**Laporan Tugas Kecil 3**

**IF2211 Strategi Algoritma**

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

****

Disusun Oleh:

Ikmal Alfaozi 13520125

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2021/2022**

**Daftar Isi**

**Algoritma Branch and Bound**

Algoritma Branch and Bound dalam Penyelesaian Persoalan 15-Puzzle

**Kode Program**

Modul Graph.py

Modul main.py

**Data Uji**

**Screenshot Input dan Output Program**

**Alamat Drive Kode Program**

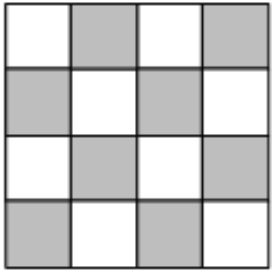
1. **Algoritma Branch and Bound**

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

1. Periksa apakah masukkan 15-puzzle dapat mencapai *goal state* (susunan akhir 15-puzzle) dengan menggunakan .

KURANG(i) adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) adalah posisi ubin bernomor i pada susunan yang diperiksa.

X bernilai 1 jika berada pada sel yang diarsir.



*Goal state* hanya dapat dicapai dari status awal jika hasil dari bernilai genap.

* Jika *goal state* tidak dapat dicapai, maka berhenti dan keluarkan pesan bahwa susunan akhir puzzle tidak dapat dicapai.
* Jika *goal state* dapat dicapai, lanjut ke langkah kedua.

1. Masukkan puzzle sekarang ke dalam himpunan simpul mati dan periksa apakah puzzle sama dengan *goal state*.

* Jika sama, berhenti dan tampilkan proses puzzle dari simpul awal sampai *goal state*.
* Jika tidak sama, lanjut ke langkah ketiga.

1. Bangkitkan simpul anak yang mungkin dari matriks 15-puzzle sekarang.
2. Hitung cost (biaya) tiap simpul anak yang dibangkitkan dengan formula berikut.

|  |
| --- |
| .  panjang lintasan dari simpul akar ke P.  jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. |

1. Masukkan simpul anak ke dalam antrian simpul hidup berdasarkan urutan cost dari terkecil sampai terbesar.
2. Ambil simpul dengan cost terkecil dari antrian simpul hidup dan periksa apakah simpul tersebut ada di himpunan simpul mati.

* Jika simpul tersebut ada di himpunan simpul mati (berarti sudah pernah diperiksa sebelumnya), perika simpul dengan cost terkecil selanjutnya dari antrian simpul hidup.
* Jika simpul tersebut belum ada di himpunan simpul mati, ulangi langkah kedua sampai mencapai *goal state* atau antrian simpul hidupnya kosong.

1. **Kode Program**
2. **Modul Graph.py**

Modul ini berisi kelas Node dan LiveNodeArray. Kelas Node berfungsi sebagai *blueprint* dari objek node, sedangkan kelas LiveNodeArray berfungsi sebagai *blueprint* dari objek antrian simpul hidup.

|  |
| --- |
| class Node:      def \_\_init\_\_(self, parent, puzzle, child, level, cost, blank, previous\_step):          self.parent = parent          self.puzzle = puzzle          self.child = child          self.level = level          self.cost = cost          self.blank = blank # (x, y) blank puzzle          self.previous\_step = previous\_step        def show(self):      # Menampilkan puzzle          for row in self.puzzle:              for col in row:                  if (col != 16):                      if (col < 10):                          print(col,end='   ')                      else:                          print(col, end='  ')                  else:                      print('  ', end='  ')              print()        def isGoal(self):      # Memeriksa apakah node merupakan goal state          goal = True          i = 0          while (i < 4 and goal):              j = 0              while (j < 4 and goal):                  if (self.puzzle[i,j] != i\*4 + j + 1):                      goal = False                  j += 1              i += 1          return goal  class LiveNodeArray:      def \_\_init\_\_(self, array, length):          self.array = array          self.length = length      def append(self, s):      # Menambahkan simpul s ke dalam array berdasarkan prioritasnya (cost)          found = False          i = 0          while (not found and i < self.length):              if (self.array[i].cost >= s.cost):                  found = True              else:                  i += 1          self.array.insert(i, s)          self.length += 1        def pop(self):      # Menghapus dan mengembalikan elemen pertama array          first = self.array[0]          self.array = self.array[1:]          self.length -= 1          return first        def display(self):      # Menampilkan seluruh isi array          for i in range(self.length):              self.array[i].show()              print(self.array[i].cost)              print() |

