BAB III

METODE PENELITIAN

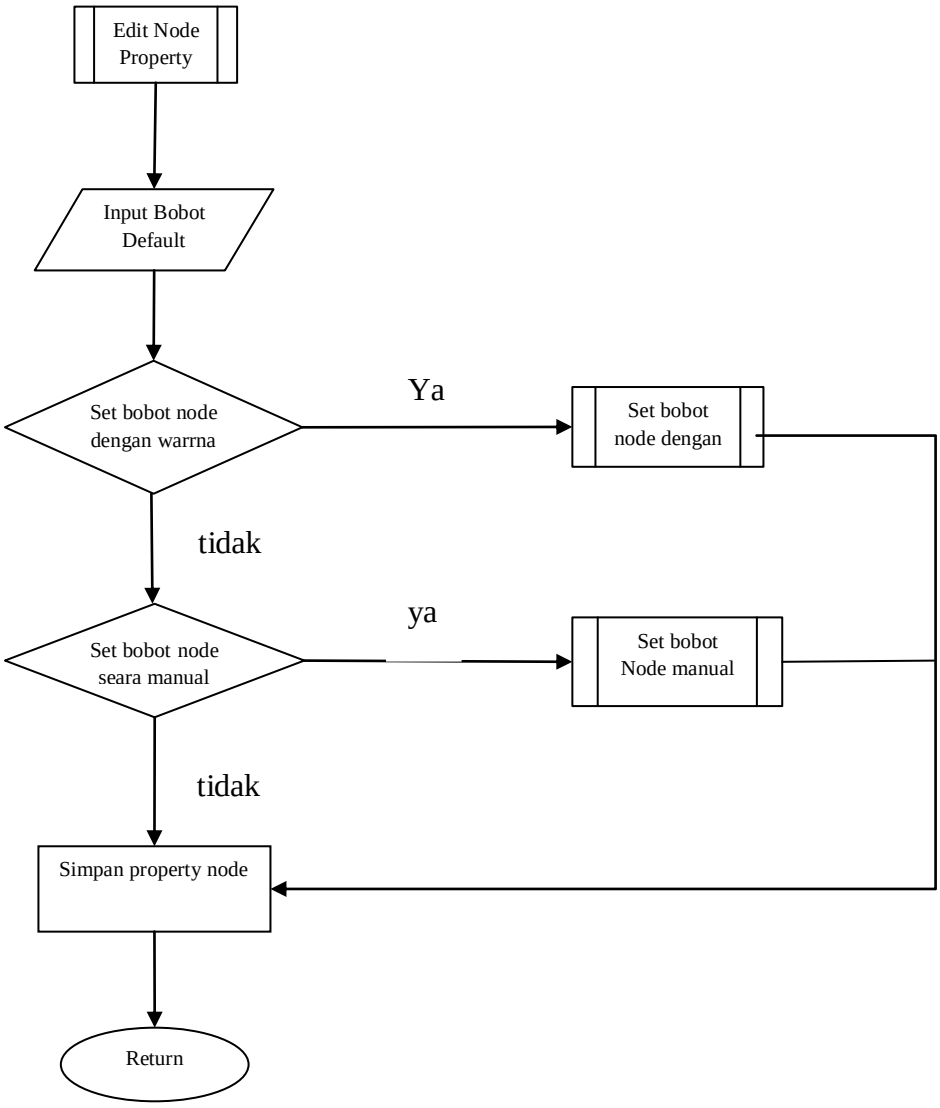
3.1 Analisis dan Desain

3.1.1 Metode

Media yang digunakan dalam aplikasi ini adalah berbasis java. *Output* dari implementasi, berupa informasi simulasi peta rute jalan yang dinginkan beserta jalur terdekat yang ditempuh.

3.1.2 Pengaturan Bobot Node Peta

Pengaturan bobot *node-node* pada peta dapat dikatakan adalah salah satu hal paling penting dalam aplikasi ini. Pemberian bobot ini sangat berpengaruh pada ketepatan pencarian rute optimal pada peta yang bersangkutan. Pada pengaturan bobot *node* peta ini, pengguna dapat mengatur bobot *default node* peta. Untuk jalan yang dapat dilalui pada peta, sebaiknya bobotnya diset kecil. Sedangkan untuk daerah yang tidak dapat dilewati, bobot diset besar. Selanjutnya pengguna dapat mengatur pemberian bobot dan pengaturan properti *node* lainnya kepada *node-node* yang dipilihnya secara manual. Pengaturan properti *node* lain meliputi pemberian nama jalan, kondisi jalan dan arah jalan. Diagram alir untuk pengaturan bobot dan properti lain *node* pada peta dapat dilihat pada gambar di bawah.

