LAPORAN TUGAS KECIL

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

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Adiyansa Prasetya Wicaksana (K2) 13520142

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB 1 **3**

PENJELASAN ALGORITMA PROGRAM **3**

BAB 2 **6**

IMPLEMENTASI PROGRAM **6**

Modul puzzle 6

Modul bnb 7

Modul utils 8

BAB 3 **9**

HASIL PERCOBAAN **9**

Uji Coba Program 9

LAMPIRAN **15**

REFERENSI **16**

BAB 1
PENJELASAN ALGORITMA PROGRAM

Branch and Bound digunakan dalam persoalan optimasi yaitu untuk meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan dari persoalan. Dalam implementasinya algoritma *Branch and Bound* mirip antara gabungan antara *Breadth First Search* dengan *Least Cost Search*, perbedaan terhadap BFS ada dalam simpul yang diekspansinya. Setiap simpul pada *Branch and Bound* mempunyai sebuah *cost* untuk menaksirkan terhadap *goal* yang ingin diraih. Untuk pembangkitan simpul berikutnya dipilih dari *cost* yang paling kecil (pada kasus minimasi).

Karakteristik yang lain dari *Branch and Bound* adalah adanya pemangkasan pada jalur yang dianggap sudah tidak mengarah kepada solusi. Kriteria pemangkasan secara umum:

- Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini (*best solution so far*)
- Ada batasan yang dilanggar pada simpul
- Jika solusi pada simpul hanya terdiri dari satu tidak maka bandingkan dengan solusi terbaik saat ini dan ambil yang terbaik

Algoritma *Branch and Bound* salah satunya digunakan untuk menyelesaikan permainan *fifteen puzzle*. *Fifteen Puzzle* (15 Puzzle) merupakan permainan dalam suatu “papan” dengan grid 4 x 4 yang berisikan angka dari 1 sampai 15 dan 1 tempat kosong. Elemen pada “papan” tersebut dapat diacak dan objektifnya adalah agar urutan dari elemen sesuai dengan urutan yang diinginkan. Urutan yang dipakai pada tugas kecil kali ini adalah urutan dari 1 sampai 15 dengan tempat kosong pada urutan terakhir.

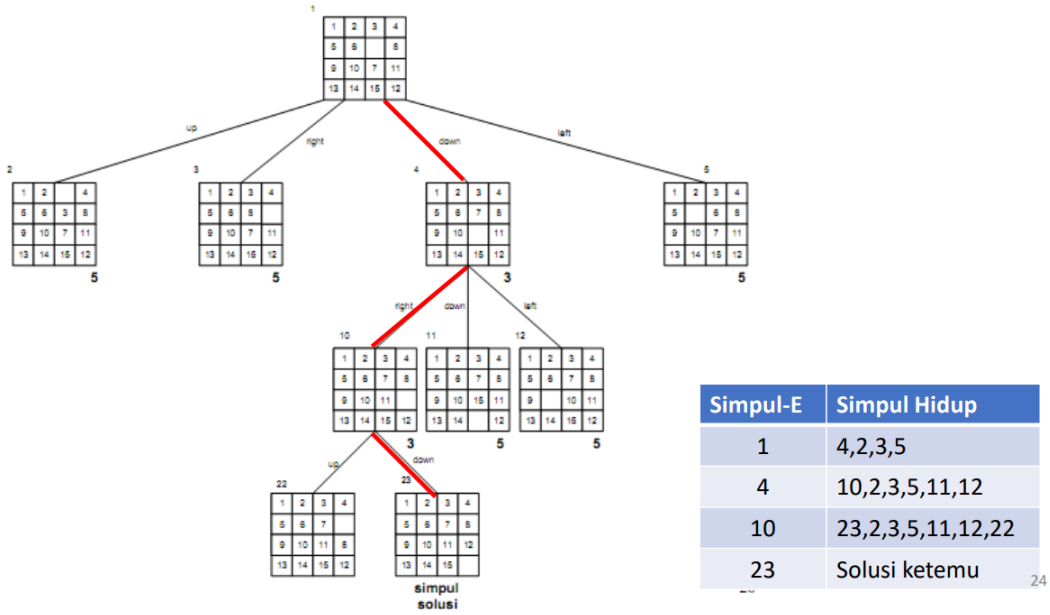
Gambar 1.1 Objektif pada permainan Fifteen Puzzle
(Sumber : https://ssaurel.medium.com/rediscover-the-joys-of-the-famous-15-puzzle-game-of-fifteen-on-android-d6a1555f5446)

