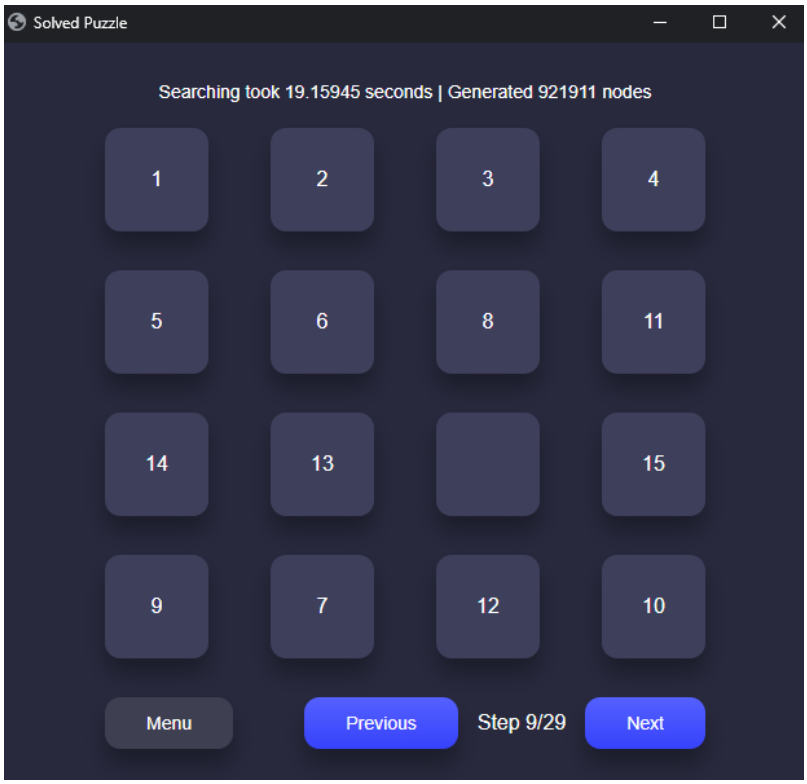
Laporan Tugas Kecil 3 Strategi Algoritma

Marchotridyo/K02/135520119



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

Cara kerja program: *Branch and Bound* yang dibuat dalam menyelesaikan persoalan

Representasi Matriks

Matriks direpresentasikan dalam bentuk array satu dimensi. Contoh, matriks berikut

```
[[2, 3, 4, 16]
[1, 5, 8, 11]
[9, 6, 10, 12]
[13, 14, 7, 15]]
```

akan direpresentasikan sebagai array satu dimensi

```
[2, 3, 4, 16, 1, 5, 8, 11, 9, 6, 10, 12, 13, 14, 7, 15]
```

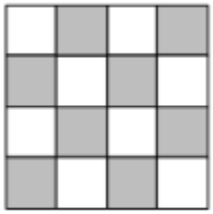
Hal ini ditujukan untuk mempermudah kalkulasi nanti saat melakukan perhitungan nilai kurang dan pergeseran petak kosong pada matriks.

Petak kosong dinyatakan sebagai petak bernilai 16.

Penentuan apakah *puzzle* bisa diselesaikan

Didefinisikan suatu fungsi Kurang(i) sebagai kunci untuk menentukan apakah suatu persoalan bisa diselesaikan atau tidak. i adalah suatu angka [1..16] yang berada pada matriks. Misalkan i disimpan dalam array pada indeks *index_i*. Kurang(i) menghitung banyaknya angka yang disimpan pada indeks > *index_i* yang memenuhi kondisi bahwa angka tersebut lebih kecil dari i.

Selain fungsi Kurang(i), ada juga suatu nilai X yang dapat ditentukan dari posisi awal petak kosong matriks. Rujuk ke gambar di bawah ini:

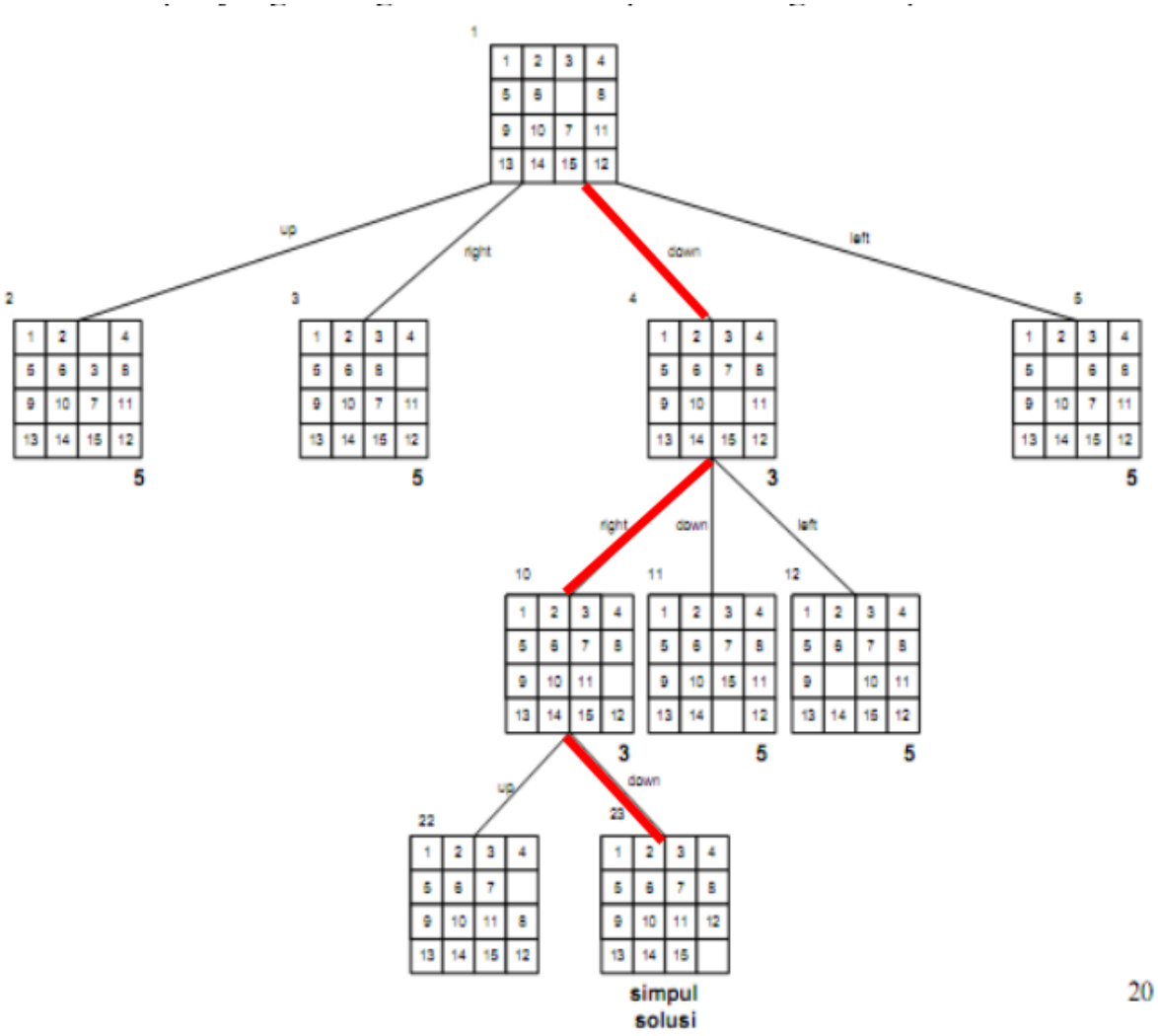


Apabila petak kosong awalnya berada pada sel yang diarsir, nilai X = 1. Jika tidak, nilai X = 0.

Matriks hanya bisa diselesaikan jika nilai penjumlahan semua Kurang(i) dengan i = [1..16] ditambah dengan nilai X bernilai genap. Jika penjumlahan tersebut bernilai ganjil, artinya *puzzle* tidak bisa diselesaikan.

Garis besar algoritma Branch and Bound yang digunakan

Fungsi penaksir *lower bound* $\hat{c}(i)$ didefinisikan sebagai $\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$. $\hat{f}(i)$ menyatakan kedalaman simpul yang sedang diperiksa dan $\hat{g}(i)$ menyatakan banyaknya petak (selain petak kosong) yang berada di posisi yang salah.



Sebagai contoh, akan dihitung $\hat{c}(i)$ dari simpul bernomor 10. Perhatikan bahwa simpul ini terletak di kedalaman 2 ($\hat{f}(i) = 2$) dan petak 12 berada pada posisi yang salah ($\hat{g}(i) = 1$). Sehingga, $\hat{c}(i)$ untuk simpul bernomor 10 bernilai 2 + 1 = 3.

Implementasi Branch and Bound dalam program

Semua simpul dinyatakan sebagai sebuah instantiasi dari kelas *Matrix*. Kelas *Matrix* ini memiliki beberapa atribut,

- *arr* yaitu representasi matriks dalam sebuah array 1 dimensi
- stepsBefore untuk mengetahui langkah-langkah yang digunakan dari akar untuk mencapai kondisi sekarang. Hal ini digunakan untuk mensimulasikan pembentukan pohon
- steps untuk menyimpan kedalaman sekarang (sekaligus nilai dari $f\hat{}(i)$)
- locationMap yaitu sebuah map dengan <key, value> berbentuk <integer, integer>. Misal, locationMap[1] mengembalikan indeks letak petak 1 berada.
- kurangMap yaitu sebuah map dengan <key, value> berbentuk <integer, integer>. Misal, kurangMap[1] mengembalikan nilai Kurang(1)
- _g untuk menyimpan banyaknya petak selain petak kosong yang tidak berada pada tempat seharusnya (sekaligus nilai dari $g\hat{}(i)$)

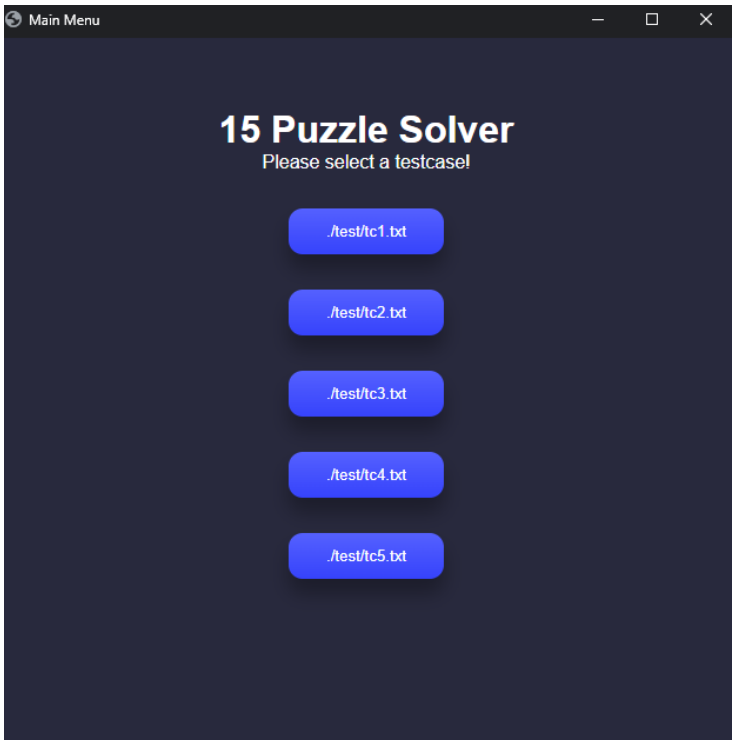
Pembentukan pohon menggunakan modifikasi BFS dengan menggunakan *priority queue*. Simpul dengan *cost* yang lebih kecil akan dimasukkan sebagai *head* dari *queue*. Apabila ada simpul dengan *cost* yang sama, simpul dengan kedalaman yang lebih dalam (nilai $f\hat{}(i)$ lebih besar) dianggap sebagai simpul yang *cost*-nya lebih kecil.

