

LAPORAN TUGAS KECIL 3 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

Implementasi Algoritma Branch and Bound pada *15-Puzzle Solver*

Tahun Akademik 2021-2022



Oleh

Ken Kalang Al Qalyubi

13520010

PROGRAM STUDI TEKNIK

INFORMATIKA INSTITUT

TEKNOLOGI BANDUNG

BANDUNG

2022

BAB I

Cara Kerja Branch and Bound pada Kasus Puzzle

1. Pada program *15-Puzzle Solver* program pertama-tama akan menentukan apakah puzzle dapat diselesaikan atau tidak dengan menghitung jumlah kurang(i) setiap ubinnya.
2. Jika puzzle dapat diselesaikan, selanjutnya program akan membuat list untuk menyimpan node kemungkinan *move* dan *cost* tiap matriksnya.
3. Program tidak akan memasukkan *move* matriks jika sudah pernah dilewati sebelumnya.
4. Selanjutnya tiap node akan dicek apakah sudah sesuai dengan hasil akhir atau tidak. Pengecekan ini dilakukan dengan mengambil cost terksecil sesuai dengan algoritma *branch and bound*.
5. Jika didapati cost yang sama maka akan terlebih dahulu akan dicek sesuai dengan node yang masuk pertama.
6. Jika sudah didapati node yang sesuai dengan solusi maka program dihentikan dan node yang masih dalam antrian akan dibunuh.

BAB II

Test Case Program

1. Test Case 1 : Puzzle tidak dapat diselesaikan

Input :

```
tc1.txt
1  1 2 3 4
2  5 6 7 8
3  13 16 10 11
4  9 14 15 12
```

Output :

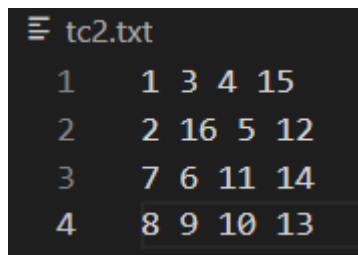
```
Masukkan nama file : tc1.txt
-----
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 13 | - | 10 | 11 |
| 9 | 14 | 15 | 12 |
-----
Jumlah nilai kurang(1) = 0
Jumlah nilai kurang(2) = 0
Jumlah nilai kurang(3) = 0
Jumlah nilai kurang(4) = 0
Jumlah nilai kurang(5) = 0
Jumlah nilai kurang(6) = 0
Jumlah nilai kurang(7) = 0
Jumlah nilai kurang(8) = 0
Jumlah nilai kurang(9) = 0
Jumlah nilai kurang(10) = 1
Jumlah nilai kurang(11) = 1
Jumlah nilai kurang(12) = 0
Jumlah nilai kurang(13) = 4
Jumlah nilai kurang(14) = 1
Jumlah nilai kurang(15) = 1
Jumlah nilai kurang(16) = 6

Jumlah kurang(i) + X adalah 15

Puzzle tidak bisa diselesaikan
Waktu berjalan program 0.010104894638061523 detik
```

