

Untuk Memenuhi Tugas Mata Kuliah Kecil 3
Strategi Algoritma (IF 2211)



Disusun Oleh:

Jaya Mangalo Soegeng Rahardjo

NIM : 13520015

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

A. Algoritma Program

Step 0.5: Display

Program mengeluarkan Basic GUI dengan input bar yang bisa digunakan untuk menulis file .txt yang terletak di folder ../test/.

Step 1: Preparation and Validity Checking

Program akan membaca input user dan mencari file dengan nama yang sama serta mengubah isi file menjadi matrix.

Program kemudian akan menggunakan fungsi *isReachable* dan *KurangIplusX* yang menghitung KurangI serta X dan menentukan apakah matriks input memiliki solusi atau tidak.

Step 2: Solving

Jika Puzzle memiliki solusi, maka program akan membuat Node *root* dengan isi matrix awal dan *path* ["none"]. Node tersebut di queue kedalam priorityqueue.

Lalu program akan masuk while-loop yang akan:

1. Mengecek jika node adalah hasil akhir yang diinginkan, jika iya loop berhenti dan *path* solusi dan *Expanded_Node* akan direturn.
2. Jika tidak, program akan mengattempt untuk menambahkan 4 node (dimana 16 bergerak atas, bawah, kiri, kanan)
 - i) Program akan mengecek jika pergerakan *directional* valid, misalnya jika 16 terletak di baris paling atas, artinya pergerakan ke atas tidak valid.
 - ii) Program juga tidak memperbolehkan pergerakan yang berlawanan dengan pergerakan sebelumnya, misalnya jika node sebelumnya bergerak ke atas, berarti pergerakan selanjutnya tidak boleh ke bawah.
 - iii) Program juga mengecek jika state tersebut sudah di-expand atau belum, program akan mengubah matrix menjadi key yang kemudian dimasukkan ke hash table. Karena hash table memiliki instant lookup, pengecekan ini sangat cepat dan efisien. Jika state sudah di-expand, maka program tidak akan menambahkan pergerakan tersebut.

Step 3: Display

Program men-display interactive 4x4 matrix yang bisa digerakkan dengan input buttons yang disediakan. Program juga men-display informasi relevan seperti KurangI, KurangI+X, Reachable, Expanded_Nodes, SearchTime.

B. Test Case

invalid1.txt

The image shows a 15-Puzzle application window and a Notepad window. The 15-Puzzle window displays a 4x4 grid with numbers 01 through 15. The bottom of the window shows navigation buttons, a progress indicator '0 of 0', and puzzle statistics: Kurangl = [0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 3, 6, 0, 4, 11, 10], Kurangl + X = 37, Solveable = False, Expanded Nodes: 0, and Search Time: 0 seconds. The Notepad window, titled 'invalid1.txt - Notepad', shows the following content:

```
File Edit Format View Help
1 3 4 15
2 16 5 12
7 6 11 14
8 9 10 13
```

The Notepad window also shows status information: Ln 1, 100%, Windows (CRLF), UTF-8.

invalid2.txt

15-Puzzle

invalid2.txt Solve

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

<< < > >>

0 of 0

Kurangl = [0, 0, 1, 2, 2, 1, 6, 4, 1, 5, 0, 8, 12, 6, 6, 12]
Kurangl + X = 67
Solveable = False
Expanded Nodes: 0
Search Time: 0 seconds

invalid2.txt - Notepad

File Edit Format View Help

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

Ln 1, 100% Windows (CRLF) UTF-8

valid1.txt

The image shows a 15-Puzzle solver application window titled "15-Puzzle" and a Notepad window titled "valid1.txt - Notepad".

The "15-Puzzle" window has a text input field containing "valid1.txt" and a "Solve" button. Below the input field is a 4x4 grid representing the puzzle state:

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

Below the grid are four navigation buttons: "<<", "<", ">", and ">>". Below these buttons is the text "1 of 4". At the bottom of the window, the following information is displayed:

Kurang1 = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 9]
Kurang1 + X = 16
Solvable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

The "valid1.txt - Notepad" window shows the contents of the file "valid1.txt":

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

The Notepad window also shows the status bar with "Ln 1, 100%, Windows (CRLF), UTF-8".

Step 1

15-Puzzle

Solve

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

<< < > >>

1 of 4

Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 9]
Kurangl + X = 16
Solvable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

Step 2

15-Puzzle

Solve

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

<< < > >>

2 of 4

Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 9]
Kurangl + X = 16
Solvable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

Step 3

15-Puzzle

valid.txt

Solve

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

<<

<

>

>>

3 of 4

Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 9]
Kurangl + X = 16
Solvable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

Step 4

15-Puzzle

valid.txt

Solve

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

<<

<

>

>>

4 of 4

Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 9]
Kurangl + X = 16
Solvable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

valid2.txt

Step tidak ditunjukkan karena terlalu besar

The image shows a 15-Puzzle solver application window titled "15-Puzzle" and a Notepad window titled "valid2.txt - Notepad".

