Pencarian Rute Terpendek untuk Robot Gerak-dan-Ganti Menggunakan Algoritme Pencarian *A-Star*

M. Abdi Haryadi. H (13519156)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: abdiharyadi.ah@gmail.com

***Abstrak*—Diberikan papan berubin dengan banyak kolom dan baris tertentu. Setiap ubin diberi warna hitam atau putih untuk posisi-posisi tertentu. Sebuah robot GG (gerak-dan-ganti) diletakkan pada salah satu ubin tertentu. Robot dapat bergerak ke arah depan, belakang, kiri, dan kanan, kemudian mengganti warna ubin yang baru saja dipijak. Tujuan dari robot adalah mengganti semua warna ubin menjadi putih. Makalah ini membahas syarat kondisi awal yang memungkinkan terselesaikannya masalah. Makalah ini juga membahas penyelesaian masalah menggunakan algoritme pencarian *A-star*.**

Kata kunci—A-star; robot gerak-dan-ganti; pencarian rute terpendek

I. Deskripsi Permasalahan

Terdapat dua hal yang perlu didefinisikan terlebih dahulu: papan berubin dan robot GG (gerak-dan-ganti). Kedua hal ini merupakan bagian dari masalah yang akan diselesaikan.

A. Papan Berubin

Papan berubin dapat dikatakan sebagai tempat dari permasalahan ini. Papan berubin terdiri dari m baris dan n kolom, m dan n adalah bilangan asli, yang selanjutnya disebut berukuran $m \times n$. Papan terdiri dari ubin yang dapat berwarna putih atau hitam (tidak keduanya). Warna dari ubin tersebut dapat direpresentasikan sebagai matriks *boolean* dengan nilai elemen ke-(i, j) menunjukkan warna dari ubin baris ke-i, kolom ke-j. Elemen tersebut selanjutnya disebut u_{ij}. Nilai 0 menunjukkan warna putih, sedangkan nilai 1 untuk warna hitam. Gambar 1 menunjukkan contoh papan berubin yang dimaksud.

B. Robot GG

Robot GG adalah robot yang dapat bergerak dari ubin ke ubin yang bersisian pada suatu papan berubin. Dengan kata lain, robot ini dapat bergerak ke atas, ke bawah, ke kiri, dan ke kanan selama terdapat ubin di sisi arah gerak. Ubin yang baru dipijak oleh robot akan segera diganti menjadi status yang berbeda, yaitu dari putih ke hitam atau dari hitam ke putih. Gambar 1 menunjukkan salah satu aksi dari robot GG. Persegi biru adalah robot GG.

Gambar 1. Ilustrasi Papan Berubin 2 × 2 dan Aksi Robot GG

Dalam matriks *boolean* papan berubin berukuran $m \times n$, misalkanlah posisi robot GG sekarang berada pada baris ke-a, kolom ke-b, selanjutnya disebut pada posisi (a, b), a dan b bilangan asli, $a \leq m$, $b \leq n$. Tabel I. menunjukkan definisi setiap instruksi dalam algoritme.

TABEL I. Definisi Instruksi dalam Algoritme

Nama Instruksi	Syarat	Algoritme
Atas	$a > 1$	$a \leftarrow a - 1$; $u_{ab} \leftarrow \neg u_{ab}$.
Bawah	$a < m$	$a \leftarrow a + 1$; $u_{ab} \leftarrow \neg u_{ab}$.
Kiri	$b > 1$	$b \leftarrow b - 1$; $u_{ab} \leftarrow \neg u_{ab}$.
Kanan	$b < n$	$b \leftarrow b + 1$; $u_{ab} \leftarrow \neg u_{ab}$.

.

C. Rumusan Masalah

Diberikan robot GG yang berada pada suatu papan berubin berukuran $m \times n$ pada posisi (a, b), a dan b bilangan asli, $a \leq m$, $b \leq n$. Sekumpulan aksi berurutan apa yang cocok untuk dilakukan oleh apa sajakah yang perlu dilakukan oleh robot GG sehingga membentuk rute terpendek dan setiap ubin pada papan berubin berwarna putih (bernilai 0)?

