

Laporan Tugas Kecil 3
IF2211 Strategi Algoritma
Semester II Tahun 2021/2022



Oleh:
Muhammad Rakha Athaya
NIM 13520108

Institut Teknologi Bandung
2022

Bab I

Algoritma *Branch and Bound*

Tujuan

Menemukan deretan langkah untuk mencapai susunan target 15-puzzle dari susunan awal dengan panjang terpendek.

Langkah-langkah



1. Susunan angka pada papan 15-puzzle direpresentasikan menjadi sebuah array.
Contoh: [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
2. Untuk menentukan apakah sebuah susunan awal 15-puzzle dapat diselesaikan, pertama harus dicari dahulu nilai **$\Sigma kurang(i) + X$** dimana:
 - kurang(i) adalah jumlah angka yang lebih kecil dari i tetapi indeksnya dalam array lebih besar dari i
 - X bernilai 1 apabila angka 16/blank terdapat di indeks 1,3,4,6,9,11,12, atau 14

Apabila **$\Sigma kurang(i) + X$** bernilai genap, susunan 15-puzzle dapat diselesaikan.

3. Untuk setiap state P dari papan 15-puzzle, nilai *cost*-nya bisa didapatkan dengan rumus:

$$\hat{c}(P) = f(P) + \hat{g}(P)$$

Dimana $f(P)$ adalah jumlah ubin tidak kosong pada P yang tidak ada pada state awal, dan $g(P)$ adalah jumlah ubin tidak kosong pada P yang tidak ada pada state akhir.

4. Pertama, lakukan *branching* pada state awal dengan melakukan dengan menggeser ubin 16/blank ke atas, bawah, kiri, dan kanan (bisa tidak keempatnya apabila posisi ubin 16/blank berada di sisi papan).
5. Hitung *cost* dari setiap state anak yang dihasilkan. Lakukan *branching* lagi pada state anak dengan nilai *cost* terkecil.
6. State selanjutnya yang akan di-*branching* adalah state dalam kumpulan state yang belum pernah di-*branch* yang memiliki nilai *cost* terkecil.

7. Pencarian selesai ketika didapatkan state anak yang sama persisi dengan state target yang ingin didapatkan.
8. Semua state lain yang memiliki *cost* lebih besar dari state target sebelumnya ditutup (*bound*) sehingga tidak akan pernah di-*branch* lagi.

Bab II

Kode Program

Board.py

```
### Dasar-dasar papan puzzle
import copy
import random

## Papan puzzle
board = []
target = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

## Papan methods
def randomize():
    random.shuffle(board)

def posisi(x):
    return board.index(x)

def kurang(x):
    count = 0
    for i in range(16):
        if i > posisi(x):
            if board[i] < x:
                count += 1
    return count

def isBlankAtGrey():
    res = False
    if posisi(16) in [1,3,4,6,9,11,12,14]:
        res = True
    return res

# returns (bool, total count)
def isReachable():
    res = False
    count = 0
    for i in range(16):
        count += kurang(i+1)
    if (isBlankAtGrey()):
        count += 1
    if count % 2 == 0:
        res = True
    return res, count

def displayScore():
    global board
    print("\nStarting state:")
    for j in range(16):
        if j in [3,7,11,15]:
            if board[j] < 10:
```

```

        print("0"+str(board[j]))
    else:
        if board[j] == 16:
            print(" ")
        else:
            print(board[j])
    else:
        if board[j] < 10:
            print("0"+str(board[j]), end=" ")
        else:
            if board[j] == 16:
                print(" ", end=" ")
            else:
                print(board[j], end=" ")
    print("\n i | kurang(i)")
    for i in range(16):
        if i+1 < 10:
            print("0"+str(i+1),"|",kurang(i+1))
        else:
            print(str(i+1)+"|",kurang(i+1))
    print("Sigma(kurang(i)) + X =", isReachable()[1])
    if isReachable()[0]:
        print("The puzzle is solvable, solving now...\n")
    else:
        print("The puzzle is not solvable.")

def inputPuzzle(textName):
    global board
    with open("test/"+textName,"r") as file:
        temp = []
        lines = file.readlines()
        for i in lines:
            as_list = i.split(" ")
            for j in range(4):
                if j == 3:
                    temp.append(int(as_list[3].replace("\n","")))
                else:
                    temp.append(int(as_list[j]))
    board = copy.deepcopy(temp)
    file.close()

```

Search.py

```

import Board
import copy

## Important Variables
root = Board.board
final = Board.target
foundFinal = False
solution = []
activeNodes = [[root]]
activeCost = [999]
allStates = [root]