The 15-Puzzle window has a text input field containing "valid2.txt" and a "Solve" button. Below this is a 4x4 grid representing the puzzle state:

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

Below the grid are four navigation buttons: "<<", "<", ">", and ">>". Below these buttons is the text "1 of 130". At the bottom of the window, the following information is displayed:

Kurangl = [0, 0, 1, 2, 2, 1, 6, 4, 1, 5, 0, 8, 12, 6, 7, 12]
Kuragl + X = 68
Solveable = True
Expanded Nodes: 5807
Search Time: 4.67776 seconds

The Notepad window shows the contents of "valid2.txt":

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

The Notepad window also shows a status bar at the bottom with the text "Ln 1, 100% Windows (CRLF) UTF-8".

valid3.txt

Step tidak ditunjukkan karena terlalu besar

The image shows a 15-Puzzle solver application window titled "15-Puzzle" and a Notepad window titled "valid3.txt - Notepad".

The "15-Puzzle" window has a text input field containing "valid3.txt" and a "Solve" button. Below the input field is a 4x4 grid representing the puzzle state:

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

Below the grid are four navigation buttons: "<<", "<", ">", and ">>". Below these buttons, the text "1 of 109" is displayed. At the bottom of the window, the following information is shown:

Kurangl = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Kurangl + X = 120
Solveable = True
Expanded Nodes: 10358
Search Time: 8.61008 seconds

The "valid3.txt - Notepad" window shows the contents of the file "valid3.txt":

```
File Edit Format View Help
16 15 14 13
12 11 10 9
8 7 6 5
4 3 2 1
```

Below the text area, the status bar shows "Ln 1, 100% Windows (CRLF) UTF-8".

Below the Notepad window, a large block of text displays a sequence of moves for solving the puzzle, such as "left", "down", "left", "left", "up", "up", "up", etc.

C. Source Code

BranchAndBound.py

```
src > BranchAndBound.py > ...
1  from queue import PriorityQueue
2  import os
3  import copy
4
5  #Tree Nodes
6  #data is matrix of current state, path is the path to reach that state
7  #up,right,down,left are children
8  class Node:
9      def __init__(self,data,path):
10         self.data = data
11         self.path = path
12
13     def __lt__(self, other):
14         return True;
15
16     def __le__(self, other):
17         return True;
18
19 #reads a file named text from ../test/ and returns inside as a 4x4 matrix of int, invalid inputs will cause the program to fail
20 def read(text):
21     os.chdir("../test")
22     f = open(text, "r")
23
24     mat = []
25     for x in f:
26         arr = (x.strip("\n").split(" ")[4])
27         arr2 = []
28         for x in arr:
29             xInt = int(x)
30             arr2.append(xInt)
31         mat.append(arr2)
32     f.close()
33     return mat;
34
35 #turns the 4x4 matrix into an array of 16
36 def mergelist(mat):
37     arr = []
38     for l in mat:
39         arr += l
40     return arr
41
42 #returns a list consisting of Kurang(I)
43 def KurangI(arr):
44     arr_KurangI = []
45     for i in range(1,17):
46         count = 0
47         for j in range(1,i):
48             if(arr.index(j) > arr.index(i)):
49                 count+=1
50         arr_KurangI += [count]
51     return arr_KurangI;
52
```

```

53 #totals and returns KurangI + X
54 v def KurangIplusX(arr):
55     count = sum(KurangI(arr))
56 v     if(arr.index(16) in [1,3,4,6,9,11,12,14]):
57         count+=1
58     return count
59
60 #returns true if count is event
61 v def isReachable(count):
62 v     if(count%2 == 0):
63         return True
64     return False
65
66 #returns the weight or g(x) of the matrix
67 v def getweight(mat):
68     count = 0;
69 v     for i in range(4):
70         for j in range(4):
71 v             if(mat[i][j] != 4*i + j + 1):
72                 count+=1
73     return count
74
75 #returns the weight or f(x) of the node
76 v def getdepth(Current_Node):
77     return 0.001*len(Current_Node.path)
78     # return 0
79
80 #returns i,j coordinates of 16 in the 4x4 matrix
81 v def Find16(mat):
82 v     for i in range(4):
83 v         for j in range(4):
84 v             if(mat[i][j] == 16):
85                 return i,j
86
87 #turns the matrix into a string such as 01020304..... to be used as a key in a hash table
88 v def matrixtokey(mat):
89     key = ""
90 v     for i in range(4):
91 v         for j in range(4):
92             key += "{:02d}".format(mat[i][j])
93     return key
94

