []
9          for line in f.readlines():
10             m.append( [ int (x) for x in line.split(' ') ] )
11     return m
12
13 # fungsi untuk input berupa manual
14 def inputKeyboard():
15     m = [[0 for _ in range(4)] for _ in range(4)]
16     for i in range (4):
17         for j in range (4):
18             m[i][j] = int(input("Enter value for position ["+str(i)+","+str(j)+"]: "))
19             while (m[i][j]> 16 or m[i][j] == 0):
20                 print("Please only input number 1-15 or 16 for blank space!")
21             m[i][j] = int(input("Enter value for position ["+str(i)+","+str(j)+"]: "))
22     return m
23
24 # fungsi untuk mencetak semua solusi dengan mengiterasi parent tiap node
25 def displaySolution(node):
26     solutions = []
27     while (node!=None):
28         solutions.append(node)
29         node = node.parent
30
31     print("\nSTEPS:\n")
32     for i in range(len(solutions)):
33         solution = solutions.pop()
34         functions.displayInfo(i,solution)
```

```

37 print("""
38
39
40
41
42
43
44
45
46
47
48
49
50 """)
51
52 x = int(input("MAIN MENU \n1. Read .txt file \n2. Manual Input\nYour Choice: "))
53
54 while (not(x==1 or x==2)):
55     print("Invalid Choice! Please Choose either 1 or 2")
56     x = int(input("MAIN MENU \n1. Input file.txt \n2. Input Keyboard\nYour Choice: "))
57
58 if x == 1:
59     m = inputFile()
60 elif x == 2:
61     m = inputKeyboard()
62
63 print("\nPuzzle TO SOLVE")
64 functions.displayMatrix(m)
65 print("\n")
66
67 # mengecek apakah goal reachable dengan metode heuristik
68 if(not functions.reachableGoal(m)):
69     print("\nYour Puzzle can't be Solved")
70     print("""
71
72
73
74
75
76 """)
77
78 else:
79     print("\nYour Puzzle can be Solved! :) \n")
80     print("""
81
82
83
84
85
86 Solving.. please wait
87
88 """)
89     rootCost = functions.cost(m)
90     # apabila initial state puzzle sudah berupa susunan solusi maka tidak perlu menjalankan pencarian solusi
91     if rootCost==0:
92         print("THE ORIGINAL PUZZLE IS THE SOLUTION!")
93         functions.displayMatrix(m)
94     else:
95         start_time = time.time()
96         # pemanfaatan algoritma branch and bound untuk pencarian solusi
97         nodeSolution = functions.BnB(m)
98         # durasi berjalannya program BnB
99         duration = (time.time() - start_time)
100         # menampilkan semua nodes solusi ke layar
101         displaySolution(nodeSolution)
102         print("Program algorithm executed in %s seconds" % round(duration,5))
103

```


Pq.py: class pq dan node yang digunakan

```
1  from heapq import heappush, heappop
2
3
4  # kelas untuk priority queue
5  class priorityQueue:
6      #konstruktor
7      def __init__(self):
8          self.heap = []
9
10     # push pada priority queue
11     def push(self, k):
12         heappush(self.heap, k)
13
14     # pop pada priority queue
15     def pop(self):
16         return heappop(self.heap)
17
18     # mengecek apakah priority queue empty atau tidak
19     def isEmpty(self):
20         if not self.heap:
21             return True
22         else:
23             return False
24
25     class node:
26         # konstruktor
27         def __init__(self, parent, mat,
28             cost, level, move):
29
30             #m enyimpan parent node
31             self.parent = parent
32
33             # menyimpan matrix
34             self.mat = mat
35
36             # minyimpan cost (ubin yang tidak sesuai tempatnya)
37             self.cost = cost
38
39             # menyimpan depth/level/kedalaman
40             self.level = level
41
42             # menyimoan move
43             self.move = move
44
45         # pemanfaatan loading
46         def __lt__(self, next):
47             if (self.cost + self.level == next.cost + next.level):
48                 return self.cost < next.cost
49             else:
50                 return self.cost + self.level < next.cost + next.level
51
```

Functions.py: fungsi-fungsi yang dibutuhkan dalam algoritma

