Tugas Kecil 3

IF2211 Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Disusun oleh

Indra Febrio Nugroho 13518016

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2020

BAB I
DASAR TEORI

1.1 Algoritma Branch and Bound

Berikut ini adalah karakteristik dari algoritma branch and bound.

- Digunakan untuk persoalan optimisasi → meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (*constraints*) persoalan.
- B&B: BFS + least cost search
  - BFS murni: Simpul berikutnya yang akan diekspansi berdasarkan urutan pembangkitannya (FIFO)
- B&B:
  - Setiap simpul diberi sebuah nilai cost: ĉ(i) = nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i.
  - 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.

Cost dari simpul hidup dapat dihitung sebagai berikut.

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

$\hat{c}(i)$ = ongkos untuk simpul i
$\hat{f}(i)$ = ongkos mencapai simpul i dari akar
$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul i.

$f(P)$ = adalah panjang lintasan dari simpul akar ke P

$\hat{g}(P)$ = taksiran panjang lintasan terpendek dari P ke simpul solusi pada upapohon yang akarnya P.

Gambar 1. Cost simpul hidup

1.2 Garis Besar Algoritma Branch and Bound

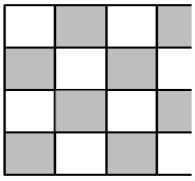
Berikut ini garis besar penyelesaian masalah dengan algoritma branch and bound.

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. 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, stop. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.

5. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

1.3 Teorema Reachable Goal

Status tujuan hanya dapat dicapai dari status awal jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap. X=1 jika pada sel yg diarsir. Kurang(i) adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan Posisi(j) > Posisi(i). POSISI(i) adalah posisi ubin bernomor i pada susunan yang diperiksa. Berikut ini gambar arsiran ubin untuk menghitung nilai X.



Gambar 2. Arsiran ubin

BAB II
PENYELESAIAN PERSOALAN

2.1 Garis Besar Program

Penulis menyelesaikan persoalan 15-Puzzle dengan Bahasa Python sesuai dengan spesifikasi tugas kecil 3. Program dapat melakukan hal-hal sebagai berikut.

1. Membaca input matriks dari file teks eksternal
2. Menghitung nilai fungsi Kurang(i)
3. Menghitung nilai ubin pertama (X)
4. Memprediksi puzzle dapat diselesaikan atau tidak
5. Menampilkan jalur penyelesaian puzzle
6. Menampilkan waktu eksekusi algoritma
7. Menghitung jumlah simpul yang di*generate* untuk menyelesaikan puzzle

2.2 Langkah Penyelesaian

Berikut ini langkah penyelesaian masalah dengan algoritma Branch and Bound

1. Program membaca masukan matriks dari file teks eksternal
2. Menetapkan matriks goal sesuai spesifikasi tugas
3. Menghitung fungsi Reachable goal yaitu fungsi Kurang(i) + X untuk menentukan apakah matriks dapat diselesaikan atau tidak
4. Apabila tidak, maka program selesai
5. Apabila iya, maka program akan memanggil fungsi solvePuzzle untuk menentukan pembangkitan simpul solusi untuk menyelesaikan persoalan.
6. Penyelesaian dimulai dengan membangkitkan PrioQueue kosong
7. Cek apakah matriks tersebut telah sesuai dengan matriks goal. Bila sesuai, program berhenti
8. Bila tidak, maka matriks ini dimasukan ke dalam prio queue.
9. Ambil elemen pertama prio queue, kemudian bangkitkan simpul dengan menggerakkan blank tile secara up, down, left, atau right. Perhatikan apakah perpindahan tidak melebihi batas matriks atau tidak. Masukkan matriks yang diekspansi ke dalam list of generated simpul.
10. Apabila simpul-simpul tersebut belum ada di dalam prio queue, masukkan simpul-simpul tersebut ke dalam prioqueue dengan menyimpan nilai cost, level, dan list dari simpul sebelumnya.
11. Ulangi langkah 9-10 sampai ditemukan simpul dengan matriks yang sesuai dengan goal state.

2.3 Struktur Program

Program ini memiliki struktur sebagai berikut

- *Penggunaan library dan pendefinisian kelas*
- *Deklarasi fungsi*
- *Main*

Pada struktur penggunaan *library*, penulis menggunakan *library time* dan *library copy*. *Library time* berfungsi untuk menghitung waktu eksekusi program. Sedangkan *library copy* berfungsi untuk melakukan *pass by value* terhadap suatu list dan/atau array dan/atau matriks dan/atau queue. Pada bagian pendefinisian kelas, penulis mengimplementasikan priority queue yang digunakan untuk menyimpan simpul-simpul hidup saat penyelesaian puzzle. Prio queue ini menyimpan sebuah tuples yang berisi cost simpul, level simpul, matrix dari simpul tersebut, dan array dari matrix simpul yang telah digenerate *sebelumnya*. Prio queue ini mengurutkan data berdasarkan cost simpul terkecil.

