PROYECTO DE LA ASIGNATURA DE INTELIGENCIA ARTIFICIAL Y SISTEMAS INTELIGENTES 2023/2024

Alumnos:

- José Manuel de Torres Domínguez - Manuel Alonso González - David de Miguel Palomino -

1. Descripción de la instalación, interfaz y manejo de la práctica

Requisitos para compilar el programa:

- Compilador compatible con el estándar ISO C++ 14
- Un computador con un compilador de C/C++
- Electricidad
- Motivación (opcional)
- Enlace de amigo de Clash Royale (enviar a josemanueldtd@gmail.com)

Instrucciones para ejecutar el proyecto:

Dos opciones:

a) Ejecutando el binario precompilado:

Dentro de la carpeta del proyecto encontrará un binario precompilado. Este binario le permitirá ejecutar el proyecto sin necesidad de compilar el programa, pero puede que sufra de problemas de incompatibilidad si la arquitectura de la computadora y el binario difieren sustancialmente.

Para ejecutar el programa, abra una ventana del terminal y sitúe la ruta seleccionada a la de la carpeta del proyecto. A continuación:

1º) De permisos de ejecución al binario:

```
chmod +x ./MiniPacman
```

2º) Ejecute el binario:

```
./MiniPacman
```

b) Compilar un nuevo binario y ejecutarlo:

Para compilar el programa, abra una ventana del terminal y sitúe la ruta seleccionada a la de la carpeta del proyecto. A continuación:

1º) Borre el binario precompilado:

```
rm MiniPacman
```

2º) Ejecute el script automatizado para compilar el programa:

```
./compilar
```

3º) Ejecute el binario:

```
./MiniPacman
```

La interfaz y el manejo del proyecto:

Al comenzar el programa, será recibido por un menú principal donde se le pedirá que introduzca, o bien un número del 1-10 para cargar uno de los archivos MINIPAC1-10.txt, o bien el nombre del archivo (sin la extensión .txt) para cargarlo.

Solo se podrán cargar los archivos que estén en la misma carpeta de ejecución que el binario. El programa no maneja los errores en la sintaxis de los archivos, por lo que se cerrará inmediatamente luego de encontrar un problema con la sintaxis del archivo que se ha cargado.

Una vez cargado el archivo, se mostrará una representación gráfica del tablero del laberinto, así como la ubicación inicial del PacMan, el Fantasma y la salida. A continuación, se ejecutarán automáticamente los algoritmos implementados y especificados en el apartado 3.

Podrá ver el nombre de cada uno de los algoritmos, así como su estado en la ejecución (o bien FINALIZADO (llegó a la salida) o FRACASO (no llegó a la salida)), los movimientos de PacMan, el Fantasma, el tiempo de ejecución y la cantidad de nodos generados.

Podrá observar un resumen de los resultados obtenidos con los 10 archivos MINIPAC y dos adicionales en el apartado 4 de este documento.

Para finalizar el programa, escriba “exit” o pulse la combinación de teclas ALT+F4.

2. Métodos de desarrollo del software

El proyecto se ha desarrollado con código propietario, desarrollado con la ayuda auxiliar de algunas funciones o ejemplos de código en Internet limitadas a búsquedas de las cuales solo hemos extraído información para realizar el código oportuno.

3. Descripción de los algoritmos de resolución usados en la solución

Se han desarrollado los algoritmos de Escalada Simple, algoritmo de Primer Mejor y algoritmo A*. La función heurística empleada en todos los algoritmos ha sido la distancia en línea recta desde el PacMan a la salida. Para obtener un resultado con estos algoritmos se ha empleado el uso de nodos como elemento auxiliar, cuyo conjunto consecutivo lineal forma la solución. En cada nodo almacenamos la información necesaria del tablero tal que cada nodo sea una instantánea del estado de la ejecución. Los algoritmos detallados son los siguientes:

- **Escalada simple**: para este algoritmo se ha utilizado una determinada función heurística con el objetivo de que el PacMan se mueva a una determinada celda u otra, teniendo en cuenta que si en el siguiente nodo que se genere la función heurística nos da un valor menor al que tenemos en el nodo actual, se continua por esa celda. Por tanto, si no encontramos un valor menor que el devuelto por la función heurística o se da el caso de que no podemos movernos a ninguna otra posición porque hay un obstáculo, el algoritmo termina en ese mismo instante. Cada vez que se avanza una posición la dirección que toma el PacMan se guarda en la lista de movimientos y se genera un nodo nuevo. Una vez salimos del bucle que comprueba las posibles direcciones permitidas para el PacMan desde su posición actual, se verifica sí el PacMan está en la salida en ese momento, comprobando que las coordenadas del PacMan son las mismas que la salida y entonces se establecerá el camino actual como válido.

