Laporan Tugas Kecil 3 Strategi Algoritma IF2211

Zayd Muhammad Kawakibi Zuhri

13520144

1. Algoritma *Branch and Bound* untuk 15-Puzzle

Algoritma utama program ini berada dalam fungsi solve(puzzle), yang menerima list dua dimensi dengan suatu state puzzle, berisi angka 1-16 (16 adalah tile kosong). Pertama-tama, kita deklarasi dua list, satu untuk solution tree yang akan diisi semua simpul yang dikembangkan, dan satu lagi adalah queue yang akan diisi simpul yang masih hidup. Suatu simpul direpresentasikan oleh suatu array berisi 6 elemen: state puzzle, nilai bound hasil penjumlahan taksiran cost dan depth, indeks simpul parent, kedalaman simpul, indeks simpul, dan step yang dilakukan saat simpul dikembangkan. Semua elemen ini dibutuhkan untuk algoritma dan juga pengembangan path solusi puzzle. Simpul akar diinisiasi dengan state puzzle yang didapatkan dari parameter fungsi, serta 5 elemen lainnya yang tidak terlalu relevan untuk simpul akar, sehingga diisi dengan 0. Simpul akar ini ditambahkan ke dalam list tree dan juga list queue. Setelah ini, kita dapat masuk ke dalam loop utama algoritma branch and bound.

Loop dimulai dengan mendapatkan simpul termurah dari queue simpul hidup menggunakan get_cheapest_node_index(queue). Hal ini dilakukan dengan fungsi pop, sehingga simpul tersebut akan dihapus, alias dibunuh, dari queue. Simpul termurah ini lalu dicek menggunakan is_solved() yang mengembalikan True jika state puzzle sudah solved. Tentunya, jika simpul tersebut sudah mengandung puzzle yang sudah selesai, fungsi akan mengembalikan list tree pencarian dan simpul yang didapatkan tadi, dan algoritma pun berhenti. Jika simpul termurah tersebut belum menyelesaikan puzzle, algoritma akan lanjut ke pembangkitan simpul-simpul anaknya yang akan dimasukkan ke dalam list tree dan juga queue simpul hidup.

Pembangkitan simpul dilakukan dengan fungsi get_possible_nodes() dengan beberapa parameter masukan yang akan digunakan. Fungsi ini terlebih dahulu mencari posisi kotak kosong pada puzzle menggunakan get_blank_tile(). Lalu, tergantung pada letak kotak kosong tersebut, simpul dikembangkan berdasarkan 4 gerakan puzzle yang mungkin dilakukan, yaitu menukarkan kotak kosong dengan kotak di atas, bawah, kanan, atau kiri. Setiap simpul yang dibangkitkan juga disertakan elemen-elemen lain yang dibutuhkan simpul, di antara nya adalah cost simpul tersebut. Cost ini merupakan hasil penjumlahan depth simpul dan hasil dari fungsi estimate_cost(). Fungsi ini menerima suatu state puzzle lalu menghitung banyaknya ubin tidak kosong yang tidak berada pada tempat yang benar. Nilai ini merupakan sebagai taksiran seberapa jauh suatu state berada dari goal state atau puzzle yang solved. Penjumlahannya dengan kedalaman simpul digunaka sebagai komponen Bound dalam algoritma ini.

2. *Source Code* (Dalam bahasa Python)