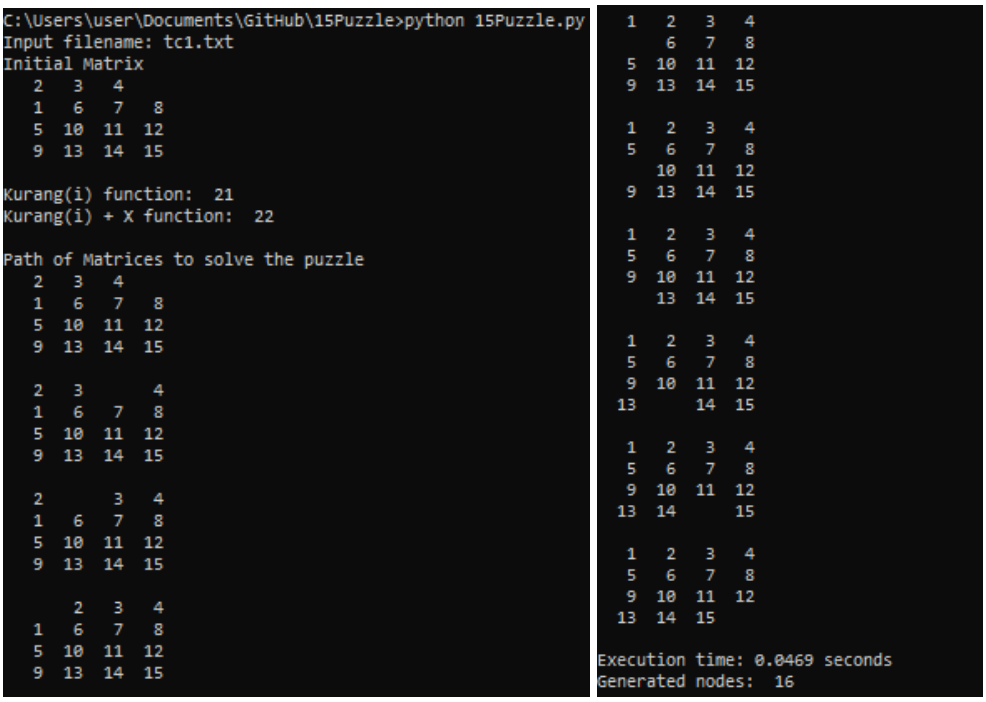
Pada bagian deklarasi fungsi, penulis menuliskan nama fungsi yang digunakan beserta implementasinya. Berikut ini fungsi-fungsi yang terdapat pada bagian ini.

- findBlankTile(m), mencari posisi blank tile di dalam matrix
- swapTiles(m, xInit, yInit, xNew, yNew), menukar dua nilai dalam matrix
- kurangFunc(m), menghitung nilai fungsi Kurang
- xFunc(x, y), menghitung nilai letak ubin kosong pertama
- isReachable(m, x, y), memprediksi apakah puzzle dapat diselesaikan atau tidak
- countMisplacedTiles(mInit, mGoal), menghitung ubin tidak kosong yang berbeda dari matrix goal
- isPuzzleSolved(mI, mG), membandingkan matrix masukan dengan matrix goal
- calcCost(l, mInit, mGoal), menghitung cost simpul
- printMatrix(m), menuliskan matrix ke layar
- printPath(mGenerated), menggambarkan jalur penyelesaian puzzle
- isMoveValid(x,y), mengecek perpindahan blank tile sesuai dengan aturan
- moveUp(m,x,y), memindahkan blank tile ke atas
- moveDown(m,x,y), memindahkan blank tile ke bawah
- moveLeft(m,x,y), memindahkan blank tile ke kiri
- moveRight(m,x,y), memindahkan blank tile ke kanan
- searchMatrix(arr,m), mencari matriks masukan di dalam array of matrix
- solvePuzzle(mInit, mGoal), menyelesaikan persoalan dengan branch and bound

Struktur *main* berisi pembacaan file input matrix, inisialisasi matrix goal, pemanggilan fungsi solvePuzzle, dan penyesuaian output program dengan spesifikasi.

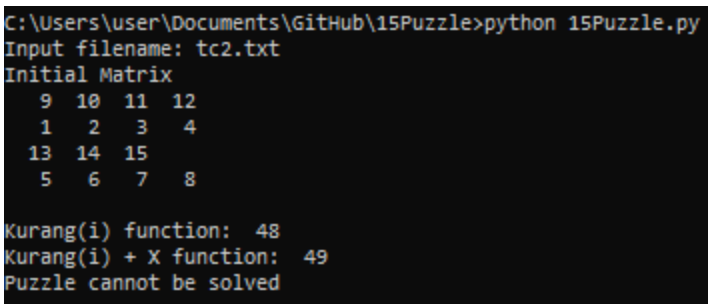
BAB III
HASIL AKHIR

3.1 Input Output Program Case 1 (berhasil)



Gambar 3 i/o case 1

3.2 Input Output Program Case 2 (gagal)



Gambar 4 i/o case 2

3.3 Input Output Program Case 3 (berhasil)

```
C:\Users\user\Documents\GitHub\15Puzzle>python 15Puzzle.py
Input filename: tc3.txt
Initial Matrix
   1   2   3   4
   5   6       8
   9  10   7  11
  13  14  15  12

Kurang(i) function:  15
Kurang(i) + X function:  16

Path of Matrices to solve the puzzle
   1   2   3   4
   5   6       8
   9  10   7  11
  13  14  15  12

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

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

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

Execution time: 0.0156 seconds
Generated nodes:  9
```

