**Tecnología en Sistemas de Información**

**Escuela de Ingeniería en Sistemas y Computación**

**Solución para un Juego de Lógica: Akari**



Fundamentos de Análisis y Diseño de Algoritmos

Diego Andrés Borrero

1227405

Cristian Fernando Jojoa

1224734

Sergio Ortiz Paz

0731354

Jhonny Reynolds Segura

1225541

Profesor:

Juan Manuel Reyes

Universidad del Valle

Facultad de Ingeniería

Santiago de Cali, Junio de 2012

**Introducción**

El funcionamiento de la solución para el juego de lógica (Akari), está basado en una única fuente la cual envía información sobre todas las posibles soluciones que puede tener el tablero de Akari, basado en este planteamiento se propuso un algoritmo ingenuo el cual evalúa todas las posibles combinaciones correspondientes a cada solución que el tablero del Akari podría tener de forma indiferente si esta solución sería correcta, de este modo es posible obtener cada solución de estas y realizar un proceso de verificación teniendo en cuenta las reglas del juego de lógica y concluir si esta solución es correcta e incorrecta. Adicionalmente se implementa un Algoritmo el cual ejecuta un procedimiento guiado con las reglas del juego Akari, obteniendo como salida una posible solución correcta del tablero correspondiente al Akari.

En este mini proyecto nuestra intención es implementar el uso de algoritmos ingenuos para la resolución del problema, esta vez justamente para la solución para el juego de lógica (Akari). También es necesario el conocimiento sobre la optimización de algoritmos y la obtención de los costos de ejecución de estos, teniendo en cuenta el lenguaje de programación utilizado en la implementación de estos algoritmos.

1. **Análisis temporal y espacial de los algoritmos implementados.**

Para el análisis de los algoritmos, se va a tener en cuenta la entrada del juego de lógica Akari que corresponde a una matriz M x N, es decir, filas y columnas respectivamente cuyos valores de esta, se han asociado a una serie de parámetros que permiten interpretar el tablero del Akari de forma natural.

Este es un algoritmo que no es óptimo ya que evalúa cada posible solución, de esta forma es un algoritmo de complejidad O(n!) donde n es el número de ------------ obteniendo así todas las posibles soluciones del juego de lógica Akari.

De esta manera, se van construyendo sucesivamente las posibles soluciones del juego de lógica, teniendo en cuenta cada combinación posible por cada celda negra con número, con todas las demás celdas negras con número y sus combinaciones como parte de una posible solución.

Para el análisis de los algoritmos, se va a tener en cuenta el número de celdas negras con número contenidas en el tablero de entrada, el número de combinaciones entre ellas correspondiente a cada posición posible que pueden tomar las bombillas en cuanto a cada celda negra con número respectivamente.

Para facilitar la comprobación del funcionamiento de los algoritmos, la aplicación permite visualizar en todo momento el contenido del archivo de entrada en formato de texto, de tal forma que sea posible interpretar la traducción de este tablero y su posible solución.

La aplicación permite buscar y abrir los archivos de entrada, los cuales son un archivo de texto plano. Las extensiones para los archivos de entrada son .txt

Para la implementación del algoritmo ingenuo, se utilizaron como estructura de datos: matriz de enteros, arreglos de enteros y arraylist de cadenas, con el fin de almacenar los elementos en posiciones contiguas de memoria, tienen un nombre relacionado a su función de variable que representa a cada celda negra con número valido en el tablero de Akari respectivamente. Para hacer referencia a esos elementos fue necesario utilizar un índice que especifica el lugar que ocupa cada elemento dentro del archivo.

El algoritmo de solución ingenua se basa en el algoritmo de combinatorias, que permite evaluar cada posible solución. De esta forma el tiempo de respuesta del algoritmo tiene relación directa con la cantidad de celdas negras con número contenidas en el tablero de entrada del Akari, ya que es necesario evaluar cada combinación posible con el fin de garantizar que el algoritmo me genera todas las posibles soluciones.

En la óptima se utilizó la funcionalidad de exclusiones en celdas en las cuales no se pueden ubicar bombillas, lo cual facilita la construcción de una solución correcta para el juego. Aparte de esta función, se utilizó colas de prioridad, arraylist de nodos y arreglos de enteros.

1. **Análisis de la solución (responde a preguntas como: ¿cuántas soluciones son posibles dado un tablero?, ¿siempre hay solución?).**

**¿Cuántas soluciones son posibles dado un tablero?**

Para el algoritmo ingenuo la cantidad de soluciones posibles depende de la cantidad de celdas negras con número contenidas en el tablero de entrada del Akari; ya que es necesario evaluar cada combinación de estas, es decir cuantas formas diferentes de ubicar las bombillas alrededor de cada celda negra con número.

--Para el algoritmo guiado el número de soluciones posible es una sola, ya que este algoritmo

**¿Siempre hay solución?**

La salida de los algoritmos no siempre genera una solución correcta. Teniendo en cuenta que el algoritmo ingenuo retorna la cantidad de posibles soluciones del tablero Akari basándose en combinatorias entre las celdas negras con números contenidas en este tablero, se tiene como consecuencia que no siempre genera una solución correcta basada en las reglas del juego de lógica.

--El algoritmo guiado

1. **Detalles principales de implementación.**

El programa realizado consta de 4 capas: lógica, algoritmos, gui y main.

