

Penyelesaian Persoalan 15-Puzzle dengan Algoritma

Branch and Bound

Dibuat sebagai Tugas Kecil 3

IF2211

Strategi Algoritma



Dikerjakan oleh:

Fernaldy 13520112

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

DAFTAR ISI

Daftar Isi

BAB I CARA KERJA PROGRAM.....	1
BAB II SCREENSHOT INPUT OUTPUT PROGRAM.....	3
BAB III KETUNTASAN TUGAS	8
BAB IV KODE PROGRAM DAN INSTANSIASI PERSOALAN.....	9
BAB V ALAMAT GOOGLE DRIVE DAN GITHUB	16

BAB I

CARA KERJA PROGRAM

Berikut adalah langkah-langkah cara kerja program *branch and bound* yang dibuat dalam penyelesaian masalah:

1. Program membaca *file input* dan memasukkan bilangan-bilangan masukan ke dalam array berukuran 16
2. Perhitungan waktu penyelesaian persoalan dimulai
3. Menghitung nilai Kurang(i) untuk tiap elemen array. Nilai Kurang(i) adalah jumlah ubin bernomor j sedemikian sehingga $j < i$ dan Posisi(j) > Posisi(i). Posisi(i) adalah posisi ubin bernomor i pada susunan yang sedang diperiksa.
4. Menentukan nilai X. Nilai X adalah 1 jika ubin kosong terletak pada indeks 1, 3, 4, 6, 9, 11, 12, atau 14. Selain itu, X akan bernilai 0
5. Menghitung nilai $\sum_{i=1}^{16} \text{Kurang}(i) + X$. Jika nilai tersebut genap, berarti persoalan dapat diselesaikan dan lanjut pada poin 6. Namun, apabila nilai tersebut ganjil, persoalan tidak dapat diselesaikan dan lanjut ke poin 11.
6. Buat list kosong dan antrean kosong. List akan berisi semua susunan *puzzle* yang sudah pernah dihasilkan. Antrean kosong berisi simpul-simpul susunan *puzzle* yang menunggu untuk di-*expand*. Jenis antrean yang digunakan adalah *priority queue* yang elemen-elemennya disusun secara menaik berdasarkan estimasi *cost* simpul tersebut.
7. Masukkan konfigurasi array awal sebagai anggota list dan antrean.
8. Ambil elemen pertama dari antrean. Apabila simpul tersebut adalah solusi, maka proses iterasi dihentikan dan lanjut ke poin 11. Jika simpul tersebut bukan solusi, hasilkan semua simpul baru yang bisa diturunkan dari simpul tersebut. Untuk setiap simpul baru yang dihasilkan dan belum pernah dihasilkan sebelumnya, hitung estimasi *cost* simpul tersebut. Estimasi *cost* yang digunakan adalah *cost* yang diperlukan untuk sampai ke suatu simpul x dari akar ditambah dengan jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir. Simpul-simpul tersebut lalu dimasukkan ke dalam list dan antrean.
9. Simpul-simpul baru yang bisa dihasilkan dari suatu simpul adalah simpul dengan konfigurasi *puzzle* yang diperoleh dengan menggerakkan ubin kosong ke atas, bawah, kanan, dan kiri. Syarat ubin kosong bisa digerakkan ke arah tertentu adalah jika ubin

kosong tidak berada pada sisi tertentu *puzzle* sedemikian sehingga ubin kosong tidak lagi bisa digerakkan ke arah tersebut dan gerakan sebelumnya yang dilakukan pada ubin kosong tersebut bukan ke arah sebaliknya dari gerakan yang akan dilakukan. Sebagai contoh, ubin kosong bisa digerakkan ke arah atas adalah jika ubin kosong tidak berada pada sisi paling atas *puzzle* dan gerakan sebelumnya yang dilakukan pada ubin kosong tersebut bukan ke bawah.

