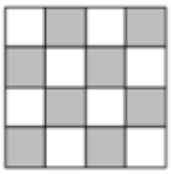
LAPORAN TUCIL 3

ALGORITMA BRANCH AND BOUND

Muhammad Naufal Satriandana (13520068)

A. Algoritma *Branch and Bound*

Program ini adalah program untuk menyelesaikan persoalan 15 puzzle dengan pendekatan branch and bound. Branch and bound adalah algoritma yang memanfaatkan pembangkitan pohon ruang status untuk mencari solusi persoalan. Dalam branch and bound, setiap simpul memiliki cost yang menentukan simpul mana yang akan diekspansi selanjutnya, yang dalam hal ini simpul dengan cost terendah. Dalam program ini, penentuan cost untuk simpul ke i menggunakan rumus c(i) = f(i)+g(i) dimana f(i) merupakan panjang lintasan dari simpul akar ke simpul i, dan g(i) merupakan jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. Namun, penggunaan f(i) untuk beberapa kasus pada implementasi ini akan menyebabkan program berjalan dengan waktu yang sangat lama dan memakan memori yang sangat besar, (bahkan program bisa tidak selesai karena keterbatasan memori), rumus cost yang digunakan adalah c(i) = n*f(i)+g(i), yang dalam hal ini n adalah faktor pengali f yang menunjukan seberapa berpengaruh f terhadap cost. Dalam implementasi ini, n adalah 0. Pemilihan n = 0 dikarenakan program tidak berfokus pada pencarian rute terpendek, melainkan hanya pencarian jawaban saja. Langkahnya sebagai berikut:



Gambar 1. Posisi Ubin Kosong untuk Penentuan Nilai X

1. Program menerima suatu matriks yang merupakan posisi awal puzzle dengan ukuran 4x4
2. Program menentukan apakah dapat dicari solusi akhir dari matriks tersebut dengan menggunakan rumus Sigma(KURANG(i)) + X, yakni penjumlahan nilai kurang(i) dari 0 sampai 16 dan ditambahkan X (bernilai 1 jika sel kosong pada posisi awal ada pada sel yang diarsir pada gambar 1)
3. Jika nilai tersebut ganjil, program menuliskan pesan bahwa persoalan tidak bisa diselesaikan
4. Jika genap, program membangkitkan simpul anak dari pohon tersebut dengan memeriksa 4 arah. Sebagai heuristik, simpul tersebut tidak akan dibangkitkan jika arahnya berkebalikan dengan arah sebelumnya (contohnya atas lalu bawah) dan juga tidak akan dibangkitkan jika gerakan tersebut tidak valid (contohnya ubin kosong di pojok kiri atas digerakkan ke kiri atau ke atas)
5. Untuk setiap simpul yang dibuat, program memasukkan posisi puzzle ke dalam sebuah set agar tidak kembali ke susunan yang sama pada simpul berbeda. Kemudian, jika susunan puzzle simpul tersebut belum ada dalam set, program memasukkan simpul kedalam sebuah senarai terurut berdasarkan cost

6. Program membangkitkan simpul berikutnya (cost terkecil) dalam senarai, mengeluarkannya dari senarai, dan kembali menjalankan langkah 4 hingga ditemukan solusi.

B. Kode program

Program ini terdiri dari dua file, Puzzle.py dan GUI.py.

Puzzle.py:

```
import timeit
import bisect
import numpy as np

class Tree:
    matriksKunjung = set()
    simpulHidup = []
    nodeCount = [0]
    distanceFactor = 0
    def __init__(self, matriks, parent, emptyPosRow, emptyPosCol, direction,
cost, distance):
        self.matriks = matriks
        self.parent = parent
        self.emptyPosRow = emptyPosRow
        self.emptyPosCol = emptyPosCol
        self.direction = direction
        self.cost = cost
        self.distance = distance
        converted = ConvertToString(self.matriks)
        if converted not in self.matriksKunjung:
            bisect.insort_right(self.simpulHidup, self, key=SortFunction)
            self.matriksKunjung.add(converted)
            self.nodeCount[0] += 1
    def Generate(self):
        self.simpulHidup.remove(self)
        if self.direction != "down" and self.emptyPosRow != 0:
            temp = Move(self.matriks, "up", self.emptyPosRow,
self.emptyPosCol)
            self.upChild = Tree(temp, self, self.emptyPosRow-1,
self.emptyPosCol, "up", self.distanceFactor*(self.distance+1)+CountG(temp),
self.distance+1)
        if self.direction != "up" and self.emptyPosRow != 3:
            temp = Move(self.matriks, "down", self.emptyPosRow,
self.emptyPosCol)
            self.downChild = Tree(temp, self, self.emptyPosRow+1,
self.emptyPosCol, "down", self.distanceFactor*(self.distance+1)+CountG(temp),
self.distance+1)
        if self.direction != "right" and self.emptyPosCol != 0:
```

```
            temp = Move(self.matriks, "left", self.emptyPosRow,
self.emptyPosCol)
            self.leftChild = Tree(temp, self, self.emptyPosRow,
self.emptyPosCol-1, "left",
self.distanceFactor*(self.distance+1)+CountG(temp), self.distance+1)
        if self.direction != "left" and self.emptyPosCol != 3:
            temp = Move(self.matriks, "right", self.emptyPosRow,
self.emptyPosCol)
            self.rightChild = Tree(temp, self, self.emptyPosRow,
self.emptyPosCol+1, "right",
self.distanceFactor*(self.distance+1)+CountG(temp), self.distance+1)
    def PrintAll(self, current):
        jawaban = []
        p = current
        while p != None:
            jawaban.append(p)
            p = p.parent
        for i in range(len(jawaban)-1, -1, -1):
            print (jawaban[i].direction)
            print (jawaban[i].matriks)
        return jawaban
    def Search(self):
        while self.simpulHidup[0].cost -
self.simpulHidup[0].distanceFactor*self.simpulHidup[0].distance>0:
            self.simpulHidup[0].Generate()
        return self.simpulHidup[0]

# Functions
def ConvertToString(matriks):
    str = ""
    for i in range(4):
        for j in range(4):
            str += f"{matriks[i][j]:02d}"
    return str
def SortFunction(e):
    return e.cost
def Kurang(i,  dictArg):
    count = 0
    pos = dictArg[i]
    for x in range(i, 0, -1):
        if dictArg[x] > pos:
            count+=1
    return count
def RowIndex(i):
    return (i-1)//4
def ColIndex(i):
    return (i-1)%4
def GetX(dictArg):
```

```
    return (RowIndex(dictArg[16])+ColIndex(dictArg[16]))%2
def IsReachable(dictArg):
    arr = []
    total = 0
    for i in range(1,17):
        kurang = Kurang(i, dictArg)
        print(f"{str(i):2s}","|", kurang)
        arr.append(kurang)
        total += kurang
    sum = (total + GetX(dictArg))
    print()
    print("Sigma(KURANG(i)) + X =",sum)
    print()
    if  sum%2 == 0:
        return True, arr, sum
    else:
        return False, arr, sum
def SetArrayPosisi(matriksAwal):
    dictArg = dict()
    temp = 1
    for x in matriksAwal:
        for y in x:
            dictArg[y] = temp
            temp+=1
    return dictArg
def CountG(matriks):
    count = 0
    for i in range(4):
        for j in range(4):
            if 4*i+j+1 != matriks[i][j] and matriks[i][j] != 16:
                count+=1
    return count
def Move(matriks, direction, emptyPosRow, emptyPosCol):
    matriksCopy = np.copy(matriks)
    i = emptyPosRow
    j = emptyPosCol
    if direction == "up" and i != 0:
        matriksCopy[i-1][j],matriksCopy[i][j] =
matriksCopy[i][j],matriksCopy[i-1][j]
    elif direction == "down" and i != 3:
        matriksCopy[i+1][j],matriksCopy[i][j] =
matriksCopy[i][j],matriksCopy[i+1][j]
    elif direction == "left" and j != 0:
        matriksCopy[i][j-1],matriksCopy[i][j] =
matriksCopy[i][j],matriksCopy[i][j-1]
    elif direction == "right" and j != 3:
        matriksCopy[i][j+1],matriksCopy[i][j] =
matriksCopy[i][j],matriksCopy[i][j+1]
```

```
    return matriksCopy
def GetMatriksAwal(filename):
    try:
        return np.genfromtxt(filename).astype(int)
    except:
        return np.genfromtxt('..\\test\\' + filename).astype(int)

def Run(matriksAwal):
    jawaban = []
    print()
    print("Matriks Posisi Awal:")
    print(matriksAwal)
    start = timeit.default_timer()
    dictPosisi = SetArrayPosisi(matriksAwal)
    print(dictPosisi)
    print("i   | Kurang(i):")
    isReachable, arr, sum = IsReachable(dictPosisi)
    if isReachable:
        tree = Tree(matriksAwal, None, RowIndex(dictPosisi[16]),
ColIndex(dictPosisi[16]), "neutral", 1, 0)
        answerNode = tree.Search()
        duration = timeit.default_timer()-start
        jawaban = tree.PrintAll(answerNode)
        print("Total simpul:", tree.nodeCount[0])
        print("Total langkah:", len(jawaban)-1)
        print("Durasi Menemukan Simpul Jawaban =", duration, "detik")
    else:
        duration = timeit.default_timer()-start
        print("Persoalan tidak bisa diselesaikan")
    Tree.matriksKunjung = set()
    Tree.simpulHidup = []
    totalSimpul = Tree.nodeCount[0]
    Tree.nodeCount = [0]
    print("Durasi Total =", timeit.default_timer()-start, "detik")
    return jawaban, arr, sum, duration, totalSimpul
# main
def main():
    matriksAwal = []
    filename = input("Masukkan nama file: ")
    matriksAwal = GetMatriksAwal(filename)
    Run(matriksAwal)
```