Dalam penyelesaian permainan *Fifteen Puzzle* terdapat konfigurasi dari "papan" yang tidak dapat diselesaikan sehingga perlu pengecekan terlebih dahulu. Pengecekan ini dengan fungsi Kurang dan X. Fungsi kurang(i) menghitung banyaknya "ubin" bernomor j sehingga posisi(j) > posisi(i). Posisi(i) merupakan posisi "ubin" bernomor i pada konfigurasi yang diperiksa. Fungsi X mengebalikan nilai i + j mod 2, i dan j merupakan posisi dari "ubin" kosong dengan menggunakan larik 2 dimensi. Jika, total dari fungsi Kurang dan X bernilai genap maka objektif dapat tercapai, dan sebaliknya.

Setelah konfigurasi dapat ditentukan, selanjutnya adalah bagaimana algoritma dari *Branch and Bound* diterapkan pada permainan *Fifteen Puzzle*. *Cost* yang digunakan pada *fifteen puzzle* merupakan langkah yang telah dilakukan untuk mencapai konfigurasi saat ini ditambah *offset* dari konfigurasi sekarang dengan konfigurasi objektif. Langkah-langkah yang dilakukan adalah sebagai berikut.

1. Masukkan simpul akar ke dalam antrian PQ (digunakan PriorityQueue dalam implementasinya). Jika simpul akar merupakan solusi, maka solusi ditemukan dan hentikan pencarian.
2. Jika PQ kosong, maka hentikan pencarian.

3. Jika PQ tidak kosong, pilih dari antrian PQ simpul i yang mempunyai nilai *cost* terkecil.
4. Jika simpul i merupakan solusi, maka solusi ditemukan dan hentikan pencarian.
5. Jika simpul i bukan merupakan solusi, maka coba pindahkan “ubin” kosong ke semua arah yang memungkinkan dan jadikan “ubin” yang telah dipindahkan sebagai anak. Bangkitkan semua anak tersebut.
6. Untuk setiap anak dari simpul i, hitung *cost*-nya dan masukkan ke dalam PQ.
7. Kembali ke langkah 2.



Simpul-E	Simpul Hidup
1	4,2,3,5
4	10,2,3,5,11,12
10	23,2,3,5,11,12,22
23	Solusi ketemu

Gambar 1.2 Contoh penyelesaian Fifteen Puzzle dengan Branch and Bound

(*Sumber:* *https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf*)