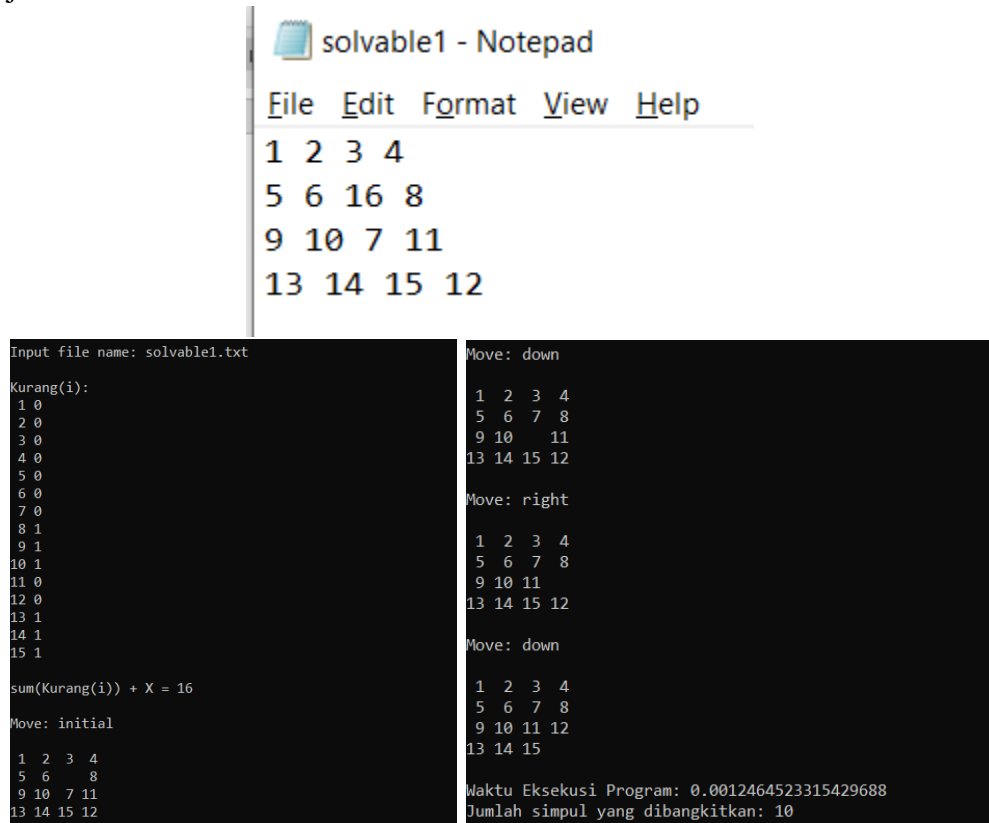
10. Ulangi poin 8 hingga menemukan solusi.
11. Perhitungan waktu penyelesaian persoalan dihentikan.
12. Jika solusi ditemukan, cetak semua rute dari konfigurasi awal hingga mencapai solusi beserta gerakan ubin kosong yang dilakukan. Selain itu, cetak juga waktu penyelesaian persoalan dan jumlah simpul yang dibangkitkan. Sementara itu, apabila *puzzle* tidak dapat diselesaikan, cetak pesan error.

BAB II

SCREENSHOT INPUT OUTPUT PROGRAM

Berikut adalah *screenshot* input output program. Program menerima masukan konfigurasi awal dari file. Bilangan 16 pada menandakan ubin kosong.

1. Data uji file solvable1.txt



```
solvable1 - Notepad
File Edit Format View Help
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

Input file name: solvable1.txt
Kurang(i):
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 1
9 1
10 1
11 0
12 0
13 1
14 1
15 1
sum(Kurang(i)) + X = 16
Move: initial
1 2 3 4
5 6 8
9 10 7 11
13 14 15 12

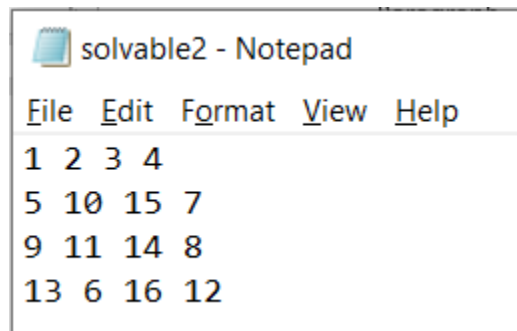
Move: down
1 2 3 4
5 6 7 8
9 10 11
13 14 15 12

Move: right
1 2 3 4
5 6 7 8
9 10 11
13 14 15 12

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

Waktu Eksekusi Program: 0.0012464523315429688
Jumlah simpul yang dibangkitkan: 10
```

