LAPORAN TUGAS KECIL 3
IF2211 STRATEGI ALGORITMA
PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN ALGORITMA BRANCH AND BOUND

Oleh

Primanda Adyatma Hafiz (13520022)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

Daftar Isi

BAB I
ALGORITMA BRANCH AND BOUND

1.1 Cara Kerja Program

1. Menerima input matriks 4x4 dengan ubin yang kosong ditandai dengan “-“
2. Menghitung total kurang[i] dari matriks pada keadaan awal dengan mengiterasi setiap ubin lalu dicek jumlah ubin yang nilainya lebih kecil dan posisinya terletak setelah ubin tersebut.
3. Nilai tersebut kemudian dijumlahkan dengan 1 bila ubin yang kosong berada pada sel yang diarsir
4. Jika totalnya ganjil maka akan ditampilkan bahwa puzzle tidak dapat diselesaikan, sedangkan bila totalnya genap maka proses akan berlanjut ke (5)
5. Inisialisasi list untuk menyimpan keseluruhan node yang telah terbentuk beserta node yang menjadi parentnya serta priority queue yang mengurutkan node berdasarkan costnya yang dihitung dari jumlah jarak dari root ke node dan total ubin yang tempatnya masih belum sesuai
6. Masukkan keadaan awal matriks ke list dan priority queue
7. Cek apakah matriks awal telah merupakan solusi, jika belum lanjut ke (8) sedangkan jika sudah lanjut ke (13)
8. Lakukan ekspan untuk node pada posisi head pada priority queue ke arah U, R ,D ,L dan simpan node hasil tersebut pada list dan priority queue
9. Ambil matriks pada posisi head priority queue
10. Cek apakah matriks telah merupakan solusi, jika belum lanjut ke (11) sedangkan jika sudah lanjut ke (13)
11. Lakukan ekspan matriks puzzle 3 arah sehingga tidak menyebabkan node kembali menjadi seperti parent nodenya
12. Kembali ke proses 9
13. Jika telah ditemukan solusi tampilkan proses penyelesaian puzzle dari list serta waktu eksekusi dan jumlah node yang terbentuk

BAB 2
KODE PROGRAM

2.1 File main.py

```
from puzzle import *
import random

def randomizeMat(mat):
    lst=[i+1 for i in range(16)]
    random.shuffle(lst)
    for i in range(16):
        mat[i//4][i%4]=lst[i]

def main():
    mat=[[0 for i in range(4)] for j in range(4)]

    while True:
        print("Pilihan input :")
        print("1. Input File")
        print("2. Input Random\n")
        inp=int(input("Masukan angka pilihan input : "))
        if(inp==1):
            inp=input("Masukkan nama file : ")
            try:
                i=0
                with open("test/"+inp) as f:
                    line=f.readline()
                    while line:
                        mat[i]=[int(j) if j!='-' else 16 for j in
line.split()]
                        line=f.readline()
                        i+=1
            except:
                print("File error")
            break
        elif(inp==2):
            randomizeMat(mat)
            break
        else:
            print("Masukan tidak valid")

    puzzleSolver(mat)

main()
```