BAB 2
IMPLEMENTASI PROGRAM

2.1 Modul puzzle

Modul puzzle berisi class Puzzle untuk merepresentasikan papan dari permainan *Fifteen Puzzle*. Kelas ini berisi method-method yang dibutuhkan untuk operasi pada elemen puzzle.

```python
from random import shuffle
import numpy as np

EMPTY = 0

"""
using 1 dimensional array because it is faster than 2 dimensional array
"""
class Puzzle:

  def __init__(self, arg = None):
    if (arg):
      self.Puzzle = self.inputPuzzle(arg)
    else:
      self.Puzzle = self.randomizePuzzle();

  def copy(self):
    copy = Puzzle()
    copy.Puzzle = np.copy(self.Puzzle)
    return copy

  def __eq__(self, other):
    return self.Puzzle.tobytes() == other.Puzzle.tobytes()

  def __hash__(self):
    return hash(bytes(self.Puzzle))

  """
  method overriding for printing Puzzle
  """
  def __str__(self):
    puzzle = np.zeros((4,4), dtype=int)
    for i in range(4):
      for j in range(4):
        puzzle[i][j] = self.Puzzle[i*4 + j]

    s = [["-" if e == 0 else str(e) for e in row] for row in puzzle]
    lens = [max(map(len, col)) for col in zip(*s)]
    fmt = '\t'.join('{{:{}}}'.format(x) for x in lens)
    table = [fmt.format(*row) for row in s]
    return('\n'.join(table) + '\n')

  """
  input Puzzle manually
  """
  def inputPuzzle(self, Puzzle):

    inputPuzz = np.zeros((16,), dtype=int)

    for i in range(4):
      for j in range(4):
        if (Puzzle[i][j] == 16):
          inputPuzz[i * 4 + j] = EMPTY;
        else:
          inputPuzz[i * 4 + j] = Puzzle[i][j]

    return inputPuzz

  """
  randomize Puzzle
  """
  def randomizePuzzle(self):
    PuzzleElmt = [i for i in range(0,16)]
    randPuzzle = np.zeros((16,), dtype=int)
    # randomize the element
    shuffle(PuzzleElmt)

    # assign each element
    for i in range(16):
      popped = PuzzleElmt.pop()
      randPuzzle[i] = popped

    return randPuzzle;

  """
  check valid
  """
  def isValid(self):
    for i in range(0, 16):
      if (self.pos(i) == -1):
        return False
    return True
```

```python
  """
  elemet getter
  """
  def getElmt(self, n):
    n -= 1;
    return self.Puzzle[n]

  """
  element setter
  """
  def setElmt(self, elmt, n):
    n -= 1;
    self.Puzzle[n] = elmt

  """
  get index of element
  """
  def pos(self, x):
    pos = 1;
    for i in range(1, 17):
      if (self.getElmt(i) == x):
        return pos
      else:
        pos += 1

    return -1

  """
  get index of empty element
  """
  def getEmpty(self):
    return self.pos(EMPTY)

  """
  move empty element
  """
  def move(self, dir, emptIndex):
    puzz = self.copy()
    # move empty element up
    if (dir == "up"):
      moveIndex = emptIndex - 4
      if (moveIndex < 1):
        return False
    # move empty element down
    elif (dir == "down"):
      moveIndex = emptIndex + 4
      if (moveIndex > 16):
        return False
    # move empty element right
    elif (dir == "right"):
      if (emptIndex % 4 == 0):
        return False
      moveIndex = emptIndex + 1
    # move empty element left
    elif (dir == "left"):
      if (emptIndex % 4 == 1):
        return False
      moveIndex = emptIndex - 1
    else:
      print("direction undefined")
      return

    temp = puzz.getElmt(moveIndex)
    # switch
    puzz.setElmt(temp, emptIndex)
    puzz.setElmt(EMPTY, moveIndex)

    return puzz


  def X(self):
    emptIndex = self.getEmpty() - 1
    i = emptIndex // 4
    j = emptIndex % 4
    return (i + j) % 2
```