```

```

## Moving methods
def moveUp(board):
    blank = board.index(16)
    board[blank] = board[blank-4]
    board[blank-4] = 16

def moveDown(board):
    blank = board.index(16)
    board[blank] = board[blank+4]
    board[blank+4] = 16

def moveRight(board):
    blank = board.index(16)
    board[blank] = board[blank+1]
    board[blank+1] = 16

def moveLeft(board):
    blank = board.index(16)
    board[blank] = board[blank-1]
    board[blank-1] = 16

## Cost methods
def cost(src, dst):
    count = 0
    for i in range(16):
        if src[i] != 16:
            if src[i] != dst[i]:
                count += 1
    return count

def totalCost(node):
    return (cost(root,node)) + (cost(node,final))

## Expand method
def expand(node):
    state = copy.deepcopy(node[-1])
    global foundFinal

    if state.index(16)-4 >= 0:
        tempState = copy.deepcopy(state)
        tempNode = copy.deepcopy(node)
        moveUp(tempState)
        if tempState not in allStates:
            allStates.append(tempState)
            tempNode.append(tempState)
            if tempState == final:
                foundFinal = True
                tempNode.append("SOLVED")
            activeNodes.append(tempNode)
            activeCost.append(totalCost(tempState))

    if state.index(16) not in [3,7,11,15]:
        tempState = copy.deepcopy(state)
        tempNode = copy.deepcopy(node)
        moveRight(tempState)

```

```

        if tempState not in allStates:
            allStates.append(tempState)
            tempNode.append(tempState)
            if tempState == final:
                foundFinal = True
                tempNode.append("SOLVED")
            activeNodes.append(tempNode)
            activeCost.append(totalCost(tempState))

    if state.index(16)+4 <= 15:
        tempState = copy.deepcopy(state)
        tempNode = copy.deepcopy(node)
        moveDown(tempState)
        if tempState not in allStates:
            allStates.append(tempState)
            tempNode.append(tempState)
            if tempState == final:
                foundFinal = True
                tempNode.append("SOLVED")
            activeNodes.append(tempNode)
            activeCost.append(totalCost(tempState))

    if state.index(16) not in [0,4,8,12]:
        tempState = copy.deepcopy(state)
        tempNode = copy.deepcopy(node)
        moveLeft(tempState)
        if tempState not in allStates:
            allStates.append(tempState)
            tempNode.append(tempState)
            if tempState == final:
                foundFinal = True
                tempNode.append("SOLVED")
            activeNodes.append(tempNode)
            activeCost.append(totalCost(tempState))

    del activeCost[activeNodes.index(node)]
    activeNodes.remove(node)

def bestNodeIdx():
    return activeCost.index(min(activeCost))

def getSolution():
    global solution
    for i in activeNodes:
        if i[-1] == "SOLVED":
            solution = i

def updateSearch():
    global root
    global activeNodes
    global allStates

    root = Board.board
    activeNodes = [[root]]
    allStates = [root]

```

Main.py

```

import Board
import Search
import time

### Algoritma utama
fileName = str(input("Input starting puzzle file name: "))
Board.inputPuzzle(fileName)
Search.updateSearch()

pick = input("Randomize puzzle board? *not recommended* [Y/N]: ")
if pick in ["Y", "y"]:
    Board.randomize()
    print("The puzzle board has been randomized")
else:
    pass

Board.displayScore()

if Board.isReachable()[0]:

    ## Searching
    start = time.time()
    Search.expand(Search.activeNodes[0])
    while not Search.foundFinal:
        Search.expand(Search.activeNodes[Search.bestNodeIdx()])
    Search.getSolution()
    end = time.time()

    ## Display solution
    for i in Search.solution:
        if i == "SOLVED":
            print(i, "dalam", round(end-start, 100)*1000, "milliseconds")
            print("Jumlah simpul dalam pohon ruang status:",
len(Search.allStates))
        else:
            for j in range(16):
                if j in [3,7,11,15]:
                    if i[j] < 10:
                        print("0"+str(i[j]))
                    else:
                        if i[j] == 16:
                            print(" ")
                        else:
                            print(i[j])
                else:
                    if i[j] < 10:
                        print("0"+str(i[j]), end=" ")
                    else:
                        if i[j] == 16:
                            print(" ", end=" ")
                        else:
                            print(i[j], end=" ")

            print()

```


Bab III

Input-Output Program

Input Program

solved1.txt

```
Input starting puzzle file name: solved1.txt
Randomize puzzle board? *not recommended* [Y/N]: N

Starting state:
01 02 03
05 07 08 04
09 06 11 12
13 10 14 15
```

solved2.txt

```
Input starting puzzle file name: solved2.txt
Randomize puzzle board? *not recommended* [Y/N]: N

Starting state:
01 02 03 04
09 05 06 07
13 10 11 08
14 15 12
```

solved3.txt

```
Input starting puzzle file name: solved3.txt
Randomize puzzle board? *not recommended* [Y/N]: N

Starting state:
02 03 04
01 05 07 08
09 06 10 12
13 14 11 15
```

unsolved1.txt

```
Input starting puzzle file name: unsolved1.txt
Randomize puzzle board? *not recommended* [Y/N]: N

Starting state:
03 09 01 15
14 11 04 06
13 10 12
02 07 08 05
```

unsolved2.txt

```

Input starting puzzle file name: unsolved2.txt
Randomize puzzle board? *not recommended* [Y/N]: N

Starting state:
15 02 01 12
08 05 06 11
04 09 10 07
03 14 13