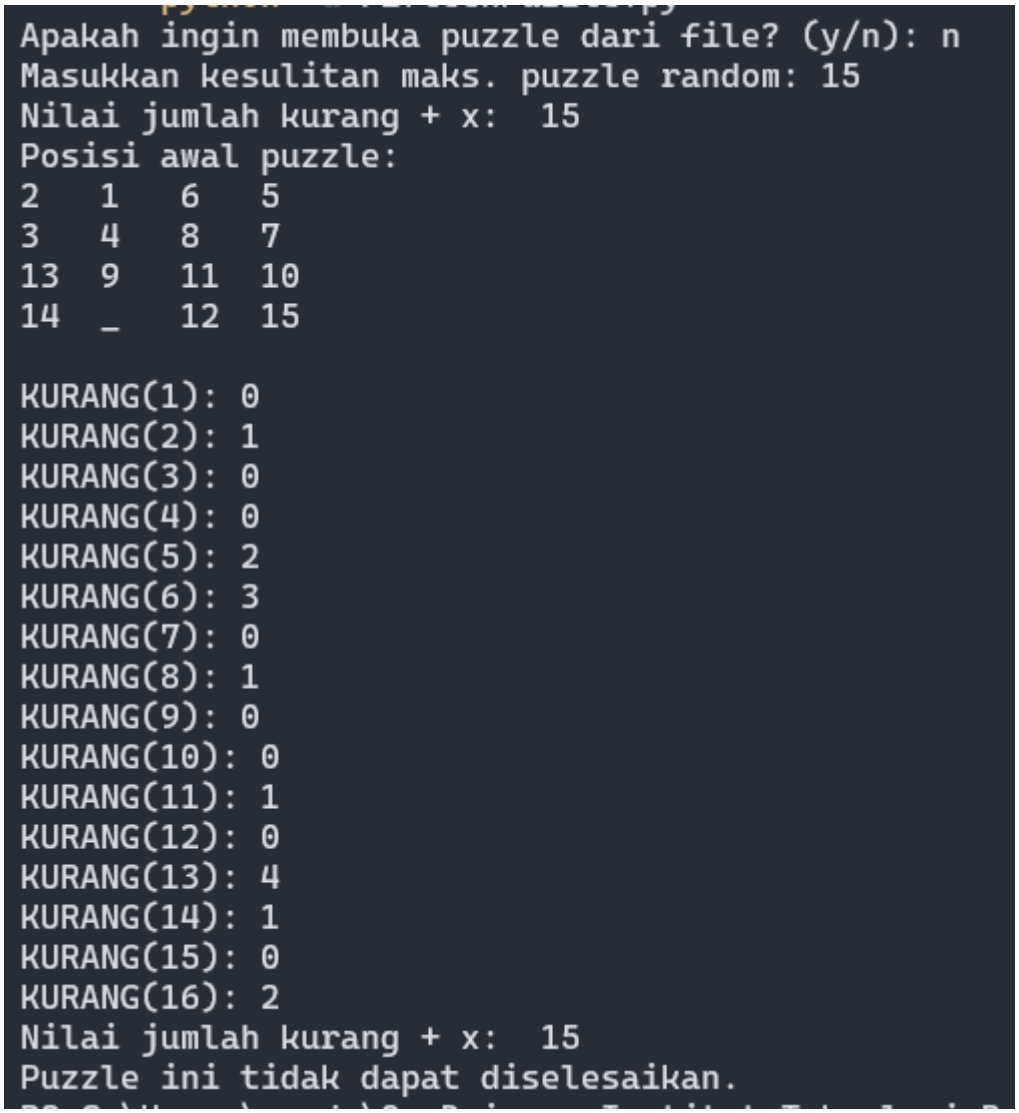
```python
import random
import copy
import time

# Mengembalikan puzzle berupa list 2 dimensi dari file di path
def read_puzzle_from_file(path):
    with open(path) as f:
        return [[int(x) for x in line.split()] for line in f]

# Mengembalikan puzzle berupa list 2 dimensi secara random
def generate_random_puzzle():
    nums = [i for i in range(1, 17)]
    return [[nums.pop(random.randint(0, len(nums) - 1)) for _ in range(4)] for _ in range(4)]

# Print matrix puzzle
def print_puzzle(puzzle):
    for row in puzzle:
        for tile in row:
            if (tile == 16):
                print("_   ", end="")
            else:
                print(str(tile) + ("   " if tile < 10 else "  "), end="")
        print()

# Print solusi dari tree dan node solusi puzzle
def print_path(tree, solved):
    path = []
    while (solved[2] != -1):
        path.append(solved)
        solved = tree[solved[2]]
    path.append(tree[0])
    path.reverse()
    for state in path:
        print("Langkah ke-" + str(state[3]) + ": " + state[5])
        print_puzzle(state[0])
        print()

# Mengembalikan nilai fungsi KURANG(i) dari puzzle
def kurang(i, puzzle):
    count = 0
    found = False
    for row in range(0, 4):
        for col in range(0, 4):
            if (not found):
                found = puzzle[row][col] == i
            else:
                if (puzzle[row][col] < i):
                    count += 1
    return count

# Mengembalikan letak kotak kosong pada puzzle
def get_blank_tile(puzzle):
    blank_row = 0
    blank_col = 0
    for row in range(0, 4):
        for col in range(0, 4):
            if (puzzle[row][col] == 16):
                blank_row = row
                blank_col = col
    return blank_row, blank_col