Untuk menghemat kalkulasi, perhitungan $f\hat{}(i)$ dan $g\hat{}(i)$ didasarkan pada kondisi simpul sebelumnya, tidak diiterasi ulang 16 petaknya setiap membangkitkan simpul baru. $f\hat{}(i)$ dari sebuah simpul selalu merupakan +1 dari simpul pembangkitnya. $g\hat{}(i)$ dikalkulasi dengan cara melihat apakah simpul yang ditukar. Apabila simpul yang ditukar sebelumnya sudah berada di posisi yang benar, nilai $g\hat{}(i)$ ditambah satu. Apabila simpul yang ditukar posisi setelah pertukaran menjadi ada di posisi yang benar, nilai $g\hat{}(i)$ dikurangi satu. Jika kedua kondisi tersebut tidak dipenuhi, nilai $g\hat{}(i)$ sama dengan simpul pembangkitnya.

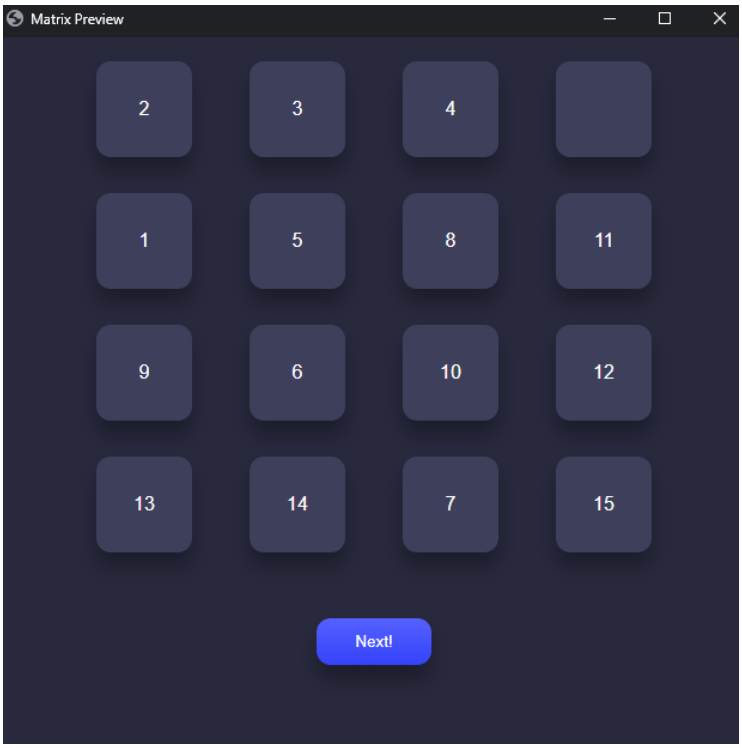
Program akan memberhentikan pencarian ketika sudah ditemukan satu solusi. Program tidak mencari solusi-solusi lainnya dengan banyak langkah yang sama yang mungkin bisa didapatkan.

