Laporan Tugas Kecil 3 Strategi Algoritma IF2211
Penerapan Algoritma Branch and Bound untuk Menyelesaikan 15 Puzzle

Vincent Prasetiya Atmadja
K03/13520099

A. Algoritma Branch and Bound

Algoritma Branch and Bound merupakan salah satu algoritma pencarian yang populer digunakan di dunia informatika. Algoritma ini digunakan untuk persoalan optimisasi, yaitu persoalan meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (constraints) persoalan. Algoritma ini mengombinasikan Breadth First Search (BFS) dan *least cost search*.

Pada algoritma Branch and Bound, ada 2 hal utama yang perlu diperhatikan, berkaitan dengan pembangkitan simpul dan proses pencarian. Kedua hal tersebut adalah sebagai berikut

1. Setiap simpul diberi sebuah nilai cost (ĉ(i)), yaitu nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i.
2. Simpul berikutnya yang akan di-expand tidak lagi berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki cost yang paling kecil (least cost search) – pada kasus minimasi.

Pada algoritma ini, ada yang dinamakan fungsi pembatas, yaitu fungsi yang digunakan untuk menerapkan pemangkasan pada jalur yang dianggap tidak lagi mengarah pada solusi. Kriteria pemangkasan secara umum yaitu

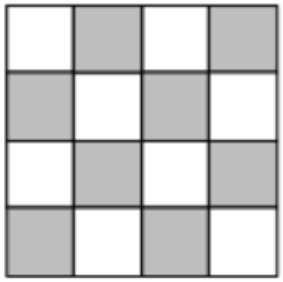
1. Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini (the best solution so far)
2. Simpul tidak merepresentasikan solusi yang ‘feasible’ karena ada batasan yang dilanggar
3. Solusi pada simpul tersebut hanya terdiri atas satu titik (tidak ada pilihan lain). Bandingkan nilai fungsi obyektif dengan solusi terbaik saat ini, yang terbaik yang diambil

Salah satu penerapan dari algoritma ini adalah penyelesaian 15 Puzzle. Puzzle ini berukuran 4 x 4 dengan satu buah sel kosong dan sel terisi lainnya diberi nomor teurut dari 1 sampai 15. Objektif dari permainan ini adalah mengolah puzzle sedemikian rupa sehingga sel kosong akan berada di pojok kanan bawah dan angka sisanya terurut. Gerakan yang bisa dilakukan adalah menggeser sel ke sel kosong.

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

Gambar 1. Susunan Akhir 15 Puzzle

Sebelum menerapkan algoritma Branch and Bound, terdapat preprocessing untuk mengecek apakah puzzle pada state awal mungkin untuk diselesaikan. Ada suatu teorema yang membantu yaitu, Status tujuan hanya dapat dicapai dari status awal jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap.



Gambar 2. Pembagian Nilai X

Nilai X akan ditentukan berdasarkan gambar di atas. Apabila kotak kosong berada di bagian yang diarsir, maka X akan bernilai 1 dan akan bernilai 0 untuk sebaliknya. Sementara itu, KURANG(i) didefinisikan sebagai, = banyaknya sel bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i), dengan POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.

Algoritma Branch and Bound secara umum dapat dituliskan langkah-langkahnya sebagai berikut.

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka stop.
2. Jika Q kosong, Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai nilai 'cost' ĉ(i) paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan. Jika satu solusi yang diinginkan, maka stop.

   Pada persoalan optimasi dengan pendekatan least cost search, periksa cost semua simpul hidup. Jika cost nya lebih besar dari cost simpul solusi, maka matikan simpul tersebut.
5. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
6. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
7. Kembali ke langkah 2.