```

```python
# Mengembalikan apakah puzzle bisa di-solve atau tidak, juga nilai jumlah KURANG + X
def is_solvable(puzzle):
    total = 0
    for i in range(1, 17):
        total += kurang(i, puzzle)

    blank_row, blank_col = get_blank_tile(puzzle)
    x = 0 if ((blank_row + blank_col) % 2 == 0) else 1

    print("Nilai jumlah kurang + x: ", total + x, end="")
    return (total + x) % 2 == 0, total + x

# Mengembalikan apakah puzzle sudah di-solve atau belum
def is_solved(puzzle):
    return puzzle == [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]

# Mengembalikan estimasi cost dari state puzzle sekarang ke solusi
def estimate_cost(puzzle):
    cost = 0
    supposed = 1
    for row in range(0, 4):
        for col in range(0, 4):
            if (puzzle[row][col] != supposed):
                cost += 1
            supposed += 1
    return cost

# Mengembalikan index node dengan nilai cost terkecil
def get_cheapest_node_index(queue):
    costs = [q[1] for q in queue]
    return costs.index(min(costs))

# Mengembalikan list dari semua node yang dapat di-capai dari dari state puzzle sekarang
def get_possible_nodes(node, tree, parent_idx, index_count):
    nodes = []
    blank_row, blank_col = get_blank_tile(node[0])
    depth = node[3] + 1
    if (blank_row > 0):
        up = copy.deepcopy(node[0])
        up[blank_row][blank_col] = up[blank_row - 1][blank_col]
        up[blank_row - 1][blank_col] = 16
        nodes.append([up, depth + estimate_cost(up), parent_idx, depth, index_count[0], "UP"])
        index_count[0] += 1
    if (blank_row < 3):
        down = copy.deepcopy(node[0])
        down[blank_row][blank_col] = down[blank_row + 1][blank_col]
        down[blank_row + 1][blank_col] = 16
        nodes.append([down, depth + estimate_cost(down), parent_idx, depth, index_count[0], "DOWN"])
        index_count[0] += 1
    if (blank_col > 0):
        left = copy.deepcopy(node[0])
        left[blank_row][blank_col] = left[blank_row][blank_col - 1]
        left[blank_row][blank_col - 1] = 16
        nodes.append([left, depth + estimate_cost(left), parent_idx, depth, index_count[0], "LEFT"])
        index_count[0] += 1
    if (blank_col < 3):
        right = copy.deepcopy(node[0])
        right[blank_row][blank_col] = right[blank_row][blank_col + 1]
        right[blank_row][blank_col + 1] = 16
        nodes.append([right, depth + estimate_cost(right), parent_idx, depth, index_count[0], "RIGHT"])
        index_count[0] += 1
    return nodes
```

```python
# Solve 15-Puzzle dengan Branch and Bound
def solve(puzzle):
    tree = []
    queue = []
    root = [puzzle, 0, -1, 0, 0, "START"] # Node = [Puzzle, Cost, Parent Node Index, Depth, Index, Step]
    tree.append(root)
    queue.append(root)
    index_count = [1]
    visited = 1
    while (len(queue) > 0):
        now = queue.pop(get_cheapest_node_index(queue))
        print("\r",end='')
        print("Nodes visited: " + str(visited)
            + " Now cost: " + str(now[1] - now[3])
            + " Depth: " + str(now[3]), end="   ")
        visited += 1
        if (is_solved(now[0])):
            return tree, now
        else:
            parent_idx = now[4]
            possible_nodes = get_possible_nodes(now, tree, parent_idx, index_count)
            for node in possible_nodes:
                tree.append(node)
                queue.append(node)
    return NULL, NULL

# Program utama
def main():
    print("Apakah ingin membuka puzzle dari file? (y/n): ", end="")
    choice = input()

    if (choice == "y"):
        print("Masukkan nama file: ", end="")
        filename = input()
        puzzle = read_puzzle_from_file(filename)
    else:
        print("Masukkan kesulitan maks. puzzle random: ", end="")
        difficulty = int(input())
        puzzle = generate_random_puzzle()
        while (is_solvable(puzzle)[1] > difficulty):
            print("\r",end='')
            puzzle = generate_random_puzzle()

    print("\nPosisi awal puzzle:")
    print_puzzle(puzzle)
    print()

    for i in range(1, 17):
        print("KURANG("+str(i)+"):", kurang(i, puzzle))

    if (not is_solvable(puzzle)[0]):
        print("\nPuzzle ini tidak dapat diselesaikan.")
    else:
        print()
        start = time.time()
        tree, solved = solve(puzzle)
        end = time.time()

        print("\n\nAlur solusi puzzle:\n")
        print_path(tree, solved)
        print("Waktu eksekusi:", round(end - start, 6), end="s\n")
        print("Jumlah simpul yang dibangkitkan:", len(tree))

if __name__ == "__main__":
    main()
```