GUI.py:

```
import tkinter as tk
from Puzzle import *
root = tk.Tk()
```

```
root.columnconfigure([0,1], weight=1)
root.rowconfigure([0,1], weight=1)
iterator = 0
jawaban = []
matriksAwal = []
def InitMatriks():
    try:
        global jawaban
        global matriksAwal
        matriksAwal = GetMatriksAwal(entry.get())
        jawaban = []
        buttonNext['state'] = 'disabled'
        buttonPrev['state'] = 'disabled'
        buttonRewind['state'] = 'disabled'
        buttonFastForward['state'] = 'disabled'
        buttonNext.grid(row=0, column=3,rowspan=4, sticky="nsew")
        buttonPrev.grid(row=0, column=1, rowspan=4, sticky="nsew")
        buttonFastForward.grid(row=0, column=4,rowspan=4, sticky="nsew")
        buttonRewind.grid(row=0, column=0, rowspan=4, sticky="nsew")


        for widget in matriksFrame.winfo_children():
            widget.destroy()
        for i in range(4):
            for j in range(4):
                relief = "raised"
                number = matriksAwal[i][j]
                if number == 16:
                    number = " "
                    relief = "sunken"
                label = tk.Label(master=matriksFrame, relief=relief,
borderwidth=10, text=number, font=('bold',50))
                label.grid(row=i, column=j,sticky="nsew")
        buttonSolve = tk.Button(master=inputFrame, text="Solve",
relief="raised", borderwidth=5, command=Solve)
        buttonSolve.grid(row=3, column=2,sticky="nsew")
    except:
        print("File tidak ditemukan")
def Update():
    global matriksFrame
    for i in range(4):
        for j in range(4):
            relief = "raised"
            number = jawaban[iterator].matriks[i][j]
            if number == 16:
                number = " "
                relief = "sunken"
            matriksFrame.winfo_children()[4*i+j]["text"] = number
```

```
            matriksFrame.winfo_children()[4*i+j]["relief"] = relief

def Next():
    global iterator
    if iterator > 0:
        iterator -= 1
        buttonPrev['state'] = 'active'
        buttonRewind['state'] = 'active'
    if iterator <= 0:
        buttonNext['state'] = 'disabled'
        buttonFastForward['state'] = 'disabled'
    Update()
def Prev():
    global iterator
    if iterator < len(jawaban)-1:
        iterator += 1
        buttonNext['state'] = 'active'
        buttonFastForward['state'] = 'active'
    if iterator >= len(jawaban)-1:
        buttonPrev['state'] = 'disabled'
        buttonRewind['state'] = 'disabled'
    Update()
def FastForward():
    buttonFastForward['state'] = 'disabled'
    if iterator > 0:
        Next()
        root.after(100, FastForward)
def Rewind():
    buttonRewind['state'] = 'disabled'
    if iterator < len(jawaban)-1:
        Prev()
        root.after(100, Rewind)
def Solve():
    global jawaban
    global iterator
    jawaban, arr, sum, duration, totalSimpul = Run(matriksAwal)
    iterator = len(jawaban)-1
    if iterator > 0:
        buttonNext['state'] = 'normal'
        buttonFastForward['state'] = 'normal'
        buttonPrev['state'] = 'disabled'
        buttonRewind['state'] = 'disabled'
    for widget in detailFrame.winfo_children():
        widget.destroy()
    tableHeadLeft = tk.Label(master=detailFrame, text = 'i', relief="groove",
borderwidth=2)
    tableHeadLeft.grid(row=0, column=0, sticky="nsew")
```