2.2 File puzzle.py

```
import heapq
import time

# variabel global untuk menghitung total node yang telah terbentuk
totalNode=1

# fungsi untuk copy matrix
def copyMat(mat):
    ret=[[0 for j in range(4)] for i in range(4)]
    for i in range(4):
        for j in range(4):
            ret[i][j]=mat[i][j]
    return ret

# fungsi untuk menghitung total KURANG(i) dengan i adalah seluruh ubin pada
matriks
def kurang(mat):
    ans=0
    for i in range(4):
        for j in range(4):
            for k in range(i*4+j+1,16):
                if(mat[k//4][k%4]<mat[i][j]):
                    ans+=1
    return ans

# Fungsi untuk mengecek apakah ubin yang kosong berada pada bagian yang diblok
def X(mat):
    for i in range(4):
        for j in range(4):
            if(mat[i][j]==16):
                return (i&1)^(j&1)

# Menampilkan nilai kurang[i]
def displayKurang(mat):
    kurangI=[0 for i in range(16)]
    for i in range(4):
        for j in range(4):
            for k in range(i*4+j+1,16):
                if(mat[k//4][k%4]<mat[i][j]):
                    kurangI[mat[i][j]-1]+=1
    for i in range(16):
        print("Kurang[{}] = {}".format(i+1,kurangI[i]))
```

```
# Menghitung setiap ubin pada puzzle yang belum pada tempatnya
def cost(mat):
    ans=0
    for i in range(4):
        for j in range(4):
            if(mat[i][j]!=i*4+j+1):
                ans+=1
    return ans

# Menampilkan matriks
def displayMat(mat):
    for i in range(4):
        for j in range(4):
            if(mat[i][j]==16):
                print("-",end=" ")
            else:
                print(mat[i][j],end=" ")
        print()

# Mengecek apakah setiap ubin pada puzzle telah berada pada posisinya
def isSolution(mat):
    for i in range(4):
        for j in range(4):
            if(i*4+j+1!=mat[i][j]):
                return False
    return True

# Mengembalikan indeks ubin yang kosong pada matriks
def findSpace(mat):
    for i in range(4):
        for j in range(4):
            if(mat[i][j]==16):
                return i,j

# Memasukkan node ke priority queue berdasarkan costnya untuk setiap command
yang valid
def pushToQueue(last,lastStep,mat,listMat,prioQueueMat,depth):
    global totalNode
    i,j=findSpace(mat)

    # Command = U
    if(lastStep!="D" and i!=0):
        totalNode+=1
        retMat4=copyMat(mat)
        retMat4[i][j],retMat4[i-1][j]=retMat4[i-1][j],retMat4[i][j]
```

```
        heapq.heappush(prioQueueMat,(cost(retMat4)+depth+1,depth+1,last,len(li
stMat),"U",copyMat(retMat4)))
        listMat.append((last,"U",retMat4))

    # Command = R
    if(lastStep!="L" and j!=3):
        totalNode+=1
        retMat1=copyMat(mat)
        retMat1[i][j],retMat1[i][j+1]=retMat1[i][j+1],retMat1[i][j]
        heapq.heappush(prioQueueMat,(cost(retMat1)+depth+1,depth+1,last,len(li
stMat),"R",copyMat(retMat1)))
        listMat.append((last,"R",retMat1))

    # Command = D
    if(lastStep!="U" and i!=3):
        totalNode+=1
        retMat3=copyMat(mat)
        retMat3[i][j],retMat3[i+1][j]=retMat3[i+1][j],retMat3[i][j]
        heapq.heappush(prioQueueMat,(cost(retMat3)+depth+1,depth+1,last,len(li
stMat),"D",copyMat(retMat3)))
        listMat.append((last,"D",retMat3))

    # Command = L
    if(lastStep!="R" and j!=0):
        totalNode+=1
        retMat2=copyMat(mat)
        retMat2[i][j],retMat2[i][j-1]=retMat2[i][j-1],retMat2[i][j]
        heapq.heappush(prioQueueMat,(cost(retMat2)+depth+1,depth+1,last,len(li
stMat),"L",copyMat(retMat2)))
        listMat.append((last,"L",retMat2))

# Menampilkan urutan command pada matriks hingga mencapai matriks solusi
def printPath(listMat,idAns):
    ret=[]
    while(idAns!=-1):
        ret.append(listMat[idAns])
        idAns=listMat[idAns][0]
    for i in range(len(ret)-2,-1,-1):
        print("LANGKAH {} =".format(len(ret)-1-i),end=" ")
        if(ret[i][1]=="U"):
            print("UP")
        elif(ret[i][1]=="D"):
            print("DOWN")
        elif(ret[i][1]=="R"):
            print("RIGHT")
        elif(ret[i][1]=="L"):
```

```
            print("LEFT")
        displayMat(ret[i][2])
        print()


def puzzleSolver(mat):
    print("\nMasukan matriks :")
    displayMat(mat)
    print()
    displayKurang(mat)
    print()
    print("Total kurang[i]+X =",kurang(mat)+X(mat),end="\n\n")

    startTime=time.time()

    if((kurang(mat)+X(mat))%2==1):
        print("Puzzle tidak bisa dipecahkan")
    else:
        # inisialisasi variabel
        listMat=[] # Berperan sebagai linked list sebagai sisi dari node yang
terbentuk
        prioQueueMat=[] # List node yang perlu diproses dan terurut
berdasarkan prioritasnya

        # cek apakah matriks merupakan solusi
        if(isSolution(mat)):
            print("Solusi telah dicapai")
        else:
            # masukkan kondisi awal matriks ke listMat
            listMat.append((-1,"",mat))
            heapq.heapify(prioQueueMat)
            pushToQueue(0,"",copyMat(mat),listMat,prioQueueMat,0)

            # lakukan iterasi hingga ditemukan solusi
            while(True):
                cur=heapq.heappop(prioQueueMat)
                if(isSolution(cur[5])):
                    print("Urutan penyelesaian puzzle :\n")
                    printPath(listMat,cur[3])
                    print("Total node terbentuk =",totalNode)
                    break
                pushToQueue(cur[3],cur[4],copyMat(cur[5]),listMat,prioQueueMat
,cur[1])
    print("Waktu eksekusi :",round(time.time()-startTime,10),"s")
```

