PENYELESAIAN 15-PUZZLE DENGAN PENDEKATAN BRANCH-AND-BOUND

Sebagai Tugas Kecil 3 IF2211 Strategi Algoritma

Disusun oleh:

Thirafi Najwan Kurniatama 13520157

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022

Checklist

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		✔

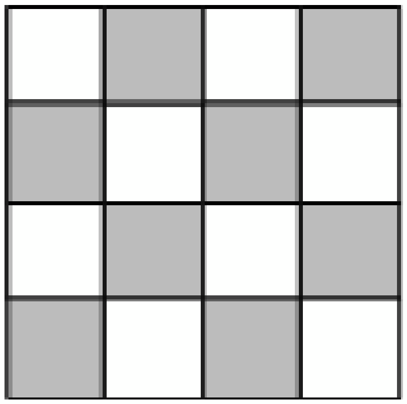
Cara Kerja Algoritma 15-Puzzle secara Branch and Bound

Algoritma branch and bound adalah algoritma DFS yang menerapkan heuristik pada pencariannya. Heuristik ini nantinya akan menghasilkan suatu nilai (biasanya berupa lower/upper bound) yang nantinya akan digunakan untuk menentukan node mana yang lebih “worth-it” untuk diteruskan pencariannya.

Pada penyelesaian 15-puzzle ini, heuristik yang digunakan sebagai cost adalah jumlah displaced tile ditambah jarak dari root (kedalaman node) serta heuristik preliminary-nya adalah dengan menggunakan total nilai dari seluruh tile untuk fungsi KURANG(i) ditambah dengan nilai variabel X.

Fungsi KURANG(i) mengembalikan banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa. Dengan catatan ubin kosong memiliki i = 16.

Nilai dari variabel X adalah 1 jika tile kosong pada root berada pada salah satu tempat yang diarsir pada gambar di bawah. jika tidak, nilainya 0.



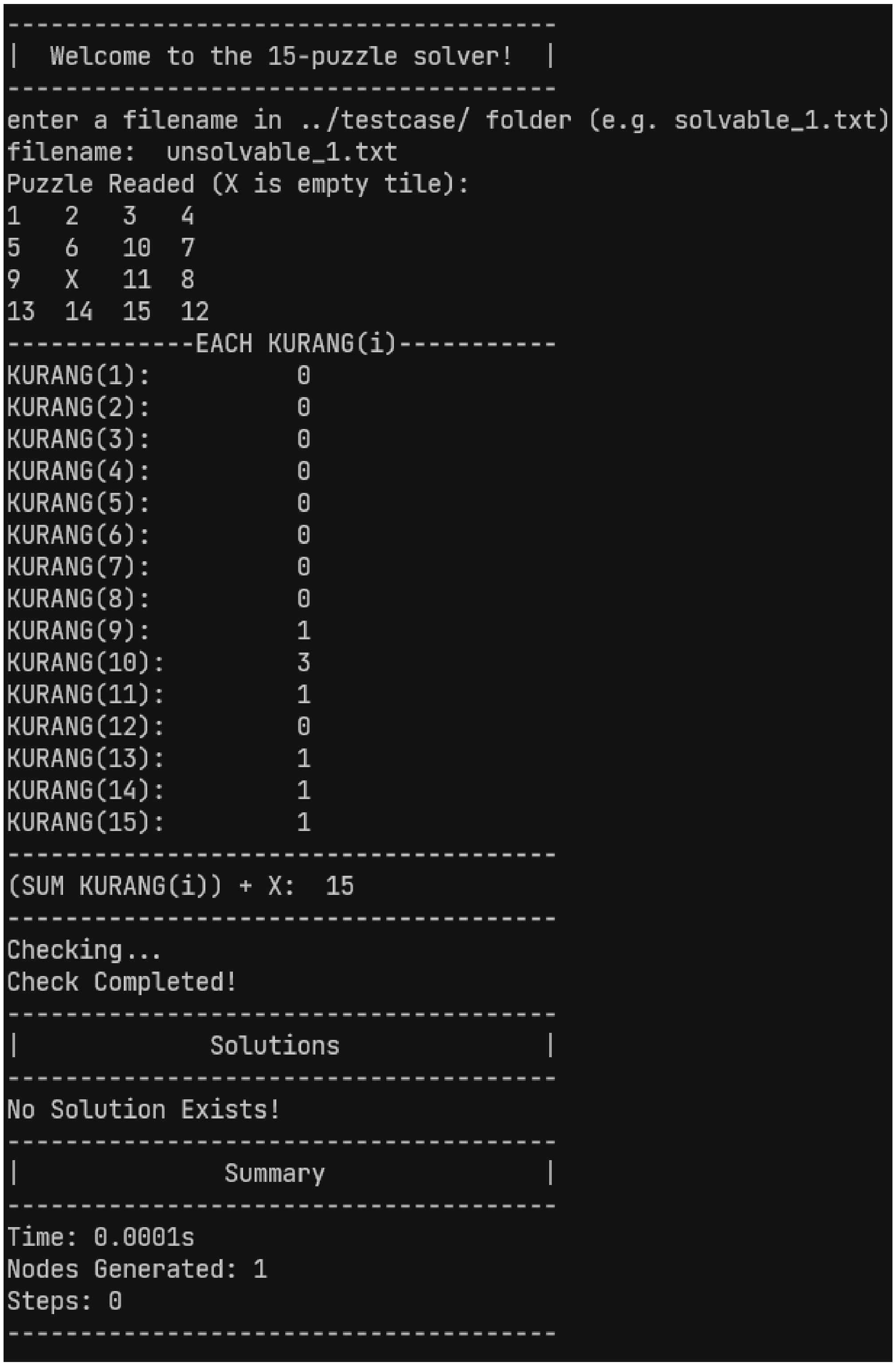
Gambar 1 Tempat tile kosong awal yang diarsir