Gambar 5 i/o case 3

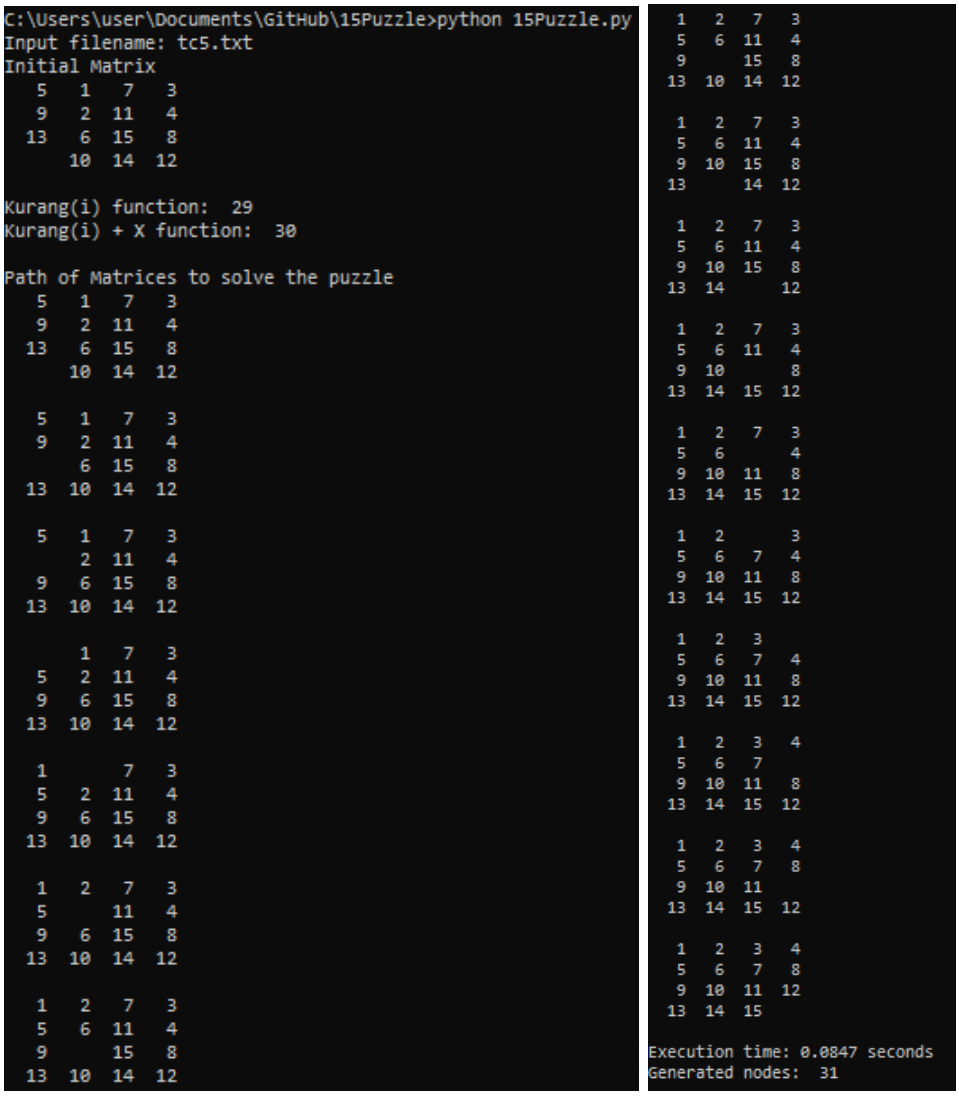
3.4 Input Output Program Case 4 (gagal)

```
C:\Users\user\Documents\GitHub\15Puzzle>python 15Puzzle.py
Input filename: tc4.txt
Initial Matrix
   2   5   6   7
   1   3   4   8
       9  10  11
  12  13  14  15

Kurang(i) function:  17
Kurang(i) + X function:  17
Puzzle cannot be solved
```

Gambar 6 i/o case 4

3.5 Input Output Program Case 5 (berhasil)



Gambar 7. i/o case 5

3.6 Kesimpulan

Luaran dari tugas ini dapat dimuat secara ringkas dalam tabel berikut

Tabel 1 Output Persoalan

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	

REFERENSI

Munir, Rinaldi. 2004. *Diktat Bahan Kuliah Strategi Algoritmik.* Bandung: Departemen Teknik Informatika, Institut Teknologi Bandung.