**TUGAS KECIL**

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

**LAPORAN**

**Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada**

**Semester II**

**Tahun Akademik 2021-2022**

**oleh**

**Owen Christian Wijaya 13520124**

**A picture containing text

Description automatically generated**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2022**

DAFTAR ISI

[BAB I. ALGORITMA *BRANCH AND BOUND* 3](#_Toc99877705)

[BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON 4](#_Toc99877706)

[2.1 algo.py 4](#_Toc99877707)

[2.2 puzzle.py 7](#_Toc99877708)

[2.3 main.py 10](#_Toc99877709)

[2.4 gui.py 12](#_Toc99877710)

[2.5 fparser.py 15](#_Toc99877711)

[BAB III. PENGUJIAN 16](#_Toc99877712)

[3.1 Pengujian 10 Langkah 16](#_Toc99877713)

[3.2 Pengujian 15 Langkah 18](#_Toc99877714)

[3.3 Pengujian 20 Langkah 19](#_Toc99877715)

[3.4 Pengujian File Gagal 1 21](#_Toc99877716)

[3.5 Pengujian File Gagal 2 22](#_Toc99877717)

[3.6 Penggunaan GUI 23](#_Toc99877718)

[BAB IV. *REPOSITORY* 23](#_Toc99877719)

# BAB I. ALGORITMA *BRANCH AND BOUND*

Dalam tugas kecil ini, algoritma *branch and bound* digunakan untuk menentukan langkah-langkah optimal untuk menyelesaikan sebuah permasalahan 15-puzzle. Dalam permainan 15-puzzle, pemain harus menyelesaikan sebuah puzzle untuk mencapai posisi akhir yang diinginkan dengan empat buah gerakan. Algoritma *branch and bound* akan melakukan pencarian terhadap semua kemungkinan gerakan, dan menentukan urutan yang diperlukan untuk menyelesaikan permasalahan tersebut.

Saat melakukan eksekusi, program akan menerima input puzzle dari pengguna baik secara langsung atau dari plainteks dan melakukan validasi. Apabila puzzle tersebut sudah mencapai posisi yang diinginkan atau ada yang invalid, maka program tidak akan melakukan pencarian. Sebaliknya, program akan memulai algoritma dengan melakukan perhitungan nilai “kurang” per ubin. Nilai “kurang” menandakan jumlah ubin yang berada pada posisi lebih dari ubin tertentu namun mempunyai nilai lebih kecil. Setelah itu, nilai jumlah ubin ini akan dijumlahkan. Posisi *empty space* (dirujuk sebagai ES lebih lanjut di dokumen ini) akan diperhitungkan juga untuk menentukan apakah puzzle tersebut dapat diselesaikan. Apabila jumlah nilai “kurang” dan nilai posisi ES puzzle bernilai ganjil, maka program tidak akan menyelesaikan puzzle. Sebaliknya, program akan melakukan inisialisasi algoritma.

Program akan menginisialisasikan sebuah *priority queue* untuk menampung *state* puzzle dan *cost* yang dibutuhkan untuk mencapai *state* tersebut. *Cost* diperhitungkan dari kedalaman *state* tersebut dan jumlah ubin yang terletak di posisi yang salah pada puzzle. Setelah itu, *state* tersebut akan di-*enqueue* ke dalam *priority queue* beserta *cost* dan informasi tentang arah yang diambil. *Priority queue* melakukan *enqueue* berdasarkan prioritas *cost. Cost* yang lebih kecil akan diurutkan di posisi yang lebih depan. Setelah semua selesai diperiksa, program akan melakukan *dequeue* untuk melakukan pengecekan terhadap *state* tersebut. Program akan melakukan *enqueue* terhadap *state* yang menggunakan semua kemungkinan arah kecuali arah yang diambil sebelumnya. Misalkan, apabila *state* yang sedang diperiksa diperoleh dari arah UP, maka program tidak akan meng-*enqueue* state DOWN karena akan kembali ke posisi awal. Selain itu, program akan melakukan pengecekan apabila suatu *state* telah pernah diperiksa sebelumnya menggunakan *dictionary*, sehingga *state* yang sudah pernah diperiksa sebelumnya tidak akan diperiksa lagi. Proses ini akan terus dilakukan sampai antara *priority queue* kosong atau ditemukan *state* selesai.