Adapun Langkah-langkah penyelesaian algoritma ini adalah sebagai berikut:

1. Push puzzle ke min-heap berdasarkan cost
2. Pop heap
3. Jika heap empty, proses selesai.
4. Cek puzzle yang dipop memiliki nilai preliminary (KURANG(i) + X) genap. jika tidak, maka unsolvable. Jika ya, lanjutkan.
5. Jika cost - level sudah nol (displaced tile nol), maka adalah solusi, print solusi. Proses selesai.
6. Jika tidak, bangkitkan semua kemungkinan puzzle setelah ubin yang kosong digeser ke 4 arah (3 arah jika sudah masuk di atas level 0)
7. Push seluruh node di no 2 ke heap
8. Kembali ke no 2 hingga menyentuh step 3 atau 5

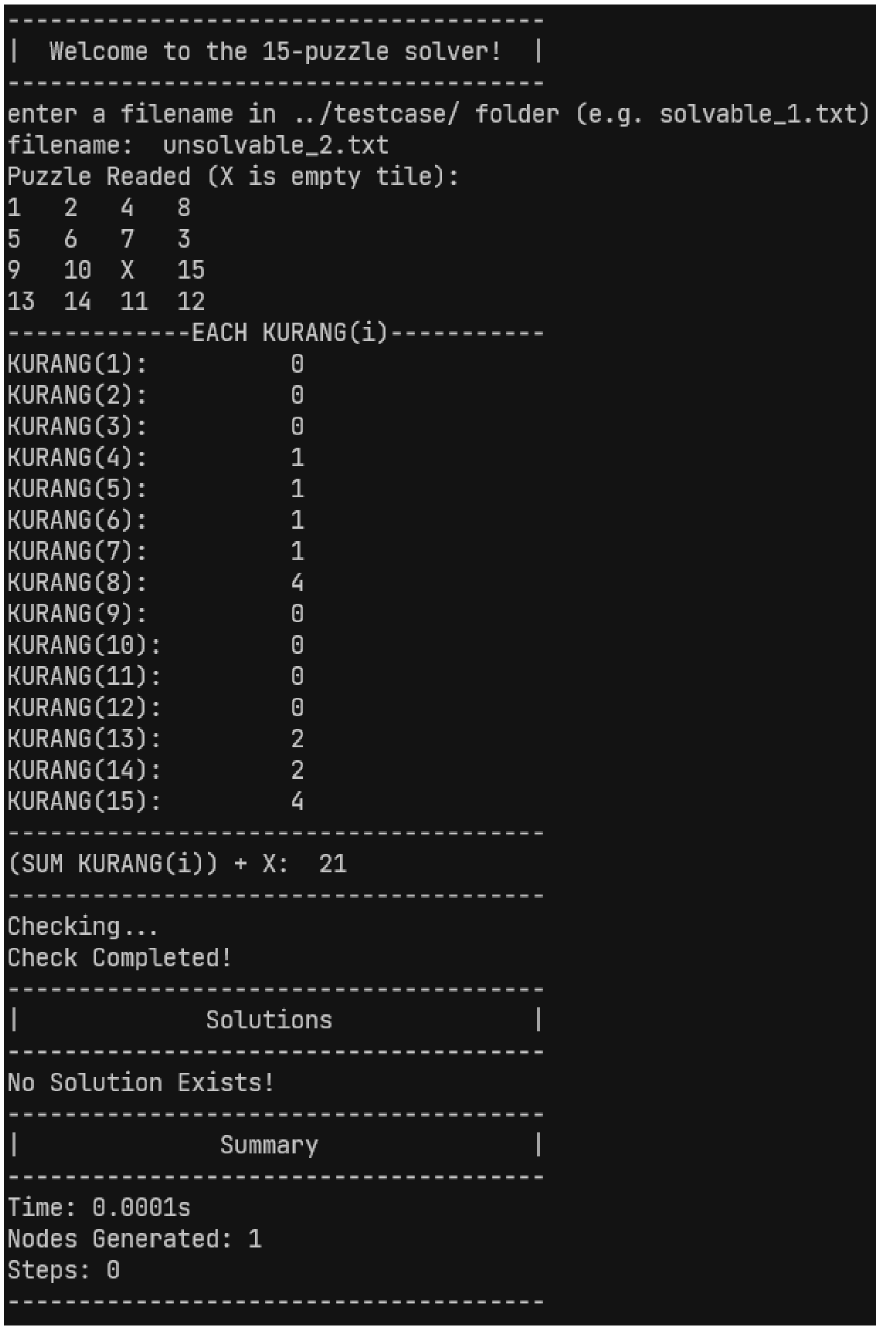
Screenshot Input-Output Program

Unsolvable Puzzle 1



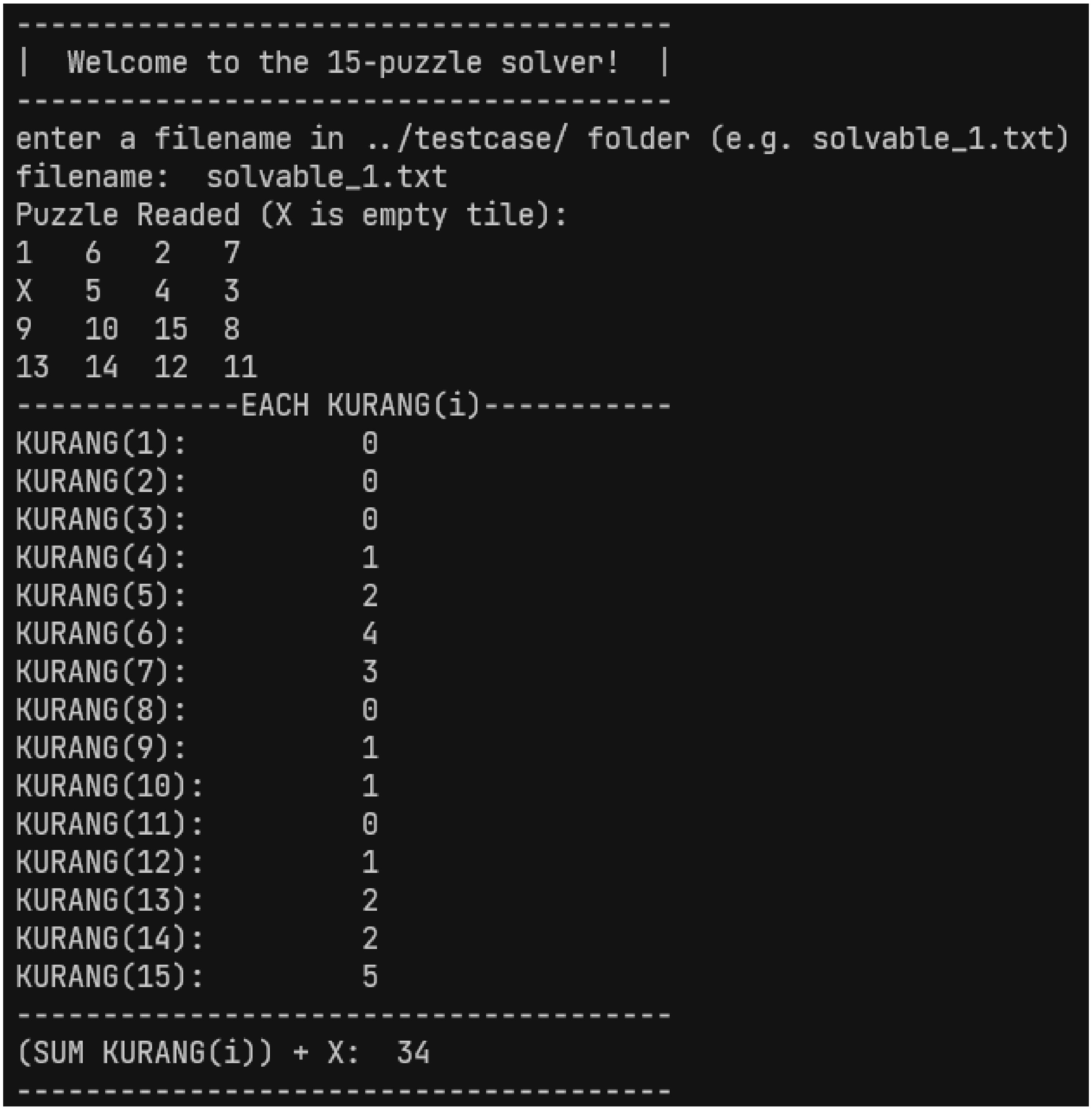
```
--------------------------------------
|  Welcome to the 15-puzzle solver!  |
--------------------------------------
enter a filename in ../testcase/ folder (e.g. solvable_1.txt)
filename:  unsolvable_1.txt
Puzzle Readed (X is empty tile):
1   2   3   4
5   6   10  7
9   X   11  8
13  14  15  12
-------------EACH KURANG(i)-----------
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):          1
KURANG(10):         3
KURANG(11):         1
KURANG(12):         0
KURANG(13):         1
KURANG(14):         1
KURANG(15):         1
--------------------------------------
(SUM KURANG(i)) + X:  15
--------------------------------------
Checking...
Check Completed!
--------------------------------------
|              Solutions             |
--------------------------------------
No Solution Exists!
--------------------------------------
|               Summary              |
--------------------------------------
Time: 0.0001s
Nodes Generated: 1
Steps: 0
--------------------------------------
```