1. **Modul main.py**

Modul ini berisi prosedur-prosedur dan fungsi-fungsi untuk menjalankan program utama.

|  |
| --- |
| from copy import deepcopy  import random  import numpy as np  from Graph import \*  from time import time  import tkinter  def randomPuzzle():  # Membuat random puzzle      array = [i for i in range(1,17)]      random.shuffle(array)      puzzle = np.array(array, dtype='int8').reshape(4,4)      return puzzle  def readFile(filename):  # Membaca input file      array = []      f = open(filename,'r')      lines = f.readlines()        for line in lines:          array.append(line.replace('\n','').replace(' ','').split(','))      f.close()      for i in range(4):          for j in range(4):              if (array[i][j] != ''):                  array[i][j] = int(array[i][j])              else:                  array[i][j] = 16        return np.array(array, dtype='int8')  def kurang(m, row, col):  # Realisasi Fungsi KURANG(i)      count = 0      e = m[row, col]      while (row < 4):          while (col < 4):              if (e > m[row, col]):                  count += 1              col += 1          row += 1          col = 0      return count  def achievable(m):  # Memeriksa apakah goal state dapat dicapai dan mengembalikan nilai dari fungsi Kurang(i) untuk setiap ubin      sum = 0      array = np.zeros(16, dtype='int8')      for row in range(4):          for col in range(4):              array[m[row, col] - 1] = kurang(m, row, col)              sum += array[m[row,col]-1]              if (m[row, col] == 16 and (row+col) % 2 == 1):                  sum += 1      return (sum % 2 == 0), sum, array  def searchBlank(m):  # Mencari blok yang kosong pada puzzle      for i in range(4):          for j in range(4):              if (m[i,j] == 16):                  return i,j  def g(m):  # Menghitung jumlah ubin tak kosong yang tidak terdapat pada susunan akhir      count = 0      for i in range(4):          for j in range(4):              if (m[i,j] != 16 and m[i,j] != i\*4+j+1):                  count += 1      return count  def swap(m, row1, col1, row2, col2):  # Menukar posisi elemen matriks      mcopy = deepcopy(m)      mcopy[row1, col1] = m[row2, col2]      mcopy[row2, col2] = m[row1,col1]      return mcopy  def checkNode(matrix, array):  # Memeriksa apakah matrix sudah ada di array      for puzzle in array:          if ((matrix==puzzle).all()):              return True      return False  def Play():  # prosedur untuk gui      global pace      if (pace != 0):          pace = 0      Play1()  def Play1():  # prosedur untuk gui      global pace      play['state'] = 'disabled'      next['state'] = 'disabled'      prev['state'] = 'disabled'      if (pace < len(steps) - 1):          Next()          main\_windows.after(1000,Play1)      else:          play['state'] = 'normal'          next['state'] = 'normal'          prev['state'] = 'normal'  def Next():  # prosedur untuk gui      global pace      if (pace < len(steps) - 1):          pace  += 1          puzzle = steps[pace].puzzle          label1.config(text=puzzle[0][0], background='#7ea098')          label2.config(text=puzzle[0][1], background='#7ea098')          label3.config(text=puzzle[0][2], background='#7ea098')          label4.config(text=puzzle[0][3], background='#7ea098')          label5.config(text=puzzle[1][0], background='#7ea098')          label6.config(text=puzzle[1][1], background='#7ea098')          label7.config(text=puzzle[1][2], background='#7ea098')          label8.config(text=puzzle[1][3], background='#7ea098')          label9.config(text=puzzle[2][0], background='#7ea098')          label10.config(text=puzzle[2][1], background='#7ea098')          label11.config(text=puzzle[2][2], background='#7ea098')          label12.config(text=puzzle[2][3], background='#7ea098')          label13.config(text=puzzle[3][0], background='#7ea098')          label14.config(text=puzzle[3][1], background='#7ea098')          label15.config(text=puzzle[3][2], background='#7ea098')          label16.config(text=puzzle[3][3], background='#7ea098')          i, j = steps[pace].blank          if (i == 0 and j == 0):              label1.config(text='', background="white")          