Tugas Week 5

Nama : Arya Putra Siahaan

Kelas : TK-45-GAB9

NIM : 110321309

Dijkstra Planner

Kode ini menerapkan algoritma Dijkstra dalam Python untuk menemukan jalur terpendek antara dua simpul pada graf berbobot. Fungsi `dijkstra` memiliki tiga parameter: `graph` (graf yang menyimpan simpul dan tetangganya dengan biaya perjalanan), `start` (simpul awal), dan `goal` (simpul tujuan). Algoritma ini menggunakan antrian prioritas (min-heap) untuk mengambil simpul dengan biaya terendah secara efisien. Pada setiap langkah, simpul terendah diproses, dan tetangganya diperiksa untuk memperbarui jarak. Jika jarak baru lebih pendek dari sebelumnya, jarak tersebut diperbarui dan simpul tetangga dimasukkan ke antrian. Proses ini berlanjut sampai simpul tujuan tercapai atau tidak ada simpul lain dalam antrian. Dalam graf contoh, jalur terpendek dari 'A' ke 'D' adalah `['A', 'B', 'C', 'D']` dengan biaya total 4. Algoritma ini memiliki kompleksitas waktu $O((V + E) \log V)$ dan ruang $O(V)$, dengan V sebagai jumlah simpul dan E sebagai jumlah sisi.

Cell Decomposition

Kode ini menggunakan algoritma dasar cell decomposition untuk perencanaan jalur, di mana area dengan rintangan dibagi menjadi beberapa sel bebas rintangan. Fungsi `cell_decomposition` menerima `obstacles` (daftar rintangan), `start` (titik awal), dan `goal` (titik tujuan). Fungsi `create_cells` membuat sel bebas rintangan berdasarkan rintangan yang ada, tetapi dalam contoh ini sel masih dibuat statis dan tidak terpetakan otomatis oleh rintangan. Fungsi `find_path_through_cells` menemukan jalur melalui sel dari titik awal ke tujuan. Pada contoh ini, rintangan didefinisikan di antara `(1, 1)` dan `(2, 2)`, dan jalur yang dihasilkan adalah dari `(0, 0)` ke `(3, 3)`. Kode ini masih dasar, karena sel dan jalur dibuat secara statis tanpa pencarian jalur dinamis.

A Planner

Kode ini mengimplementasikan algoritma A untuk menemukan jalur terpendek di grid dua dimensi dengan rintangan. Fungsi `a_star` menggunakan `open_list` (untuk simpul yang perlu diperiksa) dan `closed_list` (untuk simpul yang sudah diperiksa). Algoritma memilih simpul dengan nilai `f` terendah, yang merupakan penjumlahan dari `g` (jarak dari titik awal) dan `heuristic` (estimasi jarak ke tujuan menggunakan jarak Manhattan). Jika simpul saat ini adalah tujuan, algoritma membangun jalur dari tujuan ke awal dengan melacak asal setiap simpul di `came_from`, kemudian mengembalikan jalur tersebut. Fungsi `get_neighbors` mendapatkan tetangga dari simpul, memilih hanya simpul tanpa rintangan (ditandai `0`). Pada grid contoh, rintangan ditandai `1`, dan algoritma mencari jalur dari `(0, 0)` ke `(3, 3)`, yang kemudian dicetak.