```

```

#checks and updates the hash table
#gets the matrix mat, hashes it into a key
#if key not exist or key exist but weight (from parameter) is lower than key's value in hash table, returns true and insert/update hash table
#if key exist and key's value is lower than weight (from parameter), returns false
def CheckAndUpdateHashTable(mat,weight):
    key = matrixtokey(mat)
    global dict
    if(key not in dict.keys()):
        dict[key] = weight
        return True
    else:
        # return False
        if(dict[key] < weight):
            return False
        else:
            dict[key] = weight
            return True

#create a new branch and enqueues it (if valid from checking hash table)
def AddBranch(mat,path,i,j,direction):
    newmat = copy.deepcopy(mat)
    path2 = copy.deepcopy(path)
    if(direction == "up" and path[-1] != "down" and i!=0):
        newmat[i][j],newmat[i-1][j] = newmat[i-1][j],newmat[i][j]
        newpath = path2 + ["up"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
    elif(direction == "down" and path[-1] != "up" and i!=3):
        newmat[i][j],newmat[i+1][j] = newmat[i+1][j],newmat[i][j]
        newpath = path2 + ["down"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
    elif(direction == "left" and path[-1] != "right" and j!=0):
        newmat[i][j],newmat[i][j-1] = newmat[i][j-1],newmat[i][j]
        newpath = path2 + ["left"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
    elif(direction == "right" and path[-1] != "left" and j!=3):
        newmat[i][j],newmat[i][j+1] = newmat[i][j+1],newmat[i][j]
        newpath = path2 + ["right"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))

```

```

147 #solves the matrix and returns path(list of "left","right",etc) and number of nodes expanded
148 def solve(mat):
149     global dict
150     global q
151     global Current_Node
152
153     dict = {}
154     q = PriorityQueue()
155     Expanded_Nodes = 0
156
157     root = Node(mat,["none"])
158
159     q.put((0, root))
160
161     while not q.empty():
162         Current_Node = q.get()[1]
163         Expanded_Nodes +=1
164         mat = Current_Node.data
165         path = Current_Node.path
166
167         print(mat)
168         print(path)
169
170         weight = getweight(mat)
171         i,j = Find16(mat)
172
173         if(weight == 0 ):
174             break
175         AddBranch(mat,path,i,j,"up")
176         AddBranch(mat,path,i,j,"down")
177         AddBranch(mat,path,i,j,"left")
178         AddBranch(mat,path,i,j,"right")
179
180     return Current_Node.path,Expanded_Nodes
181
182 #uses the base matrix and a list of paths to return a matrix in step number no_state
183 def getstate(mat,path,no_state):
184     newmat = copy.deepcopy(mat)
185     i,j = Find16(newmat)
186
187     for no in range(no_state):
188         if (path[no+1] == "up"):
189             newmat[i][j],newmat[i-1][j] = newmat[i-1][j],newmat[i][j]
190             i = i-1
191         elif(path[no+1] == "down"):
192             newmat[i][j],newmat[i+1][j] = newmat[i+1][j],newmat[i][j]
193             i = i+1
194         elif(path[no+1] == "left"):
195             newmat[i][j],newmat[i][j-1] = newmat[i][j-1],newmat[i][j]
196             j = j-1
197         elif(path[no+1] == "right"):
198             newmat[i][j],newmat[i][j+1] = newmat[i][j+1],newmat[i][j]
199             j = j+1
200
201     return newmat
202

