

Laporan Tugas Kecil 3
IF 2211 Strategi Algoritma
Penyelesaian Persoalan 15-Puzzle dengan Algoritma
Branch and Bound



Disusun Oleh:
Farras Mohammad Hibban Faddila
13518017

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2019/2020

Daftar Isi

Daftar Isi	2
Daftar Gambar dan Tabel	3
Algoritma Branch and Bound	4
Pengujian Program	6
Checklist	9
Referensi	10

Daftar Gambar dan Tabel

Gambar 1.1. Pseudocode untuk algoritma Branch and Bound

Gambar 2.1. Kasus uji pertama

Gambar 2.2. Screenshot untuk kasus uji pertama (1)

Gambar 2.3. Screenshot untuk kasus uji pertama (2)

Gambar 2.4. Kasus uji kedua

Gambar 2.5. Screenshot untuk kasus uji kedua (1)

Gambar 2.6. Screenshot untuk kasus uji pertama (2)

Gambar 2.7. Screenshot untuk kasus uji pertama (3)

Tabel 3.1. Tabel ceklist pengecekan program

A. Algoritma Branch and Bound

Branch and Bound merupakan sebuah algoritma pencarian pada pohon pencarian, dan terdapat sebuah kuantitas bernama *cost* yang diberikan pada tiap simpul status di pohon pencarian tersebut. Nilai *cost* dari sebuah simpul i merupakan nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul i . Dengan kata lain, nilai *cost* akan memberikan taksiran jalur mana yang memiliki jarak terdekat dengan simpul status yang menjadi tujuan. Berdasarkan sifat tersebut, algoritma ini akan memprioritaskan simpul *child* yang memiliki *cost* terendah untuk diproses (diekspansi).

Pada permainan 15-puzzle, nilai dari *cost* untuk setiap simpul status i didefinisikan sebagai $c(i)$ sebagai berikut ini,

$$c(i) = f(i) + g(i),$$

di mana $f(i)$ merupakan banyak *step* yang dibutuhkan untuk mencapai simpul status i dari akar, dan $g(i)$ merupakan taksiran dari jarak simpul status i ke simpul status tujuan. Taksiran ini merupakan sebuah batas bawah (*bound*), dan nilai dari *bound* ini diperoleh dengan cara menghitung banyaknya kotak berisi angka pada simpul status i yang lokasinya pada matriks tidak sesuai dengan lokasi yang seharusnya. Sebagai contoh, perhatikan konfigurasi berikut ini,

$$i = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 7 & * & 8 \\ 9 & 10 & 12 & 15 \\ 6 & 11 & 14 & 13 \end{bmatrix}$$

Nilai dari $g(i)$ merupakan 7 karena terdapat 7 kotak berisi angka yang tidak sesuai dengan lokasi seharusnya, yakni kotak yang berisi angka 7, 12, 15, 6, 11, 14 dan 13 (Di sini, kotak kosong dilambangkan dengan tanda *), sedangkan nilai $f(i)$ bergantung pada akar dari pohon pencarian, yakni merupakan banyak step dari akar ke i .

Algoritma yang diimplementasikan pada program adalah sebagai berikut. Pertama, masukkan state awal permainan pada suatu Collection T , yang jenisnya akan ditentukan kemudian. Setelah itu, ambil elemen pertama pada T , dan jadikan elemen tersebut sebagai simpul yang akan diekspansi, dan masukkan simpul-simpul hasil ekspansinya pada Collection T . Hasil dari ekspansi sebuah simpul *state* merupakan kumpulan *state* yang dapat dicapai oleh *state* tersebut dalam satu langkah permainan, yakni *state* yang dibentuk dengan menggerakkan kotak kosong ke kiri, kanan, atas, atau bawah apabila valid. Berdasarkan algoritma Branch and Bound, kita ingin agar elemen yang diekspansi merupakan elemen dengan nilai *cost*

terkecil. Oleh karena itu, Collection yang digunakan adalah sebuah *priority_queue*, yakni queue yang mengurutkan elemennya berdasarkan sebuah prioritas. Di sini, prioritas dari tiap elemen queue bergantung pada nilai *cost*-nya. Semakin kecil *cost* dari sebuah elemen, maka semakin tinggi prioritasnya sehingga akan didahulukan saat melakukan ekspansi.

Pseudocode dari algoritmanya adalah sebagai berikut. Sebagai catatan, kelas Matriks merupakan sebuah kelas yang mendefinisikan *state* dari permainan n-puzzle ini, dan memuat beberapa method seperti untuk menghitung nilai dari fungsi *KURANG()*, menentukan *state* berikutnya setelah suatu langkah, menentukan apakah suatu konfigurasi permainan dapat diselesaikan atau tidak, dan lain-lain.

```
algorithm bnb is:
  input: Matriks root, yakni state awal permainan
  output: integer b, yakni jumlah simpul yang dibangkitkan
         hashmap par, yang menyimpan informasi parent dari sebuah simpul

  dist <- new hashmap of Matriks
  vis <- new hashmap of Matriks
  Par <- new hashmap of Matriks
  bangkit = 0
  ketemu <- false
  procedure ubah komparator pada kelas Matriks menjadi sesuai nilai cost

  simpul <- new priority_queue of Matriks
  simpul <- simpul U root
  dist[root] <- 0
  vis[root] <- true

  while simpul not empty and not ketemu do
    expand <- top element in simpul
    simpul <- simpul - expand
    for each step (x,y) in {(0,1),(0,-1),(1,0),(-1,0)} do
      child <- state after move with step (x,y) from expand
      if child is a legal move
        if child is not visited
          par[child] <- expand
          dist[child] <- dist[expand] + 1
          bangkit <- bangkit + 1
          vis[child] = true
          simpul <- simpul U child
          if child is target node:
            ketemu <- true

  return par, bangkit
```