```
1 import pq
2
3 def kurang(m, el, iNow, jNow):
4     # Penjelasan:
5     # fungsi ini berguna untuk menghitung banyak tile j
6     # dimana j < i dan POSISI(j) > POSISI(i)
7     # fungsi ini akan membantu perhitungan apakah puzzle dapat diselesaikan atau tidak
8     # fungsi ini akan mengembalikan nilai kurang(i)
9     less = 0
10    for i in range (iNow,4):
11        if i==iNow:
12            strj = jNow
13        else:
14            strj = 0
15        for j in range(strj,4):
16            if m[i][j] == 16:
17                continue
18            else:
19                if el > m[i][j]:
20                    less+=1
21    return less
22
23 def reachableGoal(m):
24     # Penjelasan:
25     # fungsi ini akan menentukan apakah persoalan dapat diselesaikan atau tidak
26     # x merupakan nilai 1/0, sel yang diarsir akan bernilai 1, tidak diarsir bernilai 0
27     # reachableGoal dilakukan dengan menghitung sigma Kurang(i) + X
28     # jika sigma Kurang(i) + x ganjil maka tidak dapat dicapai, akan mereturn FALSE
29     # sebaliknya, apabila hasil genap maka akan mereturn TRUE
30     totLess = 0
31     # checking x
32     kurangTable = [0 for i in range (17)]
33     for i in range(4):
34         for j in range(4):
35             if (m[i][j]>16):
36                 return False
37             if (m[i][j]==16):
38                 if ((i+j)%2==0):
39                     x = 0
40                 else:
41                     x = 1
42             less = kurang(m, m[i][j], i, j)
43             totLess += less
44             kurangTable[m[i][j]] = less
45     displayKurangTable(kurangTable, x, totLess)
46     if (x+totLess)%2==0:
47         return True
48     else:
49         return False
50
```

```

51 def cost(m):
52     # Penjelasan:
53     # fungsi ini berguna untuk menghitung cost dengan cara banyaknya ubin tidak kosong yang
54     # tidak terdapat pada susunan akhir
55     # fungsi ini mengembalikan cost suatu node
56     costs = 0
57     ans = [[1,2,3,4],
58            [5,6,7,8],
59            [9,10,11,12],
60            [13,14,15,16]]
61     for i in range(4):
62         for j in range(4):
63             if m[i][j] != 16:
64                 if m[i][j] != ans[i][j]:
65                     costs += 1
66     return costs
67
68 def copyMatrix(m):
69     # Penjelasan:
70     # fungsi ini digunakan untuk mengcopy suatu matrix ke matrix lain
71     # guna dari pengc-copy-an matrix ini adalah agar tidak merusak sebuah matrix saat dilakukan deep copy
72     # fungsi ini mengembalikan matrix yang telah tercopy
73     newM = []
74     for i in range(4):
75         newM.append([])
76         for j in range(4):
77             newM[i].append(m[i][j])
78     return newM
79
80 def moveUp(m):
81     # Penjelasan:
82     # fungsi ini ditujukan untuk mengubah posisi ubin kosong ke atas
83     # fungsi ini akan mendeeep copy matrix sesuai dengan pergeseran yang dilakukan
84     # pergeseran terlebih dahulu di cek apakah memungkinkan (masih dalam ukuran 4x4)
85     # jika tidak memenuhi maka tidak akan melakukan perubahan pada matrix
86     found = False
87     for i in range(4):
88         for j in range(4):
89             if m[i][j] == 16 and i-1 >= 0 and (not found):
90                 m[i][j], m[i-1][j] = m[i-1][j], m[i][j]
91                 found = True
92

```

```

93 def moveDown(m):
94     # Penjelasan:
95     # fungsi ini ditujukan untuk mengubah posisi ubin kosong ke bawah
96     # fungsi ini akan mendeep copy matrix sesuai dengan pergeseran yang dilakukan
97     # pergeseran terlebih dahulu di cek apakah memungkinkan (masih dalam ukuran 4x4)
98     # jika tidak memenuhi maka tidak akan melakukan perubahan pada matrix
99     found = False
100    for i in range (4):
101        for j in range (4):
102            if m[i][j] == 16 and i+1<=3 and (not found):
103                m[i][j], m[i+1][j] = m[i+1][j], m[i][j]
104                found = True
105
106
107 def moveRight(m):
108     # Penjelasan:
109     # fungsi ini ditujukan untuk mengubah posisi ubin kosong ke kanan
110     # fungsi ini akan mendeep copy matrix sesuai dengan pergeseran yang dilakukan
111     # pergeseran terlebih dahulu di cek apakah memungkinkan (masih dalam ukuran 4x4)
112     # jika tidak memenuhi maka tidak akan melakukan perubahan pada matrix
113     found = False
114     for i in range (4):
115         for j in range (4):
116             if m[i][j] == 16 and j+1<=3 and (not found):
117                 m[i][j], m[i][j+1] = m[i][j+1], m[i][j]
118                 found = True
119
120 def moveLeft(m):
121     # Penjelasan:
122     # fungsi ini ditujukan untuk mengubah posisi ubin kosong ke kiei
123     # fungsi ini akan mendeep copy matrix sesuai dengan pergeseran yang dilakukan
124     # pergeseran terlebih dahulu di cek apakah memungkinkan (masih dalam ukuran 4x4)
125     # jika tidak memenuhi maka tidak akan melakukan perubahan pada matrix
126     found = False
127     for i in range (4):
128         for j in range (4):
129             if m[i][j] == 16 and j-1>=0 and (not found):
130                 m[i][j], m[i][j-1] = m[i][j-1], m[i][j]
131                 found = True
132

```

