EIF207 –Estructuras de Datos
Proyecto de programación #2

Profesor: M.Sc. Georges Alfaro S.

Planteamiento del problema

El proyecto por completar es una aplicación de escritorio que permita generar laberintos y recorrerlos, así como manejar documentos para poder guardarlos y recuperarlos.

Objetivos del proyecto

El proyecto tiene como objetivo estudiar técnicas básicas de representación de grafos y algoritmos básicos., así como el empleo de programación por retroceso (***backtracking***). Se espera además que la aplicación esté debidamente estructurada.

Descripción del proyecto

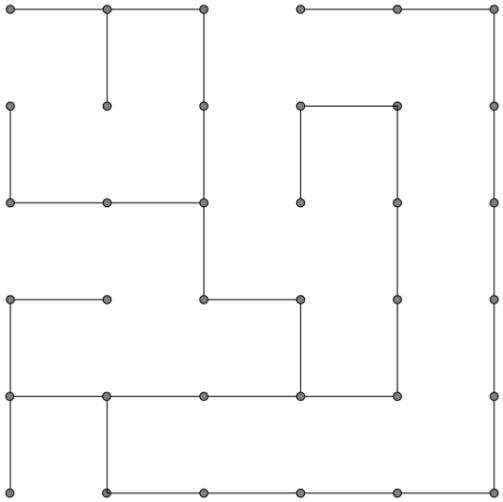
Descripción general.

El programa por desarrollar permitirá crear, guardar y recuperar laberintos generados de manera aleatoria utilizando el **algoritmo de Prim**, o la variación de Kruskal.

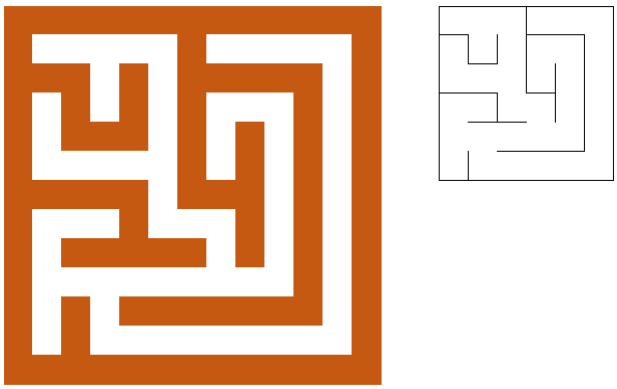
Hay muchas páginas donde puede encontrar detalles sobre la implementación del algoritmo. Al final del documento encontrará varias referencias que puede utilizar.

Existen diferentes maneras de interpretar la aplicación del algoritmo. En el caso del proyecto, cuando se vaya a generar un laberinto de dimensiones $m \times n$, se considerará una cuadrícula de nodos que indique cuáles son los espacios disponibles o “cuartos” dentro del laberinto. El algoritmo construirá un árbol que conecte los diferentes espacios, indicando en qué direcciones es posible desplazarse.

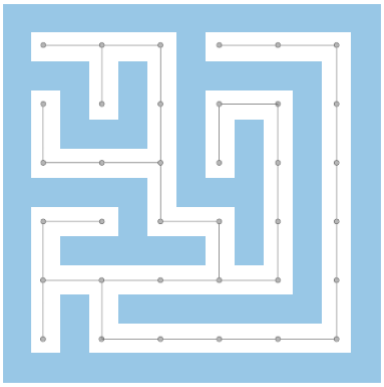
Por ejemplo, si se construye un laberinto de 6×6, los nodos podrían conectarse de la siguiente manera:



El laberinto podría dibujarse de las siguientes maneras:



Observe que el laberinto no está definido por los espacios en el primer grafo, sino por las conexiones entre cada nodo.



Si se muestra el laberinto utilizando solamente líneas de división, como se ve a la derecha en el ejemplo, el laberinto ocupará mucho menos espacio que si se dibuja usando paredes sólidas, tal como se observa al lado izquierdo. El programa tendrá una opción (menú o botón) que permita cambiar el método de despliegue a elección del usuario.

En el diagrama anterior no se muestran el punto de inicio y final de la ruta a recorrer, pero el programa deberá indicarlas utilizando algún tipo de marca que los distinga. **El punto inicial y final del recorrido deben poderse definir de manera interactiva por el usuario.**

Funcionamiento del programa.

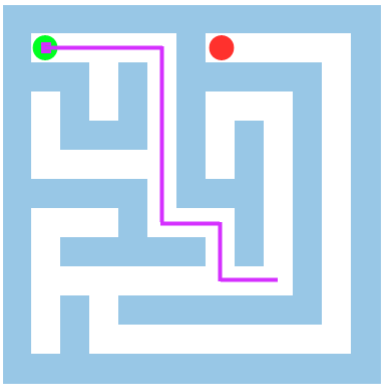
El programa permitirá abrir y guardar laberintos desde un archivo en formato XML. La estructura específica del archivo dependerá de la manera en que las clases sean definidas.

Cada laberinto se mostrará en una ventana independiente, y el programa puede tener abierta una cantidad arbitraria de ventanas. Existirá una opción en la aplicación (por medio de un menú y un botón en la ventana principal) para generar un nuevo laberinto de tamaño arbitrario (con un tamaño mínimo de 4 X 4 celdas), que se mostrará en la ventana correspondiente. **El algoritmo de generación debe garantizar que no haya rutas inaccesibles o ciclos dentro del laberinto.**

En la ventana principal se mostrará una tabla con información sobre los laberintos que se encuentren abiertos en ese momento, mostrando el nombre del archivo, la fecha de creación y el tamaño (filas y columnas) del laberinto. Si el usuario selecciona un laberinto de la lista, utilizando un doble clic, el programa traerá al frente la ventana correspondiente.

