**Laporan Tugas Kecil 3 IF2211-Strategi Algoritma**

**Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound**



**Disusun Oleh:**

**Raden Haryosatyo Wisjnunandono**

**13520070**

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2021/2022**

**BAB I**

**ALGORITMA BRANCH AND BOUND DALAM PERSOALAN 15-PUZZLE**

* 1. **Algoritma Branch and Bound Utama**

Dalam persoalan penyelesaian 15-puzzle ini dapat digunakan algoritma branch and bound sebagai algoritma utama. Pada subbab ini akan dijelaskan algoritma utama branch and bound saja dan diasumsikan state awal adalah solusi dapat dicari (solveable). Untuk algoritma untuk menentukan apakah suatu puzzle dapat diselesaikan atau tidak akan dijelaskan pada subbab selanjutnya.

Algoritma branch and bound yang diimplementasikan dalam program ini adalah sebagai berikut:

1. Inisialisasi sebuah pohon ruang status dengan state awal sebagai rootnya.
2. Masukkan node root ke dalam antrian Q
3. Jika antrian Q kosong hentikan proses
4. Pilih dan keluarkan dari antrian Q node N yang memiliki cost untuk mencapai goal yang terkecil
5. Jika node N merupakan simpul solusi maka hentikan pencarian
6. Jika node N bukan merupakan solusi, bangkitkan anak-anaknya
   * Anak yang valid dari sebuah node adalah yang jika *empty slot-*nya digeser bentuk statenya tidak kembali ke state parent dari node dan movementnya valid (jika *empty slot* sudah berada di paling kanan puzzle maka tidak bisa digerakkan ke kanan)
7. Untuk setiap node anak M dari node N hitung cost untuk mencapai goal lalu hitung cost total nya dengan menjumlahkan cost menuju goal dengan depth dari node M.
8. Masukkan node M ke dalam antrian Q
9. Kembali ke langkah 3

*\*) note: Untuk algoritma penentuan solvabilitas dan penghitungan cost akan dijelaskan pada subbab selanjutnya.*

* 1. **Algoritma Penghitungan Cost Sebuah Node Menuju Goal State**

Pada subbab ini akan dijelaskan algoritma untuk mengetahui cost dari suatu node untuk menuju ke goal state. Adapun algoritmanya adalah sebagai berikut:

1. Buat sebuah larik 1 dimensi L
2. Masukkan elemen-elemen puzzle ke larik pada indeks i dengan rumus:
   * i = no baris\*4 + no kolom, dengan i, no kolom, dan nomor baris dimulai dari 0
3. Inisialisasi varibel counter untuk menghitung banyaknya tile pada puzzle yang berada bukan pada tempatnya
4. Iterasi seluruh elemen pada larik, jika elemen larik di indeks i tidak sama dengan i+1 (kecuali untuk elemen larik di indeks i = 16), maka tambahkan 1 pada variabel counter
5. Setelah selesai melakukan iterasi, counter merupakan cost dari puzzle menuju goal state
6. Jumlahkan counter dengan depth dari node untuk mendapat total cost
   1. **Algoritma Pengecekan Solvabilitas Puzzle (Kurang(i) + X)**

Pada subbab ini akan dijelskan algoritma pencarian kurang(i) dan penentuan X

**1.3.1 Algoritma Kurang(i)**

1. Buat sebuah larik 1 dimensi L
2. Masukkan elemen-elemen puzzle ke larik pada indeks i dengan rumus:
   * i = no baris\*4 + no kolom, dengan i, no kolom, dan nomor baris dimulai dari 0
3. Inisialisasi varibel kurang untuk menghitung banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.
4. Iterasi larik sebanyak 16 kali dengan iterator i, pada tiap iterasi tersebut iterasi larik lagi dari mulai i+1 sampai 16 dengan iterator j.
5. Untuk tiap iterasi j, tambahkan 1 pada variabel kurang jika terdapat ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.

**1.3.2 Algoritma Penentuan X**

1. Tentukan letak tile kosong pada puzzle
2. Simpang posisi tile pada koordinat R,C dengan R adalah baris dan C adalah kolom tile
3. X adalah hasil dari penjumlahan R dan C yang dimodulo dengan 2

**1.3.3 Algoritma Penentuan Solvabilitas**

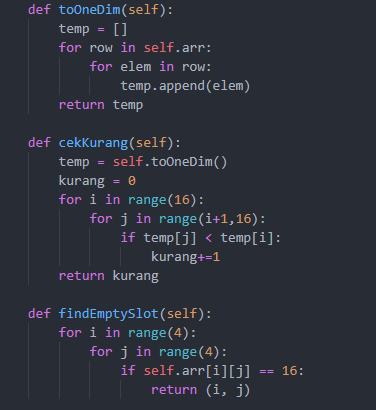
1. Jumlahkan variabel kurang pada subbab 1.3.1 dengan varibel X pada 1.3.2
2. Jika hasil penjumlahan genap maka puzzle tersebut dapat dipecahkan
3. Sebalikanya, jika hasil penjumlahan adalah ganjil maka puzzle tersebut tidak dapat dipecahkan.

**BAB II**

**SOURCE CODE**

**2.1 Kelas Matrix Untuk Menyimpan Puzzle**

****

****

**Text

Description automatically generated**

**Text

Description automatically generated**

**2.2 Kelas Tree yang Menyimpang Pohon Ruang Status**

**Text

Description automatically generated**

**2.3 Kelas PriorityQueue yang Menyimpan Antrian Q**

**Text

Description automatically generated**

**2.4 Main Program**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**BAB III**

**INPUT DAN OUTPUT**

* 1. **solveable\_01.txt**

Input File:

Graphical user interface, text

Description automatically generated

Output:

A picture containing text, electronics, keyboard

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

* 1. **solveable\_02.txt**

Input File:

A picture containing text, electronics, close

Description automatically generated

Output:

Calendar

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

* 1. **solveable\_03.txt**

Input File:

**Graphical user interface, text

Description automatically generated**

Output:

**A screenshot of a computer

Description automatically generated with low confidence** **A screenshot of a computer

Description automatically generated with medium confidence**

**Calendar

Description automatically generated**

* 1. **unsolveable\_01.txt**

Input File:

A picture containing text, electronics, close

Description automatically generated

Output:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. **unsolveable\_02.txt**

Input File:

A picture containing text, electronics

Description automatically generated

Output:

**A screenshot of a computer

Description automatically generated with medium confidence**

**BAB IV**

**KESIMPULAN**

Permasalahan persoalan penyelesaian 15-puzzle dapat diselesaikan dengan menerapkan algoritma branch and bound. Tiap-tiap state board puzzle dapat dianggap sebagai node dari sebuah pohon ruang status. Anak yang valid dari sebuah node adalah yang jika empty slot-nya digeser bentuk statenya tidak kembali ke state parent dari node dan movementnya valid (jika empty slot sudah berada di paling kanan puzzle maka tidak bisa digerakkan ke kanan). Tiap pembangkitan node baru kita hitung cost dari node tersebut unutk mencapai goal state lalu masukkan node tersebut ke dalam antrian. Pengeluaran elemen dari antrian didasarkan pada cost dari node. Semakin kecil cost nya semakin diprioritaskan.

|  |  |  |
| --- | --- | --- |
| **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 |  |  |

Link drive:

(buka dengan akun std)

Link GitHub

<https://github.com/nandono206/15-Puzzle-Branch-and-Bound>