TUGAS KECIL 1 IF2211 STRAREGI ALGORITMA

SEMESTER II TAHUN 2019/2020

PENYELESAIAN PERSOALAN 15-PUZZLE

DENGAN ALGORITMA *BRANCH AND BOUND*

Disusun oleh:
Syarifuddin Fakhri A (13518095)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2020

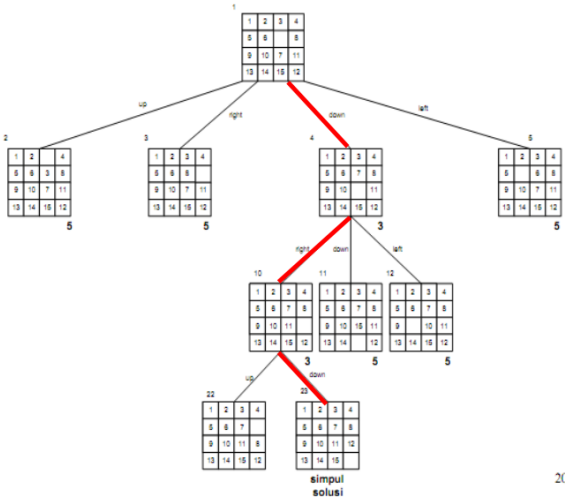
BAB 1
DESKRIPSI DAN SPESIFIKASI TUGAS

Buatlah program dalam Python untuk menyelesaikan persoalan 15-Puzzle dengan menggunakan Algoritma *Branch and Bound* seperti pada materi kuliah. Nilai *bound* tiap simpul adalah penjumlahan *cost* yang diperlukan untuk sampai suatu simpul x dari akar, dengan taksiran *cost* simpul x untuk sampai ke *goal*. Taksiran *cost* yang digunakan adalah jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir *(goal state)*. Untuk semua instansiasi persoalan 15-Puzzle, susunan akhir yang diinginkan sesuai dengan Gambar 1.1.

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

Gambar 1.1 Susunan Akhir Persoalan 15-Puzzle

Masukan: matriks yang merepresentasikan posisi awal suatu instansiasi persoalan 15-Puzzle. Matriks dibaca dari berkas teks. Program harus dapat menentukan apakah posisi awal suatu masukan dapat diselesaikan hingga mencapai susunan akhir, dengan mengimplementasikan fungsi Kurang(i) dan posisi ubin kosong di kondisi awal (X), seperti pada materi kuliah. Jika posisi awal tidak bisa mencapai susunan akhir, program akan menampilkan pesan tidak bisa diselesaikan,. Jika dapat diselesaikan, program dapat menampilkan urutan matriks rute (*path*) aksi yang dilakukan dari posisi awal ke susunan akhir. Sebagai contoh pada Gambar 1.2, matriks yang ditampilkan ke layar adalah matriks pada simpul 1, simpul 4, simpul 10 dan simpul 23.



Gambar 1.2 Contoh Pohon Ruang Status Persoalan 15-Puzzle

Keluaran Program ini adalah sebagai berikut:

1. Matriks posisi awal.
2. Nilai dari fungsi Kurang (i) untuk setiap ubin tidak kosong pada posisi awal (nilai ini tetap dikeluarkan, baik persoalan bisa diselesaikan atau tidak bisa diselesaikan).
3. Nilai dari $\sum_{i=1}^{16} KURANG(i) + X$
4. Jika persoalan tidak dapat diselesaikan (berdasarkan hasil butir 2) keluar pesan.
5. Jika persoalan dapat diselesaikan(berdasarkan hasil butir 2), menampilkan urutan matriks seperti pada penjelasan sebelumnya.
6. Waktu eksekusi
7. Jumlah simpul yang dibangkitkan dalam pohon ruang status.