```

133 def displayMatrix():
134     # Penjelasan:
135     # fungsi ini melakukan pencetakan matrix ke layar
136     print("┌───┬───┬───┬───┐")
137     for i in range(4):
138         print("│", end=" ")
139         for j in range(4):
140             el = m[i][j]
141             line = "0"+ str(el) if el < 10 else str(el)
142             if el == 16:
143                 line = " "
144             line = line + " │" if j != 3 else line + " │"
145             print(line, end=" ")
146         if i != 3:
147             print("\n┌───┬───┬───┬───┐")
148     print("\n└───┴───┴───┴───┘")
149
150 def isSame(m1, m2):
151     # Penjelasan:
152     # fungsi ini membandingkan dua buah matrix yang ada di parameter
153     # jika sama mereturn TRUE, jika berbeda mereturn FALSE
154     for i in range(4):
155         for j in range(4):
156             if m1[i][j] != m2[i][j]:
157                 return False
158     return True
159
160 def isNotInListOfNodes(nodeToCheck, listOfNodes):
161     # Penjelasan:
162     # fungsi ini mengecek apakah sebuah node tidak berada di sebuah list of nodes
163     # jika ada akan mereturn FALSE, jika tidak ada akan TRUE
164     for node in listOfNodes:
165         if isSame(node.mat, nodeToCheck.mat):
166             return False
167     return True
168
169 def displayInfo(step, node):
170     # Penjelasan:
171     # fungsi ini digunakan untuk mencetak info dari setiap pengiterasian yang digunakan ke layar
172     if step != 0:
173         print("---- S T E P ",step,"----")
174     else:
175         print("INITIAL PUZZLE STATE")
176     print("COST :", node.cost)
177     print("LEVEL :", node.level)
178     if node.move!="initial position":
179         print("MOVE :", node.move)
180     displayMatrix(node.mat)
181     print("\n")
182

```

```

183 def displayKurangTable(kurangTable,x, totless):
184     # Penjelasan:
185     # fungsi ini digunakan untuk mencetak i dan nilai masing-masing yaitu kurang(i) secara berurutan ke layar
186     # fungsi ini juga mencetak signa dari kurang(i) dan x serta pertambahan keduanya
187     print(" -----")
188     print("| i | Kurang(i) |")
189     print(" -----")
190     for i in range(len(kurangTable)):
191         if i == 0:
192             continue
193         else:
194             if i<10:
195                 if kurangTable[i] < 10:
196                     print("| 0"+str(i)+" | ", kurangTable[i], " |")
197                 else:
198                     print("| 0"+str(i)+" | ", kurangTable[i], " |")
199             else:
200                 if kurangTable[i] < 10:
201                     print("| ",i," | ",kurangTable[i]," |")
202                 else:
203                     print("| ",i," | ",kurangTable[i]," |")
204     print(" -----")
205     print("\nI Kurang(i)      =",str(totless))
206     print("Nilai x          =",str(x))
207     print("I Kurang(i) + x =", str(totless + x))
208     print("\n")
209

```