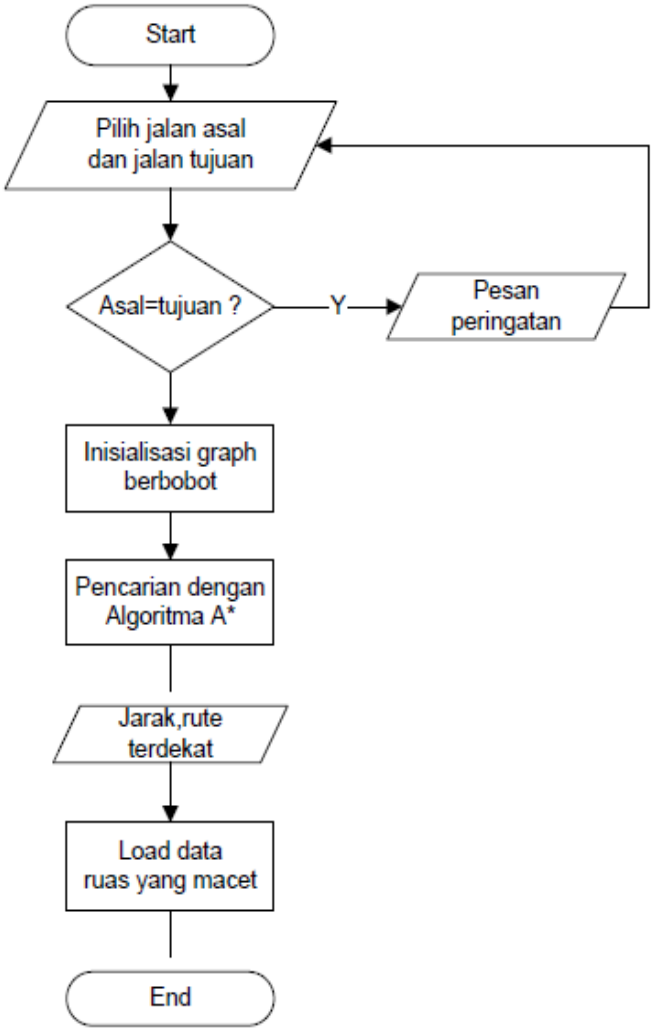


Gambar 3.1 Diagram Pengaturan Bobot Node Peta

Setelah dilakukan pengaturan bobot *node* pada peta, barulah aplikasi dapat menjalankan algoritma pencarian rute A*.

3.2 Alur Sistem

Diagram alir yang memperlihatkan tahapan pencarian rute terpendek dengan Algorima A* ditunjukkan oleh gambar di bawah ini.