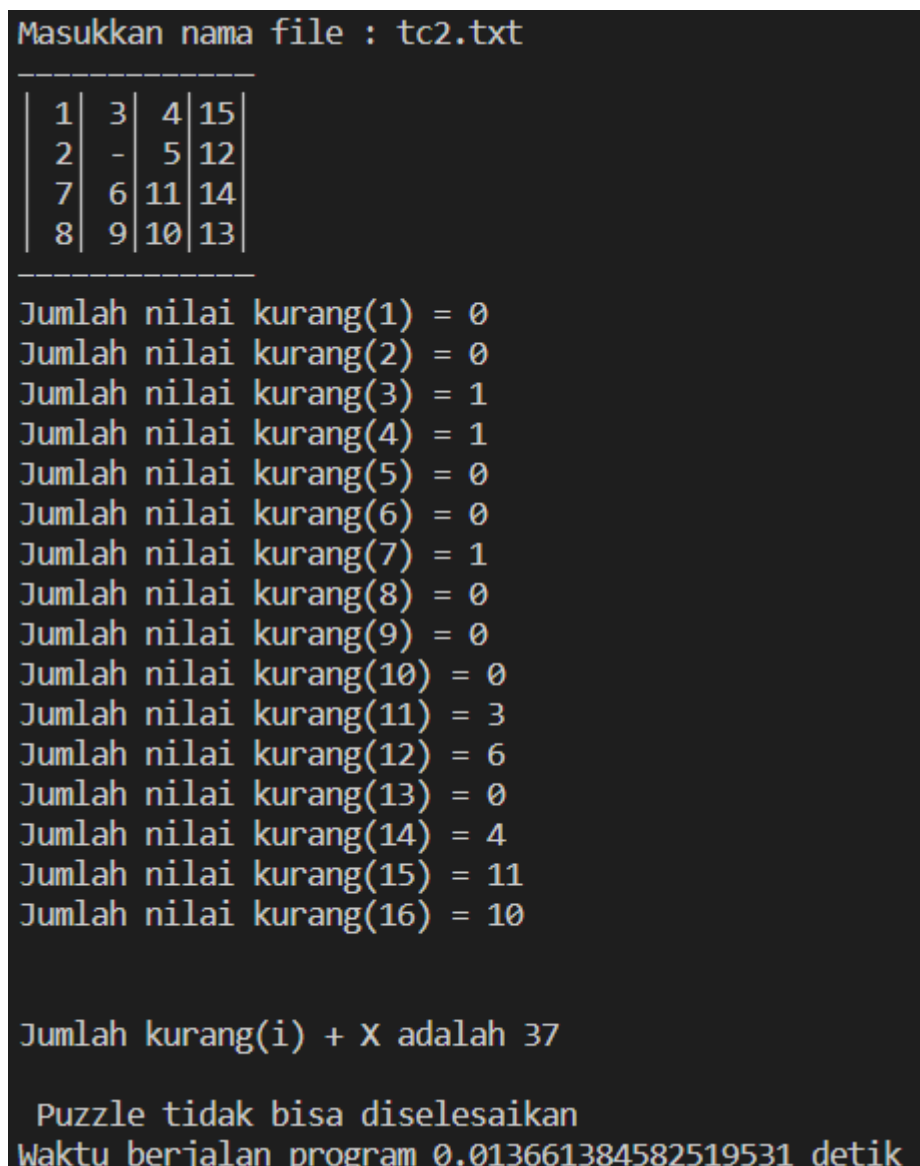
2. Test case 2 : Puzzle tidak dapat diselesaikan

Input :



```
tc2.txt
1 1 3 4 15
2 2 16 5 12
3 7 6 11 14
4 8 9 10 13
```

Output :



```
Masukkan nama file : tc2.txt
-----
| 1 | 3 | 4 | 15 |
| 2 | - | 5 | 12 |
| 7 | 6 | 11 | 14 |
| 8 | 9 | 10 | 13 |
-----
Jumlah nilai kurang(1) = 0
Jumlah nilai kurang(2) = 0
Jumlah nilai kurang(3) = 1
Jumlah nilai kurang(4) = 1
Jumlah nilai kurang(5) = 0
Jumlah nilai kurang(6) = 0
Jumlah nilai kurang(7) = 1
Jumlah nilai kurang(8) = 0
Jumlah nilai kurang(9) = 0
Jumlah nilai kurang(10) = 0
Jumlah nilai kurang(11) = 3
Jumlah nilai kurang(12) = 6
Jumlah nilai kurang(13) = 0
Jumlah nilai kurang(14) = 4
Jumlah nilai kurang(15) = 11
Jumlah nilai kurang(16) = 10

Jumlah kurang(i) + X adalah 37

Puzzle tidak bisa diselesaikan
Waktu berjalan program 0.013661384582519531 detik
```

3. Test case 3 : Puzzle dapat diselesaikan

Input :

tc3.txt

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

Output :

Masukkan nama file : tc3.txt

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

Jumlah nilai kurang(1) = 0
Jumlah nilai kurang(2) = 0
Jumlah nilai kurang(3) = 0
Jumlah nilai kurang(4) = 0
Jumlah nilai kurang(5) = 0
Jumlah nilai kurang(6) = 0
Jumlah nilai kurang(7) = 0
Jumlah nilai kurang(8) = 1
Jumlah nilai kurang(9) = 2
Jumlah nilai kurang(10) = 2
Jumlah nilai kurang(11) = 0
Jumlah nilai kurang(12) = 5
Jumlah nilai kurang(13) = 1
Jumlah nilai kurang(14) = 1
Jumlah nilai kurang(15) = 0
Jumlah nilai kurang(16) = 9

