**LAPORAN TUGAS KECIL 3 IF2211 Strategi Algoritma**

Semester II tahun 2021/2022

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

**Steven Gianmarg Haposan Siahaan**

**13520145**

**Program Studi Teknik Informatika**

**Sekolah Tinggi Elektro dan Informatika Institut Teknologi Bandung**

**2022**

# Daftar Isi

[Daftar Isi 2](#_Toc99945005)

[BAB I 3](#_Toc99945006)

[Landasan Teori 3](#_Toc99945007)

[1.1 Branch and Bound 3](#_Toc99945008)

[1.2 15-Puzzle 3](#_Toc99945009)

[BAB II 4](#_Toc99945010)

[Program dan Spesifikasi Tugas 4](#_Toc99945011)

[2.1 Spesifikasi Tugas 4](#_Toc99945012)

[2.2 Implementasi Branch and Bound pada 15-Puzzle 5](#_Toc99945013)

[2.3 Source Code 6](#_Toc99945014)

[2.4 Penjelasan Fungsi-fungsi dan Kelas yang terdapat pada Program 11](#_Toc99945015)

[2.5 Screenshoot Hasil Program 12](#_Toc99945016)

[BAB III 49](#_Toc99945017)

[Penutup 49](#_Toc99945018)

[3.1 Saran 49](#_Toc99945019)

[3.2 Kesimpulan 49](#_Toc99945020)

[DAFTAR PUSTAKA 50](#_Toc99945021)

[Lampiran 51](#_Toc99945022)

# BAB I

# Landasan Teori

## 1.1 Branch and Bound

Branch and bound adalah paradigma desain algoritma yang umumnya digunakan untuk memecahkan masalah optimasi kombinatorial. Masalah-masalah ini biasanya eksponensial dalam hal kompleksitas waktu dan mungkin perlu mengeksplorasi semua kemungkinan permutasi dalam kasus terburuk. Teknik Algoritma Cabang dan Terikat memecahkan masalah ini dengan relatif cepat. Algoritma Branch and Bound adalah suatu algoritma pencarian solusi yang digunakan untuk persoalan optimisasi. Lebih rinci untuk meminimalkan atau memaksimalkan suatu fungsi objektif sambil tidak melanggar batasan persoalan. Algoritma ini memiliki suatu fungsi pembatasan yang berguna untuk “memangkas” jalur yang dianggap tidak mengarah ke suatu solusi.

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka stop.
2. Jika Q kosong, Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai nilai ‘cost’ ĉ(i) paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan. Jika satu solusi yang diinginkan, maka stop. Pada persoalan optimasi dengan pendekatan least cost search, periksa cost semua simpul hidup. Jika cost nya lebih besar dari cost simpul solusi, maka matikan simpul tersebut.
5. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
6. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
7. Kembali ke langkah 2.

## 1.2 15-Puzzle

15-Puzzle adalah teka-teki geser yang memiliki 15 ubin persegi bernomor 1–15 dalam bingkai dengan tinggi 4 ubin dan lebar 4 ubin, menyisakan satu ubin kosong(pada program ini digunakan angka 16). posisi ubin. Ubin di baris atau kolom yang sama dari posisi terbuka dapat dipindahkan dengan menggesernya secara horizontal(kanan atau kiri) atau vertikal(atas atau bawah). Tujuan dari teka-teki adalah untuk menempatkan ubin dalam urutan numerik. Dinamakan berdasarkan jumlah ubin dalam bingkai, teka-teki 15 juga dapat disebut teka-teki 16, mengacu pada total kapasitas ubinnya.

