TALLER N°1

Rush Hour

usach

Taller de programación 2-2023

Fecha: 11-10-2023

Autor: Fabián Lizama González

TALLER N°1

Rush Hour

Explicación breve del algoritmo

Para resolver este problema se utiliza el algoritmo A*, que en palabras simples sirve para resolver cualquier problema que requiera una serie de pasos, realizando todos los pasos posibles para un estado en específico y sigue ejecutando el estado “más favorable”, a este concepto se le llama “heurística”, al poder reconocer qué casos son más favorables que otros el algoritmo encuentra la solución de manera óptima en la mayoría de los casos.

Heurísticas o técnicas utilizadas

La heurística utilizada para el algoritmo A* en este caso es la distancia del auto rojo a la salida más la cantidad de autos que bloquean la salida, más la cantidad de autos mínima que bloquean por arriba o por abajo a los autos que bloquean la salida, al probar con varias heurísticas se escogió esta porque daba los mejores resultados.

Funcionamiento del programa

El programa lee los archivos de entrada, guarda una lista de autos y una matriz donde se representa el tablero (por términos de optimización se necesitaban las dos representaciones), luego genera el estado inicial y lo agrega a una cola de prioridad de estados abiertos, mientras queden estados en la cola de abiertos el programa saca el primer estado y opera todas las posibilidades de un solo movimiento para este estado, luego calcula la heurística más el coste (cantidad de movimientos para llegar a ese estado), y se ingresan en la cola de prioridad, el proceso se repite hasta que encuentre la solución o no queden estados en la cola de abiertos.

Aspectos de implementación y eficiencia

El código es eficiente principalmente por tres puntos:

- Heurística: La heurística utilizada calcula de manera eficiente qué estados son más favorables que otros.
- Heap: La estructura de datos utilizada para guardar los estados abiertos es un heap mínimo, esto hace que la inserción y obtención de elementos tenga un coste de O(logn).
- Stack: Para imprimir la ruta de la solución se utiliza un stack donde los algoritmos de inserción y obtención de elementos tienen un coste de O(1).

La suma de estos tres factores genera que el código sea eficiente y genere soluciones en tiempos óptimos.

Ejecución del código

Para el desarrollo del código se utilizaron las librerías *iostream* para el manejo de inputs y outputs, *string* para manejo de cadenas de carácteres y *chrono* para medir el tiempo de ejecución.

Para correr el código se debe estar en un sistema operativo Linux, situarse dentro de la carpeta que contiene el código y ejecutar el siguiente comando:

make

Esto compilará el código, para luego ejecutarlo escriba el siguiente comando:

./main

Se ejecutará el programa y se verá el siguiente mensaje en consola:

Seleccione uno de los 9 tableros a resolver ingresando un número del 1 al 9:

Luego ingrese el número del tablero a resolver y el programa se ejecutará sin problemas.

Si ingresa cualquier cosa que no sea un número del 1 al 9 el programa no funcionará.

Bibliografía

De Batz De Trenquelleon, G., Choukarah, A. H., Roucairol, M., Addoum, M., & Cazenave, T. (2023). Procedural generation of rush hour levels. En Lecture Notes in Computer Science (pp. 181-190). https://doi.org/10.1007/978-3-031-34017-8_15

LaNsHoR. (2017, 30 julio). Pathfinding: A* - Lambda Lab 85. lambda lab 85. https://www.lanshor.com/pathfinding-a-estrella/

Michael Fogleman. (s. f.). https://www.michaelfogleman.com/rush/