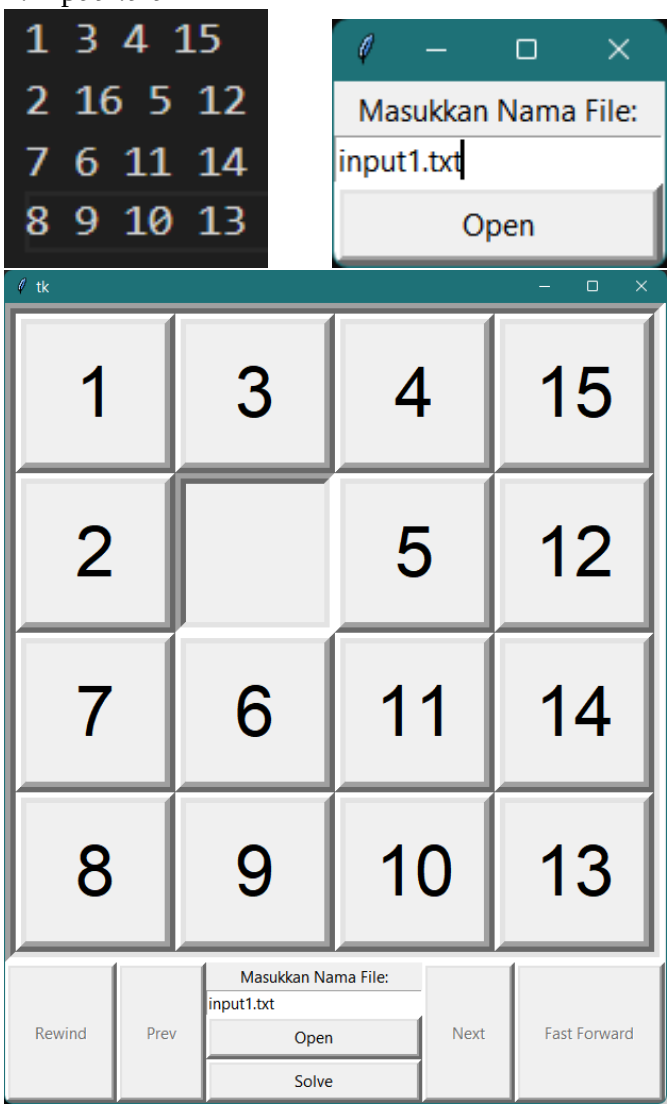
```
    tableHeadRight = tk.Label(master=detailFrame, text = 'Kurang(i):',
relief="groove", borderwidth=2)
    tableHeadRight.grid(row=0, column=1, sticky="nsew")
    for i in range(16):
        detailLabelIndex = tk.Label(master=detailFrame, text = i+1,
relief="groove", borderwidth=1)
        detailLabelIndex.grid(row=i+1, column=0, sticky="nsew")
        detailLabelValue = tk.Label(master=detailFrame, text = arr[i],
relief="groove", borderwidth=1)
        detailLabelValue.grid(row=i+1, column=1, sticky="nsew")
    total = tk.Label(master=detailFrame, text = "Sigma(KURANG(i)) + X =
"+str(sum), relief="groove", borderwidth=1)
    total.grid(row=17, column=0, columnspan= 2, sticky="nsew")
    durationLabel = tk.Label(master=detailFrame, text = "Durasi =
"+str(duration), relief="groove", borderwidth=1)
    durationLabel.grid(row=18, column=0, columnspan= 2, sticky="nsew")
    simpulText = "Total Simpul = "+str(totalSimpul)
    if totalSimpul <= 0:
        simpulText = "Persoalan tidak bisa diselesaikan"
    simpulLabel = tk.Label(master=detailFrame, text = simpulText,
relief="groove", borderwidth=1)
    simpulLabel.grid(row=19, column=0, columnspan= 2, sticky="nsew")

# inputFrame
inputFrame = tk.Frame(master=root)
inputFrame.grid(row=1, column=0, sticky="nsew")
inputFrame.columnconfigure([0,1,2,3,4], weight=1)
inputFrame.rowconfigure([0,1,2,3], weight=1)
inputLabel = tk.Label(master=inputFrame, text="Masukkan Nama File:")
inputLabel.grid(row=0, column=2,sticky="nsew")
entry = tk.Entry(master=inputFrame)
entry.grid(row=1, column=2,sticky="nsew")
buttonOpen = tk.Button(master=inputFrame, text="Open", relief="raised",
borderwidth=5, command=InitMatriks)
buttonOpen.grid(row=2, column=2,sticky="nsew")
buttonNext = tk.Button(master=inputFrame, text="Next", relief="raised",
borderwidth=5,command=Next, state="disabled")
buttonPrev = tk.Button(master=inputFrame, text="Prev", relief="raised",
borderwidth=5, command=Prev, state="disabled")
buttonFastForward = tk.Button(master=inputFrame, text="Fast Forward",
relief="raised", borderwidth=5,command=lambda:FastForward(), state="disabled")
buttonRewind = tk.Button(master=inputFrame, text="Rewind", relief="raised",
borderwidth=5, command=lambda:Rewind(), state="disabled")

# matriksFrame
matriksFrame = tk.Frame(master=root, relief="sunken", borderwidth=10)
matriksFrame.grid(row=0, column=0)
matriksFrame.rowconfigure([0,1,2,3],minsize=150,weight=1)
```

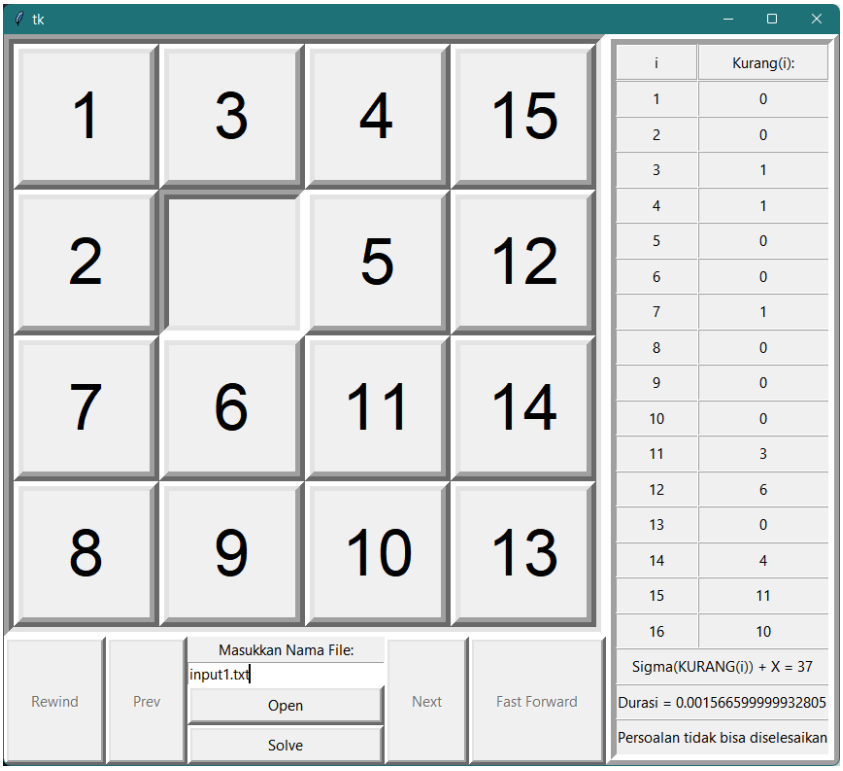
```
matriksFrame.columnconfigure([0,1,2,3],minsize=150,weight=1)

# detailFrame
detailFrame = tk.Frame(master=root, relief="ridge", borderwidth=10)
detailFrame.grid(row=0, column=1, rowspan=2, sticky="nsew")
detailFrame.rowconfigure([i for i in range(20)],weight=1)
detailFrame.columnconfigure([0,1],weight=1)

root.mainloop()
```

C. Skrinshut Input dan Output

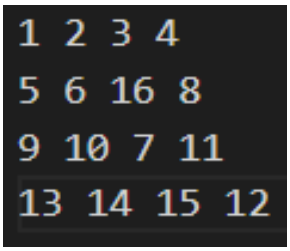
Input yang digunakan merupakan file .txt dengan format tertentu yang merepresentasikan matriks 4x4 dengan angka 16 merepresentasikan ubin kosong