```

210 def generateChildNode(parent, visitedNodes):
211     # Penjelasan:
212     # fungsi untuk mengenerate ruang pohon status sesuai parent node(cost terkecil)
213     # fungsi ini akan mengembalikan ARRAY OF GENERATED NODES
214
215     # menginisialisasi tempat penyimpanan node yaitu pada sebuah array
216     generatedNodes = []
217
218     # mengcopy matrix agar menjaga tidak terubah saat deep copy
219     parMatrix = copyMatrix(parent.mat)
220
221     # DOWN
222     # mengcopy matrix parent yang nantinya akan dilakukan pergeseran ke bawah
223     mDown = copyMatrix(parMatrix)
224     # melakukan pergeseran ke bawah
225     moveDown(mDown)
226     # membuat node baru sesuai atribut pada kelas node
227     nodeDown = pq.node(parent, mDown, cost(mDown), parent.level+1, "DOWN")
228     # jika node yang baru dibuat tidak ada di list of visitedNodes, maka node akan ditambahkan ke list of nodes
229     if isNotInListOfNodes(nodeDown, visitedNodes):
230         generatedNodes.append(nodeDown)
231
232     # UP
233     # mengcopy matrix parent yang nantinya akan dilakukan pergeseran ke atas
234     mUp = copyMatrix(parMatrix)
235     # melakukan pergeseran ke atas
236     moveUp(mUp)
237     # membuat node baru sesuai atribut pada kelas node
238     nodeUp = pq.node(parent, mUp, cost(mUp), parent.level+1, "UP")
239     # jika node yang baru dibuat tidak ada di list of visitedNodes, maka node akan ditambahkan ke list of nodes
240     if isNotInListOfNodes(nodeUp, visitedNodes):
241         generatedNodes.append(nodeUp)
242
243     # RIGHT
244     # mengcopy matrix parent yang nantinya akan dilakukan pergeseran ke kanan
245     mRight = copyMatrix(parMatrix)
246     # melakukan pergeseran ke kanan
247     moveRight(mRight)
248     # membuat node baru sesuai atribut pada kelas node
249     nodeRight = pq.node(parent, mRight, cost(mRight), parent.level+1, "RIGHT")
250     # jika node yang baru dibuat tidak ada di list of visitedNodes, maka node akan ditambahkan ke list of nodes
251     if isNotInListOfNodes(nodeRight, visitedNodes):
252         generatedNodes.append(nodeRight)
253
254     # LEFT
255     # mengcopy matrix parent yang nantinya akan dilakukan pergeseran ke kiri
256     mLeft = copyMatrix(parMatrix)
257     # melakukan pergeseran ke kiri
258     moveLeft(mLeft)
259     # membuat node baru sesuai atribut pada kelas node
260     nodeLeft = pq.node(parent, mLeft, cost(mLeft), parent.level+1, "LEFT")
261     # jika node yang baru dibuat tidak ada di list of visitedNodes, maka node akan ditambahkan ke list of nodes
262     if isNotInListOfNodes(nodeLeft, visitedNodes):
263         generatedNodes.append(nodeLeft)
264
265     # mereturn nodes yang telah dibangkitkan
266     return generatedNodes
267

```

```

def BnB(m):
    # Penjelasan:
    # Fungsi ini dilakukan untuk menyelesaikan persoalan 15 puzzle dengan algoritma branch and bound
    # Fungsi akan berhenti saat dalam keadaan goal state dan mengembalikan node solusi

    # menginisialisasikan priority queue
    PQ = pq.PriorityQueue()

    # membuat node awal yaitu node dengan matrix yang sama dengan matrix awal
    parent = pq.node(None, m, cost(m), 0, "initial position")

    # menginisialisasi array of visited nodes
    visitedNodes = []

    # menginisialisasi array of generated nodes
    generatedNodes = [parent]

    # menambahkan node awal ke priority queue
    PQ.push(parent)

    step = 0
    while not(PQ.isEmpty()):
        # pop elemen paling depan prioqueue
        parent = PQ.pop()