Setelah proses pencarian selesai, program akan menghubungkan *state-state* yang menghasilkan jawaban dengan melihat nilai ID dari *state* yang ada. Setelah itu, program akan menampilkan hasil urutan, jumlah simpul yang dibangkitkan, dan waktu eksekusi. Untuk GUI, program akan menerima *array* berisi *states* yang kemudian di-*update* per 1 detik untuk menampilkan animasi pergerakan.

# BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON

## 2.1 algo.py

File berisi algoritma *branch and bound* yang digunakan.

|  |
| --- |
| from queue import PriorityQueue  from puzzle import RED\_COLOR, RESET\_COLOR, PuzzleItem, InvItem, Puzzle  import timeit as time  def solve(p):      # displays initial puzzle      print()      print("Initial Puzzle:")      p.show()      print()        outputMessage = ""      res = []      kurangMessage = ""      # displays initial invalid values (kurang[i])      if (not p.isSolved()):          invalid\_pq = PriorityQueue()          for i in range(16):              if (p.buffer[int(i/4)][int(i%4)] == "ES"):                  invalid\_pq.put(InvItem(16, p.invalidPos(i)))              else:                  currentValue = int(p.buffer[int(i/4)][int(i%4)])                  invalid\_pq.put(InvItem(currentValue, p.invalidPos(i)))            kurangMessage += "List of Invalid Values: \n"            while (not invalid\_pq.empty()):              temp = invalid\_pq.get()              if (temp.priority == 16):                  kurangMessage += "Kurang[ES] = {}\n".format(temp.value)              else:                  kurangMessage += "Kurang[{}] = {}\n".format(temp.priority, temp.value)            kurangMessage+= "Sum of invalid values: {}\n".format(p.sumOfInvalidPos())          kurangMessage += "Sum of invalid values and whether empty space in determined position: {}\n".format(              p.sumOfInvalidPos() + p.nullPos())            print(kurangMessage)            if ((p.sumOfInvalidPos() + p.nullPos()) % 2 != 0):              raise Exception(RED\_COLOR + "This puzzle cannot be solved!\n" + RESET\_COLOR)            else:              print("\nSolving puzzle ... ")              prioqueue = PriorityQueue()              p.curr\_depth = 0              p.id = 0              # initializes initial puzzle as first item in queue              prioqueue.put(PuzzleItem(0, [p, "NONE"]))              state\_dict = {}              puzzle\_arr = []              curr\_id = 0              # starts searching process              start\_time = time.default\_timer()                while (prioqueue.qsize() != 0):                  # dequeues item with lowest priority                  puzzleItem = prioqueue.get().item                  temp = puzzleItem[0]                  prev\_direction = puzzleItem[1]                  # adds that item to the list of solved puzzles                  puzzle\_arr.append([temp, prev\_direction])                  curr\_id += 1                    if temp.isSolved():                      stop\_time = time.default\_timer()                      break                    if (temp.checkDir("UP") and prev\_direction != "DOWN"):                      puzzle\_up = Puzzle([x for arr in temp.buffer for x in arr])                      puzzle\_up.shift("UP")                      if (not puzzle\_up.stateExisted(state\_dict)):                          