II. Algoritme Pencarian *A-Star*

A-star, atau umumnya ditulis sebagai A*, adalah instansi dari pencarian *best-first*, yaitu algoritma pencarian yang simpul yang ingin diperluas berdasarkan fungsi evaluasi tertentu, dilambangkan dengan *f(n)*. Untuk *A-star*, fungsi evaluasinya ditentukan dengan menggabungkan fungsi *g(n)*, yaitu fungsi untuk biaya rute yang telah ditempuh dari simpul awal ke simpul *n*. Selain itu, fungsi evaluasi juga digabungkan dengan fungsi *h(n)*, yaitu fungsi heuristik (perkiraan) untuk

menentukan biaya rute untuk meraih tujuan dari simpul *n*. Dari gabungan tersebut, *f(n)* terdefinisi sebagai berikut: [1]

$$f(n) = g(n) + h(n) \quad (1)$$

Untuk mendapatkan solusi optimal berupa rute terpendek, fungsi *h(n)* perlu konsisten. Fungsi tersebut konsisten jika untuk setiap simpul *n* dan setiap suksesor *n'* dari *n* dibangkitkan oleh aksi *a*, berlaku pertidaksamaan berikut: [1]

$$h(n) \leq c(n, a, n') + h(n'). \quad (2)$$

Bagian III.B akan menunjukkan fungsi heuristik yang dipilih konsisten.

Adapun algoritme dasar untuk *A-star* hampir sama dengan skema pencarian graf pada umumnya. Namun, *frontier*—tempat simpul yang siap diperluas [1]—menggunakan *priority queue* untuk memperluas simpul yang mendekati solusi terlebih dahulu. Berikut adalah algoritmenya yang dasarnya berasal dari pencarian graf [1]:

```
KAMUS
    frontier: PriorityQueue of Node { Node mengandung status untuk pencarian solusi, diurut berdasarkan nilai evaluasi (f(n)) terkecil }
    visited: Set { Node yang telah dikunjungi }
    solutionFound: boolean { jika bernilai true, solusi ditemukan }
    currentNode: Node
ALGORITME
    inisialisasi frontier sebagai PriorityQueue
    masukkan Node berstatus kondisi awal masalah ke frontier (enqueue)
    visited ← {}
    stopSearch ← false
    solutionFound ← Nil { belum ada }
    while (frontier tidak kosong) and (not solutionFound) do
        currentNode ← dequeue(frontier)
        if (currentNode adalah solusi) then
            solutionFound ← true
        else
            for (setiap Node newNode hasil perluasan currentNode)
                if (newNode not in visisted) then
                    frontier.enqueue(newNode)
                { else: lewati }
            { simpul hasil perluasan currentNode yang baru telah dimasukkan ke frontier }
        visited ← visited ∪ {currentNode}
    { frontier kosong or solutionFound }
    if solutionFound then
        → solusi yang dikandung currentNode dengan melakukan backtrack hingga ke akar (kondisi awal)
    else
        → Nil { tidak ada solusi }
```

Perhatikan bahwa pada algoritme tersebut, saat solusi telah ditemukan, simpul lain di dalam *frontier* tidak digunakan. Hal ini dilakukan mengingat solusi yang didapatkan memiliki nilai yang tidak kurang dari setiap simpul dalam *frontier* tersebut. Sifat tersebut berasal dari *priority queue*. Dengan kata lain, solusi tersebut dianggap sebagai solusi yang terbaik di samping terdapat kemungkinan ada solusi terbaik lainnya dengan nilai evaluasi yang sama.

III. Perancangan Algoritme

Penyelesaian masalah dari makalah ini menggunakan algoritme pencarian *A*-star. Hal yang dirancang dalam algoritme ini adalah fungsi evaluasinya yang terdiri dari biaya rute yang telah ditempuh ditambah dengan perkiraan biaya rute ke tujuan. Tentunya, biaya rute yang telah ditempuh dipengaruhi oleh biaya aksi. Oleh karena itu, terdapat dua hal yang dirancang: biaya aksi dan fungsi heuristik.

A. Biaya Aksi

Untuk setiap aksi yang didefinisikan pada Tabel I., biayanya adalah 1. Biaya aksi ini juga berkaitan dengan jarak yang ditempuh, yaitu hanya satu ubin untuk setiap aksi.