BAB 2
DASAR TEORI

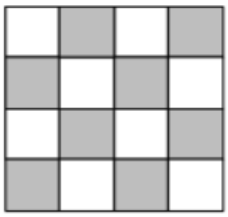
Algoritma *Branch and Bound*

Algoritma *branch and bound* adalah algoritma yang membagi permasalahan menjadi upa masalah yang lebih kecil dengan menerakan metode percabangan (*branching*) dan melakukan pembatasan (*bounding*) untuk mencapai solusi optimal. Percabangan (*branching*) merupakan metode pembentukan permasalahan ke dalam bentuk pohon pencarian (*search tree*). Proses percabangan ini diharapkan dapat membangkitkan solusi pada cabangnya dan proses pembatasan dilakukan dengan menghitung estimasi nilai (*cost*) simpul dengan memperhatikan batas. Secara umum algoritma ini adalah gabungan dari algoritma *Breadth First Search* dan algoritma *Least Cost Search* yang setiap simpul hidupnya mempunyai suatu nilai tertentu. Algoritma ini umumnya digunakan untuk memecahkan persoalan optimasi yaitu meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan persoalan.

BAB 3
ANALISIS PERSOALAN

Dalam mengerjakan tugas ini, persoalan 15-Puzzle menggunakan algoritma *Branch and Bound* untuk menyelesaikannya. Cara kerja dalam penyelesaian masalah ini adalah sebagai berikut :

1. Input data matriks *puzzle* yang akan diuji
2. Lakukan pengecekan apakah data matriks *puzzle* dapat diselesaikan atau tidak dengan rumus $\sum_{i=1}^{16} KURANG(i) + X$ dengan nilai Kurang(i) adalah banyaknya ubin bernomor j sedemikian sehingga j<i dan posisiUbin(j)> posisiUbin(i). Jika ubin kosong merupakan sel yang diarsir pada Gambar 3.1, maka nilai X=1 dan jika tidak nilai X=0. Jika tidak program akan mengeluarkan pesan dan berhenti



Gambar 3.1 Letak Arsiran Ubin Kosong

3. Jika *puzzle* dapat diselesaikan, gunakan algoritma *Branch and Bound* sebagai berikut
   a. Masukkan matriks *puzzle* yang akan diselesaikan ke dalam *queue*
   b. Cek jumlah ubin yang tidak kosong yang tidak terdapat pada susunan final pada simpul hidup, jika 0 maka *puzzle* sudah sesuai dengan simpul final dan program berhenti.
   c. Ambil matriks *puzzle* dari *queue* yang memiliki *cost* terkecil dan jadikan simpul sekarang
   d. *Push* semua cabang dari simpul sekarang yang tidak membentuk simpul sebelumnya ke dalam *queue*
   e. Hapus simpul sekarang yang berada di *queue*
   f. Ulangi langkah 3.b hingga sudah menemukan simpul final

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Program

a. Fungsi ReachableGoal

Fungsi dengan parameter matriks *puzzle* dan mengembalikan boolean apakah sebuah *puzzle* dapat diselesaikan atau tidak. Di dalam fungsi ini juga akan mencetak fungsi Kurang(i) dan Nilai Sigma(Kurang(i))+X.

```
def ReachableGoal(a) :
    kurang = [0 for i in range (16)]
    for i in range (16) :
        if (a[i] == "X") :
            idxEmp = i
            temp = 16
        else :
            temp = int(a[i])
        for j in range (i+1, 16) :
            if(a[j] == "X"):
                temp2 = 16
            else :
                temp2 = int(a[j])
            if(temp > temp2) :
                kurang[temp-1] += 1
    for i in range (16) :
        print("Kurang ("+ str(i+1)+ ") =", kurang[i])
    print()
    if ((idxEmp+1)%2 == ((idxEmp)//4+1)%2) :
        print("Nilai Sigma(Kurang(i))+X =", sum(kurang))
        print()
        return (sum(kurang)%2 == 0)
    else :
        print("Nilai Sigma(Kurang(i))+X =", (sum(kurang)+1))
        print()
        return ((sum(kurang)+1)%2 == 0)
```

b. Fungsi PrintPuzzle

Fungsi dengan parameter matriks *puzzle* dan mencetak *puzzle* ke layar

```
def PrintPuzzle(a) :
    for i in range (4) :
        for j in range (4) :
            print(a[i][j], end=" ")
               print()
```

c. Fungsi LocX

Fungsi dengan parameter matriks *puzzle* dan mengembalikan posisi X (ubin kosong) dengan nilai X antara 1 hingga 16

```
def LocX(a) :
    for i in range (4) :
        for j in range (4) :
            if (a[i][j] == "X") :
                return (i*4)+j+1
```