2. Data uji file solvable2.txt



```
solvable2 - Notepad
File Edit Format View Help
1 2 3 4
5 10 15 7
9 11 14 8
13 6 16 12
```

Input file name: solvable2.txt

Kurang(i):

```
1 0
2 0
3 0
4 0
5 0
6 0
7 1
8 1
9 2
10 4
11 2
12 0
13 2
14 4
15 8
```

sum(Kurang(i)) + X = 26

Move: initial

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

Move: down

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

Move: right

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

Move: up

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

Move: up

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

Move: up

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

Move: up

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

Move: left

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

Move: down

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

Move: left

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

Move: down

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

Move: right

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

Move: up

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

Move: right

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

Move: down

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

Move: down

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

Waktu Eksekusi Program: 0.14960193634033203
Jumlah simpul yang dibangkitkan: 1195

3. Data uji file solvable3.txt

```
solvable3 - Notepad
File Edit Format View Help
3 1 2 4
16 5 7 8
10 6 11 12
9 13 14 15
```

```
Input file name: solvable3.txt
Kurang(i):
1 0
2 0
3 2
4 0
5 0
6 0
7 1
8 1
9 0
10 2
11 1
12 1
13 0
14 0
15 0

sum(Kurang(i)) + X = 20

Move: initial
3 1 2 4
5 7 8
10 6 11 12
9 13 14 15

Move: up
1 2 4
3 6 7 8
5 10 11 12
9 13 14 15

Move: right
1 2 4
3 6 7 8
5 10 11 12
9 13 14 15

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

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

Move: right
3 1 2 4
5 7 8
10 6 11 12
9 13 14 15

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

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

Move: up
3 1 2 4
6 7 8
5 10 11 12
9 13 14 15

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

Move: right
1 6 2 4
5 3 7 8
9 10 11 12
13 14 15

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

```

Move: up
1 6 2 4
5 3 7 8
9 10 12
13 14 11 15

Move: up
1 6 2 4
5 3 7 8
9 10 7 12
13 14 11 15

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

Move: up
1 2 4
5 6 3 8
9 10 7 12
13 14 11 15

Move: right
1 2 3 4
5 6 7 8
9 10 12
13 14 11 15

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

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

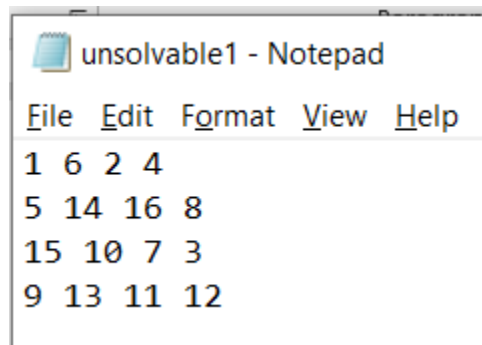
Move: down
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15

Move: right
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15

Waktu Eksekusi Program: 26.869511604309082
Jumlah simpul yang dibangkitkan: 17084

```

4. Data uji file unsolvable1.txt



```

unsolvable1 - Notepad
File Edit Format View Help
1 6 2 4
5 14 16 8
15 10 7 3
9 13 11 12

```



```

Input file name: unsolvable1.txt

Kurang(i):
1 0
2 0
3 0
4 1
5 1
6 4
7 1
8 2
9 0
10 3
11 0
12 0
13 2
14 8
15 7


sum(Kurang(i)) + X = 39

Initial matrix:
1 6 2 4
5 14 8
15 10 7 3
9 13 11 12

Puzzle tidak dapat diselesaikan

```

5. Data uji file unsolvable2.txt

 unsolvable2 - Notepad

File Edit Format View Help

```

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

```

```

Input file name: unsolvable2.txt

Kurang(i):
1 0
2 0
3 0
4 0
5 0
6 0
7 1
8 0
9 2
10 4
11 2
12 0
13 3
14 4
15 8

sum(Kurang(i)) + X = 29

Initial matrix:
1 2 3 4
5 10 15 7
9 11 14
13 6 8 12

Puzzle tidak dapat diselesaikan

```

BAB III

KETUNTASAN TUGAS

Berikut adalah ketuntasan tugas yang dibuat:

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

KODE PROGRAM DAN INSTANSIASI PERSOALAN

Kode program dibuat dalam bahasa Python.

1. File Node.py