        # menambahkan node yang sekarang akan dibangkitkan sebagai visited node
        visitedNodes.append(parent)

        # jika cost = 0 maka sudah mencapai goal state
        if parent.cost == 0:
            print("\n\n( ^_^ )/ -- YOUR PUZZLE IS SOLVED! -- ٩(๑!٧!๑)+")
            print("Total Generated Nodes:", len(generatedNodes), "nodes")
            print("Total Visited Nodes:", len(visitedNodes), "nodes")
            print("Total Steps:", parent.level)
            return parent
        else:
            # membangkitkan node child sesuai dengan parent
            childNodes = generateChildNode(parent, visitedNodes)
            # menambahkan node child ke prioqueue dengan priority berupa cost(depth+tile tidak kosong yang berbeda dengan solusi)
            for child in childNodes:
                PQ.push(child)
                generatedNodes.append(child)

```


BAB IV SCREENSHOT INPUT DAN OUPUT

MAIN MENU

LOADING (jika dapat diselesaikan)


UJI COBA TXT

Percobaan 1: Ditemukan

File	solved16.txt
Puzzle Awal	1 2 3 4 5 6 7 16 9 10 12 8 11 13 14 15
Output	

Your Choice: 1
Enter file to read: solved16.tx

PUZZLE TO SOLVE

01	02	03	04
05	06	07	
09	10	12	08
11	13	14	15

☆* . ☆
 . ^ _ n * ☆
 * ☆ (. v .) / .
 ☆ * c / * ☆
 ☆ * (? / . ☆
 /)
 Solving.. please wait

\ (" 7 ") / -- YOUR PUZZLE IS SOLVED!
 ED! -- 3 (0 1 0 0) 8
 Total Generated Nodes: 1314 node
 Total Visited Nodes: 628 nodes
 Total Steps: 16

STEPS:

INITIAL PUZZLE STATE

COST : 6
LEVEL : 0

01	02	03	04
05	06	07	
09	10	12	08
11	13	14	15

---- STEP 1 ----

COST : 5
LEVEL : 1
MOVE : DOWN

01	02	03	04
05	06	07	08

i	Kurang(i)
01	0
02	0
03	0
04	0
05	0
06	0
07	0
08	0
09	1
10	1
11	0
12	2
13	0
14	0
15	0
16	8

Σ Kurang(i) = 12
 Nilai x = 0
 Σ Kurang(i) + x = 12

Your Puzzle can be Solved! :)

STEPS:

INITIAL PUZZLE STATE

COST : 6
LEVEL : 0

01	02	03	04
05	06	07	
09	10	12	08
11	13	14	15

---- STEP 1 ----

COST : 5
LEVEL : 1
MOVE : DOWN

01	02	03	04
05	06	07	08
09	10	12	
11	13	14	15

---- STEP 2 ----

COST : 4
LEVEL : 2
MOVE : LEFT

01	02	03	04
05	06	07	08
09	10		12
11	13	14	15

---- STEP 3 ----

COST : 4
LEVEL : 3
MOVE : DOWN

01	02	03	04
05	06	07	08
09	10	14	12
11	13		15

---- STEP 4 ----

COST : 4
LEVEL : 4
MOVE : LEFT

01	02	03	04
05	06	07	08
09	10	14	12
11		13	15

---- STEP 5 ----

COST : 4
LEVEL : 5
MOVE : LEFT

01	02	03	04
05	06	07	08
09	10	14	12
	11	13	15

---- STEP 6 ----

COST : 5
LEVEL : 6
MOVE : UP

01	02	03	04
05	06	07	08
	10	14	12
09	11	13	15

---- STEP 7 ----

COST : 6
LEVEL : 7
MOVE : RIGHT

01	02	03	04
05	06	07	08
10		14	12
09	11	13	15

---- STEP 8 ----

COST : 6
LEVEL : 8
MOVE : DOWN

01	02	03	04
05	06	07	08
10	11	14	12
09		13	15

---- STEP 9 ----

COST : 6
LEVEL : 9
MOVE : RIGHT

01	02	03	04
05	06	07	08
10	11	14	12
09	13		15

<p>---- S T E P 10 ----</p> <p>COST : 6</p> <p>LEVEL : 10</p> <p>MOVE : UP</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>10</td><td>11</td><td></td><td>12</td></tr> <tr><td>09</td><td>13</td><td>14</td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08	10	11		12	09	13	14	15	<p>---- S T E P 11 ----</p> <p>COST : 5</p> <p>LEVEL : 11</p> <p>MOVE : LEFT</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>10</td><td></td><td>11</td><td>12</td></tr> <tr><td>09</td><td>13</td><td>14</td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08	10		11	12	09	13	14	15	<p>---- S T E P 13 ----</p> <p>COST : 3</p> <p>LEVEL : 13</p> <p>MOVE : DOWN</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>09</td><td>10</td><td>11</td><td>12</td></tr> <tr><td></td><td>13</td><td>14</td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08	09	10	11	12		13	14	15	<p>---- S T E P 15 ----</p> <p>COST : 1</p> <p>LEVEL : 15</p> <p>MOVE : RIGHT</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>09</td><td>10</td><td>11</td><td>12</td></tr> <tr><td>13</td><td>14</td><td></td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08	09	10	11	12	13	14		15
01	02	03	04																																																																
05	06	07	08																																																																
10	11		12																																																																
09	13	14	15																																																																
01	02	03	04																																																																
05	06	07	08																																																																
10		11	12																																																																
09	13	14	15																																																																
01	02	03	04																																																																
05	06	07	08																																																																
09	10	11	12																																																																
	13	14	15																																																																
01	02	03	04																																																																
05	06	07	08																																																																
09	10	11	12																																																																
13	14		15																																																																
<p>---- S T E P 11 ----</p> <p>COST : 5</p> <p>LEVEL : 11</p> <p>MOVE : LEFT</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>10</td><td></td><td>11</td><td>12</td></tr> <tr><td>09</td><td>13</td><td>14</td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08	10		11	12	09	13	14	15	<p>---- S T E P 12 ----</p> <p>COST : 4</p> <p>LEVEL : 12</p> <p>MOVE : LEFT</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td></td><td>10</td><td>11</td><td>12</td></tr> <tr><td>09</td><td>13</td><td>14</td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08		10	11	12	09	13	14	15	<p>---- S T E P 14 ----</p> <p>COST : 2</p> <p>LEVEL : 14</p> <p>MOVE : RIGHT</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>09</td><td>10</td><td>11</td><td>12</td></tr> <tr><td>13</td><td></td><td>14</td><td>15</td></tr> </table>	01	02	03	04	05	06	07	08	09	10	11	12	13		14	15	<p>---- S T E P 16 ----</p> <p>COST : 0</p> <p>LEVEL : 16</p> <p>MOVE : RIGHT</p> <table> <tr><td>01</td><td>02</td><td>03</td><td>04</td></tr> <tr><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>09</td><td>10</td><td>11</td><td>12</td></tr> <tr><td>13</td><td>14</td><td>15</td><td></td></tr> </table>	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
01	02	03	04																																																																
05	06	07	08																																																																
10		11	12																																																																
09	13	14	15																																																																
01	02	03	04																																																																
05	06	07	08																																																																
	10	11	12																																																																
09	13	14	15																																																																
01	02	03	04																																																																
05	06	07	08																																																																
09	10	11	12																																																																
13		14	15																																																																
01	02	03	04																																																																
05	06	07	08																																																																
09	10	11	12																																																																
13	14	15																																																																	
<p>Program algorithm executed in 0.37405 seconds</p>																																																																			

Percobaan 2: Ditemukan

File	solved21.txt
Puzzle Awal	2 3 4 11 1 5 10 8 9 6 12 15 13 14 16 7
Output	

1. Read .txt file
2. Manual Input
Your Choice: 1
Enter file to read: solved21.txt

$\sum \text{Kurang}(i) = 25$
Nilai x = 1
 $\sum \text{Kurang}(i) + x = 26$

PUZZLE TO SOLVE

02	03	04	11
01	05	10	08
09	06	12	15
13	14		07

Your Puzzle can be Solved! :)