puzzle\_up.curr\_depth = temp.curr\_depth + 1                          puzzle\_up.id = curr\_id                          currCost = puzzle\_up.curr\_depth + puzzle\_up.nonMatchingTile()                          prioqueue.put(PuzzleItem(currCost, [puzzle\_up, "UP"]))                          state\_dict["|".join([x for arr in puzzle\_up.buffer for x in arr])] = True                    if (temp.checkDir("LEFT") and prev\_direction != "RIGHT"):                      puzzle\_left = Puzzle([x for arr in temp.buffer for x in arr])                      puzzle\_left.shift("LEFT")                      if (not puzzle\_left.stateExisted(state\_dict)):                          puzzle\_left.curr\_depth = temp.curr\_depth + 1                          puzzle\_left.id = curr\_id                          currCost = puzzle\_left.curr\_depth + puzzle\_left.nonMatchingTile()                          prioqueue.put(PuzzleItem(currCost, [puzzle\_left, "LEFT"]))                          state\_dict["|".join([x for arr in puzzle\_left.buffer for x in arr])] = True                    if (temp.checkDir("DOWN") and prev\_direction != "UP"):                      puzzle\_down = Puzzle([x for arr in temp.buffer for x in arr])                      puzzle\_down.shift("DOWN")                      if (not puzzle\_down.stateExisted(state\_dict)):                          puzzle\_down.curr\_depth = temp.curr\_depth + 1                          puzzle\_down.id = curr\_id                          currCost = puzzle\_down.curr\_depth + puzzle\_down.nonMatchingTile()                          prioqueue.put(PuzzleItem(currCost, [puzzle\_down, "DOWN"]))                          state\_dict["|".join([x for arr in puzzle\_down.buffer for x in arr])] = True                    if (temp.checkDir("RIGHT") and prev\_direction != "LEFT"):                      puzzle\_right = Puzzle([x for arr in temp.buffer for x in arr])                      puzzle\_right.shift("RIGHT")                      if (not puzzle\_right.stateExisted(state\_dict)):                          puzzle\_right.curr\_depth = temp.curr\_depth + 1                          puzzle\_right.id = curr\_id                          currCost = puzzle\_right.curr\_depth + puzzle\_right.nonMatchingTile()                          prioqueue.put(PuzzleItem(currCost, [puzzle\_right, "RIGHT"]))                          state\_dict["|".join([x for arr in puzzle\_right.buffer for x in arr])] = True              # backtracks parent id to get the path              # of solutions              res = []              puzzle\_elmt = puzzle\_arr[-1]              while (puzzle\_elmt[0].id != 0):                  res = [puzzle\_elmt] + res                  puzzle\_elmt = puzzle\_arr[puzzle\_elmt[0].id - 1]              res = [[p, "NONE"]] + res                # outputs process information              outputMessage += "\nPuzzle solved successfully!"              outputMessage += "\nElapsed time: " + str("%.11f" % (stop\_time - start\_time)) + " seconds"              outputMessage += "\nRaised nodes: " + str(len(state\_dict))              outputMessage += "\nSteps taken : " + str(len(res) - 1)              return kurangMessage, res, outputMessage      else:          raise Exception(RED\_COLOR + "This puzzle is already solved! >:(\n" + RESET\_COLOR) |