```
class Node:

    def __init__ (self, array, level, moves):
        self.puzzle = array
        self.level = level
        self.moves = moves

    def countCost (self):
        # Menghitung estimasi cost
        g = 0
        for i in range(len(self.puzzle)):
            if (self.puzzle[i]!=16 and self.puzzle[i]!=i+1):
                g += 1
        return int(self.level + g)
```

2. File Puzzle.py

```
from Node import Node

def kurang_i(array):
    # Menghitung Kurang(i)
    kurangi = [0 for i in range(16)]
    total = 0
    for i in range(len(array)):
        for j in range(i+1, len(array)):
            if array[j] < array[i]:
                kurangi[array[i]-1] += 1
                total += 1
        if array[i] == 16:
            idx16 = i
    print("\nKurang(i):")
    for i in range(len(kurangi)):
        if (i < 9):
            print(' ', end='')
        if (i < 15):
            print(i+1, kurangi[i])
    return idx16, total
```

```

def isSolvable(idxl6, total):
    # Menentukan apakah puzzle bisa diselesaikan
    x = 0
    shaded = [1,3,4,6,9,11,12,14]
    if (idxl6 in shaded):
        x = 1
    print("\nsum(Kurang(i)) + X = " + str(total + x))
    return (total + x)%2 == 0

def isMoveableLeft(array):
    # Menentukan apakah ubin kosong bisa digerakkan ke kiri
    idx = array.index(16)
    prohibited = [0, 4, 8, 12]
    return not (idx in prohibited)

def isMoveableRight(array):
    # Menentukan apakah ubin kosong bisa digerakkan ke kanan
    idx = array.index(16)
    prohibited = [3, 7, 11, 15]
    return not (idx in prohibited)

def isMoveableUp(array):
    # Menentukan apakah ubin kosong bisa digerakkan ke atas
    idx = array.index(16)
    prohibited = [0, 1, 2, 3]
    return not (idx in prohibited)

def isMoveableDown(array):
    # Menentukan apakah ubin kosong bisa digerakkan ke bawah
    idx = array.index(16)
    prohibited = [12, 13, 14, 15]
    return not (idx in prohibited)

def moveLeft(array):
    # Menggerakkan ubin kosong ke kiri
    idx = array.index(16)
    newArray = [array[i] for i in range(16)]
    temp = newArray[idx-1]
    newArray[idx-1] = newArray[idx]
    newArray[idx] = temp
    return newArray

def moveRight(array):
    # Menggerakkan ubin kosong ke kanan
    idx = array.index(16)
    newArray = [array[i] for i in range(16)]
    temp = newArray[idx+1]
    newArray[idx+1] = newArray[idx]
    newArray[idx] = temp
    return newArray

```

```

def moveUp(array):
    # Menggerakkan ubin kosong ke atas
    idx = array.index(16)
    newArray = [array[i] for i in range(16)]
    temp = newArray[idx-4]
    newArray[idx-4] = newArray[idx]
    newArray[idx] = temp
    return newArray

def moveDown(array):
    # Menggerakkan ubin kosong ke bawah
    idx = array.index(16)
    newArray = [array[i] for i in range(16)]
    temp = newArray[idx+4]
    newArray[idx+4] = newArray[idx]
    newArray[idx] = temp
    return newArray

def isEverGenerated(newArray, generated):
    # Menentukan apakah newArray sudah pernah dihasilkan sebelumnya
    found = False
    i = 0
    while (i < len(generated) and not found):
        if (generated[i].puzzle == newArray):
            found = True
        else:
            i += 1
    return found

