Centro académico de Limón

IC-3002 Análisis de Algoritmos G-60

Proyecto 1: Laberinto

Profesor:

Joss Rayn Pecou Johnson

Hecho por:

Oscar Roni Ordoñez - 2023150387

Ricardo Arce Aguilar - 2023215990

2024

Semestre 2

Comparación entre los algoritmos

Algoritmos usados

El algoritmo de búsqueda de caminos implementado con backtracking se implementó desde 0 el equipo de desarrollo en una sesión de trabajo. Se tomó inspiración de los algoritmos de búsqueda de caminos en grafos, principalmente del algoritmo de Dijkstra.

Se decidió usar un enfoque iterativo basado en ir visitando cada celda del laberinto, agregar sus celdas vecinas a una pila, extraer la celda del tope de la pila y visitarla. El procedimiento se realiza en bucle hasta que la pila queda vacía, momento en el que se retornan las rutas válidas encontradas. Cada vez que se visita una celda, se le asigna a su atributo *previous* la celda visitada en la iteración anterior, de manera que al llegar al final del laberinto se pueda reconstruir el camino seguido mediante seguir la celda previa a la salida, luego la previa a esa, y así hasta llegar al inicio del laberinto.

Se contó con una gran dificultad en la implementación de este algoritmo, pues constantemente surgían errores de lógica difíciles de identificar, ya que el algoritmo se iba implementando a la par que la visualización del laberinto, por lo que no se tuvo una manera sencilla de hacer pruebas e identificar errores.

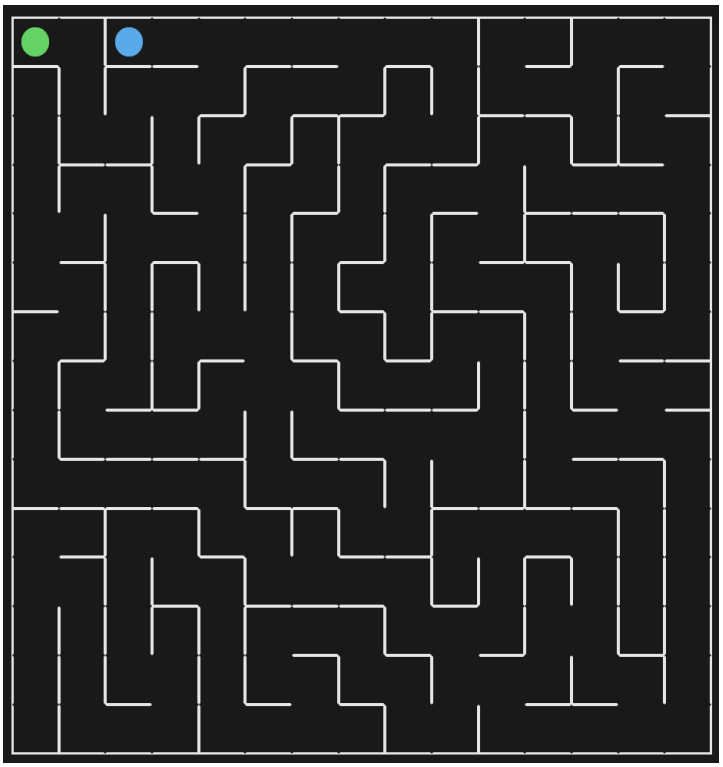
Para el algoritmo de búsqueda de caminos optimizado se decidió usar A* con la heurística de la distancia de Manhattan. Se consideró usar la heurística de la distancia euclidiana pero se notó que, por la manera de calcularla, la distancia de Manhattan era más apropiada para entornos de caminos horizontales y verticales, además de que el cálculo a realizar era más sencillo y eficiente.

Para su implementación se tomó como base el algoritmo implementado con backtracking y se agregó una serie de cálculos y procedimientos para optimizarlo en base a A*. En resumen, lo que se hizo fue no agregar las celdas vecinas a la pila de celdas siempre en el mismo orden (superior, derecha, inferior e izquierda), sino agregarlas de manera que quedara en el tope de la pila la celda con la menor distancia a la celda final o de salida. También se alteró el funcionamiento para que el algoritmo retorne únicamente la primera solución encontrada.

Comparación

A continuación se muestra un caso de búsqueda de caminos usando ambos algoritmos. Nótese la manera en la que el algoritmo implementado con A* encuentra el camino óptimo con mayor rapidez que el implementado con backtracking.