Teka-teki n adalah masalah klasik untuk algoritma pemodelan yang melibatkan heuristik. Heuristik yang umum digunakan untuk masalah ini termasuk menghitung jumlah ubin yang salah tempat dan menemukan jumlah jarak taksi antara setiap blok dan posisinya dalam konfigurasi tujuan. Perhatikan bahwa keduanya dapat diterima, yaitu mereka tidak pernah melebih-lebihkan jumlah gerakan yang tersisa, yang memastikan optimalitas untuk algoritme pencarian tertentu

# BAB II

# Program dan Spesifikasi Tugas

## 2.1 Spesifikasi Tugas

Buatlah program dalam Java/Python untuk menyelesaikan persoalan 15-Puzzle dengan menggunakan Algoritma Branch and Bound seperti pada materi kuliah. Nilai bound tiap simpul adalah penjumlahan cost yang diperlukan untuk sampai suatu simpul x dari akar, dengan taksiran cost simpul x untuk sampai ke goal. Taksiran cost yang digunakan adalah jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir (goal state). Untuk semua instansiasi persoalan 15-puzzle, susunan akhir yang diinginkan sesuai dengan Gambar 1.

Table

Description automatically generated with medium confidence

Gambar 2.1 Susunan Akhir Persoalan 15-Puzzle

Masukan: matriks yang merepresentasikan posisi awal suatu instansiasi persoalan 15-puzzle. Posisi awal 15-puzsle dibangkitkan secara acak oleh program dan/atau dimasukkan dari file teks. Program harus dapat menentukan apakah posisi awal suatu masukan dapat diselesaikan hingga mencapai susunan akhir, dengan mengimplementasikan fungsi Kurang(i) dan posisi ubin kosong di kondisi awal (X), seperti pada materi kuliah. Jika posisi awal tidak bisa mencapai susunan akhir, program akan menampilkan pesan tidak bisa diselesaikan,. Jika dapat diselesaikan, program dapat menampilkan urutan matriks rute (path) aksi yang dilakukan dari posisi awal ke susunan akhir. Sebagai contoh pada Gambar 2, matriks yang ditampilkan ke layar adalah matriks pada simpul 1, simpul 4, simpul 10 dan simpul 23. Luaran:

1. Matriks posisi awal 15-puzzle.
2. Nilai dari fungsi Kurang (i) untuk setiap ubin tidak kosong pada posisi awal (nilai ini tetap dikeluarkan, baik persoalan bisa diselesaikan atau tidak bisa diselesaikan).
3. Nilai dari
4. Jika persoalan tidak dapat diselesaikan (berdasarkan hasil butir 2) keluar pesan.
5. Jika persoalan dapat diselesaikan(berdasarkan hasil butir 2), menampilkan urutan matriks dari posisi awal ke posisi akhir seperti pada penjelasan sebelumnya.
6. Waktu eksekusi program (diluar baca input/tuis output)
7. Jumlah simpul yang dibangkitkan di dalam pohon ruang status pencarian.

Diagram, engineering drawing

Description automatically generated

Gambar 2.2 Contoh Pohon Ruang Status Persoalan 15-Puzzle

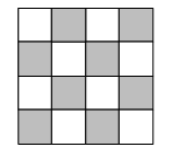
Data Uji: Buatlah 5 buah instansiasi persoalan 15-puzzle, dengan 2 kasus tidak dapat diselesaikan dan 3 kasus yang dapat diselesaikan. Instansiasi persoalan juga disertakan dalam pengumpulan Tucil 3.

Bonus: Program dibuat secara interaktif dengan graphical user interface (GUI), dapat menampilkan gambar 15-puzzle grafis yang menarik seperti di bawah ini, dan dapat menampilkan pergeseran ubin.

## 2.2 Implementasi Branch and Bound pada 15-Puzzle

Penjelasan singkat program :

1. Program menerima matriks yang berukuran 4x4(dari file masukan eksternal/input manual ).
2. Matriks tersebut akan di baca dan akan di cek apakah dapat di selesaikan atau tidak.
3. Pengecekan dilakukan dengan mengecek nilai dari fungsi kurang(i) keseluruhan dengan ditambah X ,dimana
4. Fungsi kurang(i) adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i),sedangkan X adalah nilai yang muncul akibat dari posisi blank pada puzzle, jika blank puzzle berada pada daerah yang diarsir maka X akan bernilai 1 dan jika tidak maka bernilai 0. Pada gambar ini tergambarkan jika blank berada pada (index row+index col)%2==0 maka X bernilai 0, jika (index row+index col)%2!=0 maka X bernilai 1 .