## 2.2 puzzle.py

*File* berisi *class* Puzzle yang digunakan, beserta *class PuzzleItem* yang digunakan untuk melakukan *enqueue* terhadap *state puzzle*.

|  |
| --- |
| from dataclasses import dataclass, field  from typing import Any  GREEN\_COLOR = "\u001b[32m"  RED\_COLOR = "\033[91m"  RESET\_COLOR = "\033[0m"  CYAN\_COLOR = "\u001b[36m"  @dataclass(order=True)  class PuzzleItem:      priority: int      item: Any=field(compare=False)    @dataclass(order=True)  class InvItem:      priority: int      value: Any=field(compare=False)    class Puzzle:      # data members      ROW\_SIZE = 4      COL\_SIZE = 4      NULL\_I = 0      NULL\_J = 0      curr\_depth = 0      id = 0      buffer = []        '''      Constructor for the puzzle matrix      '''      def \_\_init\_\_(self, puzzle\_string):          self.buffer = [[0 for \_ in range(self.COL\_SIZE)] for \_ in range(self.ROW\_SIZE)]          for i in range(self.ROW\_SIZE):              for j in range(self.COL\_SIZE):                  elmt = puzzle\_string[i \* self.COL\_SIZE + j]                  self.buffer[i][j] = elmt                  if (elmt == "ES"):                      self.NULL\_I = i                      self.NULL\_J = j        '''      Prints the puzzle matrix in a readable format      '''      def show(self):          for i in range(self.ROW\_SIZE):              for j in range(self.COL\_SIZE):                  if (self.buffer[i][j] == "ES"):                      print(GREEN\_COLOR + self.buffer[i][j] + RESET\_COLOR, end="")                  else:                      print(self.buffer[i][j], end = " ")              print()        '''      Checks possible movement directions for the current position      '''      def checkDir(self, direction):          i = self.NULL\_I          j = self.NULL\_J          if (direction == "LEFT"):              return j != 0 and self.buffer[i][j - 1] != "ES"          elif (direction == "RIGHT"):              return (j != self.COL\_SIZE - 1) and self.buffer[i][j + 1] != "ES"          elif (direction == "UP"):              return i != 0 and self.buffer[i - 1][j] != "ES"          elif (direction == "DOWN"):              return (i != self.ROW\_SIZE - 1) and self.buffer[i + 1][j] != "ES"        '''      Shifts element of the puzzle matrix      '''      def shift(self, direction):          i = self.NULL\_I          j = self.NULL\_J          if (self.checkDir(direction)):              if (direction == "LEFT"): # NULL goes left                  self.buffer[i][j] = self.buffer[i][j - 1]                  self.buffer[i][j - 1] = "ES"                  self.NULL\_J -= 1              elif (direction == "RIGHT"): # NULL goes right                  self.buffer[i][j] = self.buffer[i][j + 1]                  self.buffer[i][j + 1] = "ES"                  self.NULL\_J += 1              elif (direction == "UP"): # NULL goes up                  self.buffer[i][j] = self.buffer[i - 1][j]                  self.buffer[i - 1][j] = "ES"                  self.NULL\_I -= 1              elif (direction == "DOWN"): # NULL goes down                  self.buffer[i][j] = self.buffer[i + 1][j]                  self.buffer[i + 1][j] = "ES"                  self.NULL\_I += 1        '''      Checks if the puzzle is solved      '''      def isSolved(self):          # return False if last element is not NULL          if (self.buffer[self.ROW\_SIZE - 1][self.COL\_SIZE - 1] != "ES"):              return False            # else, check if all elements are in correct order, except for last element          flattened\_buffer = [x for arr in self.buffer for x in arr]          for i in range(1, len(flattened\_buffer) - 1):              # return False if any element is not in correct order              if (int(flattened\_buffer[i]) != int(flattened\_buffer[i - 1]) + 1):                  return False            # return solved if all is sorted          return True        '''      Returns 1 if:      - odd row and even column      - even row and odd column      '''      def nullPos(self):          return 1 if (self.NULL\_I % 2 != self.NULL\_J % 2) else 0        '''      Counts the appearance of invalid position where      element with less value than current element appears on a higher position      '''      def invalidPos(self, idx):          count = 0          flattened\_buffer = [x for arr in self.buffer for x in arr]          if (flattened\_buffer[idx] == "ES"):              count = self.COL\_SIZE \* self.ROW\_SIZE - idx - 1          for i in range(idx, len(flattened\_buffer)):              if (flattened\_buffer[i] != "ES" and flattened\_buffer[idx] != "ES"):                  if (int(flattened\_buffer[i]) < int(flattened\_buffer[idx]) and i > idx):                      count += 1          return count        '''      Returns the sum of invalid position      '''      def sumOfInvalidPos(self):          sum = 0          for i in range(0, self.ROW\_SIZE \* self.COL\_SIZE):              sum += self.invalidPos(i)          return sum        '''      Counts the appearance of invalid position where      tile position doesn't match the value of the tile      '''      def nonMatchingTile(self):          count = 0          flattened\_buffer = [x for arr in self.buffer for x in arr]          for i in range(0, len(flattened\_buffer)):              if (flattened\_buffer[i] != "ES" and (int(flattened\_buffer[i]) != (i + 1))):                  count += 1          return count;      '''      Checks whether current state of the puzzle has existed before      '''      def stateExisted(self, state\_dict):          state = "|".join([x for arr in self.buffer for x in arr])          return True if state in state\_dict else False |