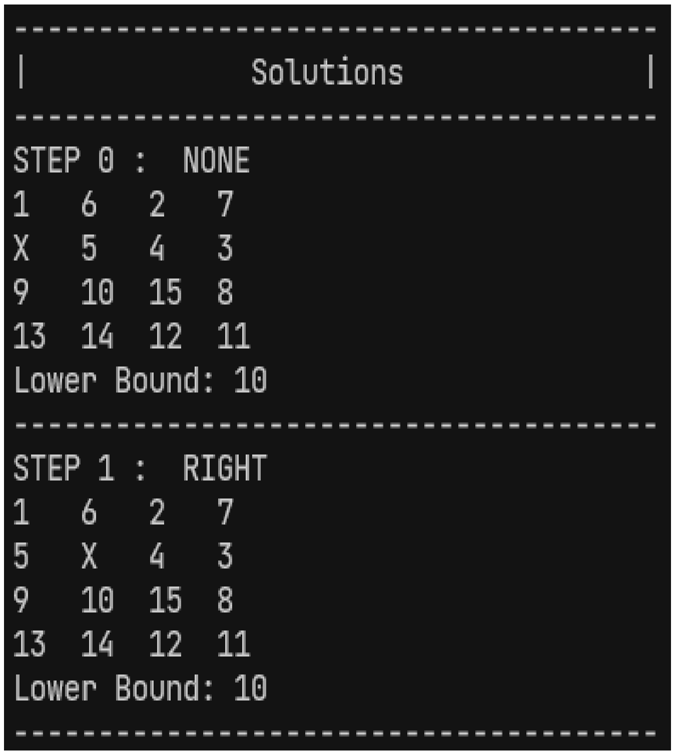
Unsolvable Puzzle 2



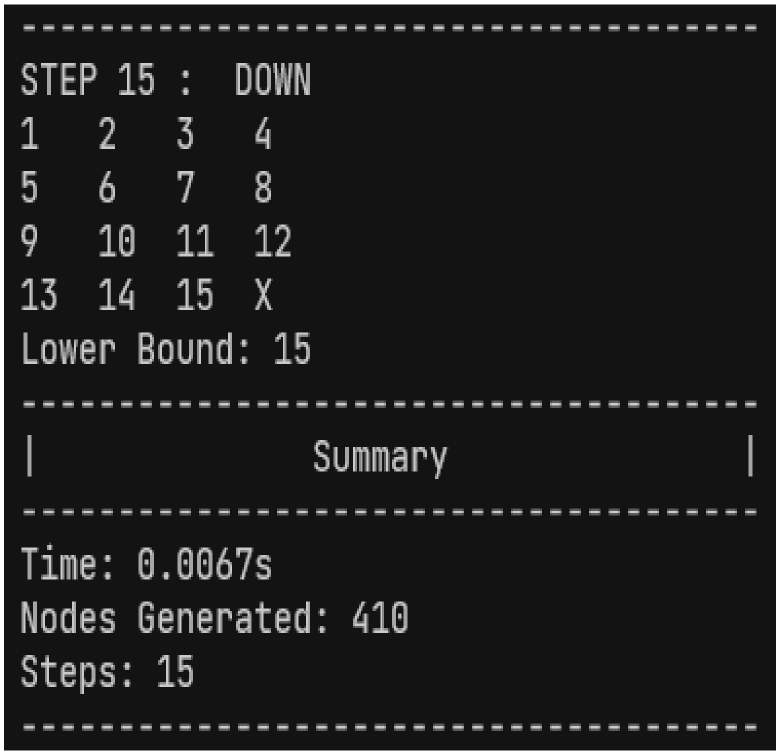
```
--------------------------------------
|  Welcome to the 15-puzzle solver!  |
--------------------------------------
enter a filename in ../testcase/ folder (e.g. solvable_1.txt)
filename:  unsolvable_2.txt
Puzzle Readed (X is empty tile):
1   2   4   8
5   6   7   3
9   10  X   15
13  14  11  12
-------------EACH KURANG(i)-----------
KURANG(1):          0
KURANG(2):          0
KURANG(3):          0
KURANG(4):          1
KURANG(5):          1
KURANG(6):          1
KURANG(7):          1
KURANG(8):          4
KURANG(9):          0
KURANG(10):         0
KURANG(11):         0
KURANG(12):         0
KURANG(13):         2
KURANG(14):         2
KURANG(15):         4
--------------------------------------
(SUM KURANG(i)) + X:  21
--------------------------------------
Checking...
Check Completed!
--------------------------------------
|              Solutions             |
--------------------------------------
No Solution Exists!
--------------------------------------
|               Summary              |
--------------------------------------
Time: 0.0001s
Nodes Generated: 1
Steps: 0
--------------------------------------
```

Solvable Puzzle 1


```
---------------------------------------
|  Welcome to the 15-puzzle solver!  |
---------------------------------------
enter a filename in ../testcase/ folder (e.g. solvable_1.txt)
filename:  solvable_1.txt
Puzzle Readed (X is empty tile):
1   6   2   7
X   5   4   3
9   10  15  8
13  14  12  11
-------------EACH KURANG(i)-----------
KURANG(1):          0
KURANG(2):          0
KURANG(3):          0
KURANG(4):          1
KURANG(5):          2
KURANG(6):          4
KURANG(7):          3
KURANG(8):          0
KURANG(9):          1
KURANG(10):         1
KURANG(11):         0
KURANG(12):         1
KURANG(13):         2
KURANG(14):         2
KURANG(15):         5
---------------------------------------
(SUM KURANG(i)) + X:  34
---------------------------------------
```