B. Fungsi Heuristik

Nilai ini didapatkan dengan memerhatikan jarak terjauh robot GG dengan salah satu ubin hitam. Jarak yang digunakan adalah jarak manhattan, yaitu jumlah antara selisih jarak antarkolom dengan selisih jarak antarbaris. [1]

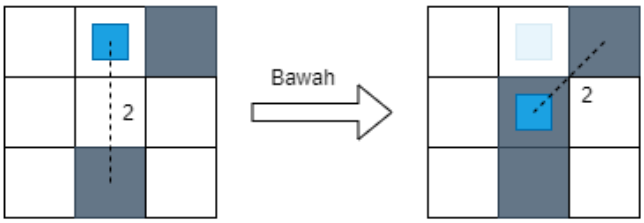
Didapatkan fungsi heuristiknya adalah

$$h(n) = max(|a \text{ - baris ubin hitam}| + |b \text{ - kolom ubin hitam}|) \quad (3)$$

Konsistensi dari fungsi heurisitik ini dapat dibuktikan dengan mudah. Perhatikan bahwa untuk setiap kolom ubin hitam, perubahan jaraknya dapat lebih jauh atau lebih dekat 1 ubin. Maka, jika hanya meninjau jarak satu ubin hitam terjauh dari robot GG, perilakunya sama. Untuk ubin hitam terjauh yang lebih dari satu, terdapat kasus tertentu yang membuat jarak sebelum dan sesudah bergerak tidak berubah, seperti pada gambar 2. Maka, dapat dikatakan

$$-1 \leq h(n') - h(n) \leq 1. \quad (4)$$

Sifat ini memenuhi pertidaksamaan (2) yang merupakan syarat fungsi heuristik konsisten.



Gambar 2. Contoh Jarak Maksimum yang Tidak Berubah Sebelum dan Setelah Aksi

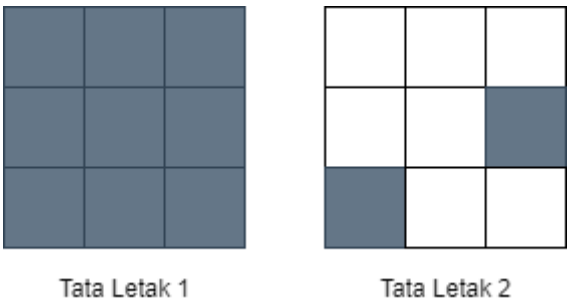
IV. Pengujian

A. *Urutan Priotitas Aksi yang Digunakan*

Untuk memperhalus gerakan robot GG, aksi yang dipilih terlebih dahulu adalah aksi dengan arah yang sama. Jika tidak ada atau untuk aksi alternatif lain, urutan prioritas aksi dimulai dari yang didahulukan adalah Atas, Bawah, Kiri, dan Kanan. Selain itu, hal ini juga dilakukan pada gerakan pertama mengingat tidak ada gerakan yang dilakukan sebelumnya.

B. *Kasus Uji yang Digunakan*

Terdapat dua hal yang diubah untuk membentuk kasus: tata letak ubin hitam dan posisi awal robot GG. Untuk tata letak, ada ubin-ubin yang semuanya hitam, dan ada juga yang hanya beberapa dan tersebar. Gambar 3 menunjukkan dua tata letak ubin hitam yang digunakan. Untuk posisi awal robot GG, terdapat dua posisi yang dipilih: (1, 1) dan (1, 2). Dari kedua hal tersebut, terbentuklah empat kasus yang akan digunakan.



Gambar 3 Tata Letak Ubin-Ubin Hitam untuk Kasus Uji yang Digunakan

C. *Hasil Implementasi*

Berikut adalah tabel hasil implementasi dari keempat kasus yang telah ditentukan. Gambar hasil implementasi dilampirkan pada bagian Lampiran. Untuk penyederhanaan, perhatikan bahwa gambar yang disediakan menampilkan simpul yang menuju ke solusi saja. Hasil yang lengkap dilampirkan dalam tautan https://github.com/AbdiHaryadi/robot-GG-simulation, folder result.

