Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

DIBUAT OLEH

William Manuel Kurniawan

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

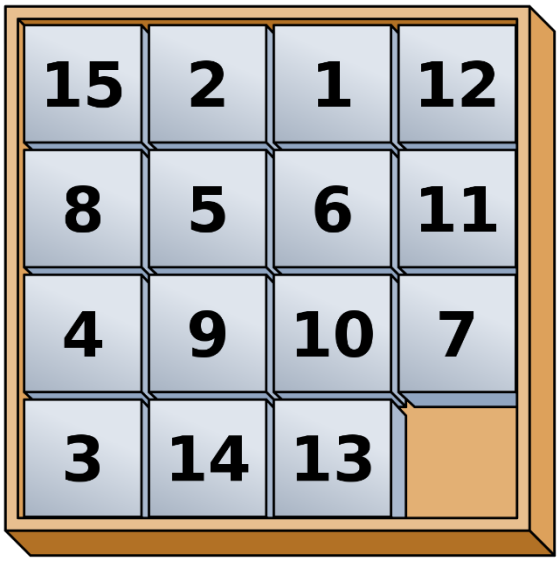
INSTITUT TEKNOLOGI BANDUNG

Semester II Tahun 2021/2022

Daftar Isi

Bab 1 Pendahuluan

15-Puzzle merupakan sebuah permainan dimana pemain harus menggeser kotak kosong yang ada pada permainan. Ketika menggeser kotak kosong, kotak berisi angka akan bertukar tempat dengan kotak yang kosong sehingga berada pada posisi yang baru. Berikut adalah tampilan permainan 15-Puzzle.



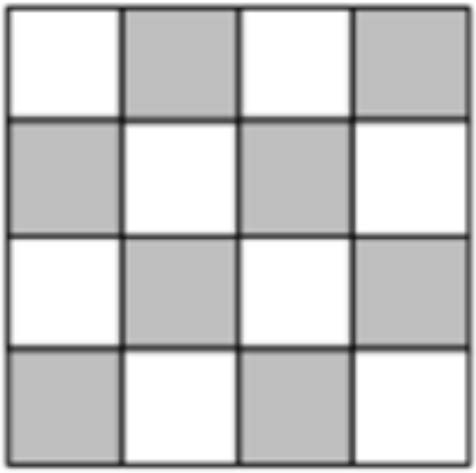
Gambar 1.1 Gambar permainan 15-Puzzle

Tujuan akhir dalam permainan adalah mengubah posisi angka yang ada sehingga terurut dengan nilai 1 dimulai dari kiri atas, nilai 2 berada di sebelah kanan nilai 1, sampai kotak kosong (dianggap sebagai nilai 16) berada pada posisi di kanan bawah. Dalam permainan 15-Puzzle, ada puzzle yang tidak dapat diselesaikan.

Algoritma *branch and bound* merupakan sebuah algoritma pencarian yang digunakan untuk persoalan optimisasi. Optimisasi dilakukan dengan memberi cost untuk setiap simpul data yang akan dicari. Dalam program, algoritma *branch and bound* dibuat menggunakan bantuan priority queue yang berisi cost dari simpul yang diperiksa. Program akan memeriksa simpul yang memiliki cost paling besar / paling kecil dalam priority queue dan menyimpan simpul hasil dari pemeriksaan ke dalam priority queue tersebut. Pemeriksaan akan terus dilakukan sampai simpul solusi ditemukan.

Bab 2 Teori Dasar Cara Kerja Program

Sebelum mencoba menyelesaikan 15-Puzzle, perlu ditentukan apakah puzzle dapat diselesaikan atau tidak. Untuk menentukan hal tersebut, dilakukan perhitungan berdasarkan nilai dan posisi kotak dalam 15-Puzzle. Perhitungan dilakukan dengan melihat nilai kotak tersebut dan melihat jika ada kotak lain setelah kotak tersebut yang memiliki nilai lebih kecil. Jika ada, maka skor untuk diperiksa ditambahkan 1. Kotak kosong juga diperiksa dengan asumsi kotak kosong memiliki nilai 16. Setelah semua kotak diperiksa, Akan diperiksa posisi kotak kosong berdasarkan gambar di bawah ini.



Gambar 2.1 Gambar kotak untuk memeriksa kotak kosong dalam permainan 15-Puzzle

Jika kotak kosong berada di posisi yang sama dengan kotak yang diarsir, maka skor untuk diperiksa ditambahkan 1. Jika nilai akhir dari skor adalah ganjil, maka puzzle tidak dapat diselesaikan tetapi jika nilai akhir dari skor adalah genap, maka puzzle dapat diselesaikan. Jika puzzle dapat diselesaikan, maka tahap selanjutnya adalah menyelesaikan puzzle tersebut. Untuk menyelesaikan puzzle, dibuat priority queue berdasarkan cost dari simpul yang diperiksa dan program akan memeriksa simpul berdasarkan simpul yang memiliki cost paling kecil. Cost dari simpul didapatkan berdasarkan gerakan yang dapat dilakukan pada kotak kosong, jumlah kotak selain kotak kosong yang tidak berada pada tempat yang sesuai, serta berapa langkah yang sudah dilakukan. Jika pada kotak kosong tidak dapat dilakukan gerakan tertentu, maka simpul hasil dari gerakan tersebut akan diberi cost yang sangat besar. Selanjutnya, untuk setiap kotak selain kotak kosong yang tidak berada pada posisinya, maka ditambahkan 1 pada cost dari simpul tersebut. Jumlah langkah yang sudah dilakukan dari simpul tersebut akan ditambahkan pada cost akhir. Program akan memeriksa simpul yang memiliki cost paling kecil

lalu memasukkan simpul hasil pemeriksaan serta cost simpul tersebut ke dalam priority queue untuk diperiksa serta menghapus simpul yang diperiksa dari queue. Program terus memeriksa simpul sampai ditemukan simpul yang merupakan simpul solusi.