d. Fungsi IsMoveToAvailable

Fungsi dengan parameter matriks *puzzle* dan char dir serta mengembalikan boolean apakah sebuah *puzzle* ubin kosongnya dapat di geser ke arah dir

```
def IsMoveToAvailable(a, dir) :
    locX = LocX(a)
    if (dir == "L") :
        if (locX == 1 or locX == 5 or locX == 9 or locX == 13) :
            return False
        else :
            return True
    elif (dir == "R") :
        if (locX == 4 or locX == 8 or locX == 12 or locX == 16) :
            return False
        else :
            return True
    elif (dir == "U") :
        if (locX > 0 and locX < 5) :
            return False
        else :
            return True
    elif (dir == "D") :
        if (locX > 12 and locX < 17) :
            return False
        else :
            return True
```

e. Fungsi MoveTo

Fungsi dengan parameter matriks *puzzle* dan char dir serta mengembalikan matriks *puzzle* setelah ubin kosongnya di geser ke arah dir

```
def MoveTo(a, dir) :
    b = copy.deepcopy(a)
    RowX = (LocX(a)-1)//4
    ColX = (LocX(a)-1)-(RowX*4)
    if (dir == "L") :
        b[RowX][ColX], b[RowX][ColX-1] = b[RowX][ColX-1], b[RowX][ColX]
    elif (dir == "R") :
        b[RowX][ColX], b[RowX][ColX+1] = b[RowX][ColX+1], b[RowX][ColX]
    elif (dir == "U") :
        b[RowX][ColX], b[RowX-1][ColX] = b[RowX-1][ColX], b[RowX][ColX]
    elif (dir == "D") :
        b[RowX][ColX], b[RowX+1][ColX] = b[RowX+1][ColX], b[RowX][ColX]
    return b
```

f. Fungsi nUbinNotFinal

Fungsi dengan parameter matriks dan mengembalikan jumlah ubin yang tidak kosong yang tidak terdapat pada susunan akhir

```
def nUbinNotFinal(a):
    temp = "1"
    count = 0
    for i in range (4) :
        for j in range (4) :
            if (temp != a[i][j] and a[i][j] != "X") :
                count += 1
            temp = str((i*4)+j+2)
    return count
```

g. Fungsi nUbinNotFinal dan Fungsi Cost

Fungsi dengan parameter matriks *puzzle* dan int depth serta mengembalikan cost simpul a dengan kedalaman depth

```
def Cost(a, depth) :
    return nUbinNotFinal(a)+depth
```

h. Main program

```
pzl = [0 for i in range (16)]
namaFile = input("Masukkan nama file puzzle: ")
print()
print("Puzzle yang akan diselesaikan : ")
file = open(namaFile, "r")
for i in range (4) :
    strLine = file.readline().replace('\n', '').replace('\r', '')
    pzl[i*4], pzl[i*4+1], pzl[i*4+2], pzl[i*4+3] = strLine.split(" ")
    print(pzl[i*4], pzl[i*4+1], pzl[i*4+2], pzl[i*4+3])
print()
timeStart = perf_counter_ns()
countNode = 0
if (not ReachableGoal(pzl)) :
    print("Persoalan tidak dapat diselesaikan")
else :
    puzzle = [[0 for i in range (4)] for j in range (4)]
    for i in range (4):
        for j in range (4):
            puzzle[i][j] = pzl[i*4+j]
    if (nUbinNotFinal(puzzle) == 0) :
        print("Puzzle ini sudah pada susunan akhir! :")
        PrintPuzzle(puzzle)
    else :
        print("Urutan penyelesaian puzzle :")
        queue = []
        tplCostPzl = (Cost(puzzle, 0), puzzle, "Root", 0)
        queue.append(tplCostPzl)
        goal = False
        while (not goal) :
            temp = min(queue)
            if (nUbinNotFinal(temp[1]) == 0) :
                PrintPuzzle(temp[1])
                queue.remove(temp)
                goal = True
            else :
                print(temp[3])
                PrintPuzzle(temp[1])
                depth = temp[3]+1
                if (IsMoveToAvailable(temp[1], "L") and temp[2] != "R") :
                    tempPzl = MoveTo(temp[1], "L")
                    tplCostPzl = (Cost(tempPzl, depth), tempPzl, "L", depth)
                    queue.append(tplCostPzl)
                    countNode +=1
                if (IsMoveToAvailable(temp[1], "R") and temp[2] != "L") :
                    tempPzl = MoveTo(temp[1], "R")
                    tplCostPzl = (Cost(tempPzl, depth), tempPzl, "R", depth)
                    queue.append(tplCostPzl)
                    countNode +=1
                if (IsMoveToAvailable(temp[1], "U") and temp[2] != "D") :
                    tempPzl = MoveTo(temp[1], "U")
                    tplCostPzl = (Cost(tempPzl, depth), tempPzl, "U", depth)
                    queue.append(tplCostPzl)
                    countNode +=1
                if (IsMoveToAvailable(temp[1], "D") and temp[2] != "U") :
                    tempPzl = MoveTo(temp[1], "D")
                    tplCostPzl = (Cost(tempPzl, depth), tempPzl, "D", depth)
```

```
                    queue.append(tplCostPzl)
                    countNode +=1                queue.remove(temp)
                print("   ->   ")
print()
timeEnd = perf_counter_ns()
print("Waktu eksekusi :", (timeEnd-timeStart)/pow(10,6), "milidetik")
print()
print("Jumlah simpul yang dibangkitkan :", countNode)
```