3. Contoh-Contoh Penyelesaian 15-Puzzle

a. Random puzzle yang menghasilkan unsolvable puzzle:



b. Puzzle unsolvable yang dibaca dari file:

```
Apakah ingin membuka puzzle dari file? (y/n): y
Masukkan nama file: unsolvable1.txt

Posisi awal puzzle:
2   1   3   5
7   4   10  8
9   6   11  12
13  14  _   15

KURANG(1): 0
KURANG(2): 1
KURANG(3): 0
KURANG(4): 0
KURANG(5): 1
KURANG(6): 0
KURANG(7): 2
KURANG(8): 1
KURANG(9): 1
KURANG(10): 3
KURANG(11): 0
KURANG(12): 0
KURANG(13): 0
KURANG(14): 0
KURANG(15): 0
KURANG(16): 1
Nilai jumlah kurang + x:  11
Puzzle ini tidak dapat diselesaikan.
```

c. Puzzle dengan 12 langkah penyelesaian

```
Apakah ingin membuka puzzle dari file? (y/n): y
Masukkan nama file: solvable12.txt

Posisi awal puzzle:
1   2   3   4
5   14  6   _
9   7   12  8
13  11  10  15

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): 2
KURANG(10): 0
KURANG(11): 1
KURANG(12): 3
KURANG(13): 2
KURANG(14): 8
KURANG(15): 0
KURANG(16): 8
Nilai jumlah kurang + x:  24
Nodes visited: 415 Now cost: 0 Depth: 12
```

```
Alur solusi puzzle:

Langkah ke-0: START
1   2   3   4
5   14  6   _
9   7   12  8
13  11  10  15

Langkah ke-1: DOWN
1   2   3   4
5   14  6   8
9   7   12  _
13  11  10  15

Langkah ke-2: LEFT
1   2   3   4
5   14  6   8
9   7   _   12
13  11  10  15

Langkah ke-3: LEFT
1   2   3   4
5   14  6   8
9   _   7   12
13  11  10  15

Langkah ke-4: UP
1   2   3   4
5   _   6   8
9   14  7   12
13  11  10  15

Langkah ke-5: RIGHT
1   2   3   4
5   6   _   8
9   14  7   12
13  11  10  15

Langkah ke-6: DOWN
1   2   3   4
5   6   7   8
9   14  _   12
13  11  10  15
```

```
Langkah ke-7: DOWN
1   2   3   4
5   6   7   8
9   14  10  12
13  11  _   15

Langkah ke-8: LEFT
1   2   3   4
5   6   7   8
9   14  10  12
13  _   11  15

Langkah ke-9: UP
1   2   3   4
5   6   7   8
9   _   10  12
13  14  11  15

Langkah ke-10: RIGHT
1   2   3   4
5   6   7   8
9   10  _   12
13  14  11  15

Langkah ke-11: DOWN
1   2   3   4
5   6   7   8
9   10  11  12
13  14  _   15

Langkah ke-12: RIGHT
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  _

Waktu eksekusi: 0.083004s