```

GUI.py

```
1  ~ from tkinter import *
2  ~ from BranchAndBound import *
3  ~ import time
4
5
6  ~ #arranges the labels to match mat
7  ~ def display(mat):
8  ~     for i in range(4):
9  ~         for j in range(4):
10 ~             Labels[int(mat[i][j])-1].grid(row=i+1,column=j)
11
12 ~ #updates the display based on the command
13 ~ def play(command):
14 ~     global path
15 ~     global total_steps
16 ~     global state
17 ~     global myLabel17
18
19 ~     if(path == None):
20 ~         return
21 ~     if(command == "first"):
22 ~         state = 1;
23 ~     elif(command == "back"):
24 ~         state = max(1,state-1)
25 ~     elif(command == "next"):
26 ~         state = min(total_steps,state+1)
27 ~     elif(command == "last"):
28 ~         state = total_steps
29 ~     newmat = getstate(mat,path,state-1)
30 ~     myLabel17["text"] = str(state) + " of " + str(total_steps)
31 ~     display(newmat)
32 ~     return
33
```

```

34 #solve the matrix
35 def run(myEntry):
36     global mat
37     global state
38     global total_steps
39     global path
40     global statslabel
41
42     text = myEntry.get()
43     mat = read(text)
44
45     arr = mergelist(mat)
46     ListKurangI = KurangI(arr)
47     KurangIplusx = KurangIplusX(arr)
48     if(isReachable(KurangIplusx)):
49         start_time = time.time()
50         path,expanded = solve(mat)
51         runtime = round((time.time() - start_time),5)
52
53         total_steps = len(path)          #1 to total_steps, not 0 to total
54         state = 1;
55     else:
56         expanded = 0
57         runtime = 0
58         state = 0
59         total_steps = 0
60
61     stats1 = "KurangI = "+str(ListKurangI)+ "\nKurangI + X = " + str(KurangIplusx) + "\nSolveable = "+str(isReachable(KurangIplusx))
62     stats2 = "\nExpanded Nodes: "+str(expanded)+"\nSearch Time: "+str(runtime) + " seconds"
63
64     stats = stats1 + stats2
65     display(mat)
66     myLabel17["text"] = str(state) +" of " + str(total_steps)
67     statslabel["text"] = stats
68
69

```

```

#gui interface
def main():
    root = Tk()
    root.title("15-Puzzle")
    myEntry = Entry(root, text="Enter Filename here", width=40)
    myEntry.grid(row=0, column=0, pady=10, columnspan=3)

    myButton = Button(root, text="Solve", command=lambda: run(myEntry))
    myButton.grid(row=0, column=3, pady=10)

    myLabel1 = Label(root, text="01", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel2 = Label(root, text="02", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel3 = Label(root, text="03", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel4 = Label(root, text="04", height=5, width=10, borderwidth=2, relief="solid", font=(30))

    myLabel5 = Label(root, text="05", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel6 = Label(root, text="06", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel7 = Label(root, text="07", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel8 = Label(root, text="08", height=5, width=10, borderwidth=2, relief="solid", font=(30))

    myLabel9 = Label(root, text="09", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel10 = Label(root, text="10", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel11 = Label(root, text="11", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel12 = Label(root, text="12", height=5, width=10, borderwidth=2, relief="solid", font=(30))

    myLabel13 = Label(root, text="13", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel14 = Label(root, text="14", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel15 = Label(root, text="15", height=5, width=10, borderwidth=2, relief="solid", font=(30))
    myLabel16 = Label(root, text=" ", height=5, width=10, borderwidth=2, relief="solid", font=(30))

    Button_first = Button(root, text="<<", command=lambda: play("first"))
    Button_back = Button(root, text="<", command=lambda: play("back"))
    Button_next = Button(root, text=">", command=lambda: play("next"))
    Button_last = Button(root, text=">>>", command=lambda: play("last"))

    Button_first.grid(row=5, column=0, pady=10, padx=20)
    Button_back.grid(row=5, column=1, pady=10, padx=20)
    Button_next.grid(row=5, column=2, pady=10, padx=20)
    Button_last.grid(row=5, column=3, pady=10, padx=20)

    global path
    path = None
    state = 0;
    total_steps = 0;
    mat = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
    stats = ""

    global myLabel17
    myLabel17 = Label(root, text=str(state) + " of " + str(total_steps))
    myLabel17.grid(row=6, column=0, pady=10, columnspan=4)

    global Labels
    Labels = [myLabel1, myLabel2, myLabel3, myLabel4, myLabel5, myLabel6, myLabel7, myLabel8, myLabel9, myLabel10, myLabel11, myLabel12, myLabel13, myLabel14, myLabel15, myLabel16]

    display(mat)

    global statslabel
    statslabel = Label(root, text=stats)
    statslabel.grid(row=7, column=0, pady=10, padx=20, columnspan=4)
    root.mainloop()

#starts the gui
if __name__ == "__main__":
    main()

```


D. Test Case Files

valid1.txt

1 2 3 4

5 6 16 8

9 10 7 11

13 14 15 12

valid2.txt

13 7 1 16

4 12 8 10

15 14 5 3

6 9 2 11

valid3.txt

16 15 14 13

12 11 10 9

8 7 6 5

4 3 2 1

invalid1.txt

1 3 4 15

2 16 5 12

7 6 11 14

8 9 10 13

invalid2.txt

13 7 1 16

4 12 8 10

14 15 5 3

6 9 2 11

E. Checklist

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	✓	

F. Github Repository

<https://github.com/JayaMangalo/Tucil-3-Stima>