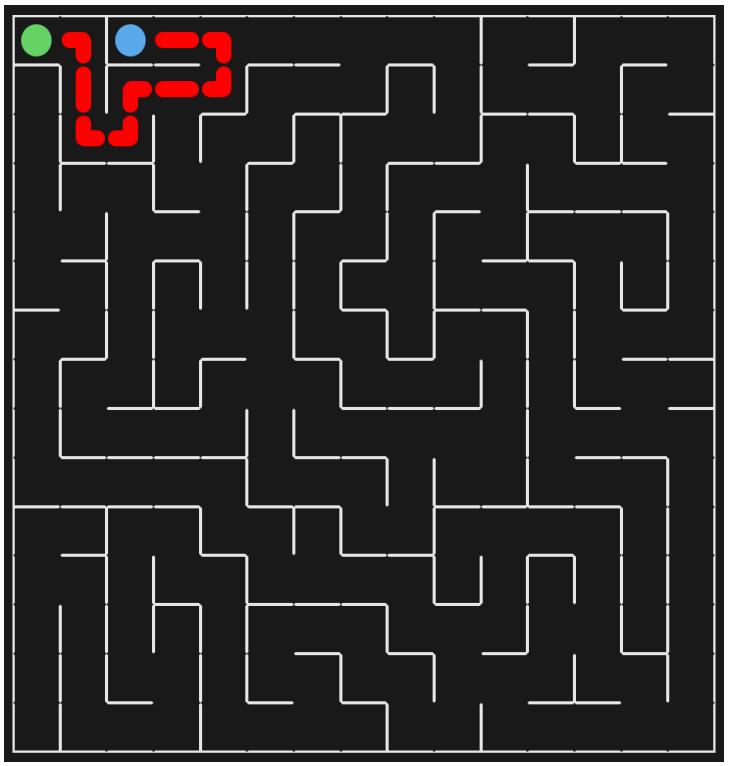
Se tomará como ejemplo el siguiente algoritmo generado aleatoriamente:



Se colocaron sus puntos de inicio y fin en las celdas (0,0) y (0,2) respectivamente. Al realizar la búsqueda de caminos mediante el algoritmo

implementado con backtracking, se midió su tiempo de ejecución y la duración fue de aproximadamente 9,5 segundos. Realizando la misma búsqueda con el algoritmo implementado con A*, se encontró que el tiempo de ejecución es casi instantáneo, quedando siempre por debajo de medio segundo.

Ambos algoritmos encontraron el siguiente camino como mejor solución:



La gran diferencia de tiempo se debe a que el algoritmo de backtracking siempre visita las celdas vecinas en el mismo orden (arriba, derecha, abajo e izquierda), mientras que el algoritmo optimizado con A* visita las celdas vecinas empezando por la que esté a una menor distancia del final. Por lo anterior, el algoritmo de backtracking, luego de llegar a la celda (0,4), continúa buscando caminos hacia la derecha, lo cual lo obliga a recorrer toda la parte derecha e inferior del laberinto antes de regresar a probar con la celda vecina izquierda de la celda (0,4), en cambio el algoritmo optimizado, al llegar a la celda (0,4), calcula la

distancia de sus celdas vecinas hasta el punto final y decide avanzar a la izquierda, decisión que lo lleva a la mejor solución en muy poco tiempo.

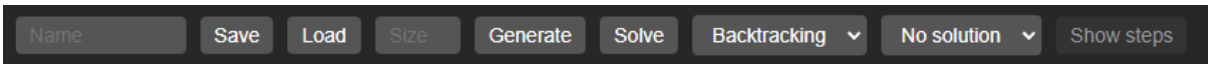
Instrucciones de uso

Ejecución de la aplicación

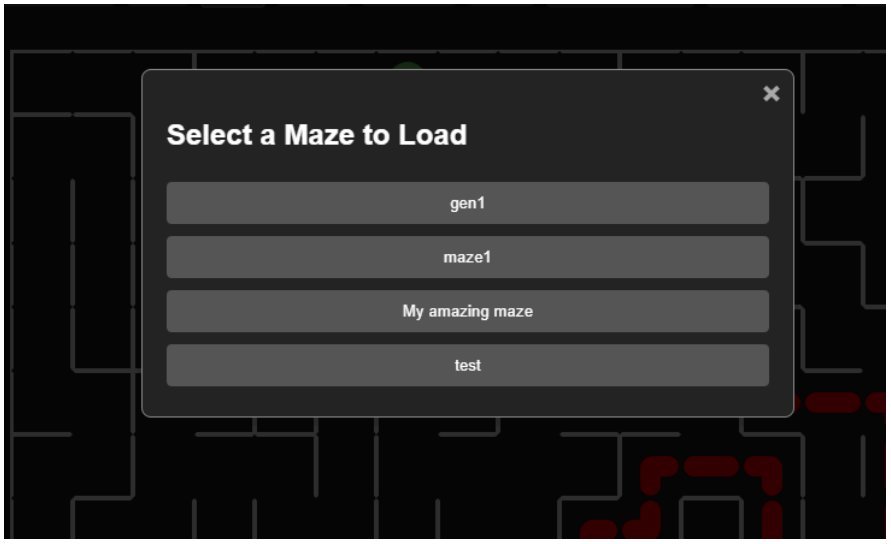
Una vez clonado el repositorio (disponible en la dirección https://github.com/okRoni/p1-laberinto-backtraking), siga las instrucciones en el archivo README.md para inicializar el entorno virtual y ejecutar el archivo. Al ejecutar el comando *python app.py* aparecerá un texto en el que especifica la URL a la que hay que entrar para acceder a la aplicación, seguramente será http://127.0.0.1:5000 o similar.