def generate(node, generated):
    # Menghasilkan semua anak yang mungkin dari suatu simpul
    lastmove = node.moves[len(node.moves)-1]
    newArray = []
    newMoves = []
    if (isMoveableUp(node.puzzle) and lastmove != "down"):
        newArray = moveUp(node.puzzle)
        if not isEverGenerated(newArray, generated):
            newArray.append(newArray)
            newMoves.append("up")
    if (isMoveableRight(node.puzzle) and lastmove != "left"):
        newArray = moveRight(node.puzzle)
        if not isEverGenerated(newArray, generated):
            newArray.append(newArray)
            newMoves.append("right")
    if (isMoveableDown(node.puzzle) and lastmove != "up"):
        newArray = moveDown(node.puzzle)
        if not isEverGenerated(newArray, generated):
            newArray.append(newArray)
            newMoves.append("down")
    if (isMoveableLeft(node.puzzle) and lastmove != "right"):
        newArray = moveLeft(node.puzzle)
        if not isEverGenerated(newArray, generated):
            newArray.append(newArray)
            newMoves.append("left")
    return newArray, newMoves

```

```

def action(array, direction):
    # Menggerakkan ubin kosong ke arah direction
    if (direction == "initial"):
        return array
    elif (direction == "left"):
        return moveLeft(array)
    elif (direction == "right"):
        return moveRight(array)
    elif (direction == "up"):
        return moveUp(array)
    elif (direction == "down"):
        return moveDown(array)

def display(array):
    # Mencetak array
    for i in range(4):
        for j in range(4):
            if (array[i*4+j] == 16):
                print(" ", end = ' ')
            elif (array[i*4+j] < 10):
                print(" " + str(array[i*4+j]), end = ' ')
            else:
                print(str(array[i*4+j]), end = ' ')
        print()
    print()

```

3. File main.py

```
from Puzzle import *
from queue import PriorityQueue
import time

def readFile():
    # Melakukan pembacaan file
    filename = input("\nInput file name: ")
    f = open("../test/" + filename, "r")
    lines = f.read()
    array = []
    temp = ""
    for text in lines:
        if (text != ' ' and text != '\n'):
            temp += text
        else:
            array.append(int(temp))
            temp = ""
    array.append(int(temp))
    return array

def solve(array):
    # Menyelesaikan puzzle
    idx16, total = kurang_i(array)
    found = False
    if (isSolvable(idx16, total)):
        generated = []
        q = PriorityQueue()
        newNode = Node(array, 0, ["initial"])
        solution = [i+1 for i in range(16)]
        generated.append(newNode)
        q.put((newNode.countCost(), len(generated), newNode))
        while not(q.empty()) and not(found):
            currentNode = q.get()[2]
            if (currentNode.puzzle == solution):
                found = True
            else:
                newArray, newMoves = generate(currentNode,
generated)
                i = 0
                for newArray in newArray:
                    newNode = Node(newArray, currentNode.level + 1,
currentNode.moves + [newMoves[i]])
                    generated.append(newNode)
                    q.put((newNode.countCost(), len(generated),
newNode))
                    i += 1
                return currentNode, found, len(generated)
        else:
            dummyNode = Node(array, 0, ["initial"])
            return dummyNode, found, 0
```

```

if __name__ == '__main__':
    array = readFile()
    start = time.time()
    resultNode, solved, n = solve(array)
    end = time.time()
    if solved:
        tempArray = [array[i] for i in range(16)]
        print()
        for i in range(len(resultNode.moves)):
            print("Move: " + resultNode.moves[i])
            print()
            tempArray = action(tempArray, resultNode.moves[i])
            display(tempArray)
        print("Waktu Eksekusi Program: " + str(end-start))
        print("Jumlah simpul yang dibangkitkan: " + str(n))
        print()
    else:
        print("\nInitial matrix:")
        display(array)
        print("Puzzle tidak dapat diselesaikan\n")

```

4. Instansiasi perosalan file solvable1.txt

```

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

```

5. Instansiasi perosalan file solvable2.txt

```

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

```

6. Instansiasi perosalan file solvable3.txt

```

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

```

7. Instansiasi perosalan file unsolvable1.txt

```

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

```


8. Instansiasi perosalan file unsolvable2.txt

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

BAB V

ALAMAT GOOGLE DRIVE DAN GITHUB

Link google drive:

https://drive.google.com/drive/folders/1gj5tHKSPiNzuP7by5qcwSMRTF0Md_QBx?usp=sharing

Link repository github:

https://github.com/fernaldy112/Tucil3_13520112