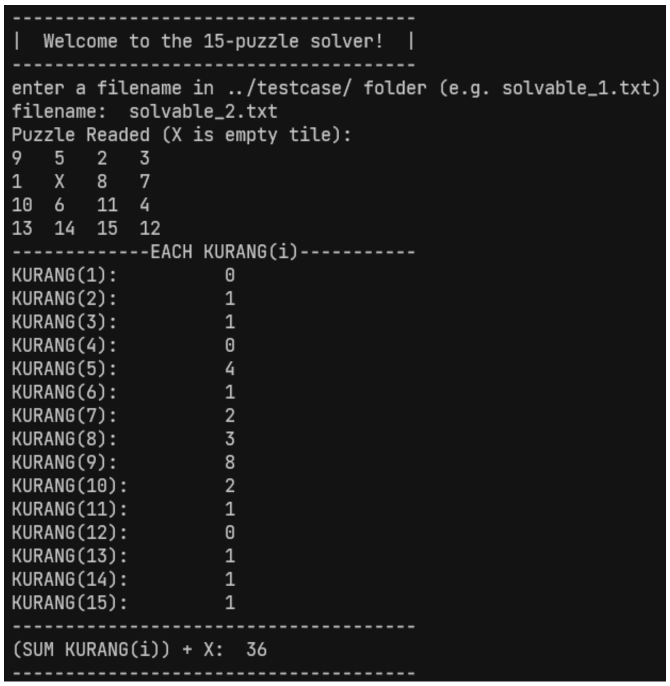


```
---------------------------------------
|              Solutions              |
---------------------------------------
STEP 0 :  NONE
1   6   2   7
X   5   4   3
9   10  15  8
13  14  12  11
Lower Bound: 10
---------------------------------------
STEP 1 :  RIGHT
1   6   2   7
5   X   4   3
9   10  15  8
13  14  12  11
Lower Bound: 10
---------------------------------------
```

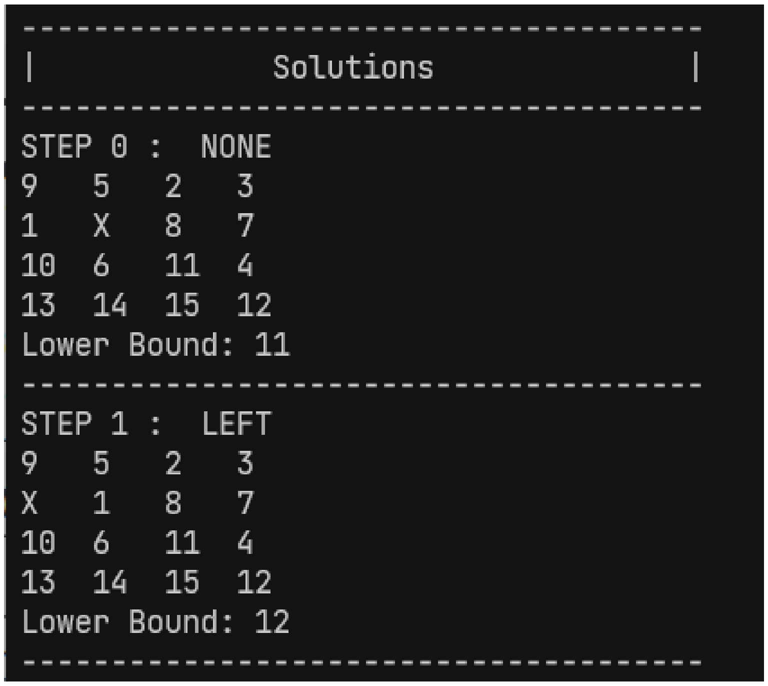


```
---------------------------------------
STEP 15 :  DOWN
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  X
Lower Bound: 15
---------------------------------------
|               Summary               |
---------------------------------------
Time: 0.0067s
Nodes Generated: 410
Steps: 15
---------------------------------------
```

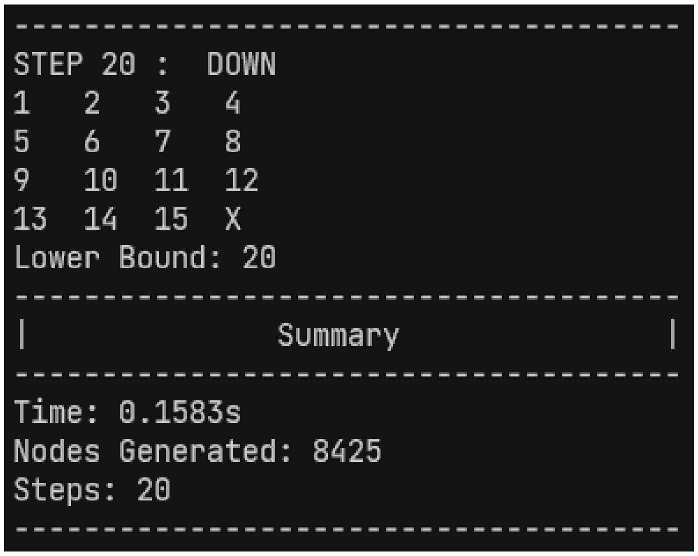
Solvable Puzzle 2



```
--------------------------------------
|  Welcome to the 15-puzzle solver!  |
--------------------------------------
enter a filename in ../testcase/ folder (e.g. solvable_1.txt)
filename:  solvable_2.txt
Puzzle Readed (X is empty tile):
9   5   2   3
1   X   8   7
10  6   11  4
13  14  15  12
-------------EACH KURANG(i)-----------
KURANG(1):          0
KURANG(2):          1
KURANG(3):          1
KURANG(4):          0
KURANG(5):          4
KURANG(6):          1
KURANG(7):          2
KURANG(8):          3
KURANG(9):          8
KURANG(10):         2
KURANG(11):         1
KURANG(12):         0
KURANG(13):         1
KURANG(14):         1
KURANG(15):         1
--------------------------------------
(SUM KURANG(i)) + X:  36
--------------------------------------
```