Jumlah kurang(i) + X adalah 22

Puzzle dapat diselesaikan
Solusi ditemukan

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

1	2	3	4
5	6	8	12
9	10	-	7
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	-	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	-

Jumlah simpul yang dibangkitkan = 39
waktu berjalan program 0.033408164978027344 detik

4. Test case 4 : Puzzle dapat diselesaikan

Input :

tc4.txt

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

Output :

Masukkan nama file : tc4.txt

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

Jumlah nilai kurang(1) = 0
Jumlah nilai kurang(2) = 0
Jumlah nilai kurang(3) = 0
Jumlah nilai kurang(4) = 0
Jumlah nilai kurang(5) = 0
Jumlah nilai kurang(6) = 0
Jumlah nilai kurang(7) = 0
Jumlah nilai kurang(8) = 0
Jumlah nilai kurang(9) = 1
Jumlah nilai kurang(10) = 1
Jumlah nilai kurang(11) = 0
Jumlah nilai kurang(12) = 2
Jumlah nilai kurang(13) = 0
Jumlah nilai kurang(14) = 0
Jumlah nilai kurang(15) = 0
Jumlah nilai kurang(16) = 8

Jumlah kurang(i) + X adalah 12

Puzzle dapat diselesaikan

Solusi ditemukan

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

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

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

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

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

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

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

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

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

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

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	-

Jumlah simpul yang dibangkitkan = 2003
Waktu berjalan program 9.448330879211426 detik

5. Test case 5: Puzzle dapat diselesaikan

Input :

tc5.txt				
1	16	1	3	4
2	9	2	6	7
3	10	5	11	8
4	13	14	15	12

Output :

Masukkan nama file : tc5.txt

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

Jumlah nilai kurang(1) = 0
Jumlah nilai kurang(2) = 0
Jumlah nilai kurang(3) = 1
Jumlah nilai kurang(4) = 1
Jumlah nilai kurang(5) = 0
Jumlah nilai kurang(6) = 1
Jumlah nilai kurang(7) = 1
Jumlah nilai kurang(8) = 0
Jumlah nilai kurang(9) = 5
Jumlah nilai kurang(10) = 2
Jumlah nilai kurang(11) = 1
Jumlah nilai kurang(12) = 0
Jumlah nilai kurang(13) = 1
Jumlah nilai kurang(14) = 1
Jumlah nilai kurang(15) = 1
Jumlah nilai kurang(16) = 15

Jumlah kurang(i) + X adalah 30

Puzzle dapat diselesaikan
Solusi ditemukan

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

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

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

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

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

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

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

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

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

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

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

Jumlah simpul yang dibangkitkan = 37
Waktu berjalan program 0.04587817192077637 detik

Kode Program Dalam Bahasa Python

```
import time
import numpy as np
from copy import deepcopy
import sys
sys.setrecursionlimit(2000)

def getCost(start):
    return start.cost

# nge append + sort berdasar cost
def add(queue, start):
    queue.append(start)
    queue.sort(key=getCost)
```

```

# buat pergerakan
def ke_kiri(mat, x,y):
    a = mat[x][y]
    b = mat[x][y-1]
    mat[x][y] = b
    mat[x][y-1] = a
    return mat

def ke_kanan(mat,x,y):
    a = mat[x][y]
    b = mat[x][y+1]
    mat[x][y] = b
    mat[x][y+1] = a
    return mat

def ke_bawah(mat,x,y):
    a = mat[x][y]
    b = mat[x+1][y]
    mat[x][y] = b
    mat[x+1][y] = a
    return mat

def ke_atas(mat,x,y):
    a = mat[x][y]
    b = mat[x-1][y]
    mat[x][y] = b
    mat[x-1][y] = a
    return mat

# boolean cek bisa solve apa ga
def isSolvable(mat):
    count = 0
    list = mat.flatten()
    cek = True
    jumlahkurang = [0 for i in range(16)]
    for i in range (16):
        kurang = 0
        for j in range (i+1,16):
            if list[i] > list[j]:
                kurang += 1
                count += 1
                jumlahkurang[list[i]-1] +=1

    for i in range(16):
        print("Jumlah nilai kurang(" + str(i+1) + ") = " +
str(jumlahkurang[i]))