Gambar 2.3 Posisi Blank pada Puzzle

1. Jika nilai dari adalah genap maka puzzle dapat diselesaikan. Jika bernilai ganjil, maka tidak dapat diselesaikan.
2. Jika dapat diselesaikan, maka matriks tersebut dijadikan sebagai elemen simpul akar dari tree Branch and Bound. Lalu setelah itu akan dibangkitkan simpul lainnya yang merupakan child dari simpul akar saat ini, dengan menggunakan sistem priority queue(antrian).Antrian ini akan memprioritaskan nilai fungsi cost terkecil.
3. Simpul-simpul yang dibangkitkan tersebut berdasarkan arah pergeserah ubin kosong(blank) atau dalam program ini ditandai dengan 16. Pergerakan yang diatur dalam program ini secara berurutan adalah up,right,down,left. Pada proses pergerakan ini harus berhati-hati agar pergerakan tidak bolak-balik(naik lalu turun, atau kanan lalu kiri, karena akan sama saja balik ke posisi awal). Selain itu juga harus diperhatikan, apakah posisi blank(yang ditandai dengan 16) masih berada di dalam index matrix yang aman untuk digeser sesuai pergerakan yang di mau.
4. Semua simpul yang telah dibangkitkan akan di masukkan ke dalam priority queue dengan prioritas adalah nilai cost yang lebih kecil.
5. Prinsipnya tetap sama seperti queue pada biasanya yakni First In First Out, namun dengan adanya tambahan parameter cost sebagai prioritas.
6. Melakukan aksi pop elemen, apabila matriks simpul yang dipop dari PrioQueue maka puzzle sudah terselesaikan. Jika berbeda,ulangi Langkah ke -6.
7. Untuk setiap simpul yang di pop akan di masukkan ke dalam visited agar tidak terjadi pemeriksaan terhadap simpula yang sudah dilewati(menghindari pembangkitan simpul dengan matriks duplikat.

Text, letter

Description automatically generated

Gambar 2. Pendekatan fungsi cost untuk persoalan

## 2.3 Source Code

import copy

import time

import numpy as np

import queue

# Jika menginginkan ukuran puzzle lebih besar

N=4 #Ubah angka pada N= sesuai ukuran puzzle,pada tugas ini diminta 15-Puzzle

final = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]] #Puzzle akhir

matrix=[[0 for i in range(N)] for i in range (N)] #Membuat array yang berisikan 0 sebanyak 16 elemen

def getBlankIndex(matrix):

    for i in range(N):

        for j in range(N):

            if(matrix[i][j]==16):

                return [i,j]

def kurang(matrix):

    index=getBlankIndex(matrix)

    mat1 = []

    # ubah jadi 1d

    for a in matrix:

        for b in a:

            mat1.append(b)

    matrix = mat1

    kurang=[]

    pos=0

    sigma\_kurang=0

    for i in range(N\*N):

        tmp=matrix[i]

        count\_kurang=0

        for j in range(i+1,16):

            if(tmp > matrix[j] and tmp!=0):

                sigma\_kurang+=1

    if((index[0]+index[1])%2==0):

        sigma\_kurang=sigma\_kurang+0

    else :

        sigma\_kurang=sigma\_kurang+1

    return (sigma\_kurang)

def display\_kurang(matrix):

    index=getBlankIndex(matrix)

    mat1 = []