Screenshot input-output program

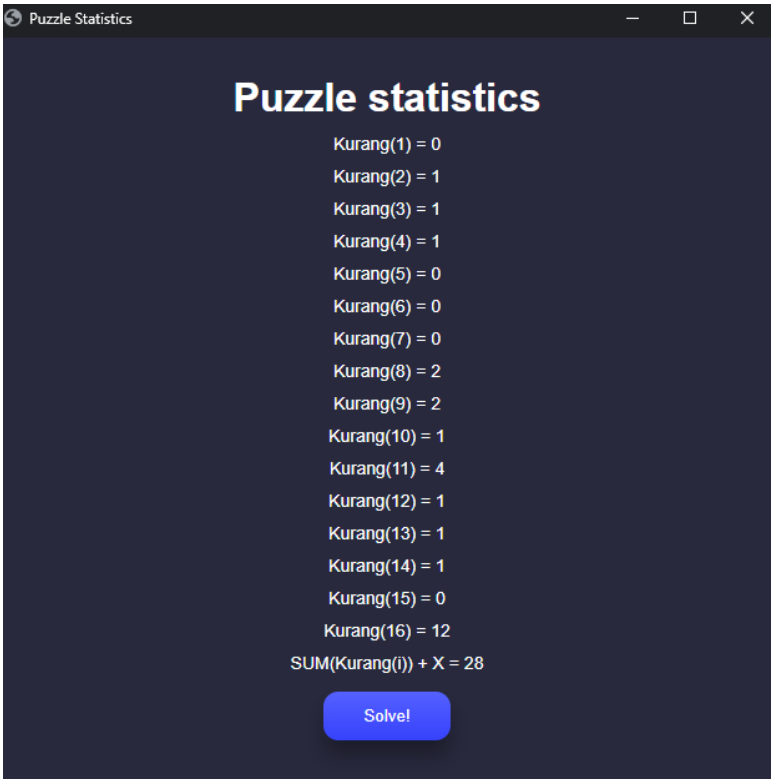
Main menu untuk memilih *test case*



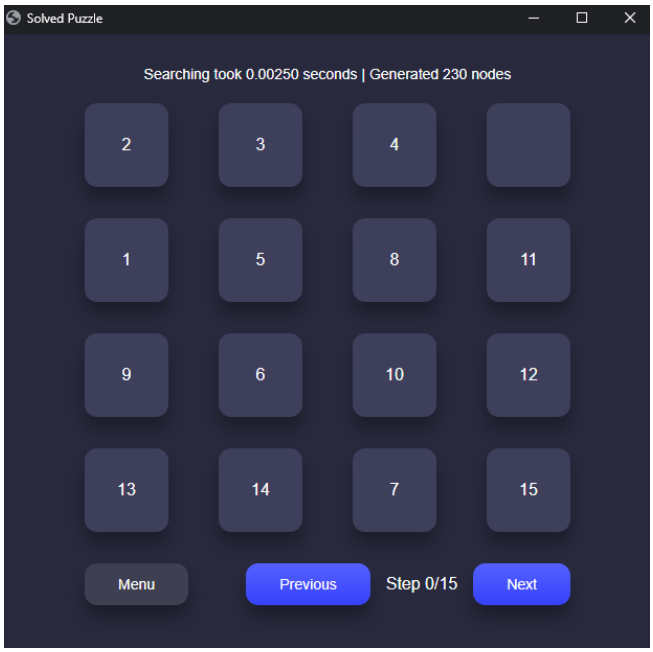
M*atrix preview* setelah memilih salah satu *test case*