Pada penyelesaian 15 Puzzle yang dapat diselesaikan, pembentukan dan penjelajahan simpulnya akan mengikuti langkah di atas. Yang membedakan adalah bagaiamana cost ĉ(i) untuk setiap simpul dihitung. Pada persoalan ini, cost ĉ(i) dihitung dengan cara sebagai berikut.

$$\hat{c}(p) = f(p) + \hat{g}(p)$$

dengan

ĉ(p) : ongkos untuk simpul p

f(p): Panjang lintasan dari simpul akar ke P
ĝ(p): jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir

B. Source Code Program

Program yang penulis buat terdiri dari 4 file Python. Board.py merupakan file yang berisi definisi kelas Board beserta dengan method-methodnya. PrioQueue.py merupakan file yang berisi implementasi kelas PrioQueue. ioHandler.py merupakan file yang mengolah input dan output. main.py merupakan kode utama program. Kode selengkapnya dapat dilihat pada repository penulis, yaitu (https://github.com/TheOne28/BnB_15Puzzle)

Board.py

```
from copy import deepcopy      Vincent Atmadja, 2 hours ago • Forget to make remote git, a

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

Vincent Atmadja, 2 hours ago | 1 author (Vincent Atmadja)
class Board:
    '''
        table -> papan permainan
        indX -> index X yang kosong
        indY -> index Y yang kosong
        depth -> kedalaman sekarang
    '''

    def __init__(self, table, indX, indY, depth, parent=None ,moveBefore=None) :
        self.table = table
        self.indX = indX
        self.indY = indY
        self.depth = depth
        self.parent = parent
        self.cost = depth + self.find_cost(table)
        self.moveBefore = moveBefore

    def find_cost(self, table):
        count = 0
        for i in range(4):
            for j in range(4):
                if(table[i][j] != 16 and target[i][j] != 16):
                    if(table[i][j] != target[i][j]):
                        count += 1
        return count


    def find_Kurangi(self, indX, indY):
        count = 0
        compare = self.table[indX][indY]
```

```
        for i in range(indY, 4):
            if(self.table[indX][i] < compare ):
                count += 1

        for i in range(indX + 1, 4):
            for j in range(4):
                if(self.table[i][j] < compare):
                    count += 1

        return count

    def is_solveable(self):
        varX = 0
        totalKurang = 0
        listKurangi = {}

        for i in range(4):
            for j in range(4):
                kurangi = self.find_Kurangi(i, j)
                listKurangi[self.table[i][j]] = kurangi
                totalKurang += kurangi

        if(self.indX % 2 == 0 and (self.indY == 1 or self.indY == 3)):
            varX = 1

        if(self.indX % 2 == 1 and (self.indY  == 0 or self.indY == 2)):
            varX = 1

        totalKurang += varX

        if(totalKurang % 2 == 1):
            return listKurangi, totalKurang, False
        else:
            return listKurangi, totalKurang, True

    def get_table(self):
        return self.table
```

```
def get_cost(self):
    return self.cost

def get_blankX(self):
    return self.indX

def get_blankY(self):
    return self.indY

def get_depth(self):
    return self.depth

def get_move_before(self):
    return self.moveBefore

def get_parent(self):
    return self.parent

def is_target(self):
    for i in range(4):
        for j in range(4):
            if(self.table[i][j] != target[i][j]):
                return False
    return True

def moveBlank(self, deltaX, deltaY):
    newX = self.indX + deltaX
    newY = self.indY + deltaY

    if(newX >=0 and newX <= 3 and newY >= 0 and newY <= 3):
        newTable = deepcopy(self.table)

        newTable[newX][newY], newTable[self.indX][self.indY] = newTable[self.indX][self.indY], newTable[newX][newY]
        return newTable
    else:
        return None
```

```
def printTable(self):
    if(self.moveBefore == None):
        print("Root")
    else:
        print("Move: ", self.moveBefore)

    for row in self.table:
        for cell in row:
            if(cell == 16):
                print("-", end="\t")
            else:
                print(cell, end = "\t")
        print()

def __lt__(self, other):
    return self.cost < other.get_cost()
```

