Proyecto Final

Algoritmia y Programación II

Laberinto Recursivo.

Por: Carlos Rojas, Sebastián Mercado, Juan Contreras.

El proyecto final de Algoritmia y Programación II fue un reto para nosotros, puesto que evaluaba no solo lo aprendido en clase, si no también cómo interpretamos la información que podemos encontrar para nuestro propio uso. Debíamos hacer uso de recursividad para generar un laberinto, darle una entrada y salida, y mostrar el camino que debería de recorrerse. Nuestro primer enfoque fue la generación del laberinto, fijándonos en la generación aleatoria de este, de ahí, nuestro primer problema y posiblemente el más importante.

**“¿Qué problemas encontró al momento de desarrollar el proyecto?, ¿cómo los solucionó?”**

El proyecto mencionaba como bono un laberinto aleatorio, nos enfocamos en este punto y esto causó nuestro mayor problema en el desarrollo del código. Al ser un laberinto aleatorio, necesitábamos de una manera de generar aleatoriamente un laberinto. Decidimos hacer una matriz binaria que generase aleatoriamente 0 y 1 usando la clase “Random” de Java, la cual, es cercana a lo que tratábamos de conseguir. Explicando esta clase, lo que hace es que toma la hora actual, la convierte en una serie de números y a través de procesos matemáticos, genera números pseudoaleatorios. No podrían técnicamente llamarse aleatorios como tal, pero en este contexto, es perfecta para lo que necesitamos.

La complejidad radicó en la generación del camino del laberinto al ser aleatorio, necesitábamos crearle al laberinto la manera de pensar la solución, pero al ser aleatorio, era complicado darle reglas y condiciones, pues podría haber casos donde no podríamos saber con qué se encontraría el usuario. Quizás podría generarse un laberinto lleno de salidas y entradas, quizás podría haber uno donde haya entradas y salidas en la frontera, pero ningún camino que las una, quizás un laberinto donde no haya ninguna salida ni entrada, quizás uno donde no haya directamente entrada, ni salida, ni camino o uno donde simplemente haya camino en el medio de este sin entrada o salida.

El sistema aleatorio nos hizo tener que pensar varias reglas que el algoritmo debía hacer para poder idear cómo debería pensar la computadora y eso nos consumió mentalmente en el proyecto, sin embargo, nos dimos cuenta de que la solución era bastante sencilla. No necesitábamos nosotros enseñarle cómo hacer el camino, simplemente necesitábamos hacer que en prueba y error lo encontrase. El algoritmo de manera recursiva lo que hace es, generar un caso inicial donde ya se encontró el final del laberinto dados los puntos iniciales y finales, después empieza a moverse a través del laberinto por los lugares donde halla camino, y usando Backtracking, desmarca la posición actual para evaluar otros posibles caminos. Esto lo hace hasta que él logre llegar al punto final dado. Esto simplificó nuestro desarrollo en gran medida, debido a que nunca vimos el problema con el enfoque recursivo.

Cuando desarrollábamos el proyecto, lo resolvíamos como un problema procedimental, no recursivo, por lo que creábamos nosotros mismos nuestros propios obstáculos. Gracias a esto, pudimos desarrollar el proyecto satisfactoriamente,

**“¿Cómo fue distribuido el trabajo entre los integrantes del equipo?”**

El desarrollo del proyecto trató de ser mayoritariamente en grupo, pero había ocasiones donde no siempre se podía lograr esto debido al tiempo. Necesitábamos desarrollar el código, así que nos separamos en secciones. Carlos y Juan se encargaron de la lógica del código, sus reglas y movimientos dentro del laberinto, y Sebastián se encargó de indagar en interfaz gráfica y generar cada avance que lográbamos, también apoyando la lógica del código al ver qué problemas podrían generarse. Esta división del trabajo no solo nos permitía desarrollar el código, si no, ver nuestros problemas más allá de un mensaje de error en la consola, veíamos qué errores de lógica había dentro de este programa que nos servían de retroalimentación.

**“¿Cree que este problema hubiese sido más fácil de desarrollar usando programación procedimental en vez de recursiva? Justifique su respuesta.”**

No, en absoluto. Si este proyecto hubiera sido realizado sin recursividad, su dificultad hubiera sido muchísimo más alta. Si hubiéramos usado programación procedimental, tendríamos que, con el uso de ciclos, indicarle movimientos, razonamientos, análisis al código que serían complicados. La recursividad en el desarrollo del proyecto nos mostró cómo puede simplificar el procedimiento, sin la necesidad de una estructura de código tan densa. Además, si fuese el caso de programación procedimental, subiría más la dificultad debido a nuestro enfoque en el laberinto aleatorio. Simplemente no sería óptimo optar por programación procedimental en este caso.