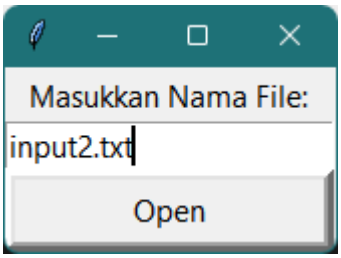
1. input1.txt

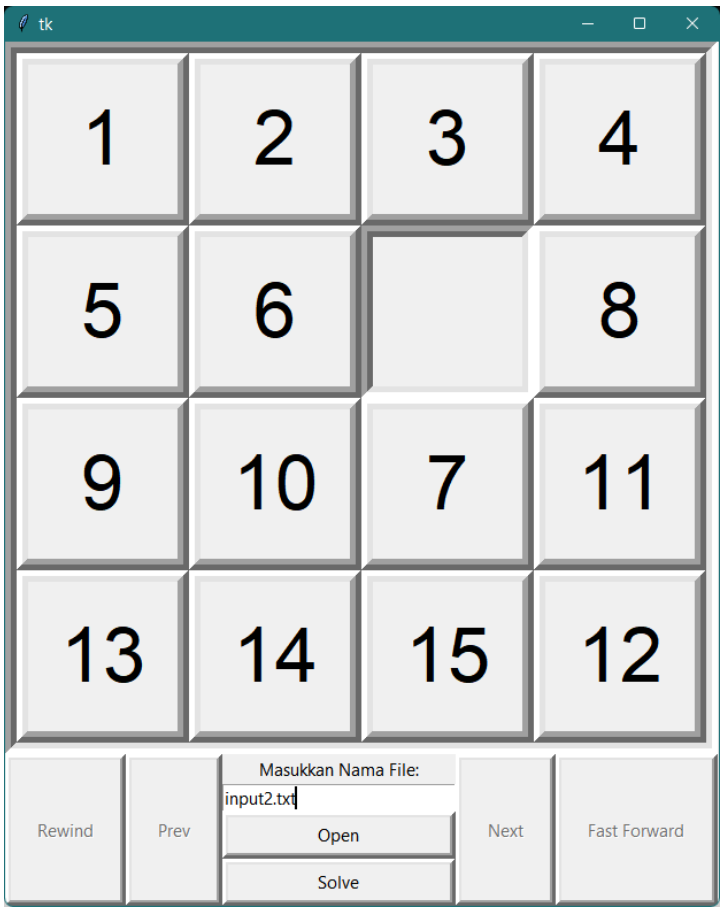


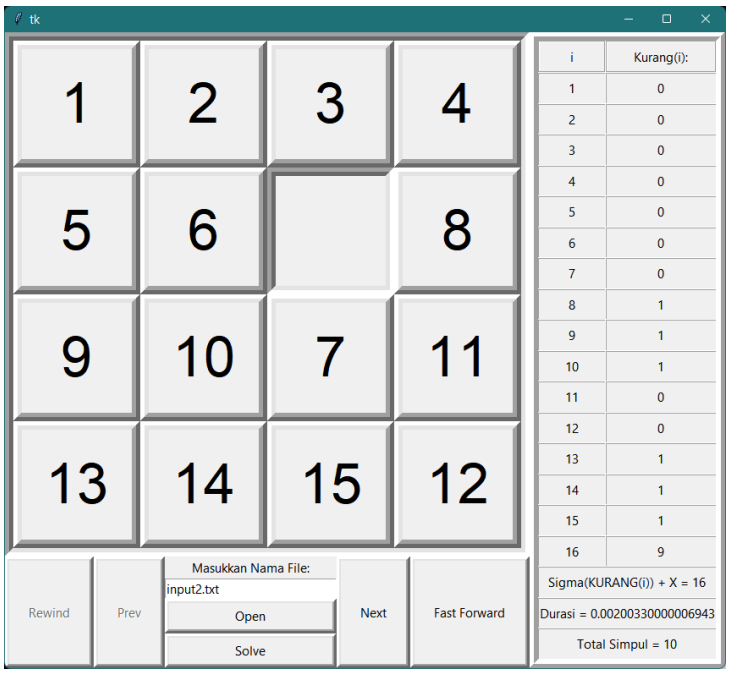

tk
1 3 4 15
2 5 12
7 6 11 14
8 9 10 13
Rewind
Prev
Masukkan Nama File:
input1.txt
Open
Solve
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 1
4 1
5 0
6 0
7 1
8 0
9 0
10 0
11 3
12 6
13 0
14 4
15 11
16 10
Sigma(KURANG(i)) + X = 37
Durasi = 0.001566599999932805
Persoalan tidak bisa diselesaikan