Bab 3 Source Program

Algoritma penyelelsaian 15-Puzzle dibuat menggunakan bahasa pemrograman Python yang akan menampilkan skor yang menentukan apakah puzzle bisa diselesaikan, susunan puzzle awal, setiap gerakan yang dilakukan beserta gambar puzzle setelah gerakan, jumlah langkah yang dibutuhkan, waktu yang dibutuhkan, dan jumlah simpul yang dibangkitkan. Berikut adalah isi dari file puzzleFunction.py

```
def isDone(arrData):
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            checkResult = i * 4 + j + 1
            if (arrData[i][j] != checkResult) :
                return False
            j += 1
        i += 1
    return True

def findEmpty(arrData):
    resultTemp = [-1, -1]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            if (arrData[i][j] == 16) :
                result = [i, j]
                return result
            j += 1
        i += 1
    return resultTemp

def findScore(arrData, level):
    finalResult = 0
    position = findEmpty(arrData)
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            if (i != position[0] or j != position[1]) :
```

```
                if (arrData[i][j] != i * 4 + j + 1) :
                    finalResult += 1
            j += 1
        i += 1
    finalResult += level
    return finalResult

def moveLeftScore(arrData, level):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[1] > 0) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0]][position[1]
- 1]
        arrData2[position[0]][position[1] - 1] = saveData
        score = findScore(arrData2, level)
        return score
    else :
        return 99999

def moveRightScore(arrData, level):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[1] < 3) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0]][position[1]
+ 1]
        arrData2[position[0]][position[1] + 1] = saveData
        score = findScore(arrData2, level)
        return score
    else :
        return 99999

def moveUpScore(arrData, level):
```

```
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[0] > 0) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0] -
1][position[1]]
        arrData2[position[0] - 1][position[1]] = saveData
        score = findScore(arrData2, level)
        return score
    else :
        return 99999

def moveDownScore(arrData, level):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[0] < 3) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0] +
1][position[1]]
        arrData2[position[0] + 1][position[1]] = saveData
        score = findScore(arrData2, level)
        return score
    else :
        return 99999

def moveLeft(arrData):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
```

```
    position = findEmpty(arrData2)
    if (position[1] > 0) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0]][position[1]
- 1]
        arrData2[position[0]][position[1] - 1] = saveData
    return arrData2

def moveRight(arrData):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[1] < 3) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0]][position[1]
+ 1]
        arrData2[position[0]][position[1] + 1] = saveData
    return arrData2

def moveUp(arrData):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[0] > 0) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0] -
1][position[1]]
        arrData2[position[0] - 1][position[1]] = saveData
    return arrData2

def moveDown(arrData):
    arrData2 = [[0] * (4) for _ in range(4)]
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
```

```
            arrData2[i][j] = arrData[i][j]
            j += 1
        i += 1
    position = findEmpty(arrData2)
    if (position[0] < 3) :
        saveData = arrData2[position[0]][position[1]]
        arrData2[position[0]][position[1]] = arrData2[position[0] +
1][position[1]]
        arrData2[position[0] + 1][position[1]] = saveData
    return arrData2

def readFileInput(filename) :
    arrData2 = [[0] * (4) for _ in range(4)]
    f = open(filename, "r")
    for i in range (0,4):
        line = f.readline()
        arrLine = line.split()
        for j in range (0,4):
            if (arrLine[j] == "-"):
                arrData2[i][j] = 16
            else:
                arrData2[i][j] = int(arrLine[j])

    return arrData2

def printArr(arrData):
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
            if (arrData[i][j] == 16):
                print("- ", end = "")
            else:
                print(str(arrData[i][j]) + " ", end ="")
            j += 1
        print()
        i += 1
    print()

def isRun(arrData):
    simpanScore = [0] * (16)
    countRun = 0
    positionXStart = -1
    positionYStart = -1
    i = 0
    while (i < 4) :
        j = 0
        while (j < 4) :
```

```
            checkScore = 0
            if (arrData[i][j] == 16) :
                positionYStart = i + 1
                positionXStart = j + 1
            m = j
            while (m < 4) :
                if (arrData[i][m] < arrData[i][j]) :
                    checkScore += 1
                m += 1
            k = i + 1
            while (k < 4) :
                l = 0
                while (l < 4) :
                    if (arrData[k][l] < arrData[i][j]) :
                        checkScore += 1
                    l += 1
                k += 1
            countRun += checkScore
            simpanScore[arrData[i][j]-1] = checkScore
            j += 1
        i += 1
    X = 0
    if ((positionXStart % 2 == 0 and positionYStart % 2 == 1) or
(positionXStart % 2 == 1 and positionYStart % 2 == 0)) :
        countRun += 1
        X = 1
    print("Nilai Kurang(i): ")
    for i in range (0, 16):
        print("Dari " + str(i+1) + ": " + str(simpanScore[i]))
    print("Nilai X: " + str(X))
    print("Nilai Akhir: " + str(countRun))
    print()
    if (countRun % 2 == 0) :
        return True
    else :
        return False

def addDataQueue(arrDataQueue, score, levelQueue, lastCommandTemp, idTemp):
    i = 0
    arrDataQueue2 = []
    while (i < len(arrDataQueue) and arrDataQueue[i][0] < score):
        if (arrDataQueue[i] not in arrDataQueue2):
            arrDataQueue2.append(arrDataQueue[i])
        i += 1

    listInsert = [score,levelQueue,lastCommandTemp,idTemp]
    if (listInsert not in arrDataQueue2):
        arrDataQueue2.append(listInsert)
```

```
    for j in range (i, len(arrDataQueue)):
        if (arrDataQueue[j] not in arrDataQueue2):
            arrDataQueue2.append(arrDataQueue[j])

    return arrDataQueue2

def dataQueuePos(arrDataQueue, score):
    i = 0
    while (i < len(arrDataQueue) and arrDataQueue[i][0] < score):
        i += 1
    return i

def addQueuePos(arrQueue, arrData, posArr):
    if (arrQueue == None or len(arrQueue) == 0) :
        arrQueue2 = [None] * (1)
        arrQueue2[0] = arrData
        return arrQueue2
    else :
        arrQueue2 = [None] * (len(arrQueue) + 1)
        i = 0
        while (i < posArr) :
            arrQueue2[i] = arrQueue[i]
            i += 1

        arrQueue2[posArr] = arrData
        i = len(arrQueue2) - 1

        while (i > posArr) :
            arrQueue2[i] = arrQueue[i - 1]
            i -= 1
        return arrQueue2

def delQueue(arrQueue):
    if (len(arrQueue) == 1):
        return []

    else:
        for i in range (0, len(arrQueue)-1):
            arrQueue[i] = arrQueue[i+1]

        return arrQueue

def delDataQueue(arrDataQueue):
    if (len(arrDataQueue) == 1):
        return []

    else:
```

```
        for i in range (0, len(arrDataQueue)-1):
            arrDataQueue[i] = arrDataQueue[i+1]

        return arrDataQueue
```