Jumlah simpul yang dibangkitkan: 1383
```

d. Puzzle dengan 18 langkah penyelesaian

```
Apakah ingin membuka puzzle dari file? (y/n): y
Masukkan nama file: solvable18.txt

Posisi awal puzzle:
1   2   4   7
10  _   9   3
5   12  6   8
13  14  11  15

KURANG(1): 0
KURANG(2): 0
KURANG(3): 0
KURANG(4): 1
KURANG(5): 0
KURANG(6): 0
KURANG(7): 3
KURANG(8): 0
KURANG(9): 4
KURANG(10): 5
KURANG(11): 0
KURANG(12): 3
KURANG(13): 1
KURANG(14): 1
KURANG(15): 0
KURANG(16): 10
Nilai jumlah kurang + x:  28
Nodes visited: 12737 Now cost: 0 Depth: 18
```

```
Alur solusi puzzle:

Langkah ke-0: START
1   2   4   7
10  _   9   3
5   12  6   8
13  14  11  15

Langkah ke-1: RIGHT
1   2   4   7
10  9   _   3
5   12  6   8
13  14  11  15

Langkah ke-2: DOWN
1   2   4   7
10  9   6   3
5   12  _   8
13  14  11  15

Langkah ke-3: LEFT
1   2   4   7
10  9   6   3
5   _   12  8
13  14  11  15

Langkah ke-4: UP
1   2   4   7
10  _   6   3
5   9   12  8
13  14  11  15

Langkah ke-5: LEFT
1   2   4   7
_   10  6   3
5   9   12  8
13  14  11  15

Langkah ke-6: DOWN
1   2   4   7
5   10  6   3
_   9   12  8
13  14  11  15

Langkah ke-7: RIGHT
1   2   4   7
5   10  6   3
9   _   12  8
13  14  11  15

Langkah ke-8: UP
1   2   4   7
5   _   6   3
9   10  12  8
13  14  11  15

Langkah ke-9: RIGHT
1   2   4   7
5   6   _   3
9   10  12  8
13  14  11  15

Langkah ke-10: RIGHT
1   2   4   7
5   6   3   _
9   10  12  8
13  14  11  15

Langkah ke-11: UP
1   2   4   _
5   6   3   7
9   10  12  8
13  14  11  15

Langkah ke-12: LEFT
1   2   _   4
5   6   3   7
9   10  12  8
13  14  11  15

Langkah ke-13: DOWN
1   2   3   4
5   6   _   7
9   10  12  8
13  14  11  15

Langkah ke-14: RIGHT
1   2   3   4
5   6   7   _
9   10  12  8
13  14  11  15

Langkah ke-15: DOWN
1   2   3   4
5   6   7   8
9   10  12  _
13  14  11  15

Langkah ke-16: LEFT
1   2   3   4
5   6   7   8
9   10  _   12
13  14  11  15

Langkah ke-17: DOWN
1   2   3   4
5   6   7   8
9   10  11  12
13  14  _   15

Langkah ke-18: RIGHT
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  _

Waktu eksekusi: 16.063903s
Jumlah simpul yang dibangkitkan: 38974
```

