**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)** Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

**3. Contoh-Contoh Penyelesaian 15-Puzzle**

a. Random puzzle yang menghasilkan unsolvable puzzle:

Text

Description automatically generated

b. Puzzle unsolvable yang dibaca dari file:

Text

Description automatically generated

c. Puzzle dengan 12 langkah penyelesaian

Text

Description automatically generated

Calendar

Description automatically generated with medium confidenceA screen shot of a computer

Description automatically generated with low confidence

d. Puzzle dengan 18 langkah penyelesaian

Text

Description automatically generated

Calendar

Description automatically generatedCalendar

Description automatically generated

e. Puzzle dengan 20 langkah penyelesaian

Text

Description automatically generated

Calendar

Description automatically generatedCalendar

Description automatically generated

**4. *Link GitHub Repository Source Code***

[**https://github.com/zaydzuhri/Tucil3\_13520144**](https://github.com/zaydzuhri/Tucil3_13520144)

**Table

Description automatically generated**