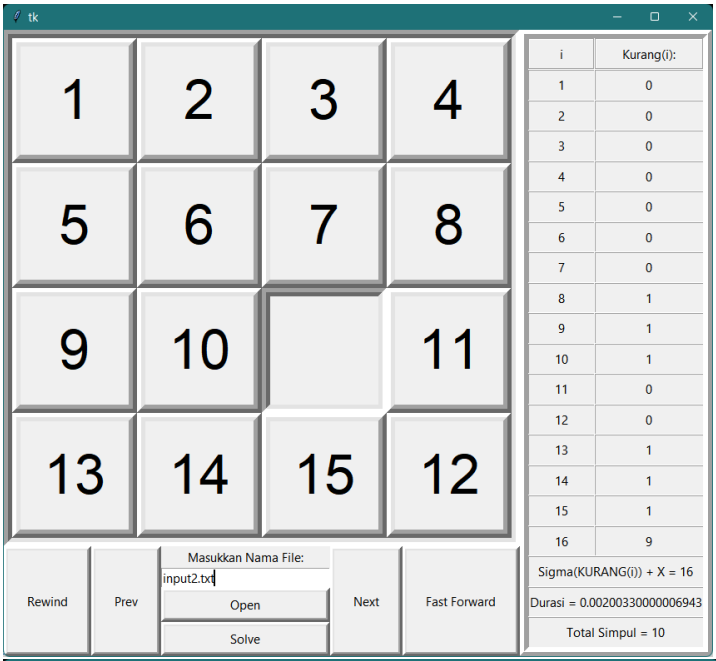
2. input2.txt

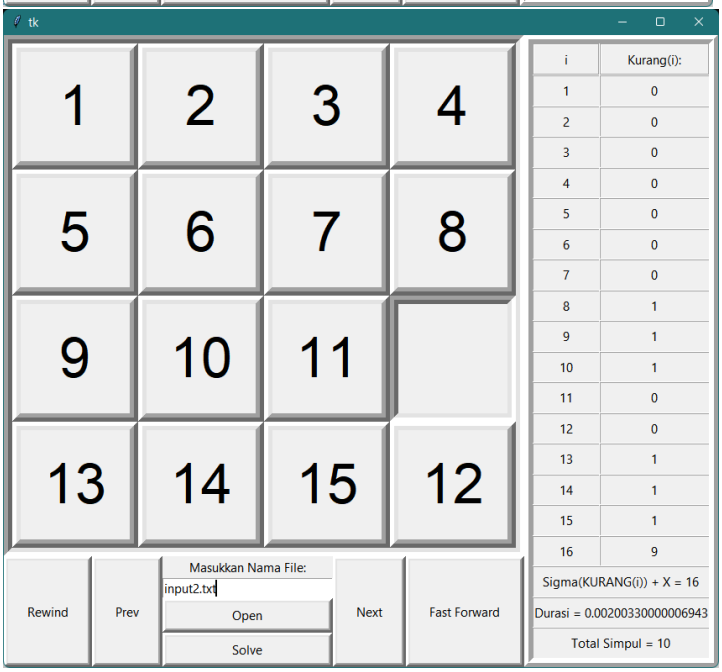

1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

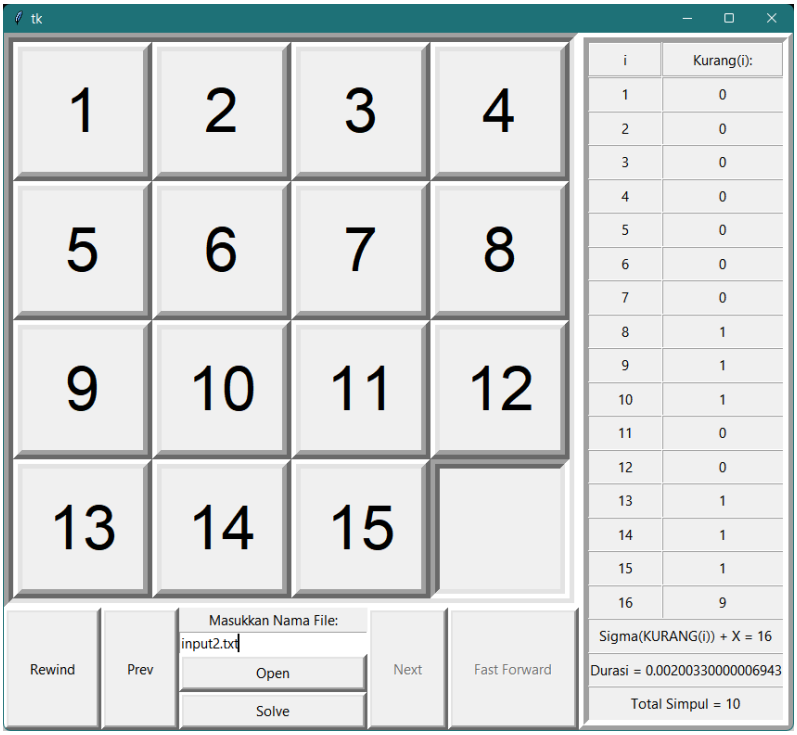

Masukkan Nama File:
input2.txt
Open


tk
1
2
3
4
5
6
8
9
10
7
11
13
14
15
12
Rewind
Prev
Masukkan Nama File:
input2.txt
Open
Solve
Next
Fast Forward

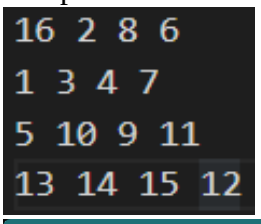

tk
1
2
3
4
5
6
8
9
10
7
11
13
14
15
12
Rewind
Prev
Masukkan Nama File:
input2.txt
Open
Solve
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 1
9 1
10 1
11 0
12 0
13 1
14 1
15 1
16 9
Sigma(KURANG(i)) + X = 16
Durasi = 0.00200330000006943
Total Simpul = 10

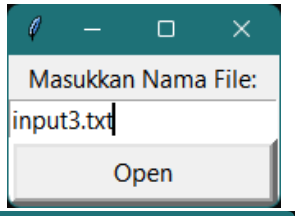

tk
1
2
3
4
5
6
7
8
9
10
11
13
14
15
12
Rewind
Prev
Masukkan Nama File:
input2.txt
Open
Solve
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 1
9 1
10 1
11 0
12 0
13 1
14 1
15 1
16 9
Sigma(KURANG(i)) + X = 16
Durasi = 0.00200330000006943
Total Simpul = 10

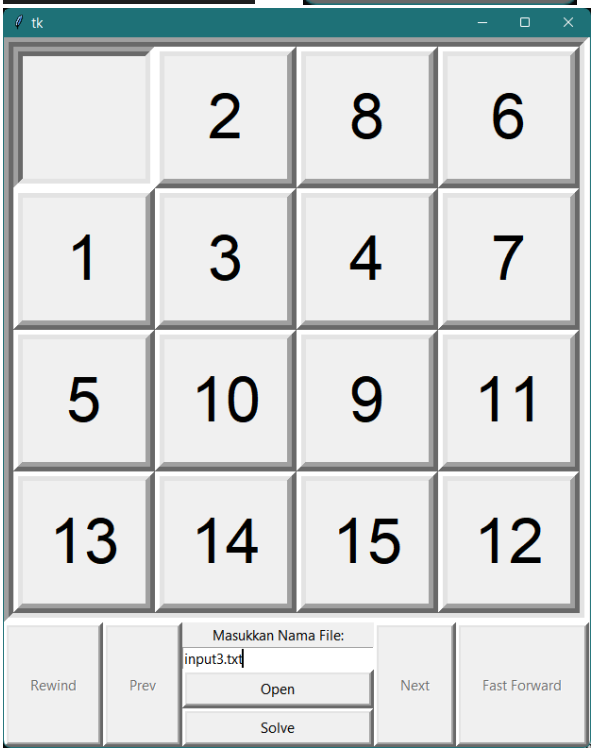

tk
1
2
3
4
5
6
7
8
9
10
11
13
14
15
12
Rewind
Prev
Masukkan Nama File:
input2.txt
Open
Solve
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 1
9 1
10 1
11 0
12 0
13 1
14 1
15 1
16 9
Sigma(KURANG(i)) + X = 16
Durasi = 0.00200330000006943
Total Simpul = 10

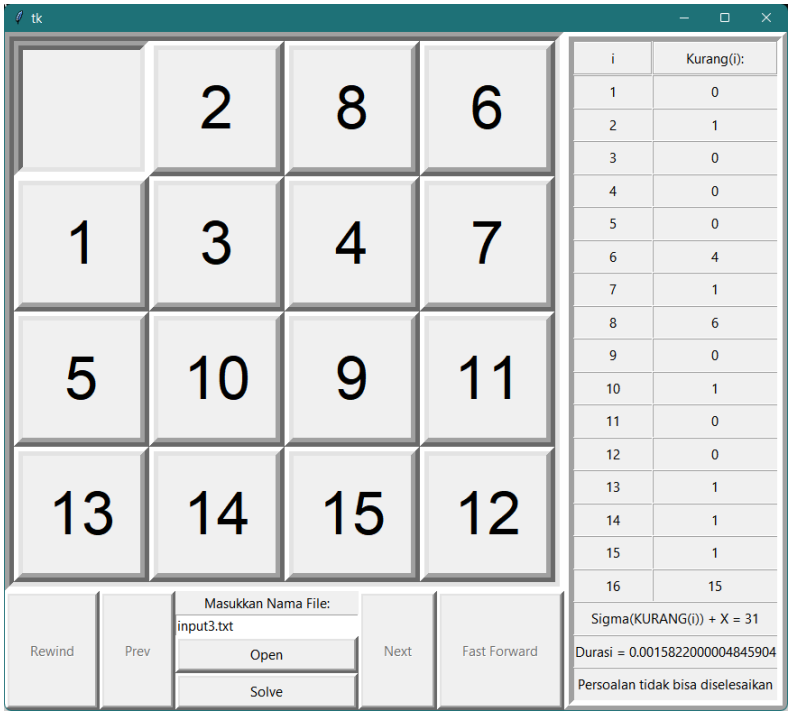

tk
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15
i
Kurang(i):
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 1
9 1
10 1
11 0
12 0
13 1
14 1
15 1
16 9
Sigma(KURANG(i)) + X = 16
Durasi = 0.00200330000006943
Total Simpul = 10
Rewind
Prev
Masukkan Nama File:
input2.txt
Open
Solve
Next
Fast Forward