Cargado y guardado de laberintos

Una vez ejecutada la aplicación, acceda al dirección especificada anteriormente. Para cargar y guardar laberintos use la barra de opciones ubicada en la parte superior de la página, la cual contiene las siguientes opciones:



Oprimiendo el botón *Load* se mostrará una lista de laberintos guardados (estos se guardan en la carpeta src/mazes).



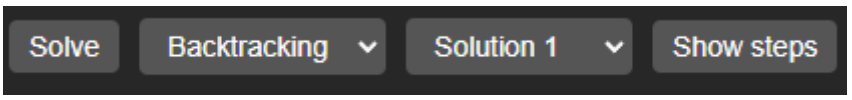
Puede guardar el laberinto actual asignándole un nombre en el campo *Name* y oprimiendo el botón *Save*.

Generación y solución de laberintos

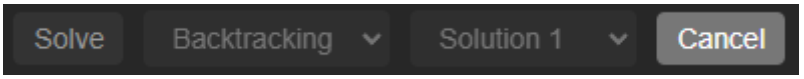
En la barra de opciones puede ingresar el tamaño deseado en el campo *Size*, debe ser un número entre 5 y 50, para generar el laberinto puede oprimir el botón *Generate*. Debe seleccionar un punto de inicio y de final seleccionando las celdas de su preferencia. Para generar los caminos puede seleccionar el algoritmo que desee usar y oprimiendo el botón Solve. En la lista desplegable del final puede seleccionar la solución que desee mostrar.

Visualización del proceso de solución

Para hacer esto es necesario haber ejecutado uno de los tipos de algoritmos de solución. Una vez hecho esto se habilitará una opción para mostrar los pasos "*Show Steps*", que al hacer click empezará una animación del proceso de solución con el paso a paso que tomó el algoritmo escogido.



En lo que dure esta animación no se puede realizar otra tarea, pero puede cancelarla en cualquier momento presionando el nuevo botón de "*Cancel*".



Bitácora Oscar

Fecha: 16/9

Horas trabajadas: 1,5

Progreso logrado:

Se realizó una reunión donde se revisaron los requerimientos iniciales del proyecto y se acordaron cuestiones básicas del formato de trabajo. Entre estos acuerdos se encuentra el formato del laberinto (interno), formato de guardado y cargado, dudas compartidas, desglose de tareas, asignación de prioridades, asignación de tareas y la fecha de la siguiente reunión.

Fecha: 17/9

Horas trabajadas: 2,5

Progreso logrado:

Se inicializó el proyecto de Flask, junto con su repositorio en GitHub. Además se colocaron instrucciones para copiar e iniciar el programa. La mayor parte del tiempo se gastó investigando sobre el framework y prácticas comunes al trabajar con este.

Fecha: 18/9

Horas trabajadas: 2,5

Progreso logrado:

Nos reunimos para planear el comportamiento del algoritmo y estudiar posibles soluciones al problema.

Fecha: 21/9

Horas trabajadas: 4,5

Progreso logrado:

Se realizó un prototipo para la funcionalidad de renderizado del laberinto y una solución. Todavía hay que investigar como hacerlo dinámico, pero con esto ya podemos probar cuestiones de algoritmos. Los avances fueron subidos a una nueva rama para el desarrollo de la interfaz.

Fecha: 29/9

Horas trabajadas: 11

Progreso logrado:

El esfuerzo de hoy ha sido principalmente para hacer el renderizado del laberinto dinámico, con la idea de que facilite la implementación de otras funcionalidades. La mayor parte del tiempo se invirtió en investigación y prototipado. Aunque también se cambió un poco el estilizado de los laberintos para que no cambiará de tamaño dependiendo de la cantidad de casillas.