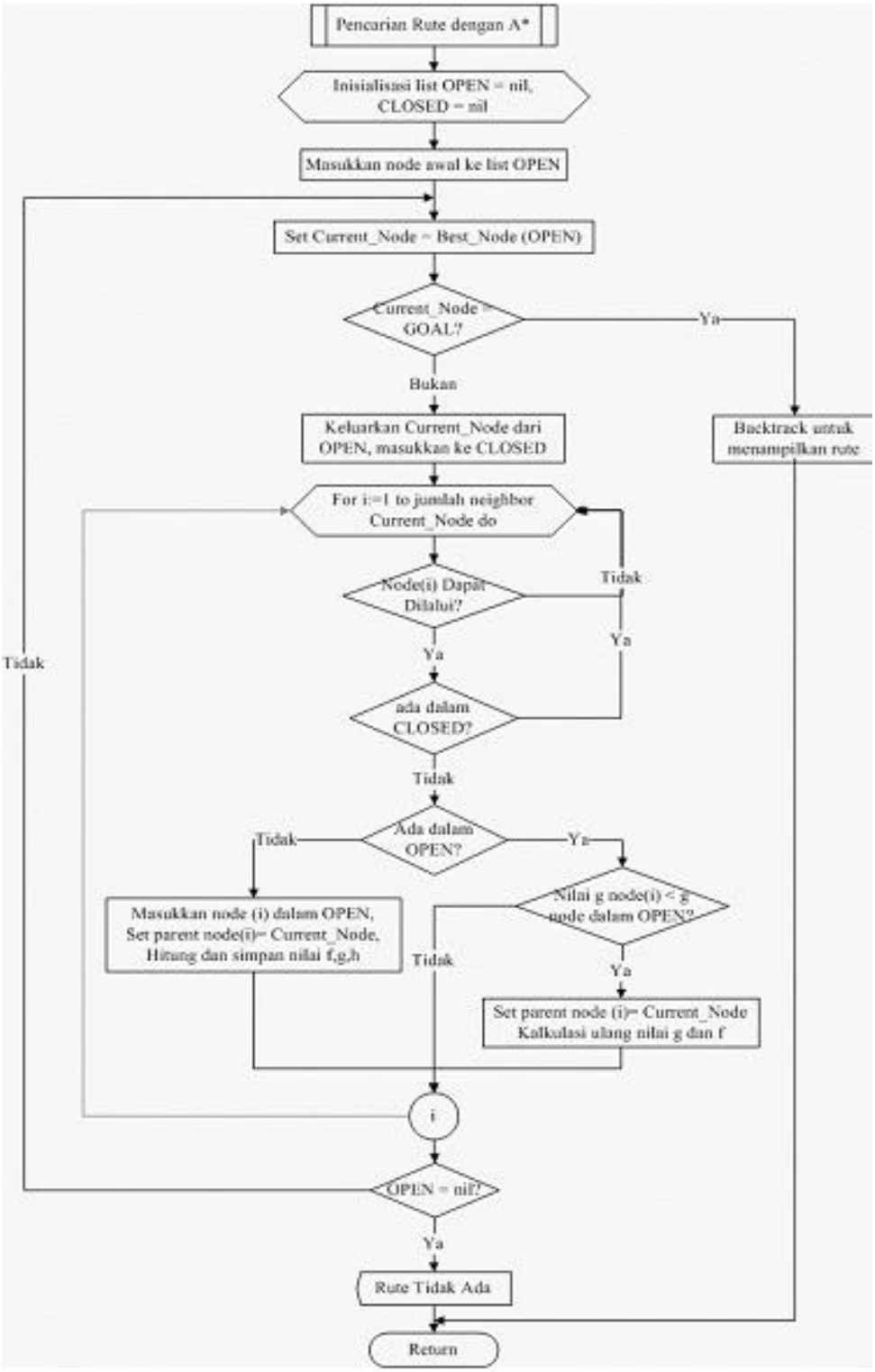
Gambar 3.2 Diagram alir pencarian rute jalan terdekat.

Penjelasan diagram alir pencarian rute jalan terdekat diatas adalah sebagai berikut:

1. *User* menentukan jalan asal dan jalan tujuan sebagai inputan dalam pencarian dengan menggunakan algortima *A**. Ruas jalan asal dan tujuan tidak boleh sama, jika sama akan muncul pesan *error*.
2. *Graph* berbobot disini dipresentasikan sebagai struktur data bertipe *array* 2 dimensi (Matrik ketetanggaan) yang menggambarkan hubungan *node-node* yang saling berkaitan yang nantinya digunakan sebagai inputan pencarian dengan algortima *A**. Dalam permasalahan kali ini bobot dari *graph* adalah jarak antara *node-node* yang

saling berhubungan Semua data *graph* berbobot (*node*, bobot / jarak antar *node*) ini didapat dari data yang sudah tersimpan di dalam *database*.

3. Melakuan pencarian dengan algoritma A*. Inputan dari pencarian dengan algortima A* adalah *graph* berbobot, *node* asal, *node* tujuan. Dan *output* nya atau hasil pencarian adalah jarak dan rute jalan terdekat.
4. Data ruas yang macet akan digunakan sebagai pertimbangan dalam menentukan rute solusi. Jika didalam hasil pencarian rute terdekat terdapat ruas macet, maka program otomatis akan memberikan rute solusi untuk menghindari kemacetan. Dengan melakukan pencarian ulang, hanya saja yang membedakan adalah *graph* berbobotnya. *Node-node* yang terdapat pada ruas kemacetan akan dimasukkan ke dalam *graph* berbobot.
5. Hasil pencarian dan ruas-ruas kemacetan akan ditampilkan pada peta.



Gambar 3.3 Diagram alir Algoritma A*.

Algoritma A* akan memulai pencarian dengan *looping* pada semua *node/vertex* yang terhubung dengan *node* di posisi sekarang (berawal dari posisi asal). Kemudian cari bobot/jarak tekecil dari *node-node* yang terhubung, dimana *node* yang terhubung belum pernah dikunjungi atau bobot yang baru lebih kecil dari bobot yang lama. kemudian

menyimpan jarak/bobot dan rute pada *array*. Setiap *node/vertex* yang telah dipilih akan disimpan guna menghindari *back step* (kembali ke *node* sebelumnya). Dilakukan berulang-ulang hingga semua *node* yang ada telah dipilih/dikunjungi semua. Pada kasus kali ini pencarian akan dilakukan ke semua pasang *node*/simpul/vertex (*all pairs shortest path*), jadi semua *node* akan diuji sehingga benar-benar akan mendapatkan rute dan jarak terpendek.