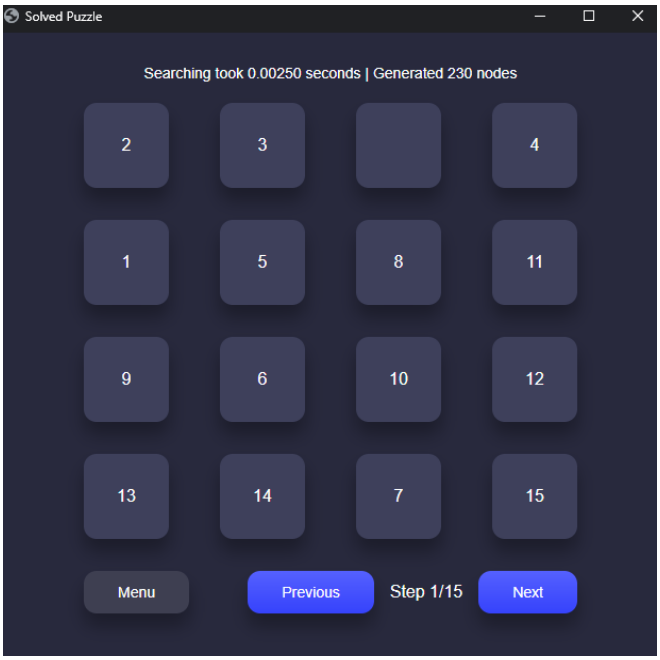


Statistik (nilai fungsi Kurang) dari *test case* yang dipilih

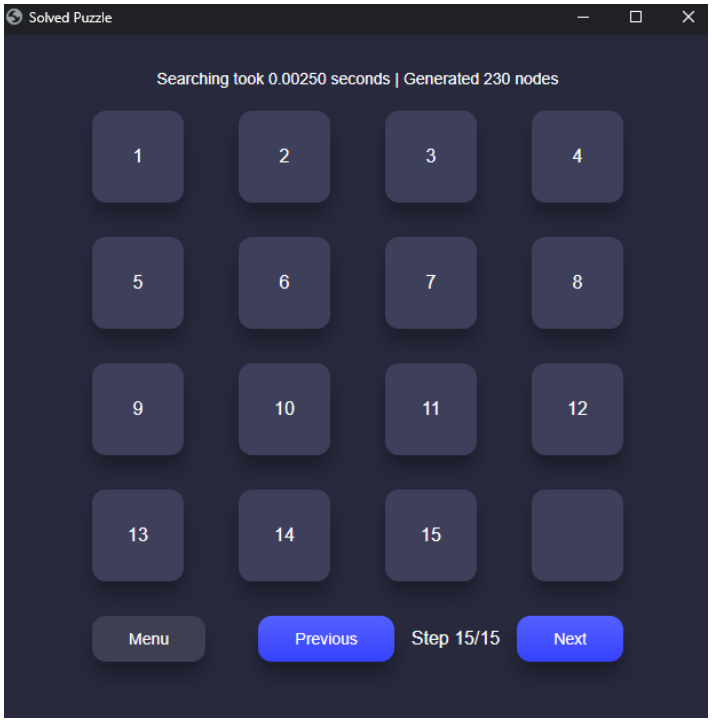


Solusi dari *puzzle* yang diberikan

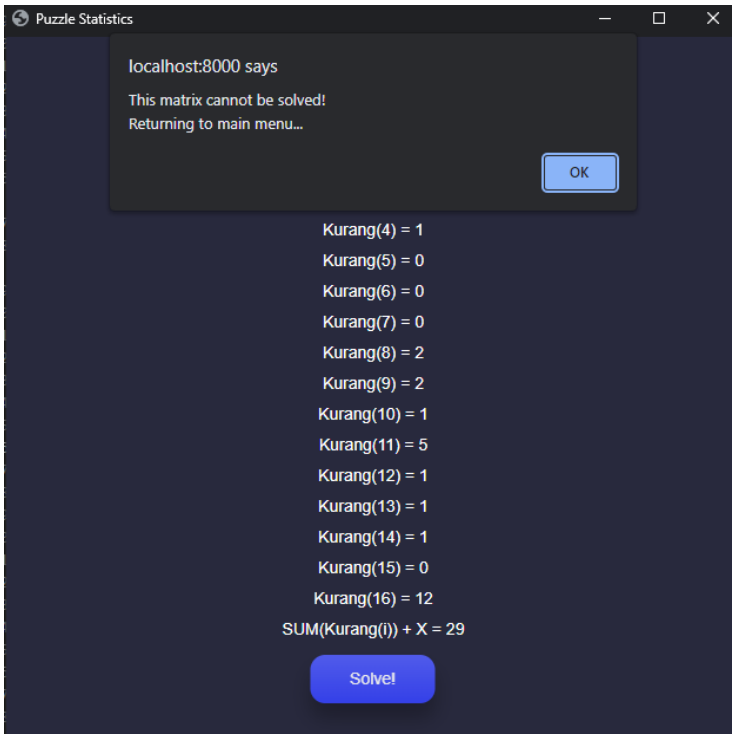




… dan seterusnya sampai step 15



Apabila *puzzle* tidak bisa diselesaikan



Kode program dalam Python

matrix.py

```
import numpy as np

class Matrix:
  def __init__(self, arr = [], stepsBefore = [], steps = 0, _g = -1, loc
= -1):
    '''
      Konstruktor untuk matriks
    '''
    self.arr = arr
    self.stepsBefore = stepsBefore
    self.steps = steps
    self.locationMap = {}
    self.kurangMap = {}
    self.kurangSum = 0
    self._g = _g
    self.loc = loc
    if (self._g == -1):
      self.fillLocationMap()
  def id(self):
    '''
      Mengembalikan ID unik untuk matriks
    '''
    return self.arr.tobytes()
  def __lt__(self, other):
    '''
      Menentukan matriks mana yang costnya lebih kecil.
      Apabila cost matriks sama, matriks yang lebih dalam (f lebih
besar) dianggap lebih kecil.
    '''
    return self.f() >= other.f() if self.c() == other.c() else self.c()
< other.c()
  def fillLocationMap(self):
    '''
      Mengisi locationMap
    '''
```

```
    self._g = 0
    for i in range(16):
      self.locationMap[self.arr[i]] = i
      if (self.arr[i] - 1 != i and self.arr[i] != 16):
        self._g += 1
    self.loc = self.locationMap[16]
  def countAllKurang(self):
    '''
      Menghitung Kurang(i) untuk setiap i
      serta mencetaknya ke layar
    '''
    for i in range(1, 17):
      # Iterasi dari [1 ... 16]
      kurang = 0
      for j in range(1, i):
        # Iterasi dari [1 ... i - 1]
        if self.locationMap[j] > self.locationMap[i]:
          kurang += 1
      self.kurangSum += kurang
      self.kurangMap[i] = kurang
  def getXValue(self):
    '''
      Menghitung nilai dari X
    '''
    pos = self.locationMap[16]
    if pos == 1 or pos == 3 or pos == 4 or pos == 6 or pos == 9 or pos
== 11 or pos == 12 or pos == 14:
      return 1
    else:
      return 0
  def getKurangPlusX(self):
    '''
      Mengembalikan nilai sum Kurang(i) + X
    '''
    return self.kurangSum + self.getXValue()
  def isGoalReachable(self):
    '''
      Mengecek apakah nilai sum Kurang(i) + X genap
```

```
    '''
    return self.getKurangPlusX() % 2 == 0
  def f(self):
    '''
      Menghitung nilai taksiran f
    '''
    return self.steps
  def g(self):
    '''
      Menghitung nilai taksiran g
    '''
    return self._g
  def c(self):
    '''
      Menghitung nilai taksiran c
    '''
    return self.f() + self.g()
  def isSolved(self):
    '''
      Menentukan apakah taksiran g = 0
    '''
    return self.g() == 0
  def canMoveUp(self):
    '''
      Menentukan apakah petak kosong bisa bergerak ke atas
    '''
    pos = self.loc
    return not (pos >= 0 and pos <= 3)
  def canMoveDown(self):
    '''
      Menentukan apakah petak kosong bisa bergerak ke bawah
    '''
    pos = self.loc
    return not (pos >= 12 and pos <= 15)
  def canMoveRight(self):
    '''
      Menentukan apakah petak kosong bisa bergerak ke kanan
    '''
```

```
    pos = self.loc
    return not (pos == 3 or pos == 7 or pos == 11 or pos == 15)
  def canMoveLeft(self):
    '''
      Menentukan apakah petak kosong bisa bergerak ke kanan
    '''
    pos = self.loc
    return not (pos == 0 or pos == 4 or pos == 8 or pos == 12)
  def createMoveUp(self):
    '''
      Mengembalikan matriks yang petak kosongnya digeser satu ke atas
    '''
    # Copy matriks lama
    newArr = np.copy(self.arr)
    pos = self.loc
    # Hitung nilai g baru
    newG = self.g()
    if (newArr[pos - 4] - 1 == pos):
      newG -= 1
    elif (newArr[pos - 4] - 1 == pos - 4):
      newG += 1
    # Swap newArr[pos] dengan newArr[pos - 4]
    newArr[pos], newArr[pos - 4] = newArr[pos - 4], newArr[pos]
    # Update pos
    pos -= 4
    return Matrix(newArr, self.stepsBefore + ["up"], self.steps + 1,
newG, pos)
  def createMoveDown(self):
    '''
      Mengembalikan matriks yang petak kosongnya digeser satu ke bawah
    '''
    # Copy matriks lama
    newArr = np.copy(self.arr)
    pos = self.loc
    # Hitung nilai g baru
    newG = self.g()
    if (newArr[pos + 4] - 1 == pos):
      newG -= 1
```

```
    elif (newArr[pos + 4] - 1 == pos + 4):
      newG += 1
    # Swap newArr[pos] dengan newArr[pos + 4]
    newArr[pos], newArr[pos + 4] = newArr[pos + 4], newArr[pos]
    # Update pos
    pos += 4
    return Matrix(newArr, self.stepsBefore + ["down"], self.steps + 1,
newG, pos)
  def createMoveRight(self):
    '''
      Mengembalikan matriks yang petak kosongnya digeser satu ke kanan
    '''
    # Copy matriks lama
    newArr = np.copy(self.arr)
    pos = self.loc
    # Hitung nilai g baru
    newG = self.g()
    if (newArr[pos + 1] - 1 == pos):
      newG -= 1
    elif (newArr[pos + 1] - 1 == pos + 1):
      newG += 1
    # Swap newArr[pos] dengan newArr[pos + 1]
    newArr[pos], newArr[pos + 1] = newArr[pos + 1], newArr[pos]
    # Update pos
    pos += 1
    return Matrix(newArr, self.stepsBefore + ["right"], self.steps + 1,
newG, pos)
  def createMoveLeft(self):
    '''
      Mengembalikan matriks yang petak kosongnya digeser satu ke kiri
    '''
    # Copy matriks lama
    newArr = np.copy(self.arr)
    pos = self.loc
    # Hitung nilai g baru
    newG = self.g()
    if (newArr[pos - 1] - 1 == pos):
      newG -= 1
```

```
    elif (newArr[pos - 1] - 1 == pos - 1):
      newG += 1
    # Swap newArr[pos] dengan newArr[pos - 1]
    newArr[pos], newArr[pos - 1] = newArr[pos - 1], newArr[pos]
    # Update pos
    pos -= 1
    return Matrix(newArr, self.stepsBefore + ["left"], self.steps + 1,
newG, pos)
```