PrioQueue.py

```
from heapq import heappush, heappop        Vincent Atmadja, 2 hours ago

Vincent Atmadja, 2 hours ago | 1 author (Vincent Atmadja)
class PriorityQueue:

    def __init__(self):
        self.queue = []

    def length(self):
        return len(self.queue)

    def front(self):
        return self.queue[0]

    def empty(self):
        return self.length() == 0

    def clear(self):
        self.queue.clear()

    def pop(self):
        item = heappop(self.queue)
        return item

    def push(self, item):
        heappush(self.queue, item)
```

ioHandler.py

```python
import os.path
from random import randint

stepAll = None

'''
Setup table
Mengecek apakah akan membaca table dari input atau membuat random table      Vincent Atmadja
'''
def setup():
    print("Welcome to 15Puzzles")
    print("Silahkan pilih mode yang anda inginkan: ")
    print("1. Random table")
    print("2. Input table")

    choice = 0

    while(choice < 1 or choice > 2):
        choice = int(input("Mode mana yang anda inginkan? "))

        if(choice < 1 or choice > 3):
            print("Input tidak valid!")

    table = None

    if(choice == 1):
        done = []
        table = []

        for i in range(4):
            intable = []
            for j in range(4):
                while(True):
                    rand = randint(1, 16)
                    if(rand not in done):
                        intable.append(rand)
```

```
                        done.append(rand)
                        break
            table.append(intable)
    else:
        table = get_input()

    return table

'''
Membaca table dari input
Input dapat berupa file maupun terminal
'''
def get_input():
    print("Untuk cell kosong, silahkan masukkan 16")
    print("Silahkan pilih cara memasukkan input: ")
    print("1. File ")
    print("2. Terminal")

    choice = 0

    table = []

    while(choice < 1 or choice > 2):
        choice = int(input("Input mana yang anda inginkan? "))

        if(choice < 1 or choice > 3):
            print("Input tidak valid!")

    if(choice == 1):
        print("File input harus terletak di folder test !!")

        name = input("Silahkan masukkan nama file (lengkap dengan ekstensi): ")

        name = "../test/" + name
        myPath = os.path.abspath(os.path.dirname(__file__))
        path = os.path.join(myPath, name)
```

```
        if(os.path.exists(path)):
            with open(path, 'r') as f:
                line = f.readline()

                while line != "":
                    inTable = [int(a) for a in line.split(" ")]
                    table.append(inTable)
                    line = f.readline()

        else:
            print("File tidak ada!!")
            return None
    else:

        print("Silahkan masukkan table di terminal: \n")

        for i in range(4):
            intable = []
            for j in range(4):
                intable.append(int(input()))
            table.append(intable)

    if(check_table_valid(table)):
        return table
    else:
        return None

'''
Mengecek bahwa input table valid

Syarat valid:
    1. Tidak ada angka kembar
    2. Semua angka berada di range 1 - 16 (inclusive) -> 16 untuk cell kosong
'''
def check_table_valid(table):
    done = []
```

```
    for row in table:
        for cell in row:
            if((cell not in done ) and cell >= 1 and cell <= 16):
                done.append(cell)
            else:
                return False
    return True

'''
Mengecek hasil Kurang(i) untuk setiap i
'''

def printKurangi(listKurangi):
    print ("   i            Kurang(i)")
    for numb in sorted(listKurangi):
        if(numb > 9):
            print("   {}              {}".format(numb, listKurangi[numb]))
        else:
            print("   {}               {}".format(numb, listKurangi[numb]))

'''
Mencetak semua Board yang dilalui sampai target
'''
      Vincent Atmadja, 2 hours ago • Forget to make remote git, add source code …
def printRoute(solutionBoard, step):
    if(solutionBoard == None):
        global stepAll
        stepAll = step
        return
    printRoute(solutionBoard.get_parent(), step + 1)
    solutionBoard.printTable()
```