Berikut adalah isi dari file puzzleSolver.py

```
import time
import sys
import puzzleFunction

filename = sys.argv[1]
arrData = puzzleFunction.readFileInput(filename)
run = puzzleFunction.isRun(arrData)

startProcess = time.time()
arrDataQueue = [[0,0,"",""]]
arrQueue = [arrData]

arrDataStart = arrData

arrDataPrint = [[0,0,"",""]]
arrPrint = [arrDataStart]

countSimpul = 0
if (run):
    print("Penyelesaian Puzzle")
    while(puzzleFunction.isDone(arrData) == False):
        arrData = arrQueue[0]
        data = arrDataQueue[0]
        level = arrDataQueue[0][1]
        lastCommand = arrDataQueue[0][2]
        idPos = arrDataQueue[0][3]
        arrQueue = puzzleFunction.delQueue(arrQueue)
        arrDataQueue = puzzleFunction.delDataQueue(arrDataQueue)
        upScore = puzzleFunction.moveUpScore(arrData, level+1)
        leftScore = puzzleFunction.moveLeftScore(arrData, level+1)
        downScore = puzzleFunction.moveDownScore(arrData, level+1)
        rightScore = puzzleFunction.moveRightScore(arrData, level+1)
        arrDataLeft = puzzleFunction.moveLeft(arrData)
        arrDataRight = puzzleFunction.moveRight(arrData)
        arrDataUp = puzzleFunction.moveUp(arrData)
        arrDataDown = puzzleFunction.moveDown(arrData)

        if (lastCommand == "Left"):
            rightScore = 99999
```

```
        elif (lastCommand == "Right"):
            leftScore = 99999


        elif (lastCommand == "Up"):
            downScore = 99999


        elif (lastCommand == "Down"):
            upScore = 99999

        if (leftScore != 99999):
            countSimpul += 1
            idNew = idPos + "L"
            arrDataQueue = puzzleFunction.addDataQueue(arrDataQueue,
leftScore, level+1, "Left", idNew)
            posArr = puzzleFunction.dataQueuePos(arrDataQueue, leftScore)
            arrQueue = puzzleFunction.addQueuePos(arrQueue,arrDataLeft,posArr)
            arrDataPrint.append([leftScore, level+1, "Left", idNew])
            arrPrint.append(arrDataLeft)

        if (rightScore != 99999):
            countSimpul += 1
            idNew = idPos + "R"
            arrDataQueue = puzzleFunction.addDataQueue(arrDataQueue,
rightScore, level+1, "Right", idNew)
            posArr = puzzleFunction.dataQueuePos(arrDataQueue, rightScore)
            arrQueue =
puzzleFunction.addQueuePos(arrQueue,arrDataRight,posArr)
            arrDataPrint.append([rightScore, level+1, "Right", idNew])
            arrPrint.append(arrDataRight)

        if (upScore != 99999):
            countSimpul += 1
            idNew = idPos + "U"
            arrDataQueue = puzzleFunction.addDataQueue(arrDataQueue, upScore,
level+1, "Up", idNew)
            posArr = puzzleFunction.dataQueuePos(arrDataQueue, upScore)
            arrQueue = puzzleFunction.addQueuePos(arrQueue,arrDataUp,posArr)
            arrDataPrint.append([upScore, level+1, "Up", idNew])
            arrPrint.append(arrDataUp)

        if (downScore != 99999):
            countSimpul += 1
            idNew = idPos + "D"
            arrDataQueue = puzzleFunction.addDataQueue(arrDataQueue,
downScore, level+1, "Down", idNew)
```

```
            posArr = puzzleFunction.dataQueuePos(arrDataQueue, downScore)
            arrQueue = puzzleFunction.addQueuePos(arrQueue,arrDataDown,posArr)
            arrDataPrint.append([downScore, level+1, "Down", idNew])
            arrPrint.append(arrDataDown)

        if (upScore <= leftScore and upScore <= rightScore and upScore <=
downScore):
            idNew = idPos + "U"
            data = [upScore, level+1, "Up", idNew]
            arrData = puzzleFunction.moveUp(arrData)


        elif (rightScore <= leftScore and rightScore <= upScore and rightScore
<= downScore):
            idNew = idPos + "R"
            data = [rightScore, level+1, "Right", idNew]
            arrData = puzzleFunction.moveRight(arrData)


        elif (downScore <= leftScore and downScore <= rightScore and downScore
<= upScore):
            idNew = idPos + "D"
            data = [downScore, level+1, "Down", idNew]
            arrData = puzzleFunction.moveDown(arrData)


        elif (leftScore <= rightScore and leftScore <= upScore and leftScore
<= downScore):
            idNew = idPos + "L"
            data = [leftScore, level+1, "Left", idNew]
            arrData = puzzleFunction.moveLeft(arrData)


    total = 0
    for i in range (0, len(arrDataPrint)):
        isPrint = True
        for j in range (0, len(arrDataPrint[i][3])):
            if(len(arrDataPrint[i][3]) > len(data[3])):
                isPrint = False
            elif (arrDataPrint[i][3][j] != data[3][j]):
                isPrint = False
        if (isPrint):
            total += 1
            if (arrDataPrint[i][2] == ''):
                print("Awal data: ")
            else:
                print("Perintah: " + arrDataPrint[i][2])
            puzzleFunction.printArr(arrPrint[i])
```