    # ubah jadi 1d

    for a in matrix:

        for b in a:

            mat1.append(b)

    matrix = mat1

    kurang=[]

    pos=0

    for i in range(N\*N):

        tmp=matrix[i]

        count\_kurang=0

        for j in range(i+1,16):

            if(tmp > matrix[j] and tmp!=0):

                count\_kurang += 1

        kurang+=[[tmp,count\_kurang]]

    kurang.sort() #sort fungsi kurang ke-i

    for k in range(N\*N):

        print("Nilai dari fungsi Kurang(" + str(kurang[k][0])+"):" ,kurang[k][1])

def cost(matrix): #tingkat start dari 1

    count = 0

    for i in range(N):

        for j in range(N):

            if((matrix[i][j] != final[i][j]) and matrix[i][j] != 16):

                count += 1

    return (count)

def punyasolusi(matrix):

    if(kurang(matrix)%2==0):

        return True

    else:

        return False

class Node:

    def \_\_init\_\_(self, parent, matrix, cost, blank, level,command):

        #parent

        self.parent = parent

        #matrix

        self.matrix = matrix

        # fungsi cost

        self.cost = cost

        # posisi block kosong

        self.blank = blank

        # level nodenya

        self.level = level

        # arah geraknya

        self.command= command

    def \_\_lt\_\_(self, other):

        return(self.cost+self.level <= other.cost + other.level)

def isOnMatrix(blank):

    return 0 <= blank[0] < N and 0 <= blank[1] < N

def display(matrix):

    print('\n'.join(['\t'.join([str(cell) for cell in j]) for j in matrix]))

def print\_path(root):

    if root is None:

        return

    print\_path(root.parent)

    print("--------------------------")

    display(root.matrix)

    print("Command : ",root.command,"\nLevel : " ,root.level)

    print("--------------------------")

    print()

def solve(matrix,blank):

    #up, right, down, left

    #pergerakan blank space

    # untuk row -1 berarti geser keatas,1 geser kebawah,0 diam

    # untuk col -1 berarti geser kekiri,1 geser kekanan,0 diam

    #[[up],[right],[down],[left]]-> dengan susunan [row,col]

    move=[[-1,0],[0,1],[1,0],[0,-1]]

    visited = set()

    nodes = queue.PriorityQueue()

    visited.add(tuple(np.reshape(matrix,16)))

    totalnodes=1

    #root node

    rootcost = cost(matrix) # root cost

    root = Node(None,matrix, rootcost, blank, 0,"Posisi Awal")

    nodes.put(root)

    if(punyasolusi(matrix)):

        print("Susunan Matrix Awal: ")

        print("--------------------------")

        display(matrix)

        print("--------------------------")

        print()

        print("Nilai dari fungsi Kurang (i) + X pada posisi awal :",kurang(matrix))

        display\_kurang(matrix)

        print("Puzzle Dapat Diselesaikan\n")

        start = time.time()

        while not nodes.empty() :

            current = nodes.get()

            if (current.cost) == 0:

                stop = time.time()

                print("Terdapat "+ str(current.level)+" Langkah Untuk Menyelesaikan Puzzle dari Posisi Awal(Posisi Awal tidak dihitung sebagai langkah)")

                print("")

                print("Berikut Langkah Menyelesaikan : ")

                print\_path(current)

                print("Level Akhir: ",current.level)

                print("Waktu yang dibutuhkan: ",stop - start)

                print("Jumlah Simpul yang dibangkitkan:", totalnodes)

                print("-------------------------FINISH-----------------------")

                break

            for i in range(0,4): #Ada 4 move

                #posisi 16(blank) berdasarkan array arah gerak row dan col

                child\_blank = [current.blank[0] +move[i][0],current.blank[1] + move[i][1]]

                dir=["Up","Right","Down","Left"]

                if isOnMatrix(child\_blank):

                    mat = copy.deepcopy(current.matrix)