main.py

```python
from copy import deepcopy
import timeit
import ioHandler

from Board import Board
from PrioQueue import PriorityQueue

solutionBoard = None
pq = PriorityQueue()
boardCount = 0

def generateTable(table, moveBefore=None):
    tableUp = table.moveBlank(-1, 0)
    tableDown = table.moveBlank(1, 0)
    tableRight = table.moveBlank(0, 1)
    tableLeft = table.moveBlank(0, -1)

    global boardCount
    if(tableUp != None and moveBefore != "Down"):
        boardCount += 1
        pq.push(Board(tableUp, table.get_blankX() - 1, table.get_blankY(), table.get_depth() + 1, table, "Up"))
    if(tableDown != None and moveBefore != "Up"):
        boardCount += 1
        pq.push(Board(tableDown, table.get_blankX() + 1, table.get_blankY(), table.get_depth() + 1, table, "Down"))
    if(tableRight != None and moveBefore != "Left"):
        boardCount += 1
        pq.push(Board(tableRight, table.get_blankX(), table.get_blankY() + 1, table.get_depth() + 1, table, "Right"))
    if(tableLeft != None and moveBefore != "Right"):
        boardCount += 1
        pq.push(Board(tableLeft, table.get_blankX(), table.get_blankY() - 1, table.get_depth() + 1, table, "Left"))

def solve(table):
    copyTable = deepcopy(table)
    pq.push(table)
    while(not pq.empty()):
        if(copyTable.is_target()):
            ioHandler.printRoute(copyTable, 0)
            return
        else:
            front = pq.pop()        Vincent Atmadja, 2 hours ago • Add handler in solve, if there are unsolv
            listKurangi, totalKurang, solveable = front.is_solveable()

            if(solveable):
                generateTable(front, front.get_move_before())
                copyTable = pq.front()

def main():
    table = ioHandler.setup()

    if(table == None):
        print("Input tidak valid")
        return

    start = timeit.default_timer()

    indX, indY = -1, -1

    for i in range(4):
        for j in range(4):
            if(table[i][j] == 16):
                indX, indY = i, j
                break
        if(indX != -1):
            break

    root = Board(table, indX, indY, 0)

    listKurangi, totalKurang, solveable = root.is_solveable()

    ioHandler.printKurangi(listKurangi)
```

```
    print("\u03A3Kurang(i) + X = ", (totalKurang))
    if(not solveable):
        stop = timeit.default_timer()

        print("Puzzle tidak dapat diselesaikan !!")
        print("Waktu yang dibutuhkan: ", stop - start, " sekon")
        return

    print()
    print("Cara penyelesaian: ")
    print()


    solve(root)

    step = ioHandler.stepAll
    print("Puzzle diselesaikan dalam {} langkah!!".format(step))

    stop = timeit.default_timer()
    print("Waktu yang dibutuhkan: ", stop - start, " sekon")
    print("Simpul yang dibangkitkan: ", boardCount)

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

C. Input dan Output

Berikut ini akan penulis lampirkan beberapa input dan output dari beberapa puzzle yang penulis uji. Untuk selengkapnya dapat dilihat pada folder test.

1. solveable1.txt

   Input:

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

   Output:

```
Silahkan masukkan nama file (lengkap dengan ekstensi): solveable1.txt
   i           Kurang(i)
   1              0
   2              0
   3              0
   4              0
   5              0
   6              0
   7              0
   8              1
   9              1
   10             1
   11             0
   12             0
   13             1
   14             1
   15             1
   16             9
ΣKurang(i) + X =  16

Cara penyelesaian:

Root
1       2       3       4
5       6       -       8
9       10      7       11
13      14      15      12
Move:  Down
1       2       3       4
5       6       7       8
9       10      -       11
13      14      15      12
Move:  Right
1       2       3       4
5       6       7       8
9       10      11      -
13      14      15      12
Move:  Down
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      -
Puzzle diselesaikan dalam 4 langkah!!
Waktu yang dibutuhkan:  0.005587800000284915  sekon
Simpul yang dibangkitkan:  9
```

2. solveable2.txt

   Input:

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

   Output:

```
Cara penyelesaian:

Root
1       2       3       4
5       6       12      7
9       10      -       8
13      14      11      15
Move:  Up
1       2       3       4
5       6       -       7
9       10      12      8
13      14      11      15
Move:  Right
1       2       3       4
5       6       7       -
9       10      12      8
13      14      11      15
Move:  Down
1       2       3       4
5       6       7       8
9       10      12      -
13      14      11      15
Move:  Left
1       2       3       4
5       6       7       8
9       10      -       12
13      14      11      15
Move:  Down
1       2       3       4
5       6       7       8
9       10      11      12
13      14      -       15
Move:  Right
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      -
Puzzle diselesaikan dalam 7 langkah!!
Waktu yang dibutuhkan:  0.007536099999924772  sekon
Simpul yang dibangkitkan:  26
```

3. solveable3.txt

   Input:

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

   Output:

```
Cara penyelesaian:

Root
3       1       2       4
-       5       7       8
10      6       11      12
9       13      14      15
Move:  Right
3       1       2       4
5       -       7       8
10      6       11      12
9       13      14      15
Move:  Down
3       1       2       4
5       6       7       8
10      -       11      12
9       13      14      15
Move:  Left
3       1       2       4
5       6       7       8
-       10      11      12
9       13      14      15
Move:  Up
3       1       2       4
-       6       7       8
5       10      11      12
9       13      14      15
Move:  Up
-       1       2       4
3       6       7       8
5       10      11      12
9       13      14      15
Move:  Right
1       -       2       4
3       6       7       8
5       10      11      12
9       13      14      15
Move:  Down
1       6       2       4
3       -       7       8
5       10      11      12
9       13      14      15
```

```
Move:  Left
1       6       2       4
-       3       7       8
5       10      11      12
9       13      14      15
Move:  Down
1       6       2       4
5       3       7       8
-       10      11      12
9       13      14      15
Move:  Down
1       6       2       4
5       3       7       8
9       10      11      12
-       13      14      15
Move:  Right
1       6       2       4
5       3       7       8
9       10      11      12
13      -       14      15
Move:  Right
1       6       2       4
5       3       7       8
9       10      11      12
13      14      -       15
Move:  Up
1       6       2       4
5       3       7       8
9       10      -       12
13      14      11      15
Move:  Up
1       6       2       4
5       3       -       8
9       10      7       12
13      14      11      15
Move:  Left
1       6       2       4
5       -       3       8
9       10      7       12
13      14      11      15
Move:  Up
1       -       2       4
5       6       3       8
9       10      7       12
13      14      11      15
```

```
Move:  Right
1       2       -       4
5       6       3       8
9       10      7       12
13      14      11      15
Move:  Down
1       2       3       4
5       6       -       8
9       10      7       12
13      14      11      15
Move:  Down
1       2       3       4
5       6       7       8
9       10      -       12
13      14      11      15
Move:  Down
1       2       3       4
5       6       7       8
9       10      11      12
13      14      -       15
Move:  Right
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      -
Puzzle diselesaikan dalam 22 langkah!!
Waktu yang dibutuhkan:  0.7927710000003572  sekon
Simpul yang dibangkitkan:  24868
```

4. unsolveable1.txt
   Input:

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

   Output:

```
   i            Kurang(i)
   1               0
   2               0
   3               1
   4               1
   5               0
   6               0
   7               1
   8               0
   9               0
   10              0
   11              3
   12              6
   13              0
   14              4
   15              11
   16              10
ΣKurang(i) + X =  37
Puzzle tidak dapat diselesaikan !!
Waktu yang dibutuhkan:  0.0017850000003818423  sekon
```

5. unsolveable2.txt

   Input:

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

   Output:

```
   i            Kurang(i)
   1               0
   2               1
   3               2
   4               3
   5               4
   6               5
   7               0
   8               1
   9               1
   10              1
   11              0
   12              0
   13              1
   14              1
   15              1
   16              9
ΣKurang(i) + X =  31
Puzzle tidak dapat diselesaikan !!
Waktu yang dibutuhkan:  0.002134799999112147  sekon
```

D. **Extra**

Github: https://github.com/TheOne28/BnB_15Puzzle

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		