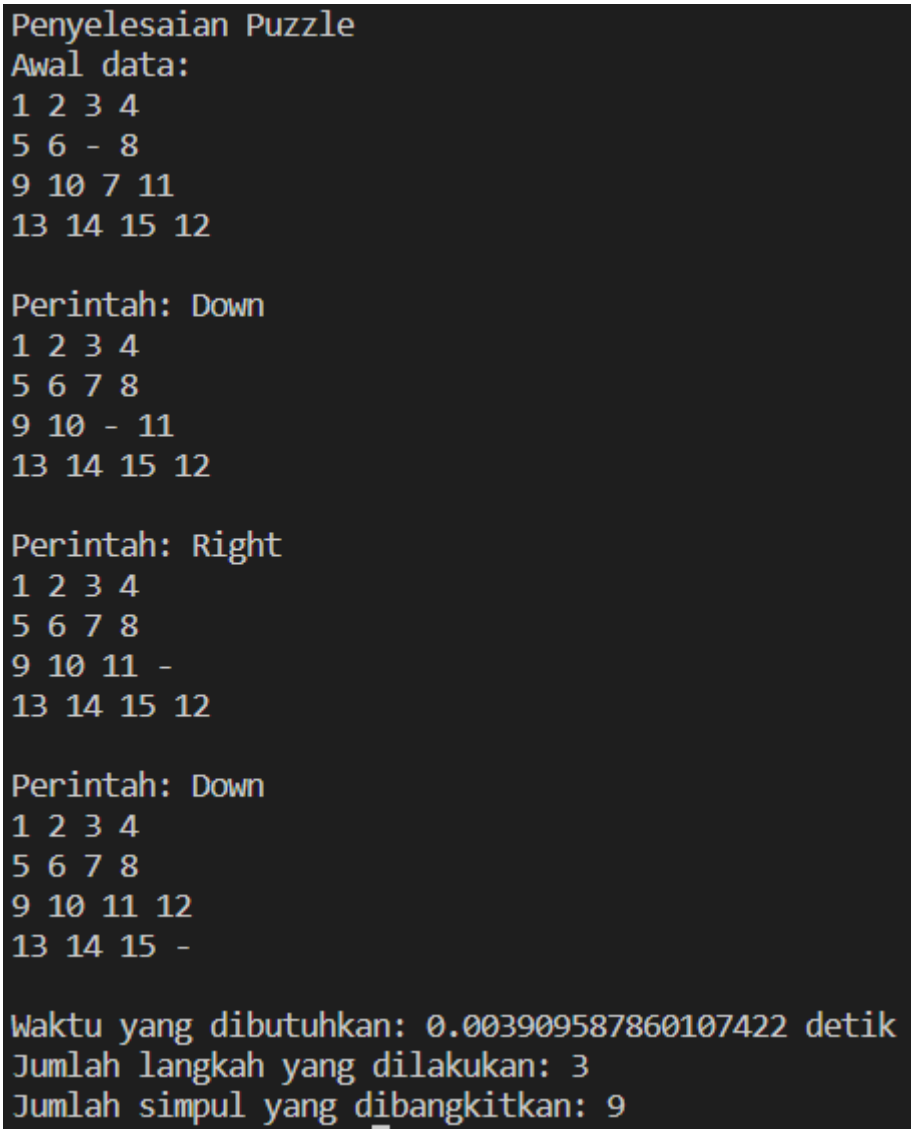
```
    endProcess = time.time()
    print("Waktu yang dibutuhkan: " + str(endProcess-startProcess) + " detik")
    print("Jumlah langkah yang dilakukan: " + str(total-1))
    print("Jumlah simpul yang dibangkitkan: " + str(countSimpul))

else:
    print("Puzzle tidak dapat diselesaikan")
```