e. Puzzle dengan 20 langkah penyelesaian

```
Apakah ingin membuka puzzle dari file? (y/n): y
Masukkan nama file: solvable20.txt

Posisi awal puzzle:
6   7   5   3
1   _   2   4
9   15  11  8
13  10  14  12

KURANG(1): 0
KURANG(2): 0
KURANG(3): 2
KURANG(4): 0
KURANG(5): 4
KURANG(6): 5
KURANG(7): 5
KURANG(8): 0
KURANG(9): 1
KURANG(10): 0
KURANG(11): 2
KURANG(12): 0
KURANG(13): 2
KURANG(14): 1
KURANG(15): 6
KURANG(16): 10
Nilai jumlah kurang + x:  38
Nodes visited: 54853 Now cost: 0 Depth: 20
```

```
Alur solusi puzzle:

Langkah ke-0: START
6   7   5   3
1   _   2   4
9   15  11  8
13  10  14  12

Langkah ke-1: UP
6   _   5   3
1   7   2   4
9   15  11  8
13  10  14  12

Langkah ke-2: RIGHT
6   5   _   3
1   7   2   4
9   15  11  8
13  10  14  12

Langkah ke-3: DOWN
6   5   2   3
1   7   _   4
9   15  11  8
13  10  14  12

Langkah ke-4: LEFT
6   5   2   3
1   _   7   4
9   15  11  8
13  10  14  12

Langkah ke-5: UP
6   _   2   3
1   5   7   4
9   15  11  8
13  10  14  12

Langkah ke-6: LEFT
_   6   2   3
1   5   7   4
9   15  11  8
13  10  14  12

Langkah ke-7: DOWN
1   6   2   3
_   5   7   4
9   15  11  8
13  10  14  12

Langkah ke-8: RIGHT
1   6   2   3
5   _   7   4
9   15  11  8
13  10  14  12

Langkah ke-9: UP
1   _   2   3
5   6   7   4
9   15  11  8
13  10  14  12

Langkah ke-10: RIGHT
1   2   _   3
5   6   7   4
9   15  11  8
13  10  14  12

Langkah ke-11: RIGHT
1   2   3   _
5   6   7   4
9   15  11  8
13  10  14  12

Langkah ke-12: DOWN
1   2   3   4
5   6   7   _
9   15  11  8
13  10  14  12

Langkah ke-13: DOWN
1   2   3   4
5   6   7   8
9   15  11  _
13  10  14  12

Langkah ke-14: LEFT
1   2   3   4
5   6   7   8
9   15  _   11
13  10  14  12

Langkah ke-15: LEFT
1   2   3   4
5   6   7   8
9   _   15  11
13  10  14  12

Langkah ke-16: DOWN
1   2   3   4
5   6   7   8
9   10  15  11
13  _   14  12

Langkah ke-17: RIGHT
1   2   3   4
5   6   7   8
9   10  15  11
13  14  _   12

Langkah ke-18: UP
1   2   3   4
5   6   7   8
9   10  _   11
13  14  15  12

Langkah ke-19: RIGHT
1   2   3   4
5   6   7   8
9   10  11  _
13  14  15  12

Langkah ke-20: DOWN
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  _

Waktu eksekusi: 268.996017s
Jumlah simpul yang dibangkitkan: 159906
```

4. *Link GitHub Repository Source Code*

https://github.com/zaydzuhri/Tucil3_13520144

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		✓