```

☆ * . ☆
      . ^ _ ^ n * ☆
    * ☆ ( . v . ) / .
      . c      / * ☆
    ☆ * ( ʘ / . ☆
      (
Solving... please wait

```

\ (' ʘ ') / -- YOUR PUZZLE IS SOLVED! -
Total Generated Nodes: 5746 nodes
Total Visited Nodes: 2718 nodes
Total Steps: 21

STEPS:

INITIAL PUZZLE STATE

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	16

---- STEP 1 ----
COST : 11
LEVEL : 1
MOVE : UP

02	03	04	11
01	05	10	08
09	06		15
13	14	12	07

---- STEP 2 ----
COST : 11
LEVEL : 2
MOVE : UP

02	03	04	11
01	05		08
09	06	10	15
13	14	12	07

---- STEP 3 ----
COST : 12
LEVEL : 3
MOVE : RIGHT

02	03	04	11
01	05	08	
09	06	10	15
13	14	12	07

---- STEP 4 ----
COST : 12
LEVEL : 4
MOVE : UP

02	03	04	
01	05	08	11
09	06	10	15
13	14	12	07

---- STEP 5 ----
COST : 11
LEVEL : 5
MOVE : LEFT

02	03		04
01	05	08	11
09	06	10	15
13	14	12	07

---- STEP 6 ----
COST : 10
LEVEL : 6
MOVE : LEFT

02		03	04
01	05	08	11
09	06	10	15
13	14	12	07

---- STEP 7 ----
COST : 9
LEVEL : 7
MOVE : LEFT

	02	03	04
01	05	08	11
09	06	10	15
13	14	12	07

---- STEP 8 ----
COST : 8
LEVEL : 8
MOVE : DOWN

01	02	03	04
	05	08	11
09	06	10	15
13	14	12	07

---- STEP 9 ----
COST : 7
LEVEL : 9
MOVE : RIGHT

01	02	03	04
05		08	11
09	06	10	15
13	14	12	07

---- STEP 10 ----
COST : 6
LEVEL : 10
MOVE : DOWN

01	02	03	04
05	06	08	11
09		10	15
13	14	12	07

---- STEP 11 ----
COST : 5
LEVEL : 11
MOVE : RIGHT

01	02	03	04
05	06	08	11
09	10		15
13	14	12	07

---- STEP 12 ----
COST : 5
LEVEL : 12
MOVE : RIGHT

01	02	03	04
05	06	08	11
09	10	15	
13	14	12	07

---- STEP 13 ----
COST : 5
LEVEL : 13
MOVE : DOWN

01	02	03	04
05	06	08	11
09	10	15	07
13	14	12	

---- STEP 14 ----
COST : 5
LEVEL : 14
MOVE : LEFT

01	02	03	04
05	06	08	11
09	10		07
13	14		12

---- STEP 15 ----
COST : 4
LEVEL : 15
MOVE : UP

01	02	03	04
05	06	08	11
09	10		07
13	14	15	12

---- STEP 16 ----
COST : 4
LEVEL : 16
MOVE : RIGHT

01	02	03	04
05	06	08	11
09	10	07	
13	14	15	12

---- STEP 17 ----
COST : 4
LEVEL : 17
MOVE : UP

01	02	03	04
05	06	08	
09	10	07	11
13	14	15	12

---- STEP 18 ----
COST : 3
LEVEL : 18
MOVE : LEFT

01	02	03	04
05	06		08
09	10	07	11
13	14	15	12

---- STEP 19 ----
COST : 2
LEVEL : 19
MOVE : DOWN

01	02	03	04
05	06	07	08
09	10		11
13	14	15	12