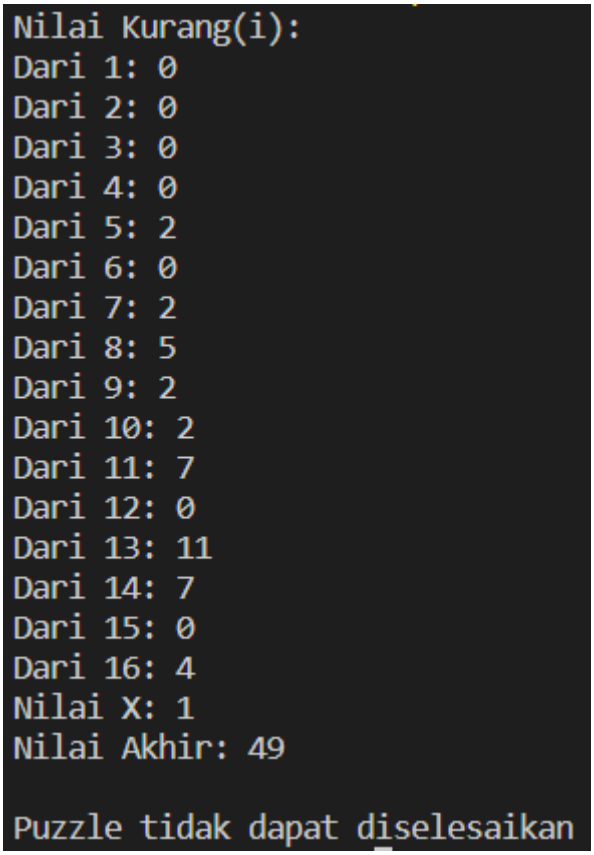
Bab 4 Hasil Pengujian

```
Nilai Kurang(i):
Dari 1: 0
Dari 2: 0
Dari 3: 0
Dari 4: 0
Dari 5: 0
Dari 6: 0
Dari 7: 0
Dari 8: 1
Dari 9: 1
Dari 10: 1
Dari 11: 0
Dari 12: 0
Dari 13: 1
Dari 14: 1
Dari 15: 1
Dari 16: 9
Nilai X: 1
Nilai Akhir: 16
```

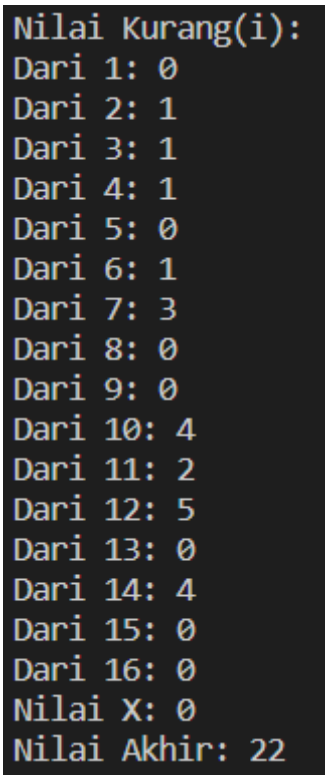
Gambar 4.1. Hasil perhitungan skor penentuan puzzle dapat diselesaikan untuk file test1.txt



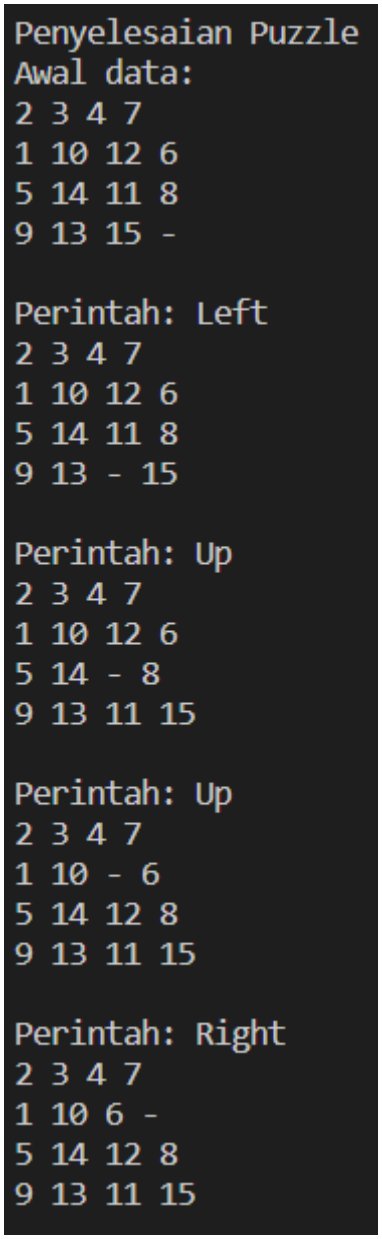
Gambar 4.2. Hasil penyelesaian puzzle dari file test1.txt



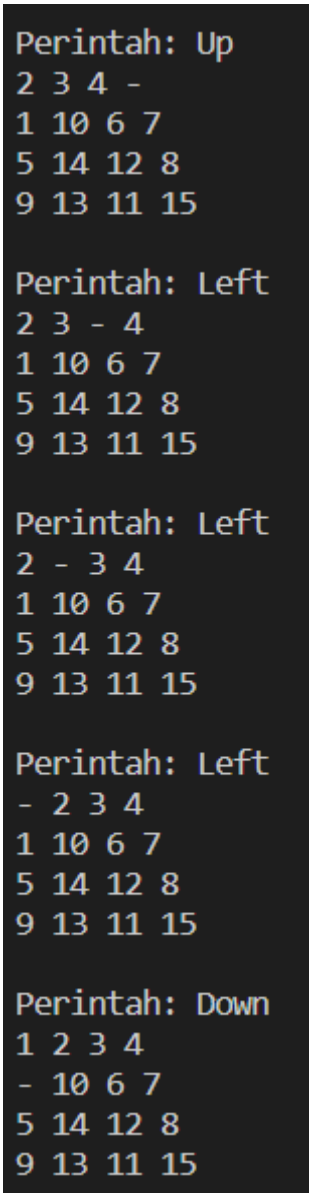
Gambar 4.3. Hasil penyelesaian puzzle untuk file test2.txt



