Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*

Fikri Ihsan Fadhiilah (13520148)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

[13520148@std.stei.itb.ac.id](mailto:13520148@std.stei.itb.ac.id)

Cara Kerja Program Branch and Bound

1. Pengecekan apakah kondisi Akhir dapat tercapai atau tidak dengan menggunakan fungsi kurangText

   Description automatically generated

Jika bernilai genap maka dapat diselesaikan, jika ganjil tidak dapat diselesaikan.

Text

Description automatically generated

1. Memasukkan sebuah simpul ke simpul hidup selama tujuan belum tercapai

Text

Description automatically generated

1. Mengurutkan simpul hidup berdasarkan cost lalu jika sama berdasarkan kedalaman setiap kali ada simpul yang di tambahkan ke simpul hidup

Text

Description automatically generated

1. Mengambil simpul dengan cost terkecil untuk melanjutkan penelusuran

Text

Description automatically generated

1. Mengulangi Langkah 2-4 sampai tercapai bentuk matrix Akhir
2. Memasukkan node ke list jawaban

Text

Description automatically generated

1. Menampilkan jawaban beserta urutan pergerakannya dari bentuk awal

SOURCE PROGRAM

import numpy as np

from copy import deepcopy

import time

import sys

#sys.setrecursionlimit(10000)

def is\_have\_solution(arr\_for\_checking,black\_tile):

que=[0 for i in range (16)]

sum=0

for i in range(np.size(arr\_for\_checking)):

#component=0

if arr\_for\_checking[i]==16 and i in black\_tile:

sum+=1

for j in range(i+1,np.size(arr\_for\_checking)):

if arr\_for\_checking[i]>arr\_for\_checking[j]:

sum+=1

que[arr\_for\_checking[i]-1]+=1

#component+=1

#print("Nilai fungsi Kurang("+str(arr\_for\_checking[i])+") = " + str(component))

for i in range(16):

print("Nilai fungsi Kurang("+str(i+1)+") = " + str(que[i]))

print("\nTotal nilai Fungsi KURANG(i) + X adalah " + str(sum)+"\n")

if sum%2==0:

return True

else:

return False

def display\_matrix(matrix):

for i in range(4):

for j in range(4):

if matrix[i][j] == 16:

print(" \t",end="")

else:

print(str(matrix[i][j])+"\t",end="")

print("")

def move\_right(state,x,y):

temp = state[x][y]

state[x][y]=state[x][y+1]

state[x][y+1]=temp

return state

def move\_up(state,x,y):

temp = state[x][y]

state[x][y]=state[x-1][y]

state[x-1][y]=temp

return state

def move\_left(state,x,y):

temp = state[x][y]

state[x][y]=state[x][y-1]

state[x][y-1]=temp

return state

def move\_down(state,x,y):

temp = state[x][y]

state[x][y]=state[x+1][y]

state[x+1][y]=temp

return state

def get\_blank\_location(arr):

for i in range(4):

for j in range(4):

if arr[i][j]==16:

x=i

y=j

break

return x,y

def count\_cost(arr):

cost=0

arr\_for\_checking=np.ravel(arr)

for i in range(np.size(arr\_for\_checking)):

if arr\_for\_checking[i] != 16 and i+1!=arr\_for\_checking[i]:

cost+=1

return cost

def get\_cost(node\_):

return node\_.cost,node\_.depth

def ins\_to\_que(que\_,node\_):

que\_.append(node\_)

que\_.sort(key=get\_cost)

def move(que\_,moved,node\_):

x,y=get\_blank\_location(moved)

moved\_node = node(moved,node\_,node\_.depth+1,x,y,count\_cost(moved)+node\_.depth+1)

ins\_to\_que(que\_,moved\_node)

def visited\_or\_not(visited,moved):

i=0

ada=False

while(i<len(visited) and ada==False):

if np.array\_equal(visited[i],moved):

ada=True

i+=1

return ada

def solve(sol\_,que\_,node\_,visited):

next\_node=node\_

global node\_generated

while not(np.array\_equal(goal\_state,next\_node.state)):

if next\_node.x != 0:

moved = move\_up(deepcopy(next\_node.state),next\_node.x,next\_node.y)

if not(visited\_or\_not(visited,moved)):

move(que\_,moved,next\_node)

node\_generated+=1

visited.append(moved)

if next\_node.y != 3:

moved = move\_right(deepcopy(next\_node.state),next\_node.x,next\_node.y)

if not(visited\_or\_not(visited,moved)):

move(que\_,moved,next\_node)

node\_generated+=1

visited.append(moved)

if next\_node.x != 3:

moved = move\_down(deepcopy(next\_node.state),next\_node.x,next\_node.y)

if not(visited\_or\_not(visited,moved)):

move(que\_,moved,next\_node)

node\_generated+=1

visited.append(moved)

if next\_node.y !=0:

moved = move\_left(deepcopy(next\_node.state),next\_node.x,next\_node.y)

if not(visited\_or\_not(visited,moved)):

move(que\_,moved,next\_node)

node\_generated+=1

visited.append(moved)

next\_node=que\_.pop(0)

