Tugas Kecil 3 IF 2211 Strategi Algoritma
Semester II Tahun 2021/2022

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

DISUSUN OLEH

Kevin Roni **13520114**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2022

BAB I
ALGORITMA BRANCH AND BOUND

Deskripsi

Algoritma *Branch and Bound* biasanya digunakan untuk menyelesaikan persoalan berupa optimisasi. Singkatnya, *Branch and Bound* atau yang biasa disingkat B&B adalah algoritma BFS (*Breath First Search)* yang digabungkan dengan *least cost search*. Jika BFS dilakukan pencarian berdasarkan urutan masuk (FIFO), maka B&B berdasarkan cost yang dimiliki tiap simpul. Ekspansi dilakukan berdasarkan simpul yang memiliki cost terbesar/terkecil (tergantung persoalan maksimasi/minimasi)

Algoritma Branch and Bound

Pada dasarnya, algoritma *Branch and Bound* memiliki ide untuk membuat antrian (priority queue) berdasarkan cost dan membangkitkan anak simpul sesuai urutan pada antrian. Secara detail, program yang saya buat dapat dijelaskan melalui langkah-langkah sebagai berikut:

1. Mula-mula, program akan membaca file input dan mengkonversi file input menjadi matriks yang disimpan di memori internal program
2. Program kemudian akan mencari nilai $\sum_{i=1}^{16} KURANG(i) + X$
3. *KURANG(i)* adalah jumlah sel bernomor j sehingga ada j < i dan posisi posisi(j) > posisi(i). Sedangkan X akan bernilai 1 apabila nilai kolom dan baris pada sel kosong berbeda ganjil-genapnya.
4. Jika bernilai genap, maka tujuan dapat dicapai. Jika ganjil, maka tujuan tidak dapat dicapai
5. Jika tujuan dapat dicapai berdasarkan poin 4, maka program akan memasukkan simpul matriks mula-mula ke dalam antrian (*priority queue*) pq sebagai simpul akar
6. Jika pq kosong, maka program berhenti
7. Jika pq tidak kosong, maka program akan mengambil simpul i dengan cost paling kecil (melakukan metode pop karena sudah terurut berdasarkan cost)
8. Cost yang dimaksud adalah panjang lintasan dari simpul akar ke simpul i ditambah dengan taksiran panjang lintasan terpendek dari p ke simpul solusi. Taksiran dapat dilakukan dengan menghitung banyaknya sel bukan kosong yang tidak berada pada tempat seharusnya berdasarkan susunan akhir
9. Jika simpul i adalah solusi, maka program akan berhenti dan mencetak langkah-langkah untuk mencapai simpul solusi dari simpul akar
10. Jika bukan solusi, maka simpul i akan membangkitkan semua anak-anak yang mungkin dan tidak repetitive
11. Yang mungkin berarti jika sel kosong sudah berada pada sel paling kanan berarti tidak menggeser sel kosong ke kanan dan begitu pula untuk 3 arah lain yang mungkin (bawah, kiri, dan atas). Tidak repetitif berarti tidak melakukan kebalikan arah dari simpul I (jika simpul i adalah hasil pergerakan ke bawah, maka simpul anaknya tidak boleh bergerak ke atas lagi)
12. Semua anak-anak yang dibangkitkan akan dimasukkan ke dalam pq
13. Ulangi langkah 6