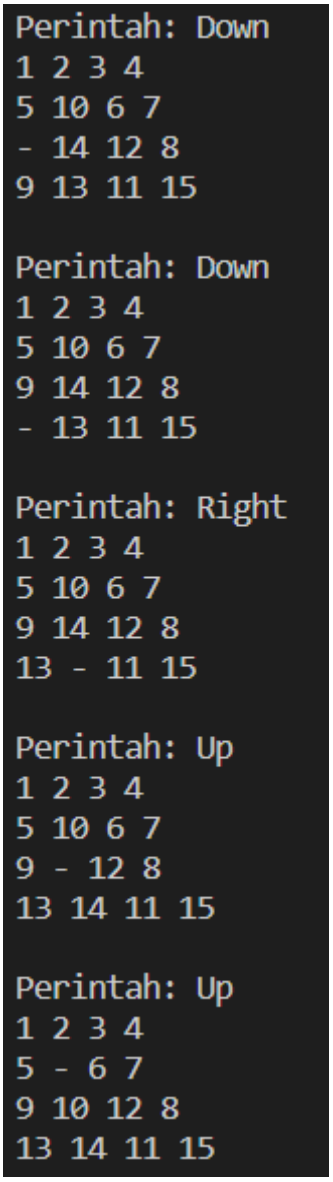
Gambar 4.4. Hasil perhitungan skor penentuan puzzle dapat diselesaikan untuk file test3.txt



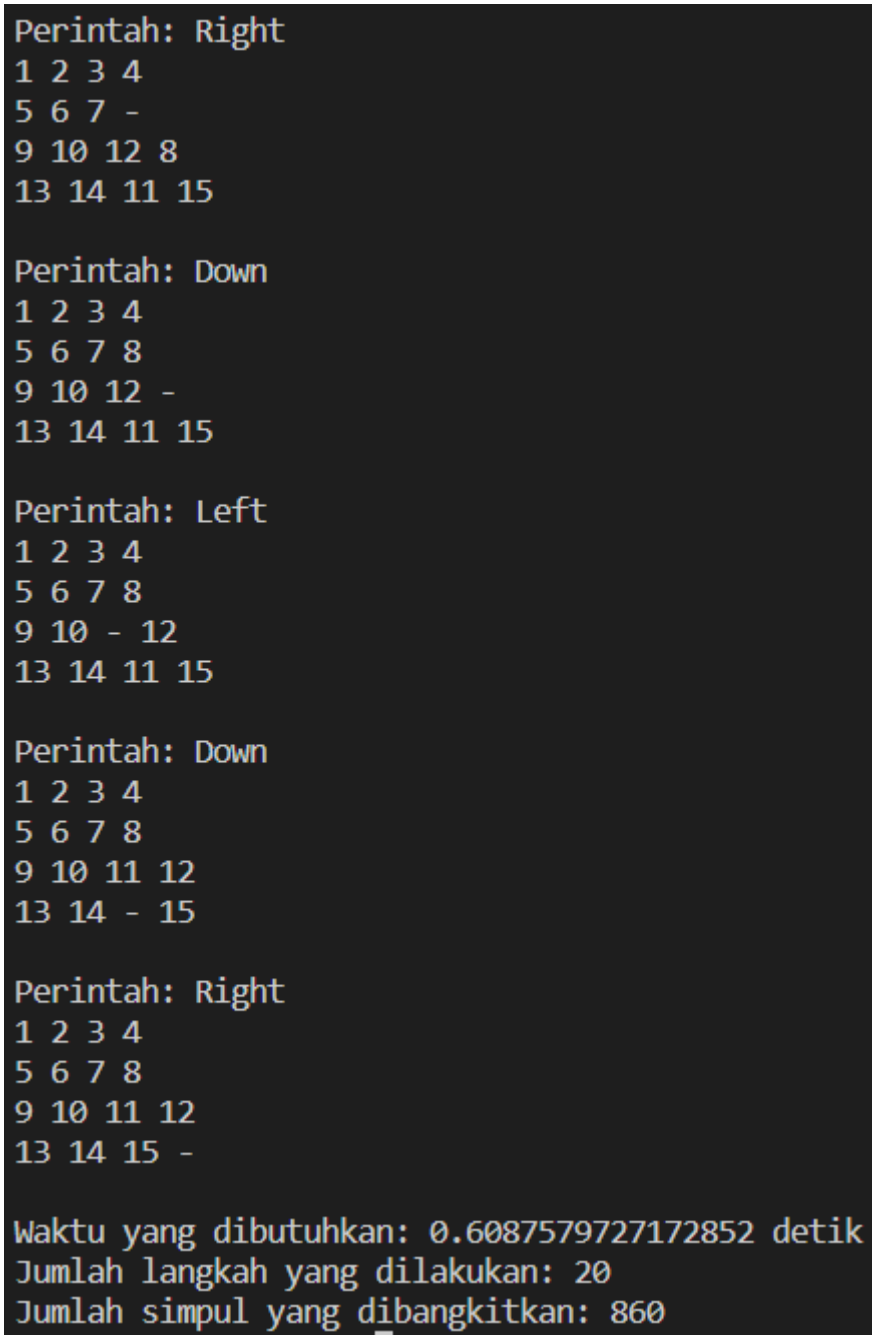
Gambar 4.5. Hasil penyelesaian puzzle dari file test3.txt (1)