BAB 3
HASIL PENGUJIAN

2.1 Contoh Input Persoalan

Nama File	Input Matriks
testSolved1.txt	1 2 3 4 5 6 - 8 9 10 7 11 13 14 15 12
testSolved2.txt	5 1 3 4 2 - 6 8 9 14 10 11 15 13 7 12
testSolved3.txt	- 1 2 3 6 7 8 4 5 9 10 11 13 14 15 12
testUnsolved1.txt	3 1 2 4 5 - 7 8 10 6 15 12 9 13 14 11
testUnsolved2.txt	15 - 1 4 8 7 6 3 14 2 9 10 11 5 13 12

2.2 Kasus 1 Dapat Diselesaikan

```
Pilihan input :
1. Input File
2. Input Random

Masukan angka pilihan input : 1
Masukkan nama file : testSolved1.txt

Masukan matriks :
1 2 3 4
5 6 - 8
9 10 7 11
13 14 15 12

Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 0
Kurang[4] = 0
Kurang[5] = 0
Kurang[6] = 0
Kurang[7] = 0
Kurang[8] = 1
Kurang[9] = 1
Kurang[10] = 1
Kurang[11] = 0
Kurang[12] = 0
Kurang[13] = 1
Kurang[14] = 1
Kurang[15] = 1
Kurang[16] = 9

Total kurang[i]+X = 16
```

```
Urutan penyelesaian puzzle :

LANGKAH 1 = DOWN
1 2 3 4
5 6 7 8
9 10 - 11
13 14 15 12

LANGKAH 2 = RIGHT
1 2 3 4
5 6 7 8
9 10 11 -
13 14 15 12

LANGKAH 3 = DOWN
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 -

Total node terbentuk = 10
Waktu eksekusi : 0.011991024 s
```

2.3 Kasus 2 Dapat Diselesaikan

```
Pilihan input :
1. Input File
2. Input Random

Masukan angka pilihan input : 1
Masukkan nama file : testSolved2.txt

Masukan matriks :
5 1 3 4
2 - 6 8
9 14 10 11
15 13 7 12

Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 1
Kurang[4] = 1
Kurang[5] = 4
Kurang[6] = 0
Kurang[7] = 0
Kurang[8] = 1
Kurang[9] = 1
Kurang[10] = 1
Kurang[11] = 1
Kurang[12] = 0
Kurang[13] = 2
Kurang[14] = 5
Kurang[15] = 3
Kurang[16] = 10

Total kurang[i]+X = 30
```

```
Urutan penyelesaian puzzle :

LANGKAH 1 = LEFT
5 1 3 4
- 2 6 8
9 14 10 11
15 13 7 12

LANGKAH 2 = DOWN
5 1 3 4
9 2 6 8
- 14 10 11
15 13 7 12

LANGKAH 3 = RIGHT
5 1 3 4
9 2 6 8
14 - 10 11
15 13 7 12

LANGKAH 4 = DOWN
5 1 3 4
9 2 6 8
14 13 10 11
15 - 7 12

LANGKAH 5 = LEFT
5 1 3 4
9 2 6 8
14 13 10 11
- 15 7 12

LANGKAH 6 = UP
5 1 3 4
9 2 6 8
- 13 10 11
14 15 7 12
```

```
LANGKAH 7 = RIGHT
5 1 3 4
9 2 6 8
13 - 10 11
14 15 7 12

LANGKAH 8 = RIGHT
5 1 3 4
9 2 6 8
13 10 - 11
14 15 7 12

LANGKAH 9 = DOWN
5 1 3 4
9 2 6 8
13 10 7 11
14 15 - 12

LANGKAH 10 = LEFT
5 1 3 4
9 2 6 8
13 10 7 11
14 - 15 12

LANGKAH 11 = LEFT
5 1 3 4
9 2 6 8
13 10 7 11
- 14 15 12

LANGKAH 12 = UP
5 1 3 4
9 2 6 8
- 10 7 11
13 14 15 12

LANGKAH 13 = UP
5 1 3 4
- 2 6 8
9 10 7 11
13 14 15 12
```

```
LANGKAH 14 = UP
- 1 3 4
5 2 6 8
9 10 7 11
13 14 15 12

LANGKAH 15 = RIGHT
1 - 3 4
5 2 6 8
9 10 7 11
13 14 15 12

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

LANGKAH 17 = RIGHT
1 2 3 4
5 6 - 8
9 10 7 11
13 14 15 12

LANGKAH 18 = DOWN
1 2 3 4
5 6 7 8
9 10 - 11
13 14 15 12

LANGKAH 19 = RIGHT
1 2 3 4
5 6 7 8
9 10 11 -
13 14 15 12

LANGKAH 20 = DOWN
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 -
```

```
Total node terbentuk = 10338
Waktu eksekusi : 0.3584201336 s
```