Al final del día, mis esfuerzos de no usar javascript han sido inútiles, esto porque al ser python la parte del servidor, no se puede crear algo genuinamente dinámico en la parte del cliente. Aunque aprendí un poco de AJAX, htmx, y otras herramientas, todas terminaron siendo insuficientes y poco viables para el desarrollo. Por lo que al final implementé javascript en la parte del cliente para darle cualidades dinámicas al renderizado y mejorar el UX. Mi único lamento es haber seguido intentando por varias horas las otras alternativas incluso después de haber conseguido el permiso del profesor para usar javascript.

Se implementó también las funciones de guardado y cargado en la interfaz de la parte del cliente.

Fecha: 30/9

Horas trabajadas: 4,5

Progreso logrado:

Se implementó la función para generar el laberinto con ayuda del algoritmo de *depth first search*. Además se hicieron algunos cambios a la parte del cliente y el renderizado para mejorar la experiencia un poco mientras se prueban los algoritmos.

Fecha: 2/10

Horas trabajadas: 6,5

Progreso logrado:

Se implementó las funciones para mostrar múltiples soluciones del algoritmo backtracking en la parte del cliente. Además se cambió el estilo del camino (la solución), por lo que en vez de ser simples puntos en la casilla ahora se trazará una línea punteada para apreciar mejor la dirección incluso sí están a la par.

Fecha: 3/10

Horas trabajadas: 3

Progreso logrado:

Este día fue principalmente de investigación para buscar alternativas para actualizar el laberinto y mostrar cada paso que toma el algoritmo. Sin embargo, hay problemas por cómo está montada la comunicación actual con el servidor, falta de tiempo y mi falta de experiencia con websockets y ejecución asíncrona. Por lo que al final no va a ser en tiempo real la comunicación con el proceso del algoritmo, y se implementará en la interfaz de forma más lineal, incluso sí no es lo óptimo.

Fecha: 4/10

Horas trabajadas: 10

Progreso logrado:

Se integró las opciones al cliente para poder seleccionar el tipo de algoritmo que desea escoger. También se integró en la interfaz la posibilidad de poder visualizar el proceso que siguió el algoritmo seleccionado paso por paso. Además, como es el último día se realizaron varios arreglos de bugs, se actualizaron los estilos de algunos elementos, y se pulió un poco la experiencia de usuario. También se aseguró de que la parte de javascript y los ruteos tuvieran documentación interna.

Bitácora Ricardo

Fecha: 16/9.

Horas trabajadas: 1,5.

Progreso logrado:

Se acordaron las tareas básicas a realizar, se dividieron las tareas correspondientes a cada miembro y se anotaron las preguntas que se deben hacer al profesor.

Fecha: 18/9.

Horas trabajadas: 2,5.

Progreso logrado:

Se hizo una reunión para hablar sobre el algoritmo de backtracking para la búsqueda de caminos, se plantearon diversas soluciones posibles para el problema y se llegó a una posible solución con bastantes posibilidades de funcionar. Se definió también la estructura de clases que tendrá el proyecto.

Fecha: 27/9.

Horas trabajadas: 1.

Progreso logrado:

Se terminó la clase Cell y se empezó con la programación de la clase Maze.

Fecha: 28/9.

Horas trabajadas: 2.

Progreso logrado:

Se terminó con la clase Maze. Se implementaron métodos para guardar y cargar laberintos desde archivos de texto. Se adaptó el algoritmo de renderizado para usar la nueva clase Maze.

Fecha: 30/9.

Horas trabajadas: 8.

Progreso logrado:

Se implementó el algoritmo de búsqueda de caminos mediante backtracking. Surgió un error bastante sutil en la lógica del algoritmo, lo cual tomó bastantes horas para resolver. En resumen, el error se debía a que una celda puede ser visitada desde varias celdas, por lo que no basta con simplemente usar el atributo *previous* de cada celda para ese fin. Se optó entonces por almacenar temporalmente cada celda junto con la celda desde la cual debe visitarse, de este modo se supera la limitación antes mencionada.

Fecha: 3/10.

Horas trabajadas: 2.

Progreso logrado:

Se implementó el algoritmo de búsqueda optimizada de caminos mediante A*. Se decidió usar la distancia de Manhattan como heurística. Surgió un error que tomó bastante tiempo para resolver. El error se encontraba en el orden en el que se realizan ciertas operaciones en el bucle principal.

Fecha: 4/10.

Horas trabajadas: 2,5.

Progreso logrado:

Se realizaron cambios a ambos algoritmos de búsqueda de caminos para que retornen no solo las soluciones encontradas, sino también los pasos que siguió el algoritmo para llegar a dichas soluciones. Se trabajó también en la documentación externa.