TABEL II. Hasil Implementasi

Tata Letak Awal	Posisi Awal Robot	Banyak Simpul Ekspansi	Solusi	Biaya solusi
Ubin Hitam	**GG**			
Tata Letak 1	(1, 1)	908	Bawah, Atas, Kanan, Kanan, Bawah, Bawah, Kiri, Kiri, Atas, Kanan, Kiri	11
Tata Letak 1	(1, 2)	292	Kiri, Kanan, Kanan, Bawah, Bawah, Kiri, Kiri, Atas, Kanan	9
Tata Letak 2	(1, 1)	34	Bawah, Bawah, Atas, Kanan, Kanan, Kiri	6
Tata letak 2	(1, 2)	43	Bawah, Bawah, Kiri, Kanan, Atas, Kanan	6

V. Analisis

Dari hasil implementasi, terlihat bahwa banyak simpul yang diperluas sangat jauh dengan biaya solusi. Kemungkinan besar hal tersebut terjadi disebabkan oleh fungsi heuristik yang digunakan kurang efisien. Penulis menyatakan kesulitan dalam penentuan fungsi heuristik yang lebih efisien untuk rute terpendek adalah jaminan konsisten. Dari fungsi tersebut, banyak simpul dengan nilai evaluasi yang sama. Akibatnya, performanya seperti BFS (*breadth-first-search*) yang memperluas simpul-simpul dalam ukuran eksponensial. Hal ini dapat didukung dengan perbandingan lurus antara nilai logaritma banyak simpul yang diperluas dengan biaya solusi. Rasio yang mendekati antara kedua nilai tersebut adalah 0,6. Dari keburukan tersebut, ukuran dari papan berubin beserta banyak ubin hitam sebaiknya tidak terlalu besar karena performanya sangat lama. Dari yang dialami penulis, untuk mendapatkan solusi dari papan yang semuanya terdiri dari ubin hitam pada kondisi awal dengan ukuran 5 × 5 dengan program penulis, waktunya melebihi 1 jam dan akhirnya dihentikan.

Pada kasus papan berubin yang pada awalnya semua hitam, terdapat perbedaan yang signifikan saat posisi robot GG diubah antara (1, 1) dengan (1, 2). Sebaliknya, tidak ada perubahan yang signifikan pada papan berubin yang ubin-ubin hitamnya tersebar. Selain itu, kedua kasus tata letak ubin hitam menunjukkan ubin hitam yang lebih banyak cenderung memerlukan waktu dan gerakan yang lebih banyak.

Pada kasus tata letak 1 dengan posisi awal (1, 1), terdapat hasil yang dalam arah dikatakan kurang efisien. Perhatikan bahwa solusi dimulai dengan aksi Bawah lalu Atas, kemudian dilanjutkan dengan mengubah semua ubin pinggiran. Arahnya akan lebih efisien jika menggunakan aksi Bawah dua kali pada awal. Kedua alternatif mengarahkan kepada simpul dengan iterasi ke-289, namun dengan posisi robot GG yang berbeda. Pada kondisi tersebut, robot GG cukup membuat ubin hitam yang menghubungkan ubin hitam lainnya sehingga membentuk sebuah rute. Kemudian. robot GG bergerak dan hanya mengganti ubin hitam menjadi ubin putih pada rute tersebut seperti biasa. Hal yang menarik adalah terjadi kenaikan nomor iterasi pada kondisi tersebut, yaitu 289 ke 843. Penuis berpendapat beberapa faktornya adalah sangat banyak kondisi

yang mengarahkan pada nilai 10, atau masalah simetris. Perhatikan bahwa saat menyelesaikan persoalan dengan nilai yang sama, semakin tinggi kemungkinan bentuk simetrinya, semakin banyak perluasan simpul yang ditangani. Kesamaan nilainya membuat simpul-simpul tersebut ditangani seperti BFS—tidak ada yang dipangkas. Berbeda dengan kondisi dari posisi awal (1, 1), dengan kasus tata letak yang sama, posisi awal (1, 2) menghasilkan gerakan yang lebih sedikit. Rute berubin hitam telah dideteksi oleh robot GG sehingga tinggal bergerak dan mengganti ubin-ubin hitam tersebut seperti biasa, tidak perlu berurusan dengan ubin putih.

Pada kasus tata letak 2, strategi dari robot GG terlihat sama seperti sebelumnya, yaitu membuat rute berubin hitam yang menghubungkan ubin hitam yang terpisah. Untuk posisi awal (1, 1), hal tersebut dilakukan dua kali. Di sisi lain, untuk posisi awal (1, 2), robot GG dapat menghubuugkan dua ubin hitam sekaligus sebagai satu jalur. Posisi awal (1, 1) sebenarnya memungkinkan untuk menghubungkan dua ubin yang sama, yaitu melakukan aksi Bawah dan Kanan. Akan tetapi, itu tidak menjadi solusi terbaik karena aksi yang diutamakan adalah aksi dengan arah yang sama dengan sebelumnya.