2.4 Kasus 3 Dapat Diselesaikan

```
Pilihan input :
1. Input File
2. Input Random

Masukan angka pilihan input : 1
Masukkan nama file : testSolved3.txt

Masukan matriks :
- 1 2 3
6 7 8 4
5 9 10 11
13 14 15 12

Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 0
Kurang[4] = 0
Kurang[5] = 0
Kurang[6] = 2
Kurang[7] = 2
Kurang[8] = 2
Kurang[9] = 0
Kurang[10] = 0
Kurang[11] = 0
Kurang[12] = 0
Kurang[13] = 1
Kurang[14] = 1
Kurang[15] = 1
Kurang[16] = 15

Total kurang[i]+X = 24
```

```
Urutan penyelesaian puzzle :

LANGKAH 1 = RIGHT
1 - 2 3
6 7 8 4
5 9 10 11
13 14 15 12

LANGKAH 2 = RIGHT
1 2 - 3
6 7 8 4
5 9 10 11
13 14 15 12

LANGKAH 3 = RIGHT
1 2 3 -
6 7 8 4
5 9 10 11
13 14 15 12

LANGKAH 4 = DOWN
1 2 3 4
6 7 8 -
5 9 10 11
13 14 15 12

LANGKAH 5 = LEFT
1 2 3 4
6 7 - 8
5 9 10 11
13 14 15 12

LANGKAH 6 = LEFT
1 2 3 4
6 - 7 8
5 9 10 11
13 14 15 12
```

```
LANGKAH 7 = LEFT
1 2 3 4
- 6 7 8
5 9 10 11
13 14 15 12

LANGKAH 8 = DOWN
1 2 3 4
5 6 7 8
- 9 10 11
13 14 15 12

LANGKAH 9 = RIGHT
1 2 3 4
5 6 7 8
9 - 10 11
13 14 15 12

LANGKAH 10 = RIGHT
1 2 3 4
5 6 7 8
9 10 - 11
13 14 15 12

LANGKAH 11 = RIGHT
1 2 3 4
5 6 7 8
9 10 11 -
13 14 15 12

LANGKAH 12 = DOWN
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 -

Total node terbentuk = 28
Waktu eksekusi : 0.0920450687 s
```

2.5 Kasus 1 Tidak Dapat Diselesaikan

```
Pilihan input :
1. Input File
2. Input Random

Masukan angka pilihan input : 1
Masukkan nama file : testUnsolved1.txt

Masukan matriks :
3 1 2 4
5 - 7 8
10 6 15 12
9 13 14 11

Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 2
Kurang[4] = 0
Kurang[5] = 0
Kurang[6] = 0
Kurang[7] = 1
Kurang[8] = 1
Kurang[9] = 0
Kurang[10] = 2
Kurang[11] = 0
Kurang[12] = 2
Kurang[13] = 1
Kurang[14] = 1
Kurang[15] = 5
Kurang[16] = 10

Total kurang[i]+X = 25

Puzzle tidak bisa dipecahkan
Waktu eksekusi : 0.0010006428 s
```

2.6 Kasus 2 Tidak Dapat Diselesaikan

```
Pilihan input :
1. Input File
2. Input Random

Masukan angka pilihan input : 1
Masukkan nama file : testUnsolved2.txt

Masukan matriks :
15 - 1 4
8 7 6 3
14 2 9 10
11 5 13 12

Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 1
Kurang[4] = 2
Kurang[5] = 0
Kurang[6] = 3
Kurang[7] = 4
Kurang[8] = 5
Kurang[9] = 1
Kurang[10] = 1
Kurang[11] = 1
Kurang[12] = 0
Kurang[13] = 1
Kurang[14] = 7
Kurang[15] = 14
Kurang[16] = 14

Total kurang[i]+X = 55

Puzzle tidak bisa dipecahkan
Waktu eksekusi : 0.0 s
```

LAMPIRAN

Link ke Repository Github

https://github.com/Primanda28/Tucil3_13520022

Tabel Penilaian

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