                    #swap element 16(blank) dengan elemen yang berada pada kiri atau kanan atau atas atau bawah(sesuai arah gerak)

                    temp = mat[child\_blank[0]][child\_blank[1]]

                    mat[child\_blank[0]][child\_blank[1]] = 16

                    mat[current.blank[0]][current.blank[1]] = temp

                    if tuple(np.reshape(mat,16)) not in visited:

                        visited.add(tuple(np.reshape(mat,16)))

                        totalnodes+=1

                        child\_cost = cost(mat)

                        child = Node(current, mat, child\_cost, child\_blank, current.level + 1,dir[i])

                        nodes.put(child)

    else:

        print("Susunan Matrix Awal: ")

        print("--------------------------")

        display(matrix)

        print("--------------------------")

        print()

        print("Nilai dari fungsi Kurang (i) + X pada posisi awal :",kurang(matrix))

        display\_kurang(matrix)

        print("Puzzle Tidak Dapat Diselesaikan")

def bacaFile(namaFolder,namaFile):

    with open((namaFolder+"/"+namaFile+ str(".txt")), 'r') as f:

        matrix = [[int(i) for i in line.split(' ')] for line in f]

    return matrix

def writeFile(namaFolder,namaFile):

    with open("testcase/tc10.txt", 'w') as f:

        for i in range (N) :

            row=[0 for k in range(N)]

            for j in range(N):

                elemen=int(input("Elemen Matriks Baris ke- "+str(i+1)+" Kolom ke-" + str(j+1)+ " : " ))

                if (j==3) :

                    f.write(str(elemen))

                else :

                    f.write(str(elemen)+' ')

            if(i!=3):

                f.write('\n')

    matrix=bacaFile("testcase", "tc10")

    return matrix

def main() :

    print("Cara Memasukkan Puzzle: ")

    print("1. From File")

    print("2. Input Manual")

    cara=int(input("Pilihlah cara memasukkan puzzle : "))

    if(cara==1):

        matrix=bacaFile("testcase",str(input("Masukkan Nama File (tanpa format (.txt)) : ")))

        #Jika ingin memasukkan file input dari folder lain

        #matrix=bacaFile(str(input("Masukkan Nama Folder: ")),str(input("Masukkan Nama File (tanpa format (.txt)) : ")))

    if(cara==2):

        print("Anda akan menuliskan puzzle di file pilihan anda.")

        matrix = writeFile("testcase",str(input("Masukkan Nama File (tanpa format (.txt)) : ")))

        #Jika ingin memasukkan file input dari folder lain

        #matrix=bacaFile(str(input("Masukkan Nama Folder: ")),str(input("Masukkan Nama File (tanpa format (.txt)) : ")))

    blank=getBlankIndex(matrix)

    solve(matrix,blank)

    lanjut=str(input("Apakah ingin menyelesaikan puzzle lainnya? Y/N : "  ))

    if(lanjut=="Y"):

        main()

    else :

        print("Terimakasih telah bermain game 15-Puzzle ini")

main()

## 2.4 Penjelasan Fungsi-fungsi dan Kelas yang terdapat pada Program

1. Class Node

Class Node ini adalah kelas yang akan membuat setiap Node yang diperiksa lengkap dengan atribut Node tersebut. Class node ini memiliki beberapa atribut, yakni parent,matrix,cost,blank,level,command. Pada inisialisasi, setiap atribut akan di inisialisasi sesuai dengan parameter awalnya. Pada node ini juga akan membuat fungsi lt, yang berguna sebagai priority untuk masuk ke dalam queue(antrian). Pada program ini,yang menjadi prioritas adalah fungsi cost + level terkecil.

1. Fungsi getBlankIndex(matrix)

Fungsi ini memiliki parameter matrix (array 2d), dengan tujuan untuk mencari posisi blank pada puzzle, atau pada program ini digunakan “16” untuk menggambarkan blank pada puzzle. Fungsi ini akan mengembalikan posisi blank dengan [i,j], i berarti row blank dan j berarti column blank.