Penemuan yang menarik dari hasil implementasi ini adalah jika robot GG berada pada posisi genap dan ingin mengubah ubin hitam dengan banyak yang genap (seperti pada tata letak 1), posisi akhir robot GG adalah posisi genap juga. Posisi yang genap adalah posisi yang membuat jumlah baris dan kolom tempat robot GG berada adalah genap. Jika banyak ubin hitam adalah ganjil, posisi awal yang genap berakhir pada posisi yang ganjil, dan sebaliknya. Hal tersebut terjadi karena setiap robot GG bergerak, banyak ubin hitam dapat berkurang atau bertambah tepat satu. Akibatnya, jika gerakan optimal memiliki biaya solusi lebih dari banyak ubin hitam pada kondisi awal, selisihnya pasti genap dan hanya digunakan untuk mengganti ubin putih menjadi ubin hitam dan mengubahnya kembali. Akibat dari sifat ini adalah tidak semua kondisi awal memiliki solusi ke kondisi akhir tertentu (termasuk posisi robot GG).

Kesimetrisan dari kondisi awal sangat penting untuk dioptimasi agar tidak terjadi perhitungan yang redundan. Adapun salah satu solusi yang ditawarkan penulis adalah memperluas simpul dengan mengabaikan simpul yang sama dengan bentuk simetri simpul hasil perluasan lainnya. Hal itu juga perlu diterapkan untuk simpul yang telah dikunjungi. Sebagai contoh, pada kasus tata letak 1 dengan posisi awal (1, 1), aksi Kanan sebagai aksi pertama tidak perlu ditinjau lagi karena telah diwakili oleh aksi Bawah. Setelah aksi Bawah pertama, aksi Bawah kedua juga tidak perlu ditinjau lagi karena sudah diwakili oleh aksi Atas sebagai aksi kedua. Sebagai pendukung, perhatikan bahwa tata letak 2 yang asimetris memilliki simpul ekspansi yang lebih sedikit daripada tata letak 1.

VI. Kesimpulan dan Penggunaan Lebih Lanjut

Algoritme pencarian *A-star* dapat digunakan untuk masalah robot GG. Akan tetapi, fungsi heuristik perlu dirancang dengan lebih baik lagi untuk menciptakan performa yang lebih efisien. Solusi dari permasalahan ini dapat dibuat lebih efisien lagi dan bervariatif. Oleh karena itu, diharapkan masalah ini dapat menjadi *open-problem* bagi pembaca yang tertarik untuk menyelesaikannya dengan cara yang berbeda.

Perilaku robot GG dapat digunakan sebagai mekanisme baru dalam gim, terutama untuk gim teka-teki. Kesulitan dapat dipertimbangkan dengan menentukan rute terpendek dari penyelesaian suatu tata letak ubin-ubin hitam, baik jarak maupun kerumitan dari rute tersebut. Selain itu, permasalahan yang dibahas oleh makalah ini tidak terlalu kompleks. Masalah ini diharapkan dapat berkembang menjadi masalah yang bervariasi seperti keberadaan dua robot GG, dinding, dan lain-lain. Representasi dari masalah ini juga dapat dibalik, seperti diberikan papan berubin berukuran tertentu yang mula-mula terdiri dari ubin putih, tentnukan rute terpendek sehingga papan berubin menggambarkan suatu pola atau gambar tertentu. Hal itu tentunya menambah ketertarikan dari gim teka-teki yang menerapkan solusi dari masalah ini.

Tautan Video YouTube

Tautan: https://youtu.be/FynwwRJYgZY

Video ini berisi algoritme pencarian *A-star* terhadap permasalahan rute terpendek untuk robot GG beserta implementasinya.

Ucapan Terima Kasih

Permasalahan ini terinspirasi oleh teka-teki dari gim catur Nintendo DS. Tujuannya adalah menggerakkan komponen catur yang disediakan seperti kuda hingga semua ubin dari papan catur tersebut sama. Sayangnya, penulis tidak mengetahui dengan jelas nama gim dan pembuatnya.

Penulis juga berterima kasih kepada dosen-dosen pengampu mata kuliah IF2211 Strategi Algoritma Semester 2 Tahun 2020/2021, terutama Bapak Dwi Hendratmo Widyantoro yang telah memberikan bimbingan selama satu semester di kelas K-3. Selain itu, penulis berterima kasih kepada keluarga dan teman-teman yang telah memberikan umpan balik dan dukungan sehingga saya dapat menyelesaikan makalah ini dengan baik. Penulis juga menyadari bahwa konten dalam makalah ini masih kurang karena keterbatasan waktu ataupun wawasan dari penulis itu sendiri.