2.2 Modul bnb

Modul bnb berisi algoritma utama dalam penyelesaian *fifteen puzzle*. Pada modul ini terdapat dua kelas yaitu PriorityQueue dan Node. PriorityQueue merupakan implementasi dari Priority Queue dan Node merupakan implementasi simpul yang digunakan pada langkah-langkah penyelesaian.

```python
from puzzle import *
from heapq import heappush, heappop

"""
priority queue for determining the next node to be expanded
"""
class PriorityQueue:
  def __init__(self):
    self.heap = []

  def push(self, k):
    heappush(self.heap, k)

  def pop(self):
    return heappop(self.heap)

  def empty(self):
    if not self.heap:
      return True
    else :
      return False

"""
node that can expand when the Puzzle is moved
"""
class Node:

  def __init__(self, parent, puzzle, cost, level, emptyPos, dir):
    self.parent = parent
    self.puzzle = puzzle
    self.cost = cost
    self.level = level
    self.emptyPos = emptyPos
    self.dir = dir

  def __lt__(self, nxt):
    return self.cost + self.level < nxt.cost + nxt.level


"""
create node
"""
def createNode(parent, puzz, movedPuzz, dir, emptyPos, level) -> Node:

  if (dir == "up"):
    newEmptyPos = emptyPos - 4
  elif (dir == "down"):
    newEmptyPos = emptyPos + 4
  elif (dir == "right"):
    newEmptyPos = emptyPos + 1
  else:
    newEmptyPos = emptyPos - 1

  # O(1) cost finding
  cost = parent.cost
  if (puzz.getElmt(newEmptyPos) != newEmptyPos):
    cost -= 1
  if (puzz.getElmt(newEmptyPos) != emptyPos):
    cost += 1

  newNode = Node(parent, movedPuzz, cost, level, newEmptyPos, dir)

  return newNode

"""
print steps to goal
"""
def printPath(root):
  if (root == None):
    return

  printPath(root.parent)
  print(f"dir: {root.dir}")
  print(root.puzzle)
```

```python
"""
find Puzzle elmt
that is j < i and posisi(j) > posisi(i)
"""
def kurang(Puzzle, i):
  if (i == 16):
    iPos = Puzzle.getEmpty()
  else:
    iPos = Puzzle.pos(i)
  kur = 0

  for x in range(1,i):
    if (Puzzle.pos(x) > iPos):
      kur += 1;

  print(f"kurang[{i}]\t: {kur}")
  return kur

"""
check if Puzzle is solveable
"""
def checkBnB(Puzzle):
  if (Puzzle.isValid()):
    kur = 0

    for i in range(1, 17):
      kur += kurang(Puzzle, i)

    check = kur + Puzzle.X()
    print(f"\nsigma(kurang): {kur}, X: {Puzzle.X()}")

    if (check % 2 == 0):
      return True
    else:
      return False
  else:
    return False

"""
cost from nodes to goal
"""
def costGoal(cur):
  cost = 0
  for i in range(1, 16):
    if (cur.getElmt == EMPTY):
      if (cur.getElmt(i) != i):
        cost += 1

  return cost

"""
different cost function
"""
def costGoalManhattan(cur):
  cost = 0
  for i in range(1, 16):
    if (cur.getElmt(i) != i):
      curIndex = cur.pos(i)
      xDiff = abs(curIndex // 4 - i // 4)
      yDiff = abs(curIndex % 4 - curIndex % 4)
      cost += xDiff + yDiff

  return cost


"""
solve the puzzle using branch and bound
"""
def solve(initial):

  if (checkBnB(initial)):
    puzzleIter = ["up", "right", "down", "left"]
    memory = set()
    pq = PriorityQueue()

    cost = costGoal(initial)
    root = Node(None, initial, cost, 0, initial.getEmpty(), "-")

    pq.push(root)
    count = 0

    while not pq.empty():
      minimum = pq.pop()
      count += 1

      if minimum.cost == 0:
        print()
        print("-------------------Solution-------------------")
        printPath(minimum)
        return minimum, len(memory)

      memory.add(minimum.puzzle)

      for i in puzzleIter:
        movedPuzzle = minimum.puzzle.move(i, minimum.emptyPos)
        if (movedPuzzle):
          if movedPuzzle not in memory:
            child = createNode(minimum, minimum.puzzle, movedPuzzle, i, minimum.emptyPos, minimum.level + 1)
            pq.push(child)

  else:
    print("puzzle unsolveable")

  return None, 0
```

2.3 Modul utils

Modul utils berisi fungsi-fungsi pembantu yang akan digunakan di main. Fungsi-fungsi yang terdapat pada modul ini adalah untuk pembacaan dari file dan mengeluarkan output.

```python
from bnb import *
import time

def readFromFile(fileName):
  path = "./test/" + fileName

  try:
    matr = []
    countLine = 0
    with(open(path)) as f:
      for line in f:
        splitted = line.split(" ")
        try:
          row = [int(x) for x in splitted]
          matr.append(row)
          countLine += 1
        except:
          print("Element invalid")

    if countLine != 4:
      raise Exception("Invalid txt file")
    else:
      return matr

  except:
    print(f"{path} not found")
    print("Will use randomize matrix")


def start():
  fileInput = input("enter file name: ")
  puzzle = Puzzle(readFromFile(fileInput))

  print("\ncurrent puzzle:")
  print(puzzle)

  start_time = time.time()
  solution, nodes = solve(puzzle)
  end_time = time.time()
  if (solution):
    print(f"{solution.level} steps and {nodes} nodes explored")
    print("time taken: %.4f seconds" %(end_time - start_time))
```

BAB 3

HASIL PERCOBAAN

3.1. Uji Coba Program

Percobaan 1

<table>
<tr><th>Nama File</th><td>5step.txt</td></tr>
<tr><th>Bentuk Awal</th><td>current puzzle:
1 2 3 -
5 6 7 4
9 10 12 8
13 14 11 15</td></tr>
<tr><th colspan="2">Hasil</th></tr>
<tr><td colspan="2">kurang[1] : 0
kurang[2] : 0
kurang[3] : 0
kurang[4] : 0
kurang[5] : 1
kurang[6] : 1
kurang[7] : 1
kurang[8] : 0
kurang[9] : 1
kurang[10] : 1
kurang[11] : 0
kurang[12] : 2
kurang[13] : 1
kurang[14] : 1
kurang[15] : 0
kurang[16] : 12

sigma(kurang): 21, X: 1</td></tr>
<tr><td colspan="2">dir: down
1 2 3 4
5 6 7 -
9 10 12 8
13 14 11 15

dir: down
1 2 3 4
5 6 7 8
9 10 12 -
13 14 11 15

dir: left
1 2 3 4
5 6 7 8
9 10 - 12
13 14 11 15

dir: down
1 2 3 4
5 6 7 8
9 10 11 12
13 14 - 15

dir: right
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 -</td></tr>
<tr><td colspan="2">5 steps and 5 nodes explored
time taken: 0.0030 seconds</td></tr>
</table>