1. Fungsi kurang(matrix)

Fungsi ini memiliki parameter berupa matrix (array 2d) dan akan mengembalikan nilai sigma dari fungsi kurang(i) +X.

1. Fungsi display\_kurang(matrix)

Fungsi ini memiliki parameter berupa matrix (array 2d) dan akan melakukan print fungsi kurang(i) untuk setiap i yang berada pada puzzle.

1. Fungsi cost(matrix)

Fungsi ini memiliki parameter berupa matrix (array 2d) dan akan menghitung cost dari puzzle, namun untuk blank tidak perlu dihitung. Fungsi cost sendiri adalah fungsi yang menunjukkan posisi ubin yang belum sesuai posisi yang diinginkan pada matrix final (hasil akhir),namun blank tidak perlu dihitung. Fungsi ini akan mereturn nilai dari fungsi cost dari suatu matrix (array 2d).

1. Fungsi punyasolusi(matrix)

Fungsi ini memiliki parameter matrix (array 2d) dan akan mengembalikan true jika fungsi kurang dari matrix (array 2d) tersebut merupakan bilangan genap dan false jika merupakan bilangan ganjil.

1. Fungsi isOnMatrix(blank)

Fungsi ini memiliki parameter berupa blank,dimana blank sendiri adalah array yang berisikan [row,col] yang menunjukkan posisi dari blank. Fungsi ini akan memeriksa apakah blank masih berada di dalam range row col dari matrix atau tidak (dalam kasus program ini adalah 4x4).Fungsi ini akan mereturn true jika blank masih berada di dalam matrix. Fungsi ini akan digunakan saat memeriksa apakah pergerakan masih valid (masih berada di dalam matrix).

1. Fungsi display(matrix)

Fungsi display memiliki parameter matrix (array 2d) fungsi ini akan melakukan print terhadap setiap elemen matrix.

1. Fungsi print\_path(root)

Fungsi ini memiliki parameter berupa root dan akan melakukan print terhadap setiap matrix yang menjadi rute untuk ke solusi(goal). Pada fungsi ini juga akan ditampilkan level serta command yang dilakukan.

1. Fungsi solve (matrix,blank)

Fungsi ini memiliki parameter berupa matrix (array 2d) serta blank (array) fungsi ini akan menyelesaikan puzzle dengan cara branch and bound.

1. Fungsi bacaFile (namaFolder,namaFile)

Fungsi ini berparameter namaFile yang akan mengembalikan matrix yang tersimpan di dalam file yang memiliki nama adalah namaFile. Setiap line pada file tersebut akan dibaca dan di masukkan ke dalam matrix. Pada program ini juga disediakan jika ingin membaca file dari folder yang berbeda.

1. Fungsi writeFile(namaFolder,namaFile)

Fungsi ini berparameter namaFolder dan namaFile, fungsi ini akan membantu saat pengguna menuliskan puzzle secara manual ke dalam suatu file tertentu.

1. Fungsi main ()

Fungsi ini adalah fungsi utama yang akan menjalankan program dengan memanggil fungsi-fungsi diatas. Pada fungsi ini akan melakukan print-print untuk mempercantik tampilan. Pada Main ini, juga terdapat 2 pilihan cara menginputkan puzzle,yakni melalui file dan input manual. Selain itu terdapat pilihan untuk melakukan pencarian kembali.

## 2.5 Screenshoot Hasil Program

* 1. Text

     Description automatically generatedInput dari File : testcase/tc1.txt
  2. Input dari File : testcase/tc2.txt