- **Primero mejor**: en este caso, se ha llevado a cabo el desarrollo del algoritmo utilizando dos listas para almacenar los nodos abiertos y cerrados. De esta manera, el algoritmo de primero mejor funciona de la siguiente forma: primero se cierra el nodo original, guardando en la lista de nodos abiertos todas las posiciones a las que puede moverse el PacMan. Una vez hecho esto, ordenamos la lista de nodos abiertos y eliminamos aquellos nodos que estén repetidos. De esa lista, cogemos el primero y lo desarrollaremos en el siguiente paso de la ejecución repitiendo todo el proceso. En el momento en el que algún nodo llegue a la salida, se dará por concluido el algoritmo y se mostrará el camino que ha realizado PacMan para llegar a la salida. En este algoritmo, con la finalidad de que no se produzcan bucles entre varias casillas, hemos incluido la condición de que el PacMan no podrá volver a una posición en la que ya ha estado si el fantasma no se encuentra a 10 casillas de distancia de donde estaba con respecto a la primera vez que pasó el PacMan por la casilla en cuestión. Con esta medida, se evitan bucles en el movimiento del PacMan, pues estamos obligando a PacMan a recorrer tablero que aún no ha recorrido con la esperanza de que encuentre la salida. En caso de encontrarnos ante un tablero sin salida, el algoritmo lo detectará puesto que llegará un momento en el que no haya más nodos en la lista de nodos abiertos, momento en el cual se concluirá la ejecución del algoritmo.

- **A***: la lógica de este algoritmo se basa en la del primero mejor, pues en principio es igual con la salvedad de que considera los costes. En este algoritmo, la diferencia con el algoritmo A* original es que hemos decidido no actualizar los costes de los nodos, puesto que en la escala en la que estamos trabajando e, incluso, en escalas superiores la necesidad de actualizar costes es muy espontánea, tanto que parece casi inalcanzable, y sin ella hemos comprobado que nos ahorramos un gran coste computacional y tiempo. De esta manera, aunque parezca que pueda perder eficacia, el algoritmo sigue buscando la solución con menor coste posible y, según los resultados de las pruebas que se han realizado, da un resultado prácticamente inmejorable y sigue tratándose de un algoritmo eficaz.

4. Resultados obtenidos en las baterías de pruebas para cada una de las soluciones

Se ha utilizado la batería de pruebas predeterminada con 10 ficheros de laberinto para ejecutar cada una de las técnicas utilizadas, además de 2 ficheros de prueba de laberinto adicionales.

A continuación, se pueden ver los resultados obtenidos:

LABERINTO	TÉCNICA DE RESOUCIÓN	SOLUCIÓN	TIEMPO DE EJECUCIÓN	NODOS GENERADOS
MINIPAC1	Escalada simple	Solución no encontrada	23 us	23
	Primero mejor	PACMAN: D, A, D, D, D, D, D, B, D FANTASMA: D, D, D, D, B, B, I, I, I	303 us	24
	A*	PACMAN: B, D, D, D, D, D, D, A, D FANTASMA: D, D, D, D, B, B, I, I, I	643 us	43
MINIPAC2	Escalada simple	PACMAN: D A, A, D, D, D, D, D, A, D FANTASMA: B, I, I, I, D, D, D, D, D	16 us	10
	Primero mejor	PACMAN: D A, A, D, D, D, D, D, A, D FANTASMA: B, I, I, I, D, D, D, D, D	350 us	26
	A*	PACMAN: A, A, D, D, D, D, D, A, D FANTASMA: B, I, I, I, D, D, D, D, D	594 us	33
MINIPAC3	Escalada simple	Solución no encontrada	17 us	7
	Primero mejor	PACMAN: D, A, A, D, D, D, A, A, A, D, D, B, B, D FANTASMA: B, I, I, I, D, D, D, D, D, B, B, B, B, I	507 us	28
	A*	PACMAN: A, A, D, D, D, B, D, D, D, A, A, D FANTASMA: B, I, I, I, D, D, D, D, D, B, B, B	2583 us	60
MINIPAC4	Escalada simple	Solución no encontrada	9 us	2
	Primero mejor	PACMAN: A, I, B, B, B, D, D, D, D, A, A, A, D, D, D, A, A, A, A, I, I, I, I, I, I, I, I FANTASMA: D, D, I, I, I, I, I, A, D, D, D, D, D, B, I, I, I, I, I, A, D, D, D, D, D, B, I	1194 us	46
	A*	PACMAN: B, B, D, D, D, D, D, A, A, A, D, A, A, A, A, I, I, I, I, I, I, I, I FANTASMA: D, D, I, I, I, I, I, A, D, D, D, D, D, B, I, I, I, I, I, A, D, D, D	1304 us	53
MINIPAC5	Escalada simple	Solución no encontrada	11 us	3
	Primero mejor	PACMAN: A, A, A, D, D, B, B, B, D, D, D, D, D, A, A, I, I, I, A, A, A, A, A, I, I, I, I, B, I FANTASMA: D, D, D, D, D, D, D, I, I, I, I, I, I, I, D, D, D, D, D, D, D, D, I, I, I, I, I, I, I, D	710 us	37
	A*	PACMAN: A, A, A, D, D, B, B, B, D, D, D, D, D, A, A, I, I, I, A, A, A, A, A, I, I, I, I, B, I FANTASMA: D, D, D, D, D, D, D, I, I, I, I, I, I, I, D, D, D, D, D, D, D, D, I, I, I, I, I, I, I, D	838 us	40
MINIPAC6	Escalada simple	Solución no encontrada	9 us	1
	Primero mejor	Solución no encontrada	736 us	27
	A*	Solución no encontrada	668 us	27
MINIPAC7	Escalada simple	PACMAN: D, A, D, D, D, D, D, D, D FANTASMA: B, B, A, D, B, I, A, D, B	16 us	9
	Primero mejor	PACMAN: D, A, D, D, D, D, D, D, D FANTASMA: B, B, A, D, B, I, A, D, B	213 us	18
	A*	PACMAN: D, A, D, D, D, D, D, D, D FANTASMA: B, B, A, D, B, I, A, D, B	231 us	25
MINIPAC8	Escalada simple	Solución no encontrada	16 us	8
	Primero mejor	PACMAN: I, B, I, B, I, A, I, A, I, I, B, B, B, B, B, B, I, B, I FANTASMA: A, A, A, A, D, D, D, B, B, B, I, I, I, A, A, A, D, D, D	736 us	37
	A*	PACMAN: I, I, I, I, I, I, B, B, B, B, B, B, I, I FANTASMA: A, A, A, A, D, D, D, B, B, B, I, I, I, A, A	2527 us	63
MINIPAC9	Escalada simple	Solución no encontrada	5 us	0
	Primero mejor	PACMAN: D, B, B, B, B, B, I, B, I FANTASMA: A, A, A, A, D, D, D, B, B	265 us	21
	A*	PACMAN: D, B, B, B, B, B, I, B, I FANTASMA: A, A, A, A, D, D, D, B, B	449 us	24
MINIPAC10	Escalada simple	Solución no encontrada	9 us	1
	Primero mejor	PACMAN: A, A, A, I, I, B, B, I FANTASMA: I, A, A, A, A, B, B, B	176 us	15
	A*	PACMAN: B, I, I, A, A, I FANTASMA: I, A, A, A, A, B	241 us	19
MINIPACCUS1	Escalada simple	Solución no encontrada	17 us	0
	Primero mejor	Solución no encontrada	220 us	15
	A*	Solución no encontrada	193 us	15
MINIPACCUS2	Escalada simple	Solución no encontrada	4 us	0
	Primero mejor	PACMAN: D, D, D, D, D, D, D, B, B, I, I, I, I, I, I, I, B, B, D, D, D, D, D, D, D, B, B, I, I, I, I, I, B, I, I, B FANTASMA: I, A, D, D, D, D, D, D, I, I, I, I, I, I, I, D, D, D, D, D, D, D, I, I, I, I, I, I, I, D, D, D, D, D, D, I, I, I, I	892 us	39
	A*	PACMAN: D, D, D, D, D, D, D, B, B, I, I, I, I, I, I, I, B, B, D, D, D, D, D, D, D, B, B, I, I, I, I, I, B, I, I, B FANTASMA: I, A, D, D, D, D, D, D, I, I, I, I, I, I, I, D, D, D, D, D, D, D, I, I, I, I, I, I, I, D, D, D, D, D, D, I, I, I, I	1134 us	46

5. Conclusiones sobre los resultados para cada una de las técnicas de resolución empleadas.

Empecemos hablando del algoritmo de escalada simple. Este algoritmo, es el que obtiene los tiempos de ejecución más rápidos y el que genera menos nodos para encontrar la solución; sin embargo, dada su trivialidad, éste es el algoritmo que fracasa más veces al encontrar una solución. Este método de resolución heurística se debería emplear en tableros sencillos porque son en estos casos donde más destaca.

Sobre el algoritmo del primero mejor, podemos observar que en el resultado se generan más nodos que en el de escalada simple pero menos que en el de A*, lo cual indica que prueba menos casos. A pesar de generar la solución, siempre que la haya, generará una solución poco eficiente en laberintos complejos. A nivel de tiempo, calcula la solución más lenta que en el de escalada, pero más rápido que en el de A*. Como conclusión, es un algoritmo a camino entre el de escalada y el A*, obteniendo una solución siempre, pero siendo a menudo la menos eficaz.

Sobre el algoritmo A*, podemos decir que es el más costoso a nivel de nodos generados y de tiempo, pero nos da una salida, siempre que la haya, que suele ser la más eficiente o una de las más eficientes. A pesar de decir que este es el proceso más lento, con un laberinto de las características del nuestro, no llega a tardar ni medio segundo, con lo que aun así sigue siendo un algoritmo muy rápido a nivel de tiempo de ejecución.

8. Referencias.

No hemos utilizado librerías externas a las incluidas dentro del estándar de C++. En cuanto al uso de Inteligencias Artificiales, se ha utilizado ChatGPT de OpenAI para encontrar la forma óptima de calcular la función heurística, aunque ésta no proporcionó código, sino la idea general de cómo hacerlo.