elif (i == 0 and j == 1):              label2.config(text='', background="white")          elif (i == 0 and j == 2):              label3.config(text='', background="white")          elif (i == 0 and j == 3):              label4.config(text='', background="white")          elif (i == 1 and j == 0):              label5.config(text='', background="white")          elif (i == 1 and j == 1):              label6.config(text='', background="white")          elif (i == 1 and j == 2):              label7.config(text='', background="white")          elif (i == 1 and j == 3):              label8.config(text='', background="white")          elif (i == 2 and j == 0):              label9.config(text='', background="white")          elif (i == 2 and j == 1):              label10.config(text='', background="white")          elif (i == 2 and j == 2):              label11.config(text='', background="white")          elif (i == 2 and j == 3):              label12.config(text='', background="white")          elif (i == 3 and j == 0):              label13.config(text='', background="white")          elif (i == 3 and j == 1):              label14.config(text='', background="white")          elif (i == 3 and j == 2):              label15.config(text='', background="white")          elif (i == 3 and j == 3):              label16.config(text='', background="white")  def Prev():  # prosedur untuk gui      global pace      if (pace > 0):          pace -= 1          puzzle = steps[pace].puzzle          label1.config(text=puzzle[0][0], background='#7ea098')          label2.config(text=puzzle[0][1], background='#7ea098')          label3.config(text=puzzle[0][2], background='#7ea098')          label4.config(text=puzzle[0][3], background='#7ea098')          label5.config(text=puzzle[1][0], background='#7ea098')          label6.config(text=puzzle[1][1], background='#7ea098')          label7.config(text=puzzle[1][2], background='#7ea098')          label8.config(text=puzzle[1][3], background='#7ea098')          label9.config(text=puzzle[2][0], background='#7ea098')          label10.config(text=puzzle[2][1], background='#7ea098')          label11.config(text=puzzle[2][2], background='#7ea098')          label12.config(text=puzzle[2][3], background='#7ea098')          label13.config(text=puzzle[3][0], background='#7ea098')          label14.config(text=puzzle[3][1], background='#7ea098')          label15.config(text=puzzle[3][2], background='#7ea098')          label16.config(text=puzzle[3][3], background='#7ea098')            i, j = steps[pace].blank          if (i == 0 and j == 0):              label1.config(text='', background="white")          elif (i == 0 and j == 1):              label2.config(text='', background="white")          elif (i == 0 and j == 2):              label3.config(text='', background="white")          elif (i == 0 and j == 3):              label4.config(text='', background="white")          elif (i == 1 and j == 0):              label5.config(text='', background="white")          elif (i == 1 and j == 1):              label6.config(text='', background="white")          elif (i == 1 and j == 2):              label7.config(text='', background="white")          elif (i == 1 and j == 3):              label8.config(text='', background="white")          elif (i == 2 and j == 0):              label9.config(text='', background="white")          elif (i == 2 and j == 1):              label10.config(text='', background="white")          elif (i == 2 and j == 2):              label11.config(text='', background="white")          elif (i == 2 and j == 3):              label12.config(text='', background="white")          elif (i == 3 and j == 0):              label13.config(text='', background="white")          elif (i == 3 and j == 1):              label14.config(text='', background="white")          elif (i == 3 and j == 2):              label15.config(text='', background="white")          elif (i == 3 and j == 3):              label16.config(text='', background="white")  def main():  # program utama      print("1. Input File")      print("2. Random")      while (True):          try:              select = int(input("Masukkan angka pilihan: "))          except Exception:              print("Masukkan salah")          else:              break      if (select == 1):          while (True):              try:                  filename = input("Masukkan nama file: ")                  puzzle = readFile(filename)              except Exception:                  print("File tidak ditemukan")              else:                  break      else:          puzzle = randomPuzzle()        print("\nMatriks posisi awal 15-puzzle")      blank = searchBlank(puzzle)      root = Node(None, puzzle, None, 0, 0, blank, None)      root.show()      print()      start\_time = time()      solvable, sum, array = achievable(puzzle)      print("Nilai dari fungsi Kurang (i)")      for i in range(16):          print("Kurang({}) = {}".format(i+1, array[i]))      print()      print("Nilai KURANG (i) + X =", sum)      print()      if (not solvable):          print("Status tujuan tidak dapat dicapai")      else:          global main\_windows, play, prev, next, label1, label2, label3, label4, label5, label6, label7, label8, label9, label10, label11, label12, label13, label14, label15, label16, steps, pace          n = 0 # Jumlah simpul yang dibangkitkan          dNode = []          lNode = LiveNodeArray([root], 1)          while (lNode.length != 0):              solution = lNode.pop()              if (not checkNode(solution.puzzle, dNode)):                  dNode.append(solution.puzzle)              else:                  continue              if (solution.isGoal()):                  break              row, col = solution.blank              # geser blok kosong ke atas              if (row != 0 and solution.previous\_step != 'down'):                  m1 = swap(solution.puzzle, row - 1, col, row, col)                  cost = (solution.level + 1) + g(m1)                  s1 =  Node(solution, m1, None, solution.level + 1, cost, (row-1, col), 'up')                  lNode.append(s1)                  n += 1              # geser blok kosong ke kanan              if (col != 3 and solution.previous\_step != 'left'):                  m2 = swap(solution.puzzle, row, col + 1, row, col)                  cost = (solution.level + 1) + g(m2)                  s2 =  Node(solution, m2, None, solution.level + 1, cost, (row, col+1), 'right')                  lNode.append(s2)                  n += 1              # geser blok kosong ke bawah              if (row != 3 and solution.previous\_step != 'up'):                  m3 = swap(solution.puzzle, row + 1, col, row, col)                  cost = (solution.level + 1) + g(m3)                  s3 =  Node(solution, m3, None, solution.level + 1, cost, (row+1, col), 'down')                  lNode.append(s3)                  n += 1              # geser blok kosong ke kiri              if (col != 0 and solution.previous\_step != 'right'):                  m4 = swap(solution.puzzle, row, col-1, row, col)                  cost = (solution.level + 1) + g(m4)                  s4 =  Node(solution, m4, None, solution.level + 1, cost, (row, col-1), 'left')                  lNode.append(s4)                  n += 1          steps = []          p = solution          while (p is not None):              steps.insert(0, p)              p = p.parent            print("Langkah-langkah\n")          for step in steps:              step.show()              print()          execution\_time = time() - start\_time          print("Waktu eksekusi : ", execution\_time)          print("Jumlah Langkah menuju goal state :", solution.level)          print("Jumlah simpul yang dibangkitkan :",n)          main\_windows = tkinter.Tk()          main\_windows.title("PUZZLE PROCESS")          pace = 0          label1 = tkinter.Label(main\_windows, text=puzzle[0,0], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label2 = tkinter.Label(main\_windows, text=puzzle[0,1], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label3 = tkinter.Label(main\_windows, text=puzzle[0,2], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label4 = tkinter.Label(main\_windows, text=puzzle[0,3], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label5 = tkinter.Label(main\_windows, text=puzzle[1,0], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label6 = tkinter.Label(main\_windows, text=puzzle[1,1], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label7 = tkinter.Label(main\_windows, text=puzzle[1,2], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label8 = tkinter.Label(main\_windows, text=puzzle[1,3], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label9 = tkinter.Label(main\_windows, text=puzzle[2,0], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label10 = tkinter.Label(main\_windows, text=puzzle[2,1], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label11 = tkinter.Label(main\_windows, text=puzzle[2,2], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label12 = tkinter.Label(main\_windows, text=puzzle[2,3], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label13 = tkinter.Label(main\_windows, text=puzzle[3,0], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label14 = tkinter.Label(main\_windows, text=puzzle[3,1], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label15 = tkinter.Label(main\_windows, text=puzzle[3,2], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          label16 = tkinter.Label(main\_windows, text=puzzle[3,3], background='#7ea098', foreground="white", width=3, height=1, font=("Arial", 25))          i, j = steps[pace].blank          if (i == 0 and j == 0):              label1.config(text='', background="white")          elif (i == 0 and j == 1):              label2.config(text='', background="white")          elif (i == 0 and j == 2):              label3.config(text='', background="white")          elif (i == 0 and j == 3):              label4.config(text='', background="white")          elif (i == 1 and j == 0):              label5.config(text='', background="white")          elif (i == 1 and j == 1):              label6.config(text='', background="white")          elif (i == 1 and j == 2):              label7.config(text='', background="white")          elif (i == 1 and j == 3):              label8.config(text='', background="white")          elif (i == 2 and j == 0):              label9.config(text='', background="white")          elif (i == 2 and j == 1):              label10.config(text='', background="white")          elif (i == 2 and j == 2):              label11.config(text='', background="white")          elif (i == 2 and j == 3):              label12.config(text='', background="white")          elif (i == 3 and j == 0):              label13.config(text='', background="white")          elif (i == 3 and j == 1):              label14.config(text='', background="white")          elif (i == 3 and j == 2):              label15.config(text='', background="white")          elif (i == 3 and j == 3):              label16.config(text='', background="white")          next = tkinter.Button(main\_windows, text ="NEXT STEP", command = Next)          prev = tkinter.Button(main\_windows, text="PREV STEP", command = Prev)          play = tkinter.Button(main\_windows, text="PLAY", command = Play)          label1.grid(row=0,column=0,padx=1,pady=1)          label2.grid(row=0,column=1,padx=1,pady=1)          label3.grid(row=0,column=2,padx=1,pady=1)          label4.grid(row=0,column=3,padx=1,pady=1)          label5.grid(row=1,column=0,padx=1,pady=1)          label6.grid(row=1,column=1,padx=1,pady=1)          label7.grid(row=1,column=2,padx=1,pady=1)          label8.grid(row=1,column=3,padx=1,pady=1)          label9.grid(row=2,column=0,padx=1,pady=1)          label10.grid(row=2,column=1,padx=1,pady=1)          label11.grid(row=2,column=2,padx=1,pady=1)          label12.grid(row=2,column=3,padx=1,pady=1)          label13.grid(row=3,column=0,padx=1,pady=1)          label14.grid(row=3,column=1,padx=1,pady=1)          label15.grid(row=3,column=2,padx=1,pady=1)          label16.grid(row=3,column=3,padx=1,pady=1)          next.grid(row=4, column=3,padx=1,pady=1)          prev.grid(row=4, column=0,padx=1,pady=1)          prev.grid(row=4, column=0,padx=1,pady=1)          play.grid(row=4, column=1, columnspan=2 ,padx=1,pady=1)          main\_windows.mainloop()  if \_\_name\_\_ == '\_\_main\_\_':      main() |