#solve(sol\_,que\_,next\_node,visited)

sol\_.append(next\_node)

def display\_path(node\_):

if node\_.parents\_node != None:

display\_path(node\_.parents\_node)

display\_matrix(node\_.state)

print("\n")

else:

display\_matrix(node\_.state)

print("\n")

def teks\_to\_matriks(\_inputfile):

\_case = []

with open(\_inputfile) as file:

for item in file:

\_case.append([int(i) for i in item.split()])

return \_case

class node(object):

def \_\_init\_\_(self,state,parents\_node,depth,Xblank\_location,Yblank\_location,cost):

self.state=state

self.parents\_node=parents\_node

self.depth=depth

self.x=Xblank\_location

self.y=Yblank\_location

self.cost=cost

if \_\_name\_\_ == '\_\_main\_\_':

print("\n\n=== Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound ===\n")

goal\_state=np.array([

[1,2,3,4],

[5,6,7,8],

[9,10,11,12],

[13,14,15,16]

])

input\_file= input("\nMasukkan file .txt yang akan digunakan sebagai test case : ")

is\_= teks\_to\_matriks(input\_file)

black\_tile=[1,3,4,6,9,11,12,14]

arr\_for\_checking=np.ravel(is\_)

node\_generated = 0

print("Puzzle Awal : \n")

display\_matrix(is\_)

print("")

if is\_have\_solution(arr\_for\_checking,black\_tile):

back\_to\_2d = arr\_for\_checking.reshape(4,4)

#initiate root ---------------------------

urutan=[]

sol=[]

visited=[]

x\_start,y\_start=get\_blank\_location(back\_to\_2d)

start\_node = node(back\_to\_2d,None,0,x\_start,y\_start,99)

urutan.append(start\_node)

#initiate root ---------------------------

#Runtime-----------------------

start\_time = time.time()

solve(sol,urutan,start\_node,visited)

selesai=time.time()-start\_time

#Runtime-----------------------

print("\nLangkah Penyelesaian\n")

display\_path(sol[0])

print("Jumlah simpul yang dibangkitkan = "+str(node\_generated)+"\n")

urutan.clear()

print("Total waktu eksekusi penyelesaian : " + str(selesai))

else:

*Screenshoot* Input-Output Program

Input :

Text

Description automatically generatedText

Description automatically generated

Output:

Text

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

Input:

Text

Description automatically generatedText

Description automatically generated

Output:

Text

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

Input:

Text

Description automatically generatedText

Description automatically generated

Output

Text

Description automatically generatedA picture containing text, electronics, keyboard, light

Description automatically generated

A screenshot of a computer screen

Description automatically generated with medium confidence

Input:

Text

Description automatically generatedText

Description automatically generated

Outptut:

Text

Description automatically generated

Input:

Text

Description automatically generatedGraphical user interface, text

Description automatically generated

output

Text

Description automatically generated

Instansiasi Persoalan

Bisa diselesaikan :

1. 1 2 3 4

5 6 16 8

9 10 7 11

13 14 15 12

1. 1 2 3 4

5 6 7 8

9 16 10 11

13 14 15 12

1. 1 2 3 4

5 6 16 12

9 10 8 7

13 14 11 15

Tidak bisa Diselesaikan :

1. 1 3 4 15

2 16 5 12

7 6 11 14

8 9 10 13

1. 1 6 3 2

5 7 4 16

9 10 11 8

13 14 15 12

Checklist

|  |  |  |
| --- | --- | --- |
| Poin | YA | TIDAK |
| 1. Program Berhasil dikompilasi | V |  |
| 1. Program berhasil *running* | V |  |
| 1. Program dapat menerima input dan menuliskann ouput | V |  |
| 1. Luaran sudah benar untuk semua data uji | V |  |
| 1. Bonus dibuat |  | V |

Link Program : https://github.com/Fikri-IF/STIMA-TUCIL-3