```

Output Program

solved1.txt

i	kurang(i)
01	0
02	0
03	0
04	0
05	1
06	0
07	2
08	2
09	1
10	0
11	1
12	1
13	1
14	0
15	0
16	13

$\text{Sigma}(\text{kurang}(i)) + X = 22$
 The puzzle is solvable, solving now...

```

01 02 03
05 07 08 04
09 06 11 12
13 10 14 15

01 02 03
05 07 08 04
09 06 11 12
13 10 14 15

01 02 03 04
05 07 08
09 06 11 12
13 10 14 15

01 02 03 04
05 07 08
09 06 11 12
13 10 14 15

01 02 03 04
05 06 07 08
09 10 11 12
13 14 15

01 02 03 04
05 06 07 08
09 10 11 12
13 14 15

01 02 03 04
05 06 07 08
09 11 12
13 10 14 15

SOLVED dalam 2.9997825622558594 milliseconds
Jumlah simpul dalam pohon ruang status: 20

```

solved2.txt

```

i | kurang(i)
01 | 0
02 | 0
03 | 0
04 | 0
05 | 0
06 | 0
07 | 0
08 | 0
09 | 4
10 | 1
11 | 1
12 | 0
13 | 4
14 | 1
15 | 1
16 | 2
Sigma(kurang(i)) + X = 14
The puzzle is solvable, solving now...

```

```

01 02 03 04
09 05 06 07
13 10 11 08
14 15 12

01 02 03 04
09 05 06 07
13 10 11 08
14 15 12

01 02 03 04
05 06 07
09 10 11 08
13 14 15 12

01 02 03 04
05 06 07
09 10 11 08
13 14 15 12

01 02 03 04
05 06 07
09 10 11 08
13 14 15 12

```

```

01 02 03 04
05 06 07
09 10 11 08
13 14 15 12

01 02 03 04
05 06 07 08
09 10 11
13 14 15 12

01 02 03 04
05 06 07 08
09 10 11 12
13 14 15

SOLVED dalam 3.015756607055664 milliseconds
Jumlah simpul dalam pohon ruang status: 19

```

solved3.txt

```

i | kurang(i)
01 | 0
02 | 1
03 | 1
04 | 1
05 | 0
06 | 0
07 | 1
08 | 1
09 | 1
10 | 0
11 | 0
12 | 1
13 | 1
14 | 1
15 | 0
16 | 13
Sigma(kurang(i)) + X = 22
The puzzle is solvable, solving now...

```

```

02 03 04
01 05 07 08
09 06 10 12
13 14 11 15

02 03 04
01 05 07 08
09 06 10 12
13 14 11 15

02 03 04
01 05 07 08
09 06 10 12
13 14 11 15

01 02 03 04
05 07 08
09 06 10 12
13 14 11 15

01 02 03 04
05 07 08
09 06 10 12
13 14 11 15

```

```

01 02 03 04
05 06 07 08
09 10 12
13 14 11 15

01 02 03 04
05 06 07 08
09 10 11 12
13 14 15

01 02 03 04
05 06 07 08
09 10 11 12
13 14 15

SOLVED dalam 2.9959678649902344 milliseconds
Jumlah simpul dalam pohon ruang status: 20

```

unsolved1.txt

```
i | kurang(i)
01 | 0
02 | 0
03 | 2
04 | 1
05 | 0
06 | 2
07 | 1
08 | 1
09 | 7
10 | 4
11 | 7
12 | 4
13 | 6
14 | 10
15 | 11
16 | 6
Sigma(kurang(i)) + X = 63
The puzzle is not solvable.
```

unsolved2.txt

```
i | kurang(i)
01 | 0
02 | 1
03 | 0
04 | 1
05 | 2
06 | 2
07 | 1
08 | 5
09 | 2
10 | 2
11 | 5
12 | 9
13 | 0
14 | 1
15 | 14
16 | 0
Sigma(kurang(i)) + X = 45
The puzzle is not solvable.
```

Lampiran Terkait

Repository Github

https://github.com/rikurakha/Tucil3_13520108

Berkas Teks Instansiasi 15-puzzle

solved1.txt

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

solved1.txt

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

solved1.txt

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

unsolved1.txt

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

unsolved1.txt

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

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		√