3. input3.txt


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

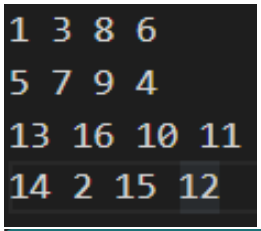

Masukkan Nama File:
input3.txt
Open

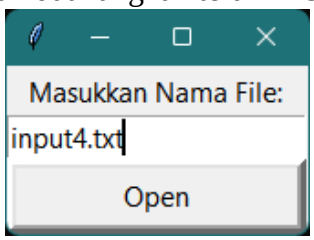

tk
2 8 6
1 3 4 7
5 10 9 11
13 14 15 12
Rewind
Prev
Masukkan Nama File:
input3.txt
Open
Solve
Next
Fast Forward

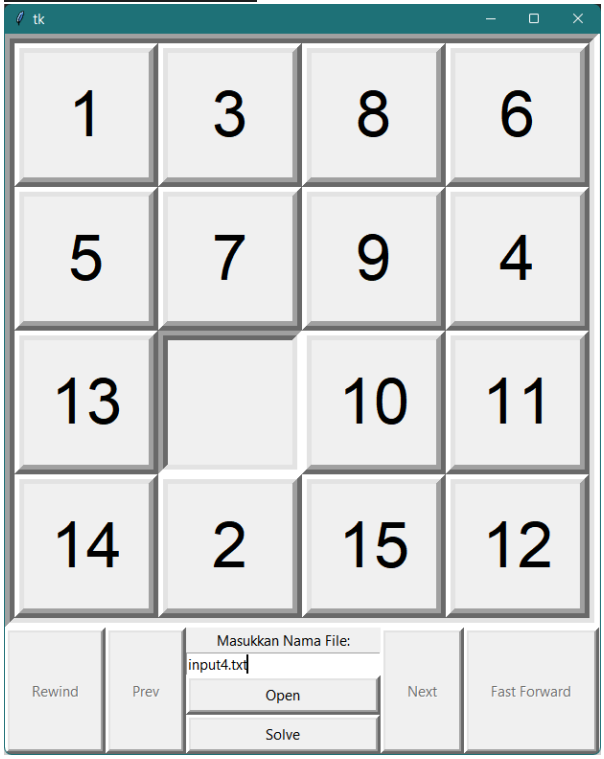


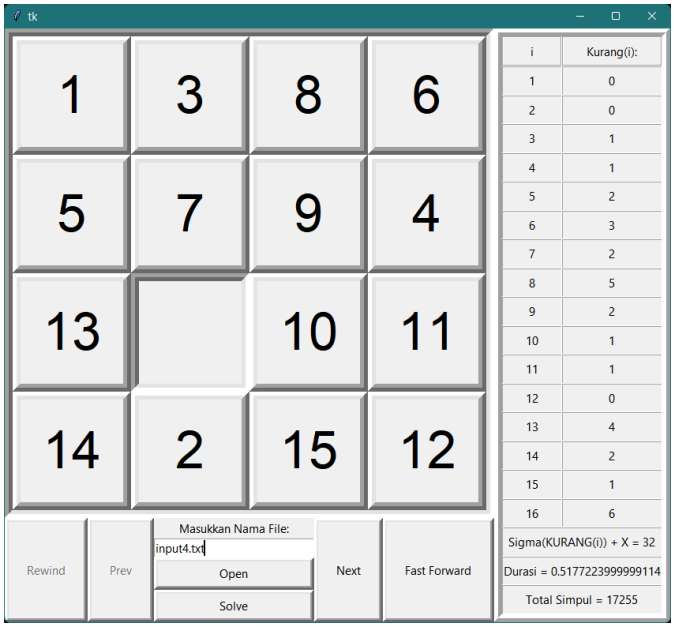
4. input4.txt

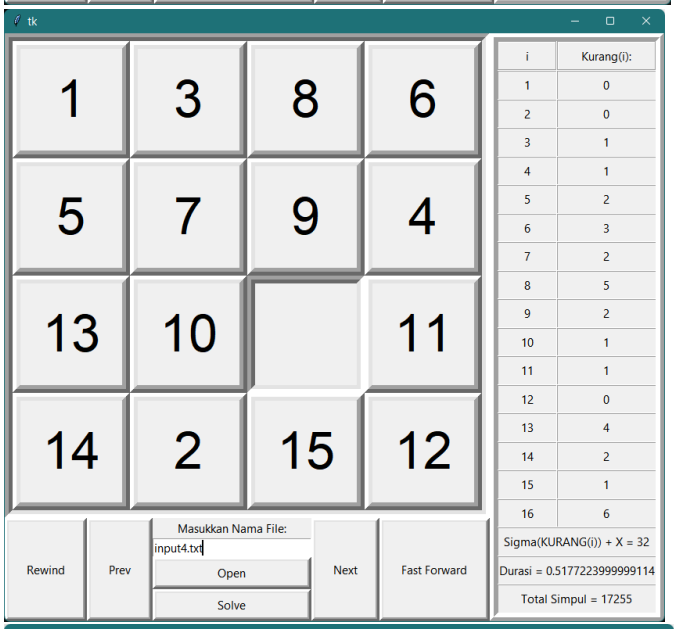
Karena penyelesaian input4.txt memiliki 147 langkah, maka hanya akan ditampilkan dua langkah pertama dan dua langkah terakhir saja.