solver.py

```
from queue import PriorityQueue
import numpy as np
import timeit
from matrix import Matrix
import eel

if __name__ == '__main__':
  matrix = None
  pq = None
  visitedMap = None
  nodeCount = None
  sol = None
  matrices = None
  currentStep = None

  @eel.expose
  def reinitialize():
    '''
      Menginisialisasi ulang semua global variable
    '''
    global matrix
    global pq
    global visitedMap
    global nodeCount
    global sol
```

```
    global matrices
    global currentStep
    matrix = None
    pq = None
    visitedMap = None
    sol = None
    nodeCount = None
    matrices = None
    currentStep = None
    eel.go_to("home.html")

def startSolver():
    '''
      Mulai GUI
    '''
    eel.init("web")
    eel.start("home.html", size = (640, 640))

@eel.expose
def pyOpenTC(tc):
    '''
      Buka TC yang sesuai
    '''
    global matrix
    f = open(f'./test/tc{tc}.txt')
    mat = []
    for i in range(4):
      mat += map(int, f.readline().strip().split())
    matrix = Matrix(np.array(mat))
    f.close()
    # Buka halaman baru yang menunjukkan kondisi matriks
    eel.go_to("matrix.html")

@eel.expose
def updateMatrixPreviewGrid():
    '''
      Memperbarui matrix pada matrix.html
    '''
```

```
    global matrix
    eel.updateGrid(matrix.arr.tolist())

@eel.expose
def showMatrixStats():
    '''
      Menunjukkan kurang.html
    '''
    eel.go_to("kurang.html")

@eel.expose
def updateKurang():
    '''
      Memperbarui kurang.html
    '''
    global matrix
    matrix.countAllKurang()
    eel.updateKurangLabels(matrix.kurangMap, matrix.getKurangPlusX())

@eel.expose
def solveMatrix():
    '''
      Mengecek apakah matrix solvable atau tidak
    '''
    global matrix
    if (not matrix.isGoalReachable()):
      eel.showErrorMessage()
      reinitialize()
    else:
      eel.go_to("solved.html")

@eel.expose
def solveGrid():
    '''
      Core programnya: Menyelesaikan grid yang diberikan oleh user.
      Hanya ditampilkan solusi pertama saja (tidak mencari semua solusi)
    '''
    global matrix
```

```
    global visitedMap
    global nodeCount
    global sol
    global matrices
    global currentStep
    # Menyelesaikan matriks
    pq = PriorityQueue()
    visitedMap = {}
    pq.put((matrix.c(), matrix))
    visitedMap[matrix.id()] = True
    nodeCount = 1
    sol = []
    matrices = []
    start = timeit.default_timer()
    # DFS dengan priority queue
    while not pq.empty():
      head = pq.get()[1]
      if head.isSolved():
        sol = head.stepsBefore
        break
      else:
        if head.canMoveUp():
          newMatrix = head.createMoveUp()
          s = newMatrix.id()
          if s not in visitedMap:
            visitedMap[s] = True
            pq.put((newMatrix.c(), newMatrix))
            nodeCount += 1
        if head.canMoveRight():
          newMatrix = head.createMoveRight()
          s = newMatrix.id()
          if s not in visitedMap:
            visitedMap[s] = True
            pq.put((newMatrix.c(), newMatrix))
            nodeCount += 1
        if head.canMoveDown():
          newMatrix = head.createMoveDown()
          s = newMatrix.id()
```

```
          if s not in visitedMap:
            visitedMap[s] = True
            pq.put((newMatrix.c(), newMatrix))
            nodeCount += 1
        if head.canMoveLeft():
          newMatrix = head.createMoveLeft()
          s = newMatrix.id()
          if s not in visitedMap:
            visitedMap[s] = True
            pq.put((newMatrix.c(), newMatrix))
            nodeCount += 1
    time = timeit.default_timer() - start
    # Bentuk solusi jawaban
    matrices.append(matrix)
    for i in range(len(sol)):
      if sol[i] == 'up':
        matrix = matrix.createMoveUp()
      elif sol[i] == 'down':
        matrix = matrix.createMoveDown()
      elif sol[i] == 'right':
        matrix = matrix.createMoveRight()
      elif sol[i] == 'left':
        matrix = matrix.createMoveLeft()
      matrices.append(matrix)
    currentStep = 0
    # Update GUI
    eel.setUpGrid(matrices[0].arr.tolist(), currentStep, len(matrices) -
1, time, nodeCount)

  @eel.expose
  def prevPage():
    '''
      Menunjukkan step sebelumnya pada solved.html
    '''
    global matrices
    global currentStep
    if (currentStep > 0):
      currentStep -= 1
```

```
      eel.updateGrid(matrices[currentStep].arr.tolist(), currentStep,
len(matrices) - 1)

  @eel.expose
  def nextPage():
    '''
      Menunjukkan step berikutnya pada solved.html
    '''
    global matrices
    global currentStep
    if (currentStep < len(matrices) - 1):
      currentStep += 1
      eel.updateGrid(matrices[currentStep].arr.tolist(), currentStep,
len(matrices) - 1)

  startSolver()
```