```



```

print("\n")

if isBlankTileBlack(mat) == True:
    count += 1

print("Jumlah kurang(i) + X adalah " + str(count) + "\n")
if count % 2 != 0:
    cek = False

return cek

# ngecek ubin kosong di arsiran atau tidak
def isBlankTileBlack(mat):
    list = mat.flatten()
    cek = False
    kosong = None
    for i in range (16):
        if list[i] == 16:
            kosong = i
            break
    for i in range (len(black_tile)):
        if kosong == black_tile[i]:
            cek = True
            break
    return cek

# hitung cost matriks belum ditambah depth
def cost(mat):
    count = 0
    cek = mat.flatten()
    sol = solution.flatten()
    for i in range (16):
        if cek[i] != 16 and cek[i] != sol[i]:
            count += 1

    return count

def getBlankx(mat):
    for i in range (4):
        for j in range (4):
            if mat[i][j] == 16:
                return i

def getBlanky(mat):
    for i in range (4):
        for j in range (4):
            if mat[i][j] == 16:

```

```

        return j

#fungsi print
def displayMat(mat):
    puzzle_string = '-' * 13 + '\n'
    for i in range(4):
        for j in range(4):
            if mat[i][j] == 16:
                puzzle_string += '|{0: >2}'.format("-")
                if j == 3:
                    puzzle_string += '|\n'
            else:
                puzzle_string += '|{0: >2}'.format(str(mat[i][j]))
                if j == 3:
                    puzzle_string += '|\n'

    puzzle_string += '-' * 13
    return puzzle_string

# untuk ngecek udah divisit belum
def isVisited(mat,queue):
    flag = False
    for i in range(len(queue)):
        if np.array_equal(mat,queue[i]):
            flag = True
    return flag

# inti program
def solving(mat,queue,solusi,start):
    if isSolvable(mat):
        print("Puzzle dapat diselesaikan")
        penyelesaian(queue,solusi,start)

    else:
        print("Puzzle tidak bisa diselesaikan")

def getBapak(start):
    return start.parent

# buat cari langkah kalo solusi dah ketemu
def cariBapak(queue,start):
    if start.parent == None:
        queue.append(start)
        return
    queue.append(start)
    cariBapak(queue,start.parent)

def getMatriks(start):

```

```

return start.matriks

#tambahan
def ngePrintJalur(list):
    list.reverse()
    for i in range(len(list)):
        path = getMatriks(list[i])
        print(displayMat(path))

# Fungsi solve pakai rekursif
# def solve(mat,queue,solusi,start):
#     global visited
#     global pembangkitan
#     if np.array_equal(mat,solusi):
#         bapakMatriks = []
#         cariBapak(bapakMatriks,start)
#         print("Solusi ditemukan")
#         ngePrintJalur(bapakMatriks)
#         print ("Langkah yang ditempuh = " + str(len(bapakMatriks)))
#         print("Jumlah pembangkitan = " + str(pembangkitan + 1))
#         return
#     else:
#         tempQueue = []
#         if start.blankx != 0:
#             next = ke_atas(deepcopy(mat),start.blankx,start.blanky)
#             if isVisited(next,visited) == False:
#                 move = puzzleSolve(next, cost(next) + start.depth + 1,
getBlankx(next),getBlanky(next), start.depth + 1, start)
#                 add(queue, move)
#                 tempQueue.append(move)

#                 pembangkitan += 1
#         if start.blankx != 3:
#             next = ke_bawah(deepcopy(mat),start.blankx,start.blanky)
#             if isVisited(next,visited) == False:
#                 move = puzzleSolve(next, cost(next) + start.depth + 1,
getBlankx(next),getBlanky(next), start.depth + 1, start)
#                 add(queue, move)
#                 tempQueue.append(move)
#                 pembangkitan += 1
#         if start.blanky != 0:
#             next = ke_kiri(deepcopy(mat),start.blankx,start.blanky)
#             if isVisited(next,visited) == False:
#                 move = puzzleSolve(next, cost(next) + start.depth + 1,
getBlankx(next),getBlanky(next), start.depth + 1, start)
#                 add(queue, move)
#                 tempQueue.append(move)
#                 pembangkitan += 1