Text

Description automatically generated

* 1. Input dari File : testcase/tc3.txt

Text

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedText

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated Text

Description automatically generated

* 1. Input dari File : testcase/tc4.txt

Text

Description automatically generated

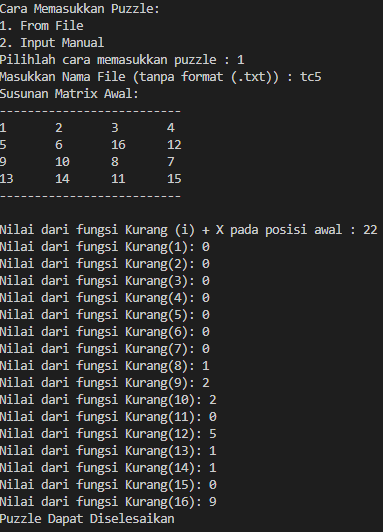
Text

Description automatically generated

Text

Description automatically generated

* 1. Input dari File : testcase/tc5.txt



Text

Description automatically generated

Graphical user interface

Description automatically generated

Text

Description automatically generated

* 1. Input dari File : testcase/tc6.txt

Text

Description automatically generated

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with low confidence

Text

Description automatically generated

* 1. Text

     Description automatically generatedInput dari File : testcase/tc7.txt

Graphical user interface

Description automatically generatedA screenshot of a computer screen

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated Graphical user interface

Description automatically generated with medium confidence A screenshot of a computer screen

Description automatically generated with medium confidence Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated

Text

Description automatically generated

* 1. Text

     Description automatically generatedInput dari File : testcase/tc8.txt

Text

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated

A screenshot of a computer screen

Description automatically generated with medium confidence Graphical user interface

Description automatically generated A picture containing calendar

Description automatically generated Text

Description automatically generated

* 1. Text

     Description automatically generatedInput dari File : testcase/tc9.txt

Text

Description automatically generated

A screenshot of a computer screen

Description automatically generated with medium confidence A screenshot of a computer screen

Description automatically generated with medium confidence Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated A picture containing graphical user interface

Description automatically generated Text

Description automatically generated with low confidence

* 1. Input manual dengan menuliskan pada file : testcase/tc10

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence Graphical user interface

Description automatically generated A screenshot of a computer screen

Description automatically generated with medium confidence Graphical user interface

Description automatically generated A screenshot of a computer screen

Description automatically generated with medium confidence Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated with medium confidence Text

Description automatically generated

# BAB III

# Penutup

## 3.1 Saran

Tidak ada saran untuk tugas kecil 3 ini, spesifikasi sudah cukup, waktu pengerjaan sudah lebih dari cukup. Terimakasih saya ucapkan kepada tim pengajar,yakni dosen dan juga para asisten yang sudah bekerja dengan sangat baik.

## 3.2 Kesimpulan

Algoritma ini cukup efektif dalam menyelesaikan 15-puzzle. Hanya saja diperlukan ide yang lebih efisien saat melakukan pencocokan antara state yang sudah diperiksa(yang berada di visited) dengan state yang akan diperiksa agar dapat menghemat waktu program, mungkin bisa saja dengan melakukan konversi dari matrix menjadi integer yang di ubah ke dalam bentuk 4 bit(nilai 1 elemen kurang dari 16). Namun ide tersebut belum sempat untuk direalisasikan untuk mempercepat proses pencarian. Namun pada program ini sudah dilakukan upaya optimasi pencarian dengan menggunakan set() yang merupakan bawaan dari python untuk melakukan pencarian.

# DAFTAR PUSTAKA

“Algoritma Branch and Bound.” Informatika, <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf>. Accessed 28 Maret 2022.

“Algoritma Branch and Bound.” Informatika, <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian2.pdf>. Accessed 28 Maret 2022.

“Algoritma Branch and Bound.” Informatika, <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian3.pdf>. Accessed 28 Maret 2022.

“Algoritma Branch and Bound.” Informatika, <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian4.pdf>. Accessed 28 Maret 2022.

# Lampiran

Link Repository : https://github.com/StevenSiahaann/Tucil3\_13520145

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi |  |  |
| 1. Program berhasil *running* |  |  |
| 1. Program dapat menerima input dan menuliskan output |  |  |
| 1. Luaran sudah benar untuk semua data uji |  |  |
| 1. Bonus dibuat |  |  |