## 2.3 main.py

*File* berisi *command-line interface* untuk penggunaan di terminal.

|  |
| --- |
| import fparser as fp  import puzzle as pc  import algo  def title():      print(" \_\_\_\_  \_     \_\_\_\_  \_\_\_\_  \_\_\_\_\_ \_\_\_\_  \_\_\_\_  \_\_\_\_  \_           \_  \_\_\_\_ ")      print("/  \_\_\\/ \\ /\\/\_   \\/\_   \\/  \_\_//  \_ \\/  \_\_\\/   \_\\/ \\ /|      / \\/ \_\_\_\\")      print("|  \\/|| | || /   / /   /|  \\  | / \\||  \\/||  /  | |\_||\_\_\_\_\_ | ||    \\")      print("|  \_\_/| \\_/|/   /\_/   /\_|  /\_ | |-|||    /|  \\_ | | ||\\_\_\_\_\| |\\_\_\_ |")      print("\\_/   \\_\_\_\_/\\_\_\_\_/\\_\_\_\_/\\_\_\_\_\_\\\_/ \|\\_/\\_/\\\_\_\_\_/\\_/ \|      \\_/\\_\_\_\_/")      print()      print(pc.GREEN\_COLOR + "A(nother) 15-Puzzle Solver" + pc.RESET\_COLOR)    def main():        while (True):          print("\nSelect your desired input method: ")          print("[1] Text file")          print("[2] Input by user")          print("[0] Exit")            try:                option = int(input("| >> "))                if (option == 1):                  print(pc.CYAN\_COLOR + "\n[SELECTED] Text file" + pc.RESET\_COLOR)                  print("Input your filename (without .txt)|")                  print("[IMPORTANT] File must be included in the test folder!")                  fname = input("| >> ")                  buffer = fp.parseText(fname)              elif (option == 2):                  print(pc.CYAN\_COLOR + "\n[SELECTED] Input by user" + pc.RESET\_COLOR)                  print("Input the desired matrix in a 4 x 4 grid style!")                  print("Fill the empty space character with '-'!")                  buffer = fp.parseInput()              elif (option == 0):                  print("Exiting program...\n")                  break                p = pc.Puzzle(buffer)              \_, res, outputMessage = algo.solve(p)              for i in range(len(res)):                  res[i][0].show()                  print("Step {} | Command: {} \n".format(i, pc.GREEN\_COLOR + res[i][1] + pc.RESET\_COLOR))              print(outputMessage)            except Exception as e:              print(e)              continue  title()  main() |

## 2.4 gui.py

*File* berisi konfigurasi *GUI* menggunakan *library* TKinter.