4.2 Pengujian

Spesifikasi komputer dalam pengujian ini sebagai berikut:

a. Prosesor : AMD A6-3400M APU with Radeon(tm) HD Graphics 1,40 GHz
b. RAM : 4 GByte (*dual chanel*)
c. System : 64-bit Operating System
d. OS : Windows 7 Professional

Dalam program yang telah dibuat, program meminta masukan sebuah string namafile yang alamat penyimpanan filenya sama dengan penyimpanan program seperti di Gambar 4.1. File tersebut mempunyai format matriks 4×4 yang berisi angka acak 1-15 dan huruf X untuk melambangkan ubin kosong. Setiap angka atau huruf X dipisahkan dengan spasi kecuali setiap simbol keempat akan dipisahkan dengan enter. Keluaran program adalah mencetak segala keluaran yang diminta pada BAB 1. Untuk pengujian ini, digunakan 5 kasus sebagai berikut:

Puzzle1.txt	Puzzle2.txt	Puzzle3.txt	Puzzle4.txt	Puzzle5.txt
1 2 3 4 5 6 X 8 9 10 7 11 13 14 15 12	X 1 5 4 7 6 9 8 11 10 13 12 15 2 3 14	5 1 7 3 9 2 11 4 13 6 10 8 14 15 X 12	3 7 2 1 9 10 5 11 6 15 13 4 8 12 14 X	5 1 2 3 6 10 8 4 13 9 7 15 14 X 12 11

```
F:\Program\stima\tucil3>python 15-Puzzle.py
Masukkan nama file puzzle: Puzzle1.txt
```

Gambar 4.1 Gambaran Masukan Program

Untuk keluaran masing-masing masukan adalah sebagai berikut:

a. File Puzzle1.txt

```
Puzzle yang akan
diselesaikan :
1 2 3 4
5 6 X 8
9 10 7 11
13 14 15 12

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

Nilai
Sigma(Kurang(i))+X
= 16

Urutan penyelesaian
puzzle :
1 2 3 4
5 6 X 8
9 10 7 11
13 14 15 12
   ->
1 2 3 4
5 6 7 8
9 10 X 11
13 14 15 12
   ->
1 2 3 4
5 6 7 8
9 10 11 X
13 14 15 12
   ->
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 X

Waktu eksekusi :
6.974191 milidetik

Jumlah simpul yang
dibangkitkan : 9
```

b. File Puzzle2.txt

```
Puzzle yang akan diselesaikan :
X 1 5 4
7 6 9 8
11 10 13 12
15 2 3 14

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

Nilai Sigma(Kurang(i))+X = 43

Persoalan tidak dapat diselesaikan

Waktu eksekusi : 1.623745 milidetik

Jumlah simpul yang dibangkitkan : 0
```