Percobaan 2

Nama File	3step.txt
Bentuk Awal	current puzzle: 1 2 3 4 5 6 - 8 9 10 7 11 13 14 15 12
Hasil	
	kurang[1] : 0 kurang[2] : 0 kurang[3] : 0 kurang[4] : 0 kurang[5] : 0 kurang[6] : 0 kurang[7] : 0 kurang[8] : 1 kurang[9] : 1 kurang[10] : 1 kurang[11] : 0 kurang[12] : 0 kurang[13] : 1 kurang[14] : 1 kurang[15] : 1 kurang[16] : 9 sigma(kurang): 15, X: 1
	dir: down 1 2 3 4 5 6 7 8 9 10 - 11 13 14 15 12 dir: right 1 2 3 4 5 6 7 8 9 10 11 - 13 14 15 12 dir: down 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 -
	3 steps and 3 nodes explored time taken: 0.0050 seconds

Percobaan 3

Nama File	16step.txt
Bentuk Awal	current puzzle: 10 2 4 8 1 5 3 - 9 7 6 12 13 14 11 15
Hasil	
	kurang[1] : 0 kurang[2] : 1 kurang[3] : 0 kurang[4] : 2 kurang[5] : 1 kurang[6] : 0 kurang[7] : 1 kurang[8] : 5 kurang[9] : 2 kurang[10] : 9 kurang[11] : 0 kurang[12] : 1 kurang[13] : 1 kurang[14] : 1 kurang[15] : 0 kurang[16] : 8 sigma(kurang): 32, X: 0
	dir: up 10 2 4 - 1 5 3 8 9 7 6 12 13 14 11 15 dir: left 10 2 - 4 1 5 3 8 9 7 6 12 13 14 11 15 dir: left 10 - 2 4 1 5 3 8 9 7 6 12 13 14 11 15 dir: left - 10 2 4 1 5 3 8 9 7 6 12 13 14 11 15 dir: down 1 10 2 4 - 5 3 8 9 7 6 12 13 14 11 15

dir: right

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

dir: up

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

dir: right

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

dir: down

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

dir: down

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

dir: left

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

dir: up

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

dir: right

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

dir: down

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

dir: down

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

dir: right

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

```
16 steps and 195 nodes explored
time taken: 0.0260 seconds
```

Percobaan 4

Nama File	unsolveable1.txt
Bentuk Awal	current puzzle: 12 11 14 - 15 2 10 4 9 7 3 6 13 5 8 1
Hasil	
	kurang[1] : 0 kurang[2] : 1 kurang[3] : 1 kurang[4] : 2 kurang[5] : 1 kurang[6] : 2 kurang[7] : 4 kurang[8] : 1 kurang[9] : 6 kurang[10] : 8 kurang[11] : 10 kurang[12] : 11 kurang[13] : 3 kurang[14] : 11 kurang[15] : 11 kurang[16] : 12 sigma(kurang): 84, X: 1 puzzle unsolveable

Percobaan 5

Nama File	unsolveable2.txt
Bentuk Awal	current puzzle: 4 9 3 14 13 6 15 10 11 8 7 1 2 - 5 12
Hasil	

```
kurang[1]       : 0
kurang[2]       : 0
kurang[3]       : 2
kurang[4]       : 3
kurang[5]       : 0
kurang[6]       : 3
kurang[7]       : 3
kurang[8]       : 4
kurang[9]       : 7
kurang[10]      : 5
kurang[11]      : 5
kurang[12]      : 0
kurang[13]      : 9
kurang[14]      : 10
kurang[15]      : 8
kurang[16]      : 2
```

```
sigma(kurang): 61, X: 0
puzzle unsolveable
```

LAMPIRAN

1. Repository Github :

LINK GITHUB

2. Checklist :

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		

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

https://www.javatpoint.com/branch-and-bound