|  |
| --- |
| from tkinter import \*  from tkinter import messagebox  from algo import solve  from puzzle import \*  from fparser import \*  import time  puzzle\_arr = []  def solveClick():        '''      Function to solve the puzzle from the GUI      '''      global puzzle\_arr      filepath = fname\_entry.get()      try:          if (len(filepath) != 0):              p = Puzzle(parseText(filepath))          else:              p = Puzzle(parseGUI(layout.getBuf()))          kurangMsg, res, outputMsg = solve(p)          ans\_text.configure(text = outputMsg)          kurang\_label.configure(text = kurangMsg)            delay\_time = 0.5          if (time\_entry.get()) != "":              delay\_time = float(time\_entry.get())          layout.renderAll(res, delay\_time)      except Exception as e:          messagebox.showerror("[ERROR]", e)  class GUIPuzzle:      '''      Initialize table for the puzzle      '''      def \_\_init\_\_(self):          for i in range(4):              for j in range(4):                  self.e = Entry(frame, width = 4, font = ('Arial', 20))                  self.e.grid(row = i, column = j)                  self.e.insert(END, "")        '''      Gets buffer value from GUI to be parsed      '''      def getBuf(self):          buffer = ""          for i in range(4):              for j in range(4):                  buffer += frame.grid\_slaves(row = i, column = j)[0].get() + " "          return buffer          '''      Clears table and resets background color      '''      def clear(self):          kurang\_label.configure(text = "")          steps\_label.configure(text = "")          ans\_text.configure(text = "Waiting for search to begin...")          for i in range(4):              for j in range(4):                  frame.grid\_slaves(row = i, column = j)[0].config({"background": "white"})                  frame.grid\_slaves(row = i, column = j)[0].delete(0, END)      '''      Renders a puzzle to the GUI      '''      def render(self, puzzle):          for i in range(4):              for j in range(4):                  self.e = Entry(frame, width = 4, font = ('Arial', 20))                  self.e.grid(row = i, column = j)                  if (puzzle.buffer[i][j] == "ES"):                      self.e.insert(END, "")                      self.e.config({"background": "gray"})                  else:                      self.e.insert(END, puzzle.buffer[i][j])      '''      Renders all puzzles in an array to the GUI      with delay time      '''      def renderAll(self, puzzle\_arr, delay\_time):          global frame          for i in range(len(puzzle\_arr)):              self.render(puzzle\_arr[i][0])              steps\_label.configure(text = "Step " + str(i + 1) + ": " + puzzle\_arr[i][1])              time.sleep(delay\_time)              window.update()  begin\_coord = 150  '''  GUI components  '''  window = Tk()  window.geometry("500x400")  window.minsize(500, 400)  window.maxsize(500, 400)  window.title("Puzzearch-15 Puzzle Solver")  frame = Frame(window)  frame.pack(fill= BOTH, expand= True, padx= 20, pady=20)  fname\_entry = Entry(frame, text = "Input file name (without \*.txt)", font = ('Arial', 10), width = 20)  fname\_entry.place(x = 0, y = begin\_coord + 40)  time\_entry = Entry(frame, text = "Time limit (in seconds)", font = ('Arial', 10), width = 6)  time\_entry.place(x = 200, y = begin\_coord + 40 )  time\_label = Label(frame, text = "Delay\ntime", font = ('Arial', 8))  time\_label.place(x = 170, y = begin\_coord + 30)  layout = GUIPuzzle()  steps\_label = Label(frame, font = ("Arial", 8))  steps\_label.place(x = 0, y = begin\_coord)  fname\_label = Label(frame, font = ("Arial, 8"), text = "Input file name (without \*.txt)")  fname\_label.place(x = 0, y = begin\_coord + 20)  kurang\_label = Label(frame, font = ("Arial, 8"), wraplength = 150)  kurang\_label.place(x = 300, y = 0)  solve\_button = Button(frame, text = "Solve", width = 16, command = solveClick)  solve\_button.place(x = 0, y = begin\_coord + 70)  clear\_button = Button(frame, text = "Clear", width = 16, command = layout.clear)  clear\_button.place(x = 128, y = begin\_coord + 70)  ans\_text = Label(frame, font = ("Arial", 8), text = "Waiting for search to begin...")  ans\_text.place(x = 25, y = begin\_coord + 100)  window.mainloop() |

## 2.5 fparser.py

*File* berisi *script* untuk melakukan *parsing* puzzle.