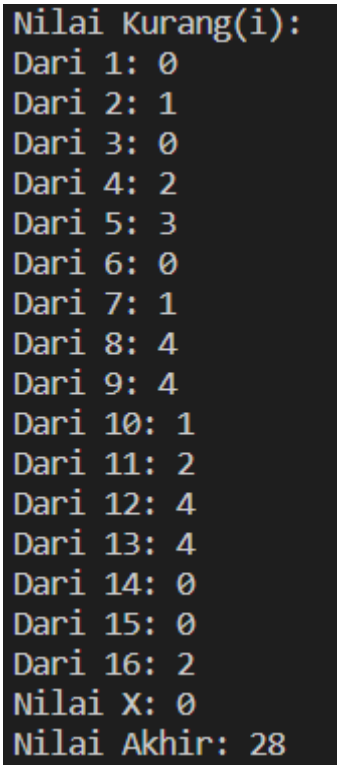
Gambar 4.6. Hasil penyelesaian puzzle dari file test3.txt (2)



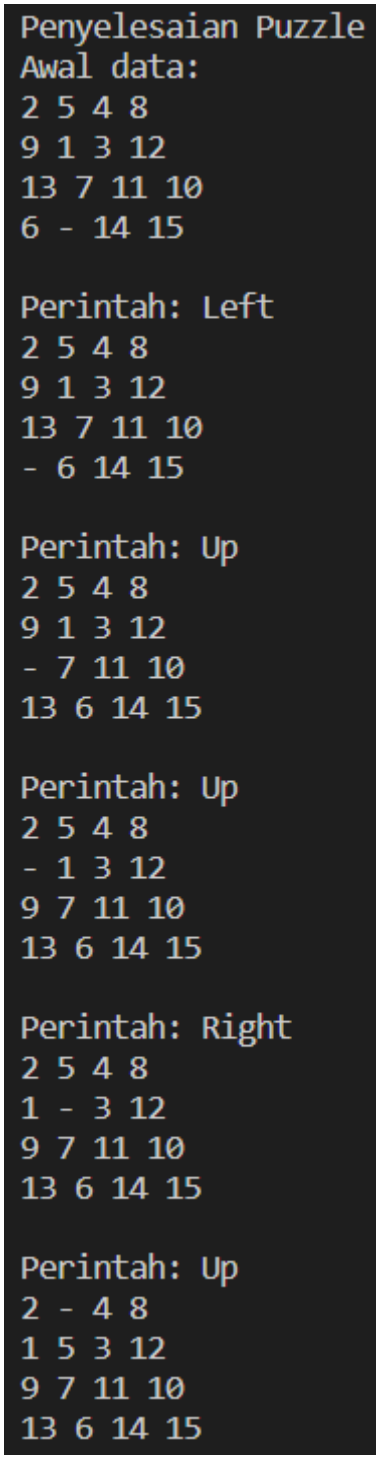
Gambar 4.7. Hasil penyelesaian puzzle dari file test3.txt (3)



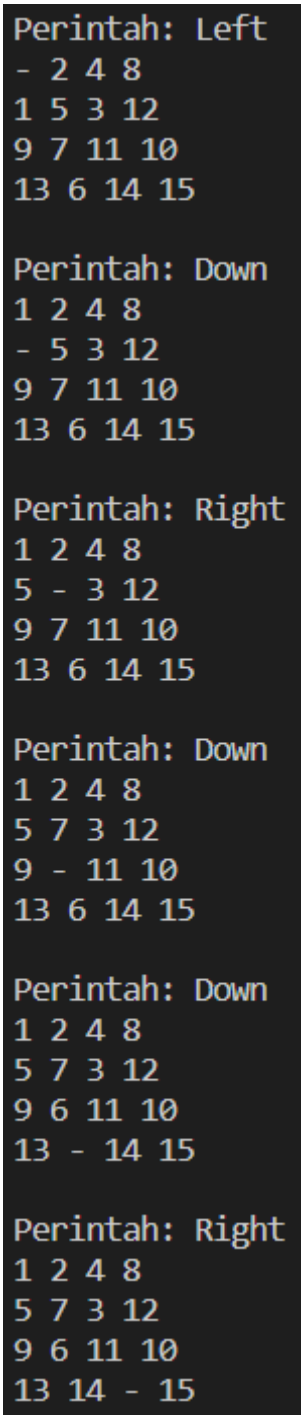
Gambar 4.8. Hasil penyelesaian puzzle dari file test3.txt (4)