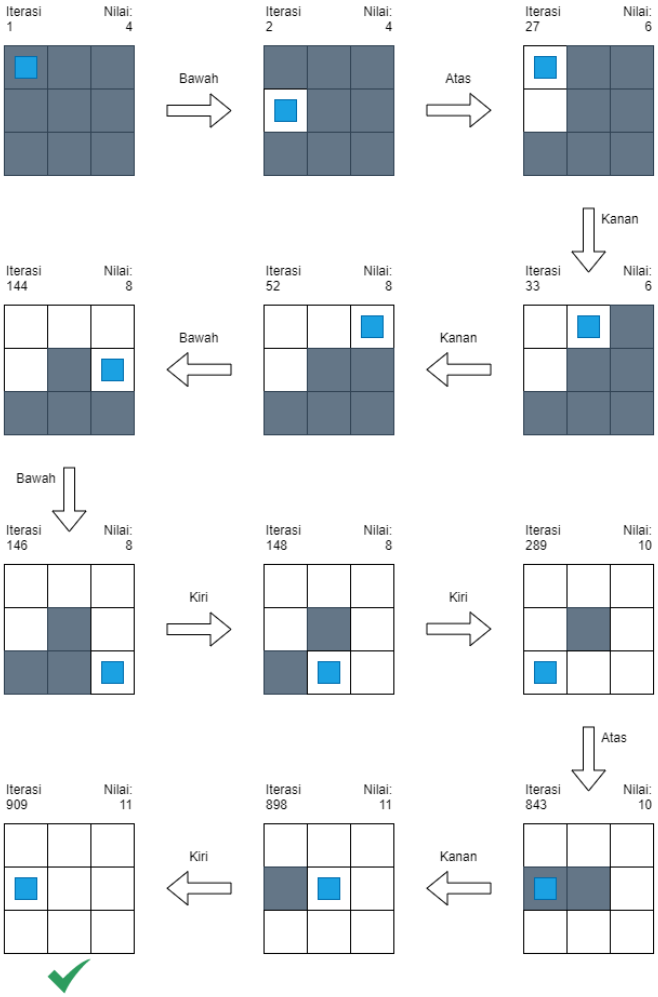
Pustaka

[1] S. Russel dan P. Norvig, Artificial Intelligence: A Modern Approach, edisi ketiga. New Jersey: Prentice Hall, 2010, hlm. 75-103