GUI menggunakan HTML/CSS/JavaScript

Komunikasi antara Python dan HTML/CSS/JavaScript menggunakan *library* Eel yang dapat diakses di ChrisKnott/Eel: A little Python library for making simple Electron-like HTML/JS GUI apps (github.com).

style.css

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Inter', sans-serif;
}

@import
url('https://fonts.googleapis.com/css2?family=Inter&display=swap');

body {
  background-color: #28293d;
}

.big-flex {
  padding: 20px 80px 40px 80px;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  gap: 30px;
}

.flex-row {
  width: 100%;
  display: flex;
  justify-content: space-between;
}

.box {
  width: 80px;
  height: 80px;
```

```
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #3e405b;
  color: #ffffff;
  border-radius: 12px;
  box-shadow: 0px 12px 16px rgba(0, 0, 0, 0.3);
}

.button-row {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  color: #ffffff;
  width: 100%;
}

.button-row .right {
  display: flex;
  align-items: center;
  gap: 15px;
}

.button-row button {
  background: linear-gradient(180deg, #5561FF 0%, #3643FC 100%, #3643FC
100%);
  color: #ffffff;
  outline: none;
  border: none;
  padding: 12px 32px;
  border-radius: 12px;
  font-size: 14px;
  transition: 0.3s all linear;
  box-shadow: 0px 12px 16px rgba(0, 0, 0, 0.3);
}

.button-row .menu-btn {
  background: rgba(255, 255, 255, 0.1);
```

```
  color: #ffffff;
  outline: none;
  border: none;
  padding: 12px 32px;
  border-radius: 12px;
  font-size: 14px;
  transition: 0.3s all linear;
  box-shadow: 0px 12px 16px rgba(0, 0, 0, 0.3);
}

.button-row button:hover {
  cursor: pointer;
  transform: scale(1.05);
  opacity: 0.9;
}

.text-container {
  padding-top: 60px;
  text-align: center;
  color: white;
}

.btn-container {
  padding-top: 30px;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 30px;
}

.btn-container-smaller-padding {
  padding-top: 15px;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 20px;
}
```

```
.prim-btn {
  padding: 12px 32px;
  color: white;
  background: linear-gradient(180deg, #5561FF 0%, #3643FC 100%, #3643FC
100%);
  box-shadow: 0px 12px 16px rgba(0, 0, 0, 0.3);
  border-radius: 12px;
  border: none;
  outline: none;
  transition: all 0.3s linear;
}

.prim-btn:hover {
  cursor: pointer;
  opacity: 0.9;
  transform: scale(1.05);
}

.stats-container {
  padding-top: 30px;
  text-align: center;
  color: white;
  display: flex;
  align-items: center;
  flex-direction: column;
  gap: 10px;
}

.stats-container p {
  font-size: 14px;
}

.calculating-container {
  padding-top: 30px;
  font-size: 14px;
  text-align: center;
  color: white;
}
```

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Main Menu</title>
  <link rel="stylesheet" href="style.css">
  <script src="eel.js"></script>
  <script>
    function openTC(tc) {
      eel.pyOpenTC(tc)
    }

    eel.expose(go_to)
    function go_to(url) {
      window.location.replace(url)
    }
  </script>
</head>
<body>
  <div class="text-container">
    <h1>15 Puzzle Solver</h1>
    <p>Please select a testcase!</p>
  </div>
  <div class="btn-container">
    <button class="prim-btn" onclick="openTC(1)">./test/tc1.txt</button>
    <button class="prim-btn" onclick="openTC(2)">./test/tc2.txt</button>
    <button class="prim-btn" onclick="openTC(3)">./test/tc3.txt</button>
    <button class="prim-btn" onclick="openTC(4)">./test/tc4.txt</button>
    <button class="prim-btn" onclick="openTC(5)">./test/tc5.txt</button>
  </div>
</body>
</html>
```

kurang.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>Puzzle Statistics</title>
  <link rel="stylesheet" href="style.css">
  <script src="eel.js"></script>
  <script>
    eel.expose(updateKurangLabels)
    function updateKurangLabels(kurangMap, kurangPlusX) {
      for (let i = 1; i <= 16; i ++) {
        document.querySelector(`.kurang-${i}`).textContent =
`Kurang(${i}) = ${kurangMap[i]}`
      }
      document.querySelector(".sum").textContent = `SUM(Kurang(i)) + X =
${kurangPlusX}`
    }

    eel.expose(go_to)
    function go_to(url) {
      window.location.replace(url)
    }

    eel.expose(showErrorMessage)
    function showErrorMessage() {
      window.alert("This matrix cannot be solved!\nReturning to main
menu...")
    }

    eel.updateKurang();
  </script>
</head>
<body>
  <div class="stats-container">
    <h1>Puzzle statistics</h1>
    <p class="kurang-1">Kurang(1) = </p>
    <p class="kurang-2">Kurang(2) = </p>
    <p class="kurang-3">Kurang(3) = </p>
    <p class="kurang-4">Kurang(4) = </p>
    <p class="kurang-5">Kurang(5) = </p>
```