Gambar 1.1. Pseudocode untuk algoritma Branch and Bound

B. Pengujian Program

1. Kasus Uji 1 (Permainan tidak *solvable*)

Pada kasus ini, konfigurasi awal permainan yang diinput pada program adalah sebagai berikut, disimpan pada file bernama `gameconfig1.txt`

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

Gambar 2.1. Kasus uji pertama

Berikut ini adalah screenshot program dengan input file tersebut.

```
Selamat datang di aplikasi 15-puzzle solver!  
Buat file konfigurasi awal permainan di file terpisah ya!  
Ketikkan nama filenya, ekstensinya juga ya:) : gameconfig1.txt  
Matriks posisi awal yang dibaca dari gameconfig1.txt adalah sebagai berikut:  
-----  
1 2 3 4  
5 6 7 *  
8 9 12 13  
10 15 14 11  
-----
```

Gambar 2.2. Screenshot untuk kasus uji pertama (1)

```
Nilai dari KURANG(1) adalah: 0
Nilai dari KURANG(2) adalah: 0
Nilai dari KURANG(3) adalah: 0
Nilai dari KURANG(4) adalah: 0
Nilai dari KURANG(5) adalah: 0
Nilai dari KURANG(6) adalah: 0
Nilai dari KURANG(7) adalah: 0
Nilai dari KURANG(8) adalah: 0
Nilai dari KURANG(9) adalah: 0
Nilai dari KURANG(10) adalah: 0
Nilai dari KURANG(11) adalah: 0
Nilai dari KURANG(12) adalah: 2
Nilai dari KURANG(13) adalah: 2
Nilai dari KURANG(14) adalah: 1
Nilai dari KURANG(15) adalah: 2
Nilai dari KURANG(16) adalah: 8

Nilai dari total fungsi KURANG dan X adalah: 15

Nilainya ganjil!.

Oleh karena itu, puzzle ini tidak mungkin diselesaikan, maaf :(
```

Gambar 2.3. Screenshot untuk kasus uji pertama (2)

2. Kasus Uji 2 (Permainan *solvable*)

Pada kasus ini, konfigurasi awal permainan yang diinput pada program adalah sebagai berikut, disimpan pada file bernama `gameconfig2.txt`

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

Gambar 2.4. Kasus uji kedua

Berikut ini adalah screenshot program dengan input file tersebut.

```

donbasta: ~/Desktop/lo/IF/semester4/Stima/stima_tucl/n-puzzle-solver-branch-and-bound
farras@donbasta:~/Desktop/lo/IF/semester4/Stima/stima_tucl/n-puzzle-solver-branch-and-bound$ python3.7 main.py
Selamat datang di aplikasi 15-puzzle solver!
Buat file konfigurasi awal permainan di file terpisah ya!
Ketikkan nama filenya, ekstensinya juga ya:) : gameconfig2.txt
Matriks posisi awal yang dibaca dari gameconfig2.txt adalah sebagai berikut:
-----
1 2 * 4
5 6 3 7
9 10 11 8
13 14 15 12
-----

Nilai dari KURANG(1) adalah: 0
Nilai dari KURANG(2) adalah: 0
Nilai dari KURANG(3) adalah: 0
Nilai dari KURANG(4) adalah: 1
Nilai dari KURANG(5) adalah: 1
Nilai dari KURANG(6) adalah: 1
Nilai dari KURANG(7) adalah: 0
Nilai dari KURANG(8) adalah: 0
Nilai dari KURANG(9) adalah: 1
Nilai dari KURANG(10) adalah: 1
Nilai dari KURANG(11) adalah: 1
Nilai dari KURANG(12) adalah: 0
Nilai dari KURANG(13) adalah: 1
Nilai dari KURANG(14) adalah: 1
Nilai dari KURANG(15) adalah: 1
Nilai dari KURANG(16) adalah: 13

```

Gambar 2.5. Screenshot untuk kasus uji kedua (1)

```

Nilai dari total fungsi KURANG dan X adalah: 22

Nilainya genap!!!!.

Yey, puzzle ini dapat diselesaikan, hore :D
Dengan menggunakan algoritma Branch n Bound, solusinya adalah sebagai berikut:
-----
Step 0:

1 2 * 4
5 6 3 7
9 10 11 8
13 14 15 12
-----
Step 1:

1 2 3 4
5 6 * 7
9 10 11 8
13 14 15 12
-----
Step 2:

1 2 3 4
5 6 7 *
9 10 11 8
13 14 15 12
-----
Step 3:

```

Gambar 2.6. Screenshot untuk kasus uji pertama (2)


```

9 10 11 8
13 14 15 12
-----
Step 2:

1 2 3 4
5 6 7 *
9 10 11 8
13 14 15 12
-----
Step 3:

1 2 3 4
5 6 7 8
9 10 11 *
13 14 15 12
-----
Step 4:

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 *
-----
Waktu eksekusi algoritmanya adalah: 5987.948 mikro sekon
Banyak simpul yang dibangkitkan saat pencarian adalah 11
farras@donbasta:~/Desktop/lol/IF/semester4/Stima/stima_tu

```

Gambar 2.7. Screenshot untuk kasus uji pertama (3)

C. Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat menerima <i>input</i> dan menuliskan <i>output</i>	✓	
4. Luaran sudah benar untuk semua data uji	✓	

Tabel 3.1. Tabel ceklist pengecekan program

D. Referensi

Munir, Rinaldi. 2018. *Algoritma Branch & Bound*. Program Studi Informatika - STEI ITB