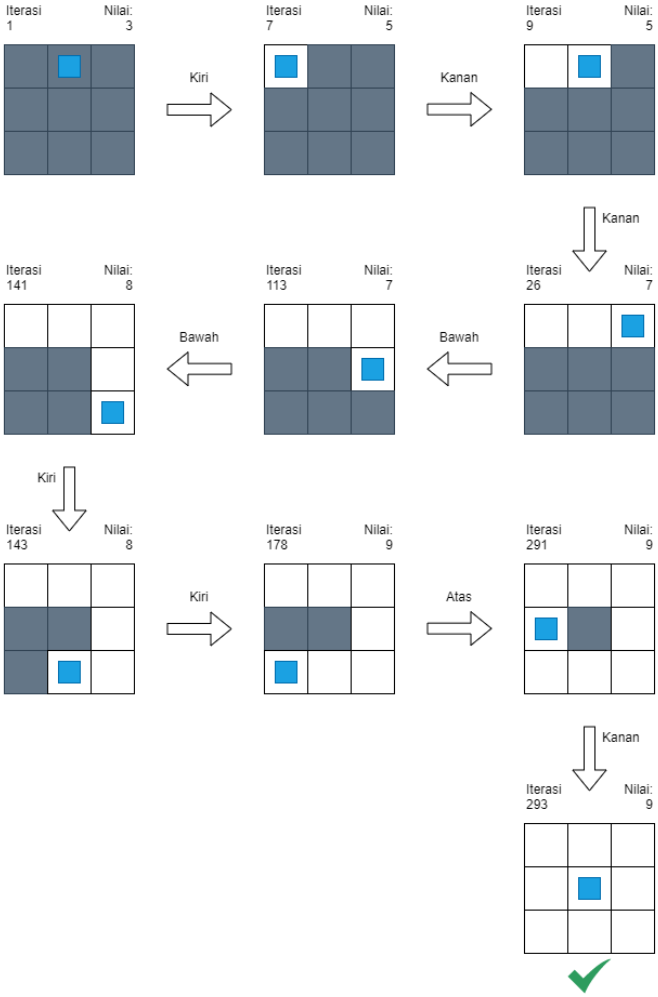
Lampiran

Lampiran ini berisi gambar solusi dari keempat kasus uji yang digunakan untuk mendukung bagian IV.C. Selain solusi, terdapat nomor iterasi simpul untuk status tersebut diperluas. Nilai untuk setiap simpul yang ditampilkan juga diperlihatkan.

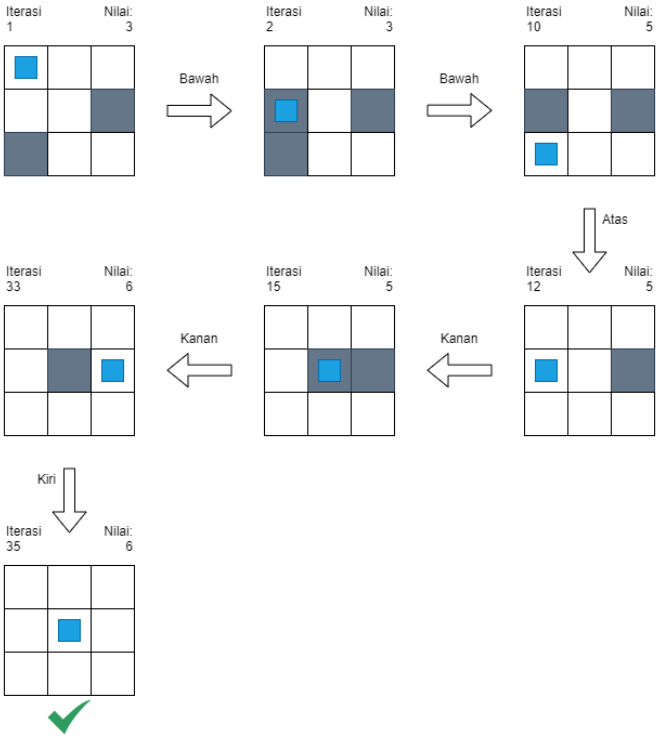
A. Solusi Tata letak 1, Posisi Awal (1, 1)



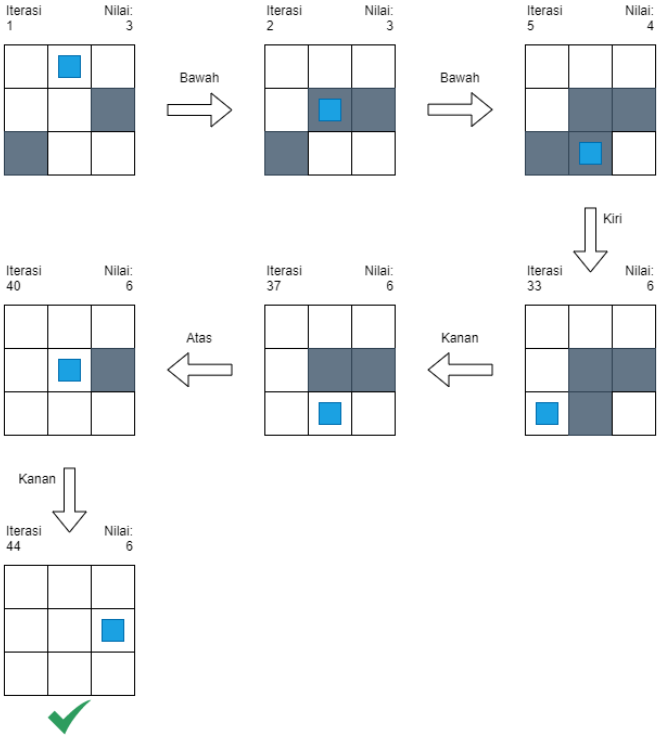
B. Solusi Tata letak 1, Posisi Awal (1, 2)



C. Solusi Tata letak 2, Posisi Awal (1, 1)



D. Solusi Tata letak 2, Posisi Awal (1, 2)



PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2021

M. Abdi Haryadi. H
(13519156)