```

---- S T E P  20 ----
COST  : 1
LEVEL : 20
MOVE  : RIGHT

```

01	02	03	04
05	06	07	08
09	10	11	
13	14	15	12

```

---- S T E P  21 ----
COST  : 0
LEVEL : 21
MOVE  : DOWN

```

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Program algorithm executed in 4.55268 seconds

Percobaan 3: Tidak Ditemukan

File	unsolved1.txt
Puzzle Awal	1 3 4 15 2 16 5 12 7 6 11 14 8 9 10 13
Output	

Your Choice: 1
Enter file to read: unsolved1.txt

PUZZLE TO SOLVE

01	03	04	15
02		05	12
07	06	11	14
08	09	10	13

i	Kurang(i)
01	0
02	0
03	1
04	1
05	0
06	0
07	1
08	0
09	0
10	0
11	3
12	6
13	0
14	4
15	11
16	10

Σ Kurang(i) = 37
Nilai x = 0
 Σ Kurang(i) + x = 37

Your Puzzle can't be Solved



UJI COBA INPUT MANUAL

Percobaan 4: Ditemukan

Puzzle Awal	<div><div>1 2 3 4 5 6 16 8 9 10 7 16 13 14 15 12</div><div><div>MAIN MENU 1. Read .txt file 2. Manual Input Your Choice: 2 Enter value for position [0,0]: 1 Enter value for position [0,1]: 2 Enter value for position [0,2]: 3 Enter value for position [0,3]: 4 Enter value for position [1,0]: 5 Enter value for position [1,1]: 6 Enter value for position [1,2]: 7 Enter value for position [1,3]: 8 Enter value for position [2,0]: 9 Enter value for position [2,1]: 10 Enter value for position [2,2]: 11 Enter value for position [2,3]: 16 Enter value for position [3,0]: 13 Enter value for position [3,1]: 14 Enter value for position [3,2]: 15 Enter value for position [3,3]: 12</div><div><div>PUZZLE TO SOLVE</div><table><tr><td>01</td><td>02</td><td>03</td><td>04</td></tr><tr><td>05</td><td>06</td><td>07</td><td>08</td></tr><tr><td>09</td><td>10</td><td>11</td><td></td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table></div></div></div>	01	02	03	04	05	06	07	08	09	10	11		13	14	15	12
01	02	03	04														
05	06	07	08														
09	10	11															
13	14	15	12														
Output																	

i	Kurang(i)
01	0
02	0
03	0
04	0
05	0
06	0
07	0
08	0
09	0
10	0
11	0
12	0
13	1
14	1
15	1
16	4

$\Sigma \text{ Kurang}(i) = 7$
 Nilai $x = 1$
 $\Sigma \text{ Kurang}(i) + x = 8$

Your Puzzle can be Solved! :)