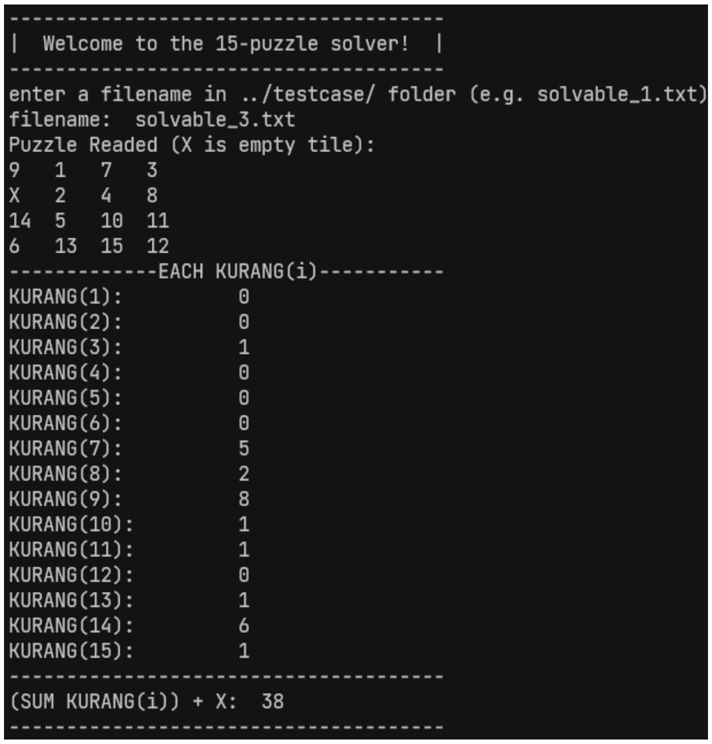


```
--------------------------------------
|              Solutions             |
--------------------------------------
STEP 0 :  NONE
9   5   2   3
1   X   8   7
10  6   11  4
13  14  15  12
Lower Bound: 11
--------------------------------------
STEP 1 :  LEFT
9   5   2   3
X   1   8   7
10  6   11  4
13  14  15  12
Lower Bound: 12
--------------------------------------
```

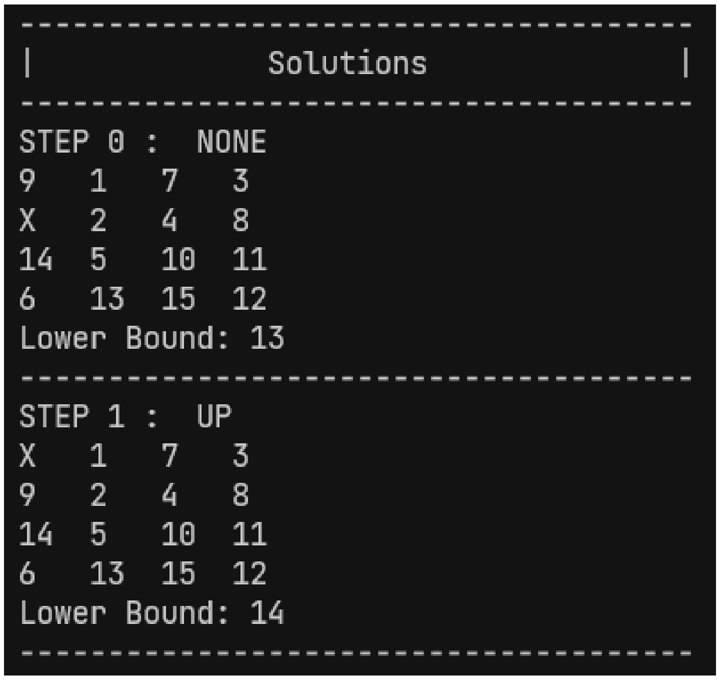


```
--------------------------------------
STEP 20 :  DOWN
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  X
Lower Bound: 20
--------------------------------------
|              Summary               |
--------------------------------------
Time: 0.1583s
Nodes Generated: 8425
Steps: 20
--------------------------------------
```