1. **Data Uji**

Data uji adalah berupa matriks puzzle yang merepresentasikan keadaan awal puzzle. Setiap elemen matriks dipisahkan dengan koma dan sel kosong puzzle direpresentasikan sebagai elemen kosong pada matriks.

1. **Data Uji 1**

14, 11, 4, 12

5, 3, 7, 1

15, 9, 8, 13

2, , 6, 10

1. **Data Uji 2**

14, 5, 7, 11

12, 6, 4, 2

3, 10, 1,

8, 9, 13, 15

1. **Data Uji 3**

10, 5, 2, 4

, 1, 3, 8

6, 14, 7, 12

9, 13, 11, 15

1. **Data Uji 4**

5, 1, 3, 4

9, 2, 7, 8

, 6, 15, 11

13, 10, 14, 12

1. **Data Uji 5**

1, 6, 2, 3

5, 10, 4,

14, 7, 8, 11

9, 13, 15, 12

1. **Screenshot Input dan Output Program**

Data uji berupa teks berada pada folder test, sedangkan program utama ada di file main.py pada folder src. Test case 1 sampai 5 berupa data uji teks, sedangkan test case 6 dan 7 berupa random matriks yang dibangkitkan oleh program.

1. **Test Case 1**

|  |
| --- |
| 1. Input File  2. Random  Masukkan angka pilihan: 1  Masukkan nama file: test/case1.txt  Matriks posisi awal 15-puzzle  14 11 4 12  5 3 7 1  15 9 8 13  2 6 10  Nilai dari fungsi Kurang (i)  Kurang(1) = 0  Kurang(2) = 0  Kurang(3) = 2  Kurang(4) = 3  Kurang(5) = 3  Kurang(6) = 0  Kurang(7) = 3  Kurang(8) = 2  Kurang(9) = 3  Kurang(10) = 0  Kurang(11) = 10  Kurang(12) = 9  Kurang(13) = 3  Kurang(14) = 13  Kurang(15) = 6  Kurang(16) = 2  Nilai KURANG (i) + X = 59  Status tujuan tidak dapat dicapai |

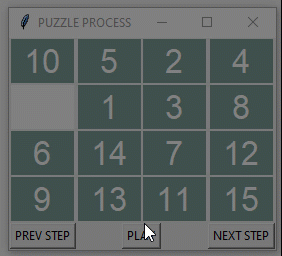
1. **Test Case 2**

|  |
| --- |
| 1. Input File  2. Random  Masukkan angka pilihan: 1  Masukkan nama file: test/case2.txt  Matriks posisi awal 15-puzzle  14 5 7 11  12 6 4 2  3 10 1  8 9 13 15  Nilai dari fungsi Kurang (i)  Kurang(1) = 0  Kurang(2) = 1  Kurang(3) = 1  Kurang(4) = 3  Kurang(5) = 4  Kurang(6) = 4  Kurang(7) = 5  Kurang(8) = 0  Kurang(9) = 0  Kurang(10) = 3  Kurang(11) = 8  Kurang(12) = 8  Kurang(13) = 0  Kurang(14) = 13  Kurang(15) = 0  Kurang(16) = 4  Nilai KURANG (i) + X = 55  Status tujuan tidak dapat dicapai |

1. **Test Case 3**

|  |
| --- |
| 1. Input File  2. Random  Masukkan angka pilihan: 1  Masukkan nama file: test/case3.txt  Matriks posisi awal 15-puzzle  10 5 2 4  1 3 8  6 14 7 12  9 13 11 15  Nilai dari fungsi Kurang (i)  Kurang(1) = 0  Kurang(2) = 1  Kurang(3) = 0  Kurang(4) = 2  Kurang(5) = 4  Kurang(6) = 0  Kurang(7) = 0  Kurang(8) = 2  Kurang(9) = 0  Kurang(10) = 9  Kurang(11) = 0  Kurang(12) = 2  Kurang(13) = 1  Kurang(14) = 5  Kurang(15) = 0  Kurang(16) = 11  Nilai KURANG (i) + X = 38  Langkah-langkah  10 5 2 4  1 3 8  6 14 7 12  9 13 11 15  5 2 4  10 1 3 8  6 14 7 12  9 13 11 15  5 2 4  10 1 3 8  6 14 7 12  9 13 11 15  5 1 2 4  10 3 8  6 14 7 12  9 13 11 15  5 1 2 4  10 3 8  6 14 7 12  9 13 11 15  5 1 2 4  6 10 3 8  14 7 12  9 13 11 15  5 1 2 4  6 10 3 8  9 14 7 12  13 11 15  5 1 2 4  6 10 3 8  9 14 7 12  13 11 15  5 1 2 4  6 10 3 8  9 7 12  13 14 11 15  5 1 2 4  6 3 8  9 10 7 12  13 14 11 15  5 1 2 4  6 3 8  9 10 7 12  13 14 11 15  1 2 4  5 6 3 8  9 10 7 12  13 14 11 15  1 2 4  5 6 3 8  9 10 7 12  13 14 11 15  1 2 4  5 6 3 8  9 10 7 12  13 14 11 15  1 2 3 4  5 6 8  9 10 7 12  13 14 11 15  1 2 3 4  5 6 7 8  9 10 12  13 14 11 15  1 2 3 4  5 6 7 8  9 10 11 12  13 14 15  1 2 3 4  5 6 7 8  9 10 11 12  13 14 15  Waktu eksekusi : 0.39127540588378906  Jumlah Langkah menuju goal state : 17  Jumlah simpul yang dibangkitkan : 795 |





1. **Alamat Drive Kode Program**

[IF2211-Strategi-Algoritma/Tucil1 at main · ikmalalfaozi/IF2211-Strategi-Algoritma (github.com)](https://github.com/ikmalalfaozi/IF2211-Strategi-Algoritma/tree/main/Tucil1)