```

☆* . ☆
      . ^ _ ^ n * ☆
    * ☆ ( . ∇ · ) / .
      . ⊂ √ * ☆
    ☆* ( √ √ . ☆
      ( √
Solving... please wait
  
```

\ (° ʌ °) / -- YOUR PUZZLE IS SOLVED! -- ʌ
 (ʌ ∇ ʌ) ʌ
 Total Generated Nodes: 4 nodes
 Total Visited Nodes: 2 nodes
 Total Steps: 1

STEPS:

INITIAL PUZZLE STATE

COST : 1

LEVEL : 0

01	02	03	04
05	06	07	08
09	10	11	
13	14	15	12

---- S T E P 1 ----

COST : 0

LEVEL : 1

MOVE : DOWN

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Percobaan 5: Ditemukan

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


```
MAIN MENU
1. Read .txt file
2. Manual Input
Your Choice: 2
Enter value for position [0,0]: 1
Enter value for position [0,1]: 2
Enter value for position [0,2]: 3
Enter value for position [0,3]: 4
Enter value for position [1,0]: 5
Enter value for position [1,1]: 6
Enter value for position [1,2]: 16
Enter value for position [1,3]: 8
Enter value for position [2,0]: 9
Enter value for position [2,1]: 10
Enter value for position [2,2]: 7
Enter value for position [2,3]: 11
Enter value for position [3,0]: 13
Enter value for position [3,1]: 14
Enter value for position [3,2]: 15
Enter value for position [3,3]: 12
```

PUZZLE TO SOLVE

01	02	03	04
05	06		08
09	10	07	11
13	14	15	12

Output

i	Kurang(i)
01	0
02	0
03	0
04	0
05	0
06	0
07	0
08	1
09	1
10	1
11	0
12	0
13	1
14	1
15	1
16	9

$\Sigma \text{ Kurang}(i) = 15$
 Nilai x = 1
 $\Sigma \text{ Kurang}(i) + x = 16$

Your Puzzle can be Solved! :)

☆* . ☆
 . ^ _ ^ n * ☆
 * ☆ (. ∇ .) / .
 . ⊂ / * ☆
 ☆* (∩ / . ☆
 (/

Solving... please wait



\(\ ^\circ \wedge^\circ)/ -- YOUR PUZZLE IS SOLVED! -- ٩(●v●)۶

Total Generated Nodes: 10 nodes
 Total Visited Nodes: 4 nodes
 Total Steps: 3

STEPS:

INITIAL PUZZLE STATE

COST : 3

LEVEL : 0

01	02	03	04
05	06		08
09	10	07	11
13	14	15	12

---- S T E P 1 ----

COST : 2

LEVEL : 1

MOVE : DOWN

01	02	03	04
05	06	07	08
09	10		11
13	14	15	12

13	14	15	12
----	----	----	----

---- S T E P 2 ----

COST : 1

LEVEL : 2

MOVE : RIGHT

01	02	03	04
05	06	07	08
09	10	11	
13	14	15	12

---- S T E P 3 ----

COST : 0

LEVEL : 3

MOVE : DOWN

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Program algorithm executed in 0.0 seconds

Percobaan 6: Tidak Ditemukan

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

MAIN MENU

1. Read .txt file

2. Manual Input

Your Choice: 2

Enter value for position [0,0]: 15

Enter value for position [0,1]: 3

Enter value for position [0,2]: 4

Enter value for position [0,3]: 1

Enter value for position [1,0]: 2

Enter value for position [1,1]: 16

Enter value for position [1,2]: 5

Enter value for position [1,3]: 12

Enter value for position [2,0]: 6

Enter value for position [2,1]: 7

Enter value for position [2,2]: 11

Enter value for position [2,3]: 14

Enter value for position [3,0]: 8

Enter value for position [3,1]: 9

Enter value for position [3,2]: 10

Enter value for position [3,3]: 13

PUZZLE TO SOLVE

15	03	04	01
02		05	12
06	07	11	14
08	09	10	13

i	Kurang(i)
01	0
02	0
03	2
04	2
05	0
06	0
07	0
08	0
09	0
10	0
11	3
12	6
13	0
14	4
15	14
16	10

$$\Sigma \text{ Kurang}(i) = 41$$

$$\text{Nilai } x = 0$$

$$\Sigma \text{ Kurang}(i) + x = 41$$

Your Puzzle can't be Solved



Lampiran

1. Link repository github: https://github.com/graceclaudia19/Tucil3_13520078

2. Checklist

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