c. File Puzzle3.txt

```
Puzzle yang akan diselesaikan :
5 1 7 3
9 2 11 4
13 6 10 8
14 15 X 12

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

Nilai Sigma(Kurang(i))+X = 26

Urutan penyelesaian puzzle :
5 1 7 3
9 2 11 4
13 6 10 8
14 15 X 12
   ->
5 1 7 3
9 2 11 4
13 6 10 8
14 X 15 12
   ->
5 1 7 3
9 2 11 4
13 6 10 8
X 14 15 12
   ->
5 1 7 3
9 2 11 4
X 6 10 8
13 14 15 12
   ->
5 1 7 3
X 2 11 4
9 6 10 8
13 14 15 12
   ->
X 1 7 3
5 2 11 4
9 6 10 8
13 14 15 12
   ->
1 X 7 3
5 2 11 4
9 6 10 8
13 14 15 12
   ->
1 2 7 3
5 X 11 4
9 6 10 8
13 14 15 12
   ->
1 2 7 3
5 6 11 4
9 X 10 8
13 14 15 12
   ->
1 2 7 3
5 6 11 4
9 10 X 8
13 14 15 12
   ->
1 2 7 3
5 6 X 4
9 10 11 8
13 14 15 12
   ->
1 2 X 3
5 6 7 4
9 10 11 8
13 14 15 12
   ->
1 2 3 X
5 6 7 4
9 10 11 8
13 14 15 12
   ->
1 2 3 4
5 6 7 X
9 10 11 8
13 14 15 12
   ->
1 2 3 4
5 6 7 8
9 10 11 X
13 14 15 12
   ->
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 X

Waktu eksekusi : 24.05063 milidetik

Jumlah simpul yang dibangkitkan : 32
```

d. File Puzzle4.txt

```
Puzzle yang akan diselesaikan :
3 7 2 1
9 10 5 11
6 15 13 4
8 12 14 X

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

Nilai Sigma(Kurang(i))+X = 29

Persoalan tidak dapat diselesaikan

Waktu eksekusi : 2.350621 milidetik
```

```
Jumlah simpul yang
dibangkitkan : 0
```

e. File Puzzle5.txt

```
Puzzle yang akan
diselesaikan :
5 1 2 3
6 10 8 4
13 9 7 15
14 X 12 11

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

Nilai
Sigma(Kurang(i))+X
= 24

Urutan penyelesaian
puzzle :
5 1 2 3
6 10 8 4
13 9 7 15
14 X 12 11
   ->
5 1 2 3
6 10 8 4
13 9 7 15
X 14 12 11
   ->
5 1 2 3
6 10 8 4
X 9 7 15
13 14 12 11
   ->
5 1 2 3
6 10 8 4
9 X 7 15
13 14 12 11
   ->
5 1 2 3
6 X 8 4
9 10 7 15
13 14 12 11
   ->
5 1 2 3
X 6 8 4
9 10 7 15
13 14 12 11
```

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

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

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

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

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

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

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

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

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

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

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

Waktu eksekusi :
131.690439
milidetik

Jumlah simpul yang
dibangkitkan : 271
```

Setelah beberapa proses pengujian, algoritma *Branch and Bound* memang lebih cepat dari pada algoritma *Breadth First Search.* Akan tetapi apabila susunan *puzzle* yang akan diselesaikan cukup acak, *puzzle* akan diselesaikan dengan waktu yang lama. Sebagai contoh apabila puzzle dengan 7 ubin yang diacak dan tiap ubin berjarak minimal 2 ubin dari posisi aslinya, dapat diselesaikan paling cepat ±2 menit dengan simpul yang dibangkitkan sejumlah 39614 simpul, sedangkan puzzle dengan 4 ubin yang diacak dan tiap ubin berjarak minimal 2 ubin dari posisi aslinya, dapat diselesaikan paling cepat ±13 mili detik dengan simpul yang dibangkitkan sejumlah 27 simpul. Dengan itu dapat disimpulkan bahwa semakin acak susunan *puzzle*-nya, semakin lama proses penyelesaiannya menjadi sebuah *puzzle* dengan susunan final. Hal ini dikarenakan masih banyaknya cabang yang dimasukkan ke dalam *queue* merupakan cabang menuju solusi yang tidak optimal. Oleh sebab itu, harus diberi batas yang lain lagi sehingga pemrosesan *puzzle* menggunakan algoritma *Branch and Bound* ini lebih optimal dan lebih cepat pemrosesannya.

BAB 5
LIST KEBERHASILAN DALAM PENGERJAAN TUGAS KECIL 3

No.	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	√	