BAB II
IMPLEMENTASI PROGRAM

solver.py

```
import numpy as np
from heapq import heappush, heappop
import time

nodeCreated = 0 #count the number of node created

def read_matrix(filename): #read the matrix from the file
    with open(filename, 'r') as fp:
        lines = fp.readlines()
    lines = [line.strip() for line in lines]
    lines = [line.split(' ') for line in lines]
    lines = [[int(x) for x in line] for line in lines]
    return np.array(lines)

def randomize(): #function to randomize the puzzle
    puzzle = np.arange(1,17)
    puzzle = puzzle.reshape(4,4)
    puzzle = np.random.permutation(puzzle)
    return puzzle

def matrixToList(arr): #convert the matrix to list
    arr1=[]
    for y in arr:
        for x in y:
            arr1.append(x)
    return arr1

def sigmaKurang(arr, listKurang): #function to calculate sigma kurang
    arr1=matrixToList(arr)
    inv_count = 0
    for i in range(16):
        curr_count = 0;
        for j in range(i + 1,16):
            if (arr1[i] > arr1[j]):
                inv_count+=1
                curr_count+=1
        listKurang[arr1[i]] = curr_count
    return inv_count

def checkEmptyPosition(arr): #check if the empty position value is zero or one
    for i in range(4):
        for j in range(4):
            if (arr[i][j] == 16):
                return int((i%2 == 0 and j%2 == 1) or (i%2 == 1 and j%2 == 0))
```

```python
def sigmaKurangIPlusX(sigmakurang, X): #function to calculate sigma kurang[I]
+ X
    return (sigmakurang + X)

def isPossible(sigmakurang, X): #check if the puzzle is possible to solve
    return (sigmakurang + X) % 2 == 0

def calculateCost(mat, final): #function to calculate the estimation step
taken
    count = 0
    for i in range(4):
        for j in range(4):
            if (mat[i][j] != final[i][j] and mat[i][j] != 16):
                count += 1
    return count

def printMatrix(result): #print the matrix
    for row in range(4):
        for col in range(4):
            if result[row][col] == 16:
                if col == 3:
                    print("-", end="\n")
                else:
                    print("-", end="\t")
            else:
                if col == 3:
                    print(result[row][col], end="\n")
                else:
                    print(result[row][col], end="\t")
    print()

def getEmptyPosition(mat): #function to get the empty position
    for i in range(4):
        for j in range(4):
            if (mat[i][j] == 16):
                return (i,j)

def printPath(root): #function to print the path from initial puzzle to final
puzzle

    if root.parent == None:
        return

    printPath(root.parent)
    printMove(root.move)
    printMatrix(root.mat)
    print()
```

```
def copyMatrix(mat): #function to copy the matrix
    newList = [[0 for j in range(4)] for i in range(4)]
    for i in range(4):
        for j in range(4):
            newList[i][j] = mat[i][j]
    return newList

def isIdxValid(X, Y): #check if the index is valid
    return X >= 0 and X < 4 and Y >= 0 and Y < 4

class node: #class for node, used to generate solution tree
    def __init__(self, parent, mat, emptyPos, cost, level, move):
        self.parent = parent
        self.mat = mat
        self.emptyPos = emptyPos
        self.cost = cost
        self.level = level
        self.move = move

    def __lt__(self, nxt): #compare the node based on the cost, so the
prioqueue will be sorted based on cost
        return self.cost < nxt.cost

class priorityQueue: #class for priority queue
    def __init__(self):
        self.heap = []

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

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

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

def createNode(mat, emptyPos, emptyPosAfter, level, parent, final, move):
    #function to create a node while moving
    global nodeCreated
    nodeCreated += 1
    newMatrix = copyMatrix(mat)
    x1 = emptyPos[0]
    y1 = emptyPos[1]
    x2 = emptyPosAfter[0]
    y2 = emptyPosAfter[1]
```

```
    newMatrix[x1][y1], newMatrix[x2][y2] = newMatrix[x2][y2],
newMatrix[x1][y1] #swap the empty position with the new position
    newlevel = level + 1
    newcost = calculateCost(newMatrix, final) + newlevel
    newNode = node(parent, newMatrix, emptyPosAfter, newcost, newlevel, move)
    return newNode

def move(x): #function to move the empty position, will be added when the
empty position is moved
    if (x == 0):
        return (-1,0)
    elif (x == 1):
        return (0,-1)
    elif (x == 2):
        return (1,0)
    elif (x == 3):
        return (0,1)

def printMove(x):
    #defined in the order of move (0 = up, 1 = left, 2 = down, 3 = right)
    if (x == None):
        print("Initial Matrix")
    elif (x == 0):
        print("Up")
    elif (x == 1):
        print("Left")
    elif (x == 2):
        print("Down")
    elif (x == 3):
        print("Right")

def solve(mat, final):
    emptyPos = getEmptyPosition(mat)
    pq = priorityQueue()
    start = time.time()
    if isPossible(sigmaKurang(mat, [0]*17), checkEmptyPosition(mat)):
        pq.push(node(None, mat, emptyPos, calculateCost(mat, final), 0, None))
#initial node
        while not pq.isEmpty():
            leastCost = pq.pop();  #pop the node with the least cost
            if (leastCost.cost == 0 or leastCost.cost == leastCost.level): #if
the cost estimation is zero, the puzzle is solved
                stop = time.time()
                print("Solution found\n")
                print("Total number of nodes created: " + str(nodeCreated))
                print("Step by step solution: \n")
                printPath(leastCost)
                print("Total time taken: " + str(stop - start))
```

```
                return
            else:
                #check if the empty position can be moved in the direction of
i
                #if the empty position can be moved, create a new node
                #in order to minimize step taken to solve the puzzle
                #program will not do any repetition move, such as move up then
move down
                for i in range(4):
                    if leastCost.move == None or leastCost.move == i:
                        emptyPosChild = leastCost.emptyPos[0] + move(i)[0],
leastCost.emptyPos[1] + move(i)[1]
                        if isIdxValid(emptyPosChild[0], emptyPosChild[1]):
                            pq.push(createNode(leastCost.mat,
leastCost.emptyPos, emptyPosChild, leastCost.level, leastCost, final, i))
                    else:
                        currmove = i%2 #since the move is in the order of up,
left, down, right, the move is categorized (by modulo 2) either 0 or 1
                        parentmove = leastCost.move % 2 #if the modulo result
is the same, the move is repetition
                        if (currmove != parentmove):
                            emptyPosChild = leastCost.emptyPos[0] +
move(i)[0], leastCost.emptyPos[1] + move(i)[1]
                            if isIdxValid(emptyPosChild[0], emptyPosChild[1]):
                                pq.push(createNode(leastCost.mat,
leastCost.emptyPos, emptyPosChild, leastCost.level, leastCost, final, i))


    else:
        print("No Possible Solution")
```