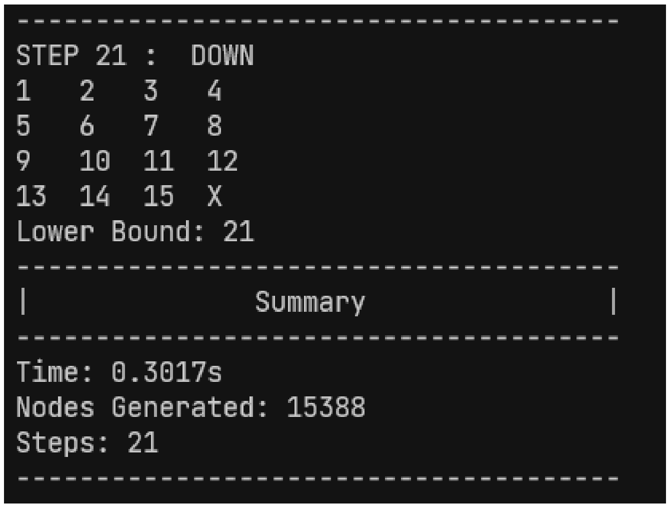
Solvable Puzzle 3



```
--------------------------------------
|  Welcome to the 15-puzzle solver!  |
--------------------------------------
enter a filename in ../testcase/ folder (e.g. solvable_1.txt)
filename:  solvable_3.txt
Puzzle Readed (X is empty tile):
9   1   7   3
X   2   4   8
14  5   10  11
6   13  15  12
-------------EACH KURANG(i)-----------
KURANG(1):          0
KURANG(2):          0
KURANG(3):          1
KURANG(4):          0
KURANG(5):          0
KURANG(6):          0
KURANG(7):          5
KURANG(8):          2
KURANG(9):          8
KURANG(10):         1
KURANG(11):         1
KURANG(12):         0
KURANG(13):         1
KURANG(14):         6
KURANG(15):         1
--------------------------------------
(SUM KURANG(i)) + X:  38
--------------------------------------
```



```
--------------------------------------
|              Solutions             |
--------------------------------------
STEP 0 :  NONE
9   1   7   3
X   2   4   8
14  5   10  11
6   13  15  12
Lower Bound: 13
--------------------------------------
STEP 1 :  UP
X   1   7   3
9   2   4   8
14  5   10  11
6   13  15  12
Lower Bound: 14
--------------------------------------
```



```
--------------------------------------
STEP 21 :  DOWN
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  X
Lower Bound: 21
--------------------------------------
|              Summary               |
--------------------------------------
Time: 0.3017s
Nodes Generated: 15388
Steps: 21
--------------------------------------
```