```

```

#         if start.blanky != 3:
#             next = ke_kanan(deepcopy(mat),start.blankx,start.blanky)
#             if isVisited(next,visited) == False:
#                 move = puzzleSolve(next, cost(next) + start.depth + 1,
getBlankx(next),getBlanky(next), start.depth + 1, start)
#                 add(queue, move)
#                 tempQueue.append(move)
#                 pembangkitan += 1
#             visited.append(mat)
#             path = dequeue(queue)
#             solve(path.matriks,queue,solusi,path)

# Fungsi solve pakai iterasi
def penyelesaian(queue,solusi,start):
    global visited
    global pembangkitan
    path = start
    visited.add(tuple(np.reshape(path.matriks,16)))
    while np.array_equal(path.matriks,solusi) == False:

        if path.blankx != 0:
            next = ke_atas(deepcopy(path.matriks),path.blankx,path.blanky)
            if tuple(np.reshape(next,16)) not in visited:
                move = puzzleSolve(next, cost(next) + path.depth + 1,
getBlankx(next),getBlanky(next), path.depth + 1, path)
                add(queue, move)
                visited.add(tuple(np.reshape(next,16)))
                pembangkitan += 1
        if path.blankx != 3:
            next = ke_bawah(deepcopy(path.matriks),path.blankx,path.blanky)
            if tuple(np.reshape(next,16)) not in visited:
                move = puzzleSolve(next, cost(next) + path.depth + 1,
getBlankx(next),getBlanky(next), path.depth + 1, path)
                add(queue, move)
                visited.add(tuple(np.reshape(next,16)))
                pembangkitan += 1
        if path.blanky != 0:
            next = ke_kiri(deepcopy(path.matriks),path.blankx,path.blanky)
            if tuple(np.reshape(next,16)) not in visited:
                move = puzzleSolve(next, cost(next) + path.depth + 1,
getBlankx(next),getBlanky(next), path.depth + 1, path)
                add(queue, move)
                visited.add(tuple(np.reshape(next,16)))
                pembangkitan += 1
        if path.blanky != 3:
            next = ke_kanan(deepcopy(path.matriks),path.blankx,path.blanky)
            if tuple(np.reshape(next,16)) not in visited:

```

```

        move = puzzleSolve(next, cost(next) + path.depth + 1,
getBlankx(next),getBlanky(next), path.depth + 1, path)
        add(queue, move)
        visited.add(tuple(np.reshape(next,16)))
        pembangkitan += 1

    path = queue.pop(0)

    bapakMatriks = []
    cariBapak(bapakMatriks,path)
    print("Solusi ditemukan")
    ngePrintJalur(bapakMatriks)
    print("Jumlah simpul yang dibangkitkan = " + str(pembangkitan + 1))
    queue.clear()

# baca file eksternal
def teks_to_matriks(file_eks):
    matriks = []
    with open(file_eks) as file:
        for char in file:
            matriks.append([int(i) for i in char.split()])
    return matriks

# buat simpen data parent dsb
class puzzleSolve:
    def __init__(self,matriks,cost,blankx,blanky,depth,parent):
        self.matriks = matriks
        self.cost = cost
        self.blanky = blanky
        self.blankx = blankx
        self.depth = depth
        self.parent = parent

black_tile = [1,3,4,6,9,11,12,14]
solution = np.array([[1,2,3,4],
                    [5,6,7,8],
                    [9,10,11,12],
                    [13,14,15,16]])

pembangkitan = 0

queue = []
visited = []

fileinput = input("Masukkan nama file : ")

```

```

dummy = np.array(teks_to_matriks("test/"+fileinput))
#buat simpul pertama
start = puzzleSolve(dummy,0,getBlankx(dummy),getBlanky(dummy),0,None)

waktu = time.time()
print(displayMat(dummy))
solving(dummy,queue,solution,start)
end = time.time()

# total time taken
print(f"Waktu berjalan program {end - waktu}" + " detik")

```

Berkas teks *test case* program :

tc1.txt :

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

tc2.txt :

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

tc3.txt :

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

tc4.txt :

1 2 3 4

5 6 7 16

9 10 12 8

11 13 14 15

tc5.txt :

16 1 3 4

9 2 6 7

10 5 11 8

13 14 15 12

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat menerima input dan menuliskan output	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat		√

Link github : <https://github.com/kenkalang/Tucil-3-STIMA-15PuzzleSolver>