```
    <p class="kurang-6">Kurang(6) = </p>
    <p class="kurang-7">Kurang(7) = </p>
    <p class="kurang-8">Kurang(8) = </p>
    <p class="kurang-9">Kurang(9) = </p>
    <p class="kurang-10">Kurang(10) = </p>
    <p class="kurang-11">Kurang(11) = </p>
    <p class="kurang-12">Kurang(12) = </p>
    <p class="kurang-13">Kurang(13) = </p>
    <p class="kurang-14">Kurang(14) = </p>
    <p class="kurang-15">Kurang(15) = </p>
    <p class="kurang-16">Kurang(16) = </p>
    <p class="sum">SUM(Kurang(i)) + X = </p>
  </div>
  <div class="btn-container-smaller-padding">
    <button class="prim-btn" onclick="eel.solveMatrix()">Solve!</button>
  </div>
  </div>
</body>
</html>
```

matrix.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Matrix Preview</title>
  <link rel="stylesheet" href="style.css">
  <script src="eel.js"></script>
  <script>
    eel.expose(updateGrid);
    function updateGrid(matrix) {
      for (let i = 0; i < 16; i ++) {
        if (matrix[i] == 16) {
          document.getElementById(`${i}`).textContent = "";
        } else {
          document.getElementById(`${i}`).textContent = matrix[i];
        }
```

```
      }
    }
    eel.expose(go_to)
    function go_to(url) {
      window.location.replace(url)
    }

    eel.updateMatrixPreviewGrid();
  </script>
</head>
<body>
  <div class="big-flex">
    <div class="flex-row">
      <div class="box" id="0"></div>
      <div class="box" id="1"></div>
      <div class="box" id="2"></div>
      <div class="box" id="3"></div>
    </div>
    <div class="flex-row">
      <div class="box" id="4"></div>
      <div class="box" id="5"></div>
      <div class="box" id="6"></div>
      <div class="box" id="7"></div>
    </div>
    <div class="flex-row">
      <div class="box" id="8"></div>
      <div class="box" id="9"></div>
      <div class="box" id="10"></div>
      <div class="box" id="11"></div>
    </div>
    <div class="flex-row">
      <div class="box" id="12"></div>
      <div class="box" id="13"></div>
      <div class="box" id="14"></div>
      <div class="box" id="15"></div>
    </div>
  </div>
  <div class="btn-container-smaller-padding">
```

```
    <button class="prim-btn"
onclick="eel.showMatrixStats()">Next!</button>
  </div>
</body>
</html>
```

solved.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Solved Puzzle</title>
  <link rel="stylesheet" href="style.css">
  <script src="eel.js"></script>
  <script>
    eel.expose(setUpGrid)
    function setUpGrid(matrix, step, steps, time, nodeCount) {
      document.querySelector(".calc").textContent = `Searching took
${time.toFixed(5)} seconds | Generated ${nodeCount} nodes`
      for (let i = 0; i < 16; i ++) {
        if (matrix[i] == 16) {
          document.getElementById(`${i}`).textContent = "";
        } else {
          document.getElementById(`${i}`).textContent = matrix[i];
        }
      }
      document.querySelector(".step").textContent = `Step
${step}/${steps}`;
    }

    eel.expose(updateGrid)
    function updateGrid(matrix, step, steps) {
      for (let i = 0; i < 16; i ++) {
        if (matrix[i] == 16) {
          document.getElementById(`${i}`).textContent = "";
        } else {
          document.getElementById(`${i}`).textContent = matrix[i];
```

```
        }
      }
      document.querySelector(".step").textContent = `Step
${step}/${steps}`;
    }

    eel.expose(go_to)
    function go_to(url) {
      window.location.replace(url)
    }

    function returnToMainMenu() {
      window.alert("Returning to main menu...")
      eel.reinitialize()
    }

    eel.solveGrid();
  </script>
</head>
<body>
  <div class="calculating-container">
    <p class="calc">Calculation in progress...</p>
  </div>
  <div class="big-flex">
    <div class="flex-row">
      <div class="box" id="0"></div>
      <div class="box" id="1"></div>
      <div class="box" id="2"></div>
      <div class="box" id="3"></div>
    </div>
    <div class="flex-row">
      <div class="box" id="4"></div>
      <div class="box" id="5"></div>
      <div class="box" id="6"></div>
      <div class="box" id="7"></div>
    </div>
    <div class="flex-row">
      <div class="box" id="8"></div>
```

```
      <div class="box" id="9"></div>
      <div class="box" id="10"></div>
      <div class="box" id="11"></div>
    </div>
    <div class="flex-row">
      <div class="box" id="12"></div>
      <div class="box" id="13"></div>
      <div class="box" id="14"></div>
      <div class="box" id="15"></div>
    </div>
    <div class="button-row">
      <div class="left">
        <button class="menu-btn"
onclick="returnToMainMenu()">Menu</button>
      </div>
      <div class="right">
        <button onclick="eel.prevPage()">Previous</button>
        <p class="step">Step x/x</p>
        <button onclick="eel.nextPage()">Next</button>
      </div>
    </div>
  </div>
</body>
</html>
```

Contoh instantiasi 5 buah persoalan 15-puzzle

Persoalan yang dapat diselesaikan

tc1.txt

```
2 3 4 16
1 5 8 11
9 6 10 12
13 14 7 15
```

tc2.txt

```
1 2 3 4
5 7 10 8
11 9 6 16
13 14 15 12
```

tc3.txt

```
1 2 3 4
5 6 11 15
9 14 13 10
16 7 8 12
```

Persoalan yang tidak dapat diselesaikan

tc4.txt

```
2 3 4 16
1 5 11 8
9 6 10 12
13 14 7 15
```

tc5.txt

```
1 3 15 4
10 2 16 11
5 14 8 6
9 7 13 12
```

Alamat GitHub kode program

acomarcho/15-puzzle-solver: 15 puzzle solver made in Python, made by applying branch and bound algorithm. (github.com)