Source Code Program

```
import os
import time
import heapq as hq




# node datatype
class CustomNode():
    def __init__(self, puz, weight, prevdir, level):
        self.puz = puz
        self.weight = weight
        self.prevdir = prevdir
        self.level = level
    def copy(self):
        return CustomNode(self.puz, self.weight, self.prevdir, self.level)
# wrapper untuk heap node
class CustomHeap(object):
    def __init__(self, initial=None, key=lambda x: x):
        self.key = key
        self.idx = 0
        if initial:
            self._data = [(key(item), i, item) for i, item in
enumerate(initial)]
            self.idx = len(initial)
        else:
            self._data = []
        hq.heapify(self._data)

    def push(self, item):
        hq.heappush(self._data, (self.key(item), self.idx, item))
        self.idx += 1

    def pop(self):
        return hq.heappop(self._data)[-1]

    def isEmpty(self):
        return len(self._data) == 0

    def elimBigger(self, cur): # eliminate with bigger key
        newList = [(k, idx, item) for k, idx, item in list(self._data) if
k <= self.key(cur)]
        self._data = newList
        hq.heapify(self._data)


# reading file
def readfileconfig(fname):
    try:
        a = [False for i in range(16)]
        cpath = os.path.dirname(__file__)
        pdir = os.path.join(cpath, "testcase", fname)
        f = open(pdir, "r")
        puz = f.read().split("\n")
        for i in range(len(puz)):
            puz[i] = puz[i].split(" ")
            for strn in puz[i]:
                if (strn != "X"):
                    if (int(strn) > 15 or int(strn) < 1):
                        raise ValueError
                    elif (a[int(strn)]):
                        raise ValueError
                    else:
                        a[int(strn)] = True
        return puz
    except:
        print("Something went wrong when reading the file, check your file
again!")
        exit()

# check if position of x is in colored
def validx(pos):
    if pos == 1 or pos == 3 or pos == 4 or pos == 6 or pos == 9 or pos ==
11 or pos == 12 or pos == 14:
        return True
    else:
        return False

# print each kurang(i)
def printeachk(puz):
    pos = [None for i in range(16)] # 0 = X  1 - 15 ubin
    for i in range(4):
        for j in range(4):
            if puz[i][j] != "X":
                pos[int(puz[i][j])] = i*4 + j
            else:
```

```python
                pos[0] = i*4 + j
    for i in range(1, 16):
        countess = 0
        for j in range(1, i):
            if pos[j] > pos[i]:
                countess += 1
        print(f"KURANG({i}): ".ljust(20), end="")
        print(str(countess).ljust(20), end="")
        print()

# kurang(i) + x
def reachable(puz):
    countess = 0
    pos = [None for i in range(16)] # 0 = X  1 - 15 ubin
    for i in range(4):
        for j in range(4):
            if puz[i][j] != "X":
                pos[int(puz[i][j])] = i*4 + j
            else:
                pos[0] = i*4 + j
    for i in range(1, 16):
        for j in range(1, i):
            if pos[j] > pos[i]:
                countess += 1

    for j in range(1, 16):
        if pos[j] > pos[0]:
            countess += 1
    if validx(pos[0]):
        countess += 1
    return countess

# for weighing
def displaced(puz):
    count = 0
    for i in range(4):
        for j in range(4):
            if puz[i][j] != "X":
                if int(puz[i][j]) != i*4 + j+1:
                    count += 1
    return count


# check if r c valid
def check(r, c):
    if (r < 0 or r > 3 or c < 0 or c > 3):
        return False
    return True

# swapping, assumes valid move
def swapblock(puz, r, c, ra, ca):
    puz[r][c], puz[ra][ca] = puz[ra][ca], puz[r][c]


#printing a puzzle
def printpuz(puz):
    for r in puz:
        for c in r:
            print(c.ljust(4), end="")
        print()

# forbid swap for opposing dir
def forbid(dirint):
    if dirint == 0:
        return 1
    elif dirint == 1:
        return 0
    elif dirint == 2:
        return 3
    elif dirint == 3:
        return 2
    else:
        return -1

# translate dircode to string
def stringdir(dirc):
    if dirc == 0:
        return "DOWN"
    elif dirc == 1:
        return "UP"
    elif dirc == 2:
        return "RIGHT"
    elif dirc == 3:
        return "LEFT"
    else:
        return "NONE"

# Solving a Puzzle
def solve(puz):
    print("Checking...")
    start = time.perf_counter()
    drc = [[1, 0], [-1, 0], [0, 1], [0, -1]]
    q = CustomHeap(None, lambda x: x.weight) # Create Queue
    prevdir = [-1]
    nodegenerated = 1
    # CHECKING
    acsols = None
    q.push(CustomNode(puz, displaced(puz), prevdir, 0)) # push initial
node
    while not q.isEmpty():
        curNode = q.pop()
        curPuz = curNode.puz
```

```python
        prevdir = curNode.prevdir
        if curNode.weight - curNode.level == 0:
            q.elimBigger(curNode) # eliminate with bigger weight
            if acsols == None: # if first solution
                acsols = curNode
            elif len(acsols.prevdir) > len(curNode.prevdir): # if new solution is shorter
                acsols = curNode
        elif reachable(curPuz) % 2 == 0:
            lenr = len(curNode.puz)
            lenc = len(curNode.puz[0])
            rx = -1
            cx = -1
            for i in range(lenr):
                for j in range(lenc):
                    if curPuz[i][j] == "X":
                        rx = i
                        cx = j
            for i in range(len(drc)):
                rnx = rx + drc[i][0]
                cnx = cx + drc[i][1]
                if check(rnx, cnx) and prevdir[-1] != forbid(i):
                    temppuz = [x[:] for x in curPuz]
                    swapblock(temppuz, rx, cx, rnx, cnx)
                    q.push(CustomNode(temppuz, displaced(temppuz) + curNode.level+1, prevdir + [i], curNode.level+1)) # push new node
                    nodegenerated += 1
    end = time.perf_counter()
    print("Check Completed!")
    printingresult(acsols, puz, [end - start, nodegenerated])

# printing solution
def printingresult(acsols, afpuz, info):
    fpuz = [x[:] for x in afpuz]
    drc = [[1, 0], [-1, 0], [0, 1], [0, -1]]
    print("--------------------------------------")
    print("|             Solutions              |")
    print("--------------------------------------")
    if (acsols == None):
        print("No Solution Exists!")
        print("--------------------------------------")
        print("|              Summary               |")
        print("--------------------------------------")
        print(f"Time: {info[0]:0.4f}s")
        print(f"Nodes Generated: {info[1]}")
        print(f"Steps: 0")
        print("--------------------------------------")
    else:
        rx = -1
        cx = -1
        for i in range(len(fpuz)):
            for j in range(len(fpuz[0])):
                if fpuz[i][j] == "X":
                    rx = i
                    cx = j
        for i in range(len(acsols.prevdir)):
            print("STEP", i, ": ", stringdir(acsols.prevdir[i]))
            if (acsols.prevdir[i] >= 0):
                swapblock(fpuz, rx, cx, rx + drc[acsols.prevdir[i]][0], cx
+ drc[acsols.prevdir[i]][1])
                rx += drc[acsols.prevdir[i]][0]
                cx += drc[acsols.prevdir[i]][1]
            printpuz(fpuz)
            print("Lower Bound:", displaced(fpuz) + i)
            print("--------------------------------------")
        print("|              Summary               |")
        print("--------------------------------------")
        print(f"Time: {info[0]:0.4f}s")
        print(f"Nodes Generated: {info[1]}")
        print(f"Steps: {len(acsols.prevdir) - 1}")
        print("--------------------------------------")

# main
def main():
    fname = "" # nama file konfigurasi
    print("--------------------------------------")
    print("|  Welcome to the 15-puzzle solver!  |")
    print("--------------------------------------")
    print("enter a filename in ../testcase/ folder (e.g. solvable_1.txt)")
    print("filename: ", end=" ")
    fname = input()
    puz = readfileconfig(fname)
    print("Puzzle Readed (X is empty tile):")
    printpuz(puz)
    print("-------------EACH KURANG(i)-----------")
    printeachk(puz)
    print("--------------------------------------")
    print("(SUM KURANG(i)) + X: ", reachable(puz))
    print("--------------------------------------")
    solve(puz)
if __name__ == "__main__":
    main()
```

Link Kode Program Beserta Testcase

https://github.com/reverseon/15-puzzle-bnb