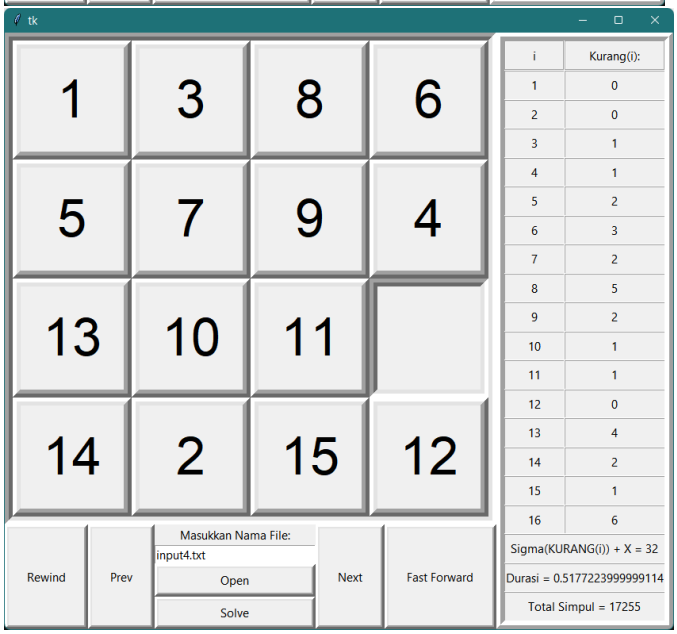




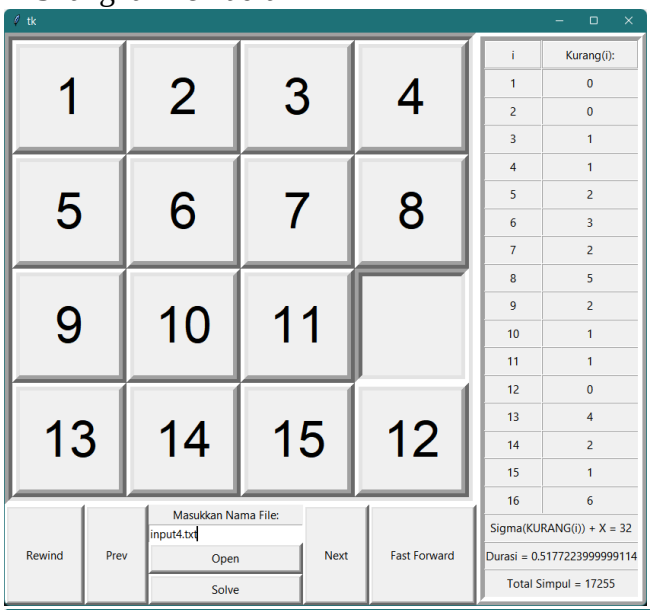


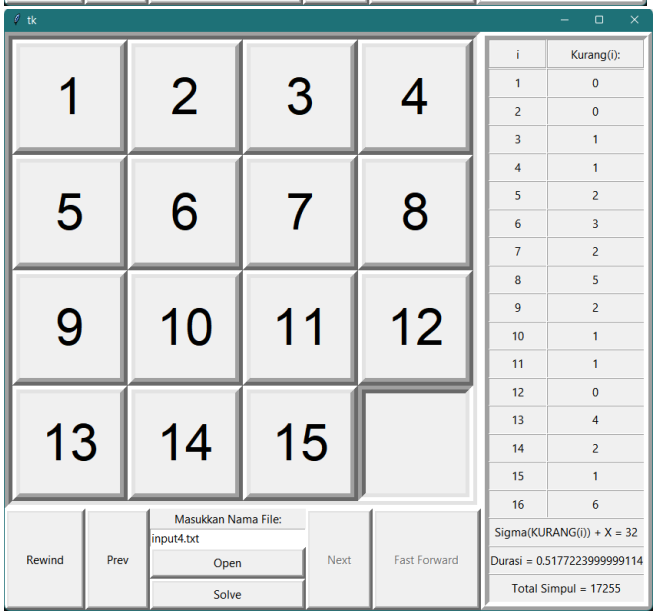

tk
1 3 8 6
5 7 9 4
13 10 11
14 2 15 12
Masukkan Nama File:
input4.txt
Open
Solve
Rewind
Prev
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 1
4 1
5 2
6 3
7 2
8 5
9 2
10 1
11 1
12 0
13 4
14 2
15 1
16 6
Sigma(KURANG(i)) + X = 32
Durasi = 0.5177223999999114
Total Simpul = 17255


tk
1 3 8 6
5 7 9 4
13 10 11
14 2 15 12
Masukkan Nama File:
input4.txt
Open
Solve
Rewind
Prev
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 1
4 1
5 2
6 3
7 2
8 5
9 2
10 1
11 1
12 0
13 4
14 2
15 1
16 6
Sigma(KURANG(i)) + X = 32
Durasi = 0.5177223999999114
Total Simpul = 17255


tk
1 3 8 6
5 7 9 4
13 10 11
14 2 15 12
Masukkan Nama File:
input4.txt
Open
Solve
Rewind
Prev
Next
Fast Forward
i
Kurang(i):
1 0
2 0
3 1
4 1
5 2
6 3
7 2
8 5
9 2
10 1
11 1
12 0
13 4
14 2
15 1
16 6
Sigma(KURANG(i)) + X = 32
Durasi = 0.5177223999999114
Total Simpul = 17255