|  |
| --- |
| import os  def checkValid(arr):      temp = [x for x in (arr)]      temp.remove("ES")      temp.append("16")      temp = [int(x) for x in temp]      temp.sort()      for i in range(len(temp)):          if (int(temp[i]) != i + 1):              raise Exception("[INVALID] Input is not valid!")      return True  def parseText(fname):      dirname = os.path.dirname(\_\_file\_\_)      path = os.path.join(dirname, '../test/')      if (os.path.exists(path + fname + ".txt")):          file = open(path + fname + ".txt", "r")          arr = file.read().replace("-", "ES").replace("\n", " ").split(" ")          return arr if checkValid(arr) else None      else:          raise Exception("[INVALID] File doesn't exist! Make sure it is stored in the 'test' folder and the filename is correct! (without .txt)")    def parseInput():      print("\n[SELECTED] Input by user")      print("Input the desired matrix in a 4 x 4 grid style!")      print("Fill the empty space character with '-'!")      buffer = [[0 for \_ in range(4)] for \_ in range(4)]      for i in range(4):          print("[ROW {}] | >> ".format(i + 1), end = " ")          buffer[i] = list(map(str, input().split()))        flattened\_buffer = ' '.join([x for arr in buffer for x in arr])      arr = flattened\_buffer.replace("-", "ES").replace("\n", " ").split(" ")      return arr if checkValid(arr) else None  def parseGUI(buffer):      arr = buffer.rstrip().replace("-", "ES").replace("\n", " ").split(" ")      return arr if checkValid(arr) else None |

# BAB III. PENGUJIAN

Pengujian dilakukan menggunakan CPU i5-9300H 2.40 GHz, sehingga kemungkinan ada perbedaan waktu eksekusi saat pengujian di komputer lain. Perlu diperhatikan juga bahwa karena pembatasan heuristik pada jumlah ubin yang tidak cocok, program ini akan memakan waktu lebih lama untuk menyelesaikan permasalahan yang membutuhkan penyelesaian di atas 20 langkah. *Test cases* yang diujikan mengambil waktu dari < 1 detik hingga 30 detik untuk penyelesaian 30 langkah.

Untuk pengujian GUI, dilakukan pengujian dari input file (succ1.txt) dan input pengguna. Animasi pada GUI dapat dilihat pada *repository* (link dilampirkan di Bab 4).

## 3.1 Pengujian 10 Langkah

|  |  |
| --- | --- |
| Berkas succ1.txt | |
|  | |
| Daftar nilai Kurang(I) dan total Kurang(I) + X | |
|  | |
| Hasil eksekusi (1) | Hasil Eksekusi (2) |
|  |  |
| Waktu eksekusi dan banyak *node* yang dibangkitkan | |
|  | |

## 3.2 Pengujian 15 Langkah

|  |  |  |
| --- | --- | --- |
| Berkas succ2.txt | | |
|  | | |
| Daftar nilai Kurang(I) dan total Kurang(I) + X | | |
|  | | |
| Hasil eksekusi (1) | Hasil eksekusi (2) | Hasil eksekusi (3) |
|  |  |  |
| Waktu eksekusi dan banyak *node* yang dibangkitkan | | |
|  | | |

## 3.3 Pengujian 20 Langkah

|  |  |
| --- | --- |
| Berkas succ3.txt | |
|  | |
| Daftar nilai Kurang(I) dan total Kurang(I) + X | |
|  | |
| Hasil eksekusi (1) | Hasil eksekusi (2) |
|  |  |
| Hasil eksekusi (3) | Hasil eksekusi (4) |
|  |  |
| Waktu eksekusi dan banyak *node* yang dibangkitkan | |
|  | |

## 3.4 Pengujian File Gagal 1

|  |
| --- |
| Berkas fail1.txt |
|  |
| Hasil pengujian (gagal karena nilai Kurang(i) + X ganjil) |
|  |

## 3.5 Pengujian File Gagal 2

|  |
| --- |
| Berkas fail2.txt |
|  |
| Hasil pengujian (gagal karena nilai Kurang(i) + X ganjil) |
|  |

## 3.6 Penggunaan GUI

|  |  |
| --- | --- |
| Pengujian GUI dengan file succ1.txt (awal) | Hasil GUI (akhir) |
|  |  |
| Pengujian GUI dengan input sendiri (mengikuti file succ2.txt) | Hasil GUI (akhir) |
|  |  |

|  |  |  |
| --- | --- | --- |
| 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 | ✓ |  |

# BAB IV. *REPOSITORY*

*Repository* dapat diakses via https://github.com/clumsyyyy/TucilStima3