main.py

```
from solver import *

try:
    print("15 Puzzle Solver Using BNB")
    print("\nChoose your input :\n1. Random By Program\n2. File Input")
    Option = int(input("\nSelect Option : "))
    if Option == 1:
        print("\nRandom By Program may take a while to generate a
solution.\n")
        print("\nContinue? (y/n)\n")
        if input("(y/n) : ") == "y":
            print("\nGenerating random puzzle...")
            puzzle = randomize()
            final = read_matrix("final.txt")
            print("Puzzle To Solve:\n")
            printMatrix(puzzle)
```

```
            print("Kurang[i] Value:\n")
            listKurang = [0]*17
            sigmakurang = sigmaKurang(puzzle, listKurang)
            for i in range(1,17):
                print("kurang[" + str(i)+ "] = " + str(listKurang[i]))
            print("\nSigma Kurang[i] + X: "+
str(sigmaKurangIPlusX(sigmakurang, checkEmptyPosition(puzzle)))+ "\n")
            solve(puzzle, final)
        else:
            print("\nProgram dihentikan\n")
            exit()
    elif Option == 2:
        filename = input("\nInput Filename: ")
        file = "../test/" + filename
        puzzle = read_matrix(file)
        final = read_matrix("final.txt")
        print("Puzzle To Solve:\n")
        printMatrix(puzzle)
        print("Kurang[i] Value:\n")
        listKurang = [0]*17
        sigmakurang = sigmaKurang(puzzle, listKurang)
        for i in range(1,17):
            print("kurang[" + str(i)+ "] = " + str(listKurang[i]))
        print("\nSigma Kurang[i] + X: "+ str(sigmaKurangIPlusX(sigmakurang,
checkEmptyPosition(puzzle)))+ "\n")
        solve(puzzle, final)
    else:
        print("\nInvalid Input. Program Stop\n")

except:
    print("\nInvalid filename input, make sure your test case is already in
test folder. Program Stop\n")
```

BAB III
INPUT OUTPUT

Berkas test-case untuk input terlampir

1. **tc1.txt**

```
15 Puzzle Solver Using BNB

Choose your input :
Puzzle To Solve:

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

Kurang[i] Value:

kurang[1] = 0
kurang[2] = 1
kurang[3] = 2
kurang[4] = 3
kurang[5] = 4
kurang[6] = 5
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[i] + X: 31

No Possible Solution
```

2. **tc2.txt**

```
Choose your input :
1. Random By Program
2. File Input

Select Option : 2

Input Filename: tc2.txt
Puzzle To Solve:

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

Kurang[i] Value:

kurang[1] = 0
kurang[2] = 1
kurang[3] = 2
kurang[4] = 3
kurang[5] = 4
kurang[6] = 4
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[i] + X: 30

Solution found
```

```
Total number of nodes created: 98203
Step by step solution:

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


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


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


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


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

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


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


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


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


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

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


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


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


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


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

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


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


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


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


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

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


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


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


Total time taken: 3.621924638748169
```

3. **tc3.txt**

```
Choose your input :
1. Random By Program
2. File Input

Select Option : 2

Input Filename: tc3.txt
Puzzle To Solve:

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

Kurang[i] Value:

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

Sigma Kurang[i] + X: 20

Solution found
```

```
Total number of nodes created: 15357
Step by step solution:

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


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


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


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


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

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


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


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


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


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

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


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


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


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


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

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


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


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


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


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

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


Total time taken: 0.4389040470123291
```

4. tc4.txt

```
Select Option : 2

Input Filename: tc4.txt
Puzzle To Solve:

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

Kurang[i] Value:

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

Sigma Kurang[i] + X: 14

Solution found

Total number of nodes created: 438
Step by step solution:
```

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


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


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


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


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

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


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


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


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


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

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


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


Total time taken: 0.0176393985748291
```

5. **tc5.txt**

```
1. Random By Program
2. File Input

Select Option : 2

Input Filename: tc5.txt
Puzzle To Solve:

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

Kurang[i] Value:

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

Sigma Kurang[i] + X: 37

No Possible Solution
```

BAB IV
PENUTUP

Link repository: https://github.com/jakartasipirok/Using-BNB-to-Solve-15-Words-Puzzle

Poin	Ya	Tidak
1. Program berhasil dikompilasi	v	
2. Program berhasil running	v	
3. Program dapat menerima input dan menuliskan output	v	
4. luaran sudah benar untuk semua data uji	v	
5. Bonus dibuat		v

BAB V
LAMPIRAN

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Tugas-Kecil-3-(2022).pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

Instansiasi berkas pengujian:

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

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

tc3.txt
3 1 2 4
16 5 7 8
10 6 11 12
9 13 14 15

tc4.txt
1 2 3 4
5 6 7 8
9 12 15 14
13 11 16 10

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