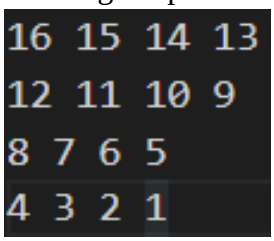
145 langkah kemudian

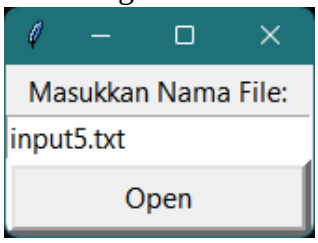


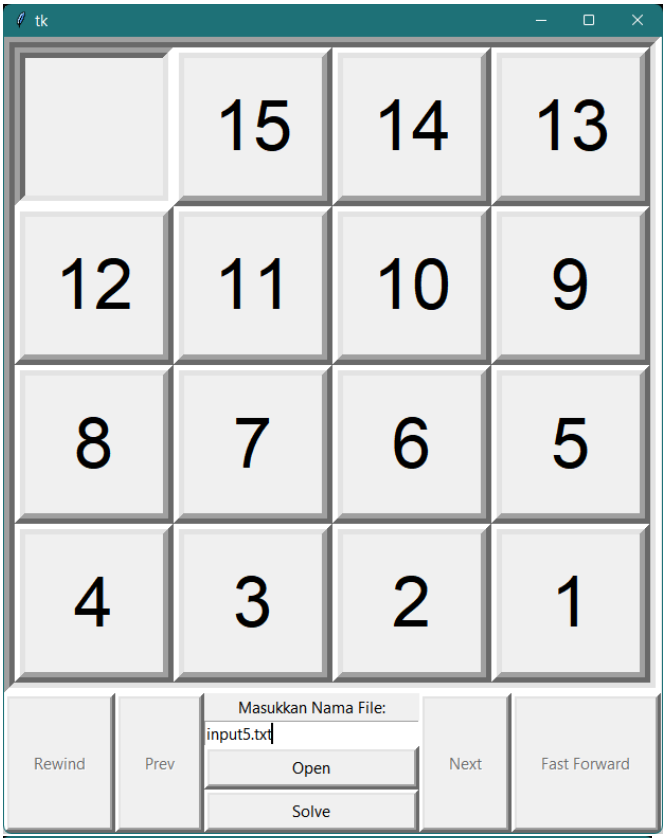


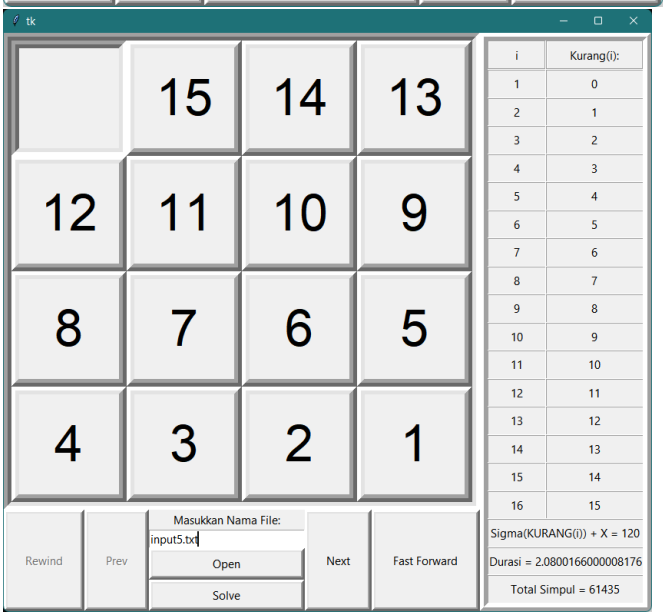
5. input5.txt

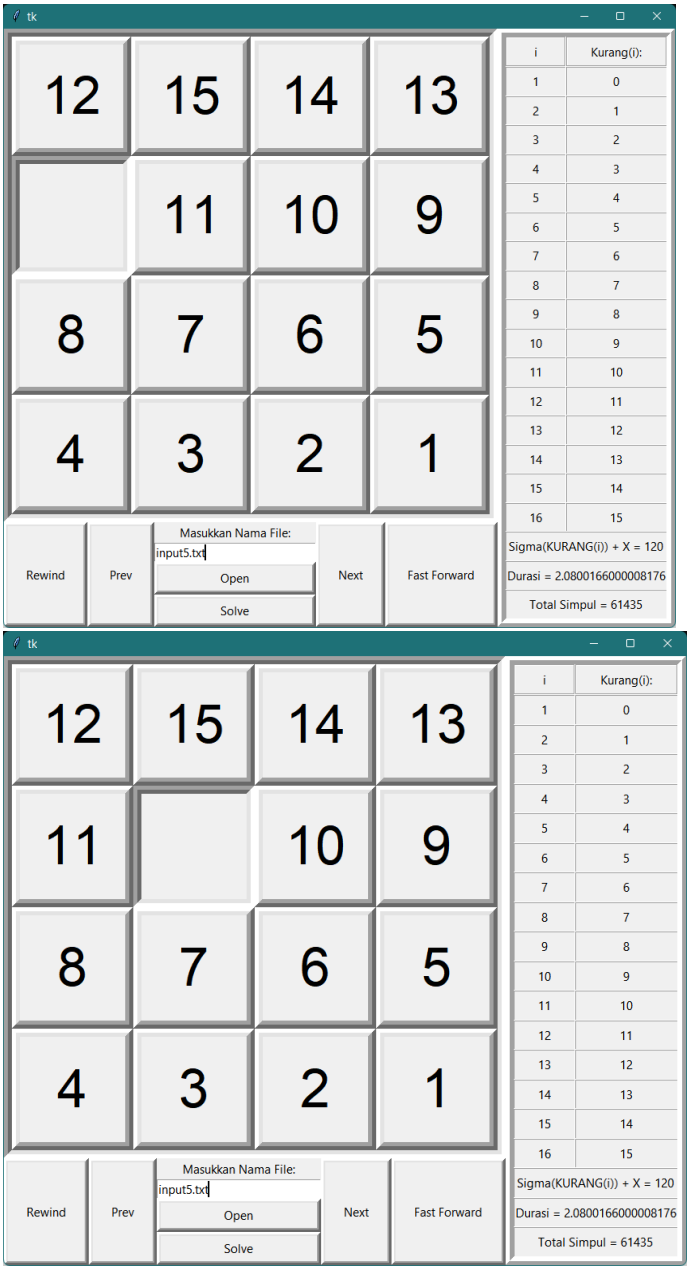
Karena penyelesaian input5.txt memiliki 144 langkah, maka hanya akan ditampilkan dua langkah pertama dan dua langkah terakhir saja.



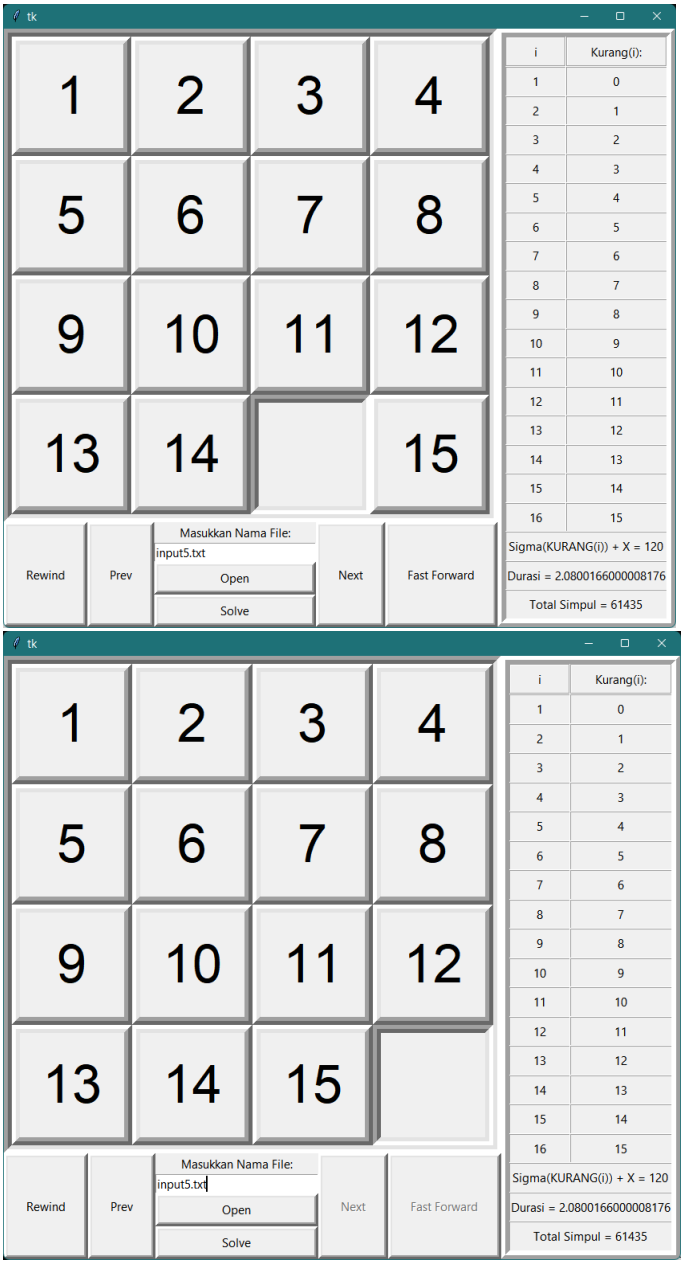



tk
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Rewind
Prev
Masukkan Nama File:
input5.txt
Open
Solve
Next
Fast Forward


tk
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Rewind
Prev
Masukkan Nama File:
input5.txt
Open
Solve
Next
Fast Forward
i
Kurang(i):
1 0
2 1
3 2
4 3
5 4
6 5
7 6
8 7
9 8
10 9
11 10
12 11
13 12
14 13
15 14
16 15
Sigma(KURANG(i)) + X = 120
Durasi = 2.0800166000008176
Total Simpul = 61435



142 langkah kemudian



D. Link Google Drive
https://drive.google.com/drive/folders/1grK943NXEs4ED-GKeqjAikx_1IwMZPJD?usp=sharing
https://github.com/FaLzNaufal/Tucil3_13520068

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	√	