Cada laberinto se mostrará en una ventana independiente, con un tamaño mínimo de 480 X 360 píxeles. El laberinto no tiene ningún tamaño máximo definido, así que, si la ventana es muy pequeña para mostrarlo completamente, se podrán utilizar barras de desplazamiento (*scroll bars*) para moverse dentro de la ventana y una opción de menú que permita cambiar la escala del dibujo. La escala del dibujo podrá variarse entre un $^1/_8$ (12.5%) y 8 veces (800%) el tamaño normal. En una escala del 100%, cada celda del laberinto se mostrará con un tamaño de 16 píxeles, aunque este tamaño podrá variar, por lo que deberá especificarlo como un valor constante dentro del código de programa.

El usuario podrá recorrer el laberinto utilizando el puntero o solicitar al programa que resuelva el laberinto. En cualquier de los dos casos, el programa mostrará la ruta entre el punto inicial y final del laberinto utilizando una línea continua de un color definido.



Consideraciones de implementación

Escriba el programa utilizando el lenguaje de programación Java. Debe utilizar una interfaz gráfica adecuada. Los programas serán probados de preferencia en la plataforma Linux, pero éstos deberán poderse ejecutar correctamente también en Windows o Mac OS X. El desarrollo de la aplicación se completará usando el IDE **NetBeans** (versión 17) y la biblioteca **Swing** para manejo de la interfaz. No se permitirá el uso de bibliotecas de terceros o código no propio, salvo cuando se haga la previa consulta al profesor sobre su uso.

Incluya métodos que permitan verificar en la consola la correctitud del algoritmo, sin considerar los métodos de despliegue y dibujo.

Entrega y evaluación

El proyecto debe entregarse **por medio del aula virtual, en el espacio asignado para ello**. La entrega se hará al finalizar la semana 16 del curso. (**sábado 18 de noviembre de 2023**). No se aceptará ningún proyecto después de esa fecha, ni se admitirá la entrega del proyecto por correo electrónico. El proyecto se puede realizar en grupos de **tres personas, como máximo**. Deberán enviar un correo al profesor indicado la lista de participantes del grupo antes del martes de la semana 14 (martes 31 de octubre 2023)

Incluya comentarios en el código de los programas y describa cada una de las clases y métodos utilizados.

En caso de que la aplicación no funcione adecuadamente, efectúe un análisis de los resultados obtenidos, indicando las razones por las cuales el programa no trabaja correctamente, y cuáles son

las posibles correcciones que se podrían hacer. Durante la revisión del proyecto, es muy importante poder defender adecuadamente la solución propuesta. De existir disponibilidad en el laboratorio, y siempre que no atrase el desarrollo normal de los temas del curso, se revisaría el proyecto en clase, durante la semana siguiente a la entrega.

El proyecto se evaluará de acuerdo con la siguiente ponderación:

Diseño de la solución y documentación		**10%**
	Documentación y análisis de resultados:	**-10%** (se penalizará solamente cuando no se haga el análisis de resultados en caso de errores o incompletitud del trabajo entregado)
Funcionalidad		**90%**
	Manejo de documentos XML (abrir y guardar laberintos generados).	15%
	Generación de laberintos nuevos (aplicación del algoritmo principal)	30%
	Validación del laberinto (accesibilidad y eliminación de ciclos)	10%
	Implementación del menú de escala	10%
	Procedimiento para recorrer el algoritmo de manera interactiva (incluye una opción para reiniciar el recorrido)	20%
	Procedimiento de solución automática del laberinto	15%

Observaciones generales:

- Los proyectos deben entregarse con toda la documentación, diagramas, código fuente y cualquier otro material solicitado.
- Se debe indicar en cada documento el nombre completo y cédula de cada participante del grupo, indicando el nombre del curso, ciclo lectivo y descripción del trabajo que se entrega. Esto incluye comentarios en cada archivo fuente entregado.
- Los trabajos no se copiarán de ninguna llave USB u otro dispositivo en el momento, sino que se deben entregar en el formato adecuado.
- Si los materiales de entrega no están completos, se penalizará hasta un 15% de la nota correspondiente. Asimismo, cualquier trabajo práctico que no sea de elaboración original de los estudiantes (plagio) se calificará con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.
- Dadas las restricciones de tiempo del ciclo lectivo, no se aceptarán trabajos después de la fecha señalada para su entrega. **La fecha indicada en el aula virtual es la fecha límite para recibir el proyecto.**

Referencias

Backtracking Maze – Path Finder. (24 de Octubre de 2023). Obtenido de 101computing.net: https://www.101computing.net/backtracking-maze-path-finder/

Kruskal's algorithm. (24 de Octubre de 2023). Obtenido de Wikipedia: https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

Maze Generation — Recursive Backtracking. (24 de Octubre de 2023). Obtenido de Medium: https://aryanab.medium.com/maze-generation-recursive-backtracking-5981bc5cc766

Maze generation algorithm. (24 de Octubre de 2023). Obtenido de Wikipedia: https://en.wikipedia.org/wiki/Maze_generation_algorithm

Rat in a Maze - Backtracking using Stack. (24 de Octubre de 2023). Obtenido de geeksForGeeks: https://www.geeksforgeeks.org/rat-in-a-maze-backtracking-using-stack/

Rat in a Maze. (24 de Octubre de 2023). Obtenido de geeksForGeeks: https://www.geeksforgeeks.org/rat-in-a-maze/