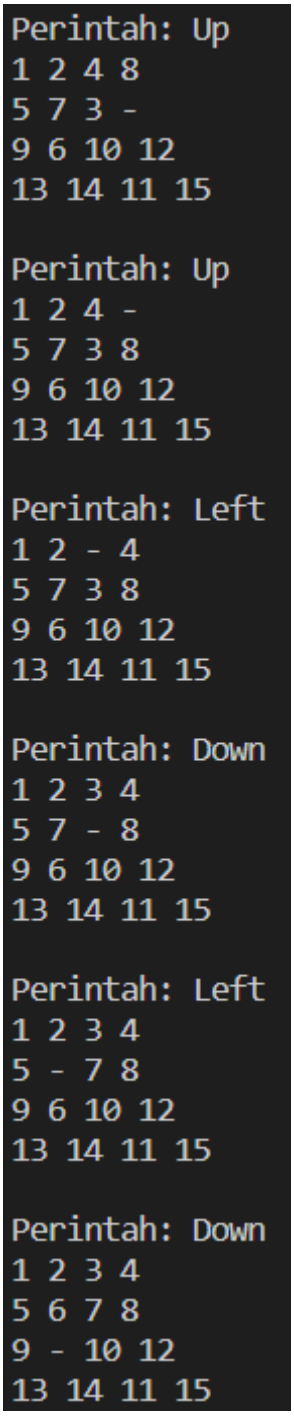
Gambar 4.9. Hasil perhitungan skor penentuan puzzle dapat diselesaikan untuk file test4.txt



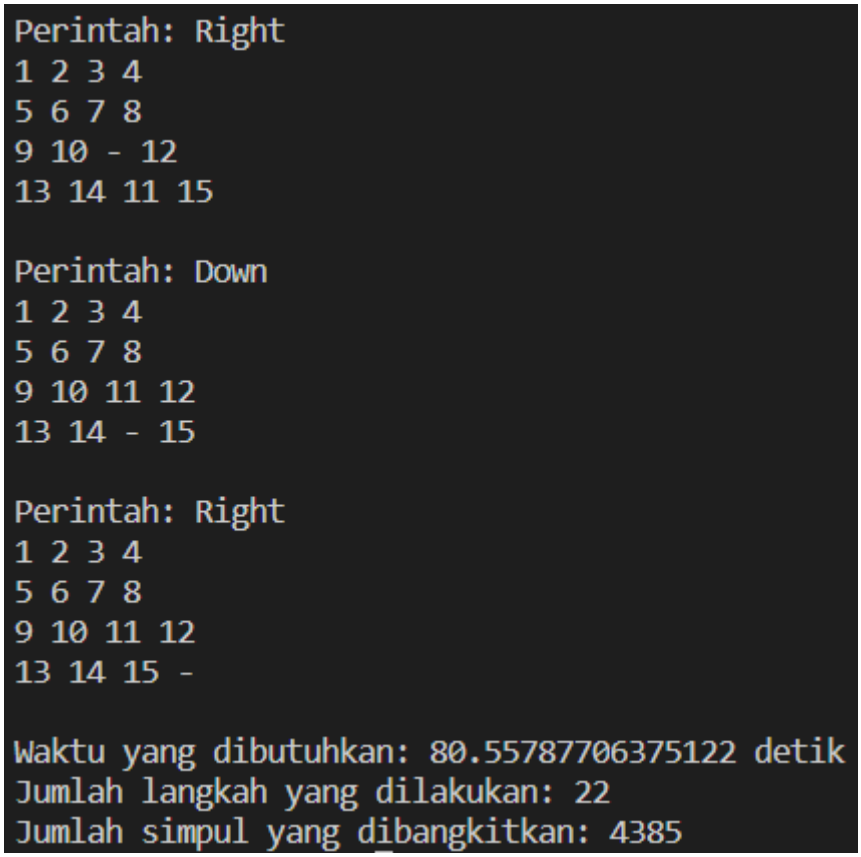
Gambar 4.10. Hasil penyelesaian puzzle dari file test4.txt (1)



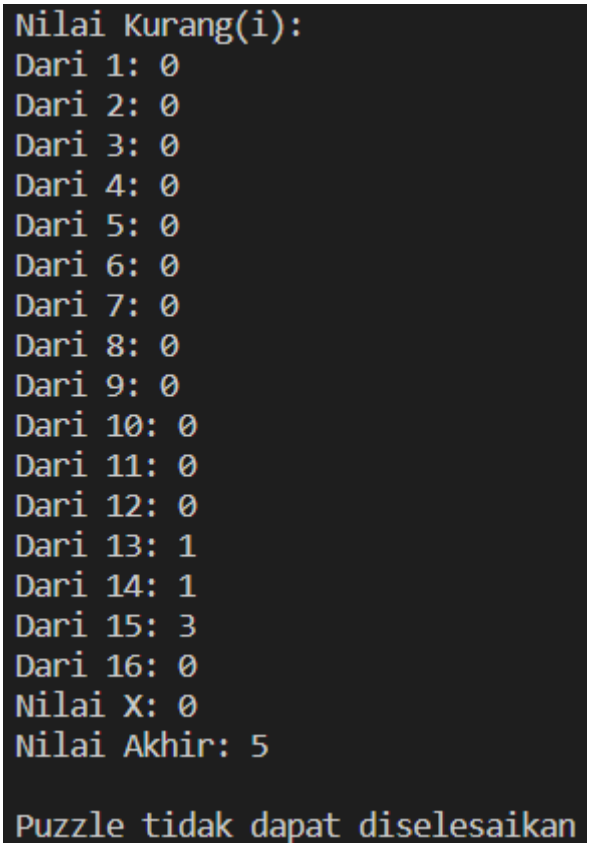
Gambar 4.11. Hasil penyelesaian puzzle dari file test4.txt (2)



Gambar 4.12. Hasil penyelesaian puzzle dari file test4.txt (3)



Gambar 4.13. Hasil penyelesaian puzzle dari file test4.txt (4)



Gambar 4.14. Hasil penyelesaian puzzle dari file test5.txt

Tabel Cek List

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

Link untuk dokumentasi dan kode di Github :

https://github.com/wmk567/15-Puzzle.git