La capa lógica comprende la parte donde se declaran las clases principales que permitirán capturar la matriz de entrada correspondiente al tablero del Akari, de un archivo .txt ; esta capa permite establecer los parámetros de entrada que tienen los algoritmos y se compone de los siguientes métodos:

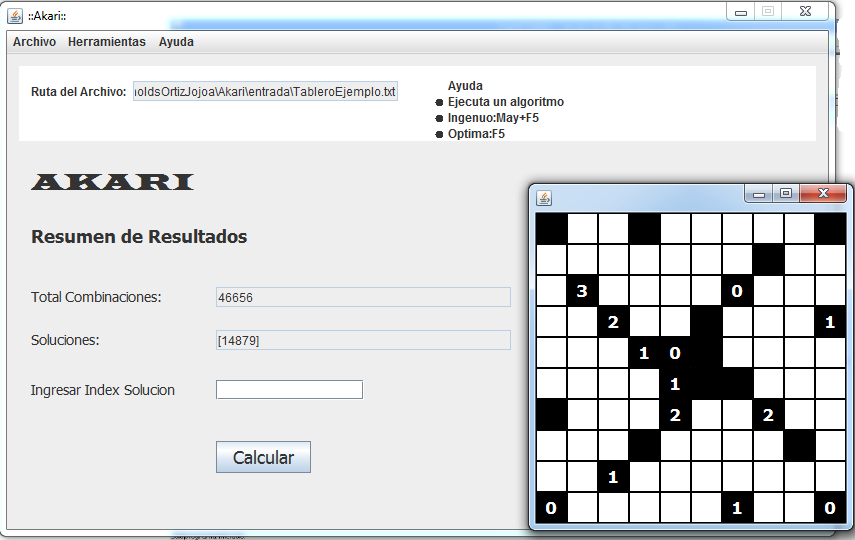
La capa de los algoritmos comprende la parte que se encarga establecer los procedimientos a seguir para dar solución al tablero Akari teniendo en cuenta los parámetros de entrada necesarios. Se compone de los siguientes métodos.

La capa gui comprende la parte la parte gráfica, el manejo de cada uno de los eventos, y se compone de los siguientes métodos:

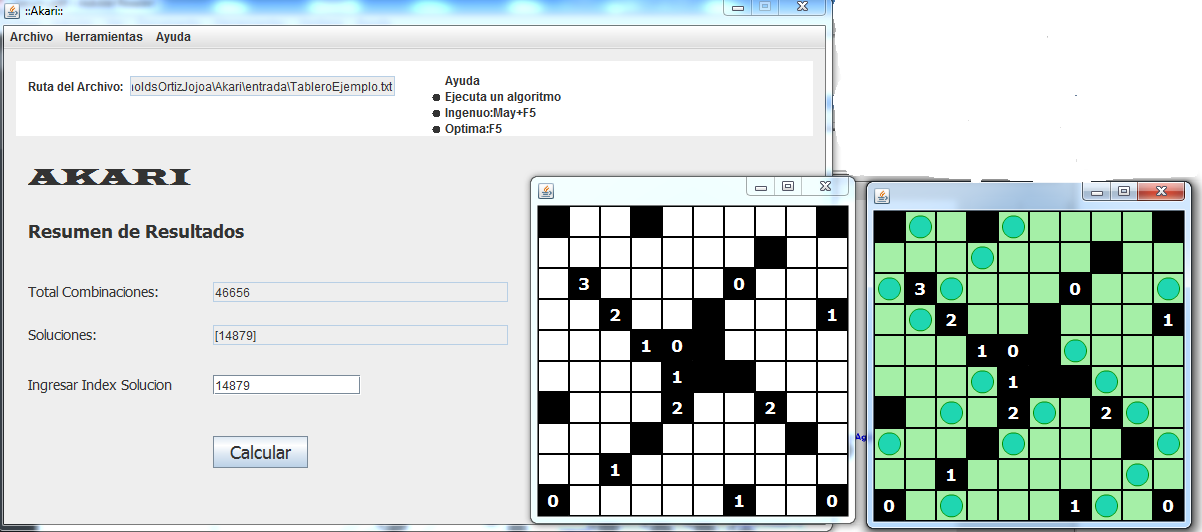
La capa main permite ejecutar la aplicación.

1. **Descripción y análisis de pruebas realizadas.**

**En la siguiente grafica se visualizan los resultados del Algoritmo Ingenuo.**

****

**En la siguiente grafica se visualiza el tablero calculado después de obtener las soluciones del Algoritmo Ingenuo.**

****

1. **Conclusiones y aspectos a mejorar.**

El algoritmo ingenuo no es óptimo ya que este tiene la cualidad de obtener todas las posibles soluciones que puede tener el juego de lógica, permitiendo identificar cada combinación posible para el número de celdas negras con número y verificar si esta posible solución es correcta de acuerdo a las reglas del juego. El algoritmo guiado puede ser un poco más eficiente ya que este realiza un procedimiento basado en las validaciones del Akari, permitiendo obtener de forma más rápida una solución que sea correcta.

Hay diferentes formas de otorgar una solución a un problema dado, cuando se crea una solución, aparte de buscar que el algoritmo implementado genere como salida una solución correcta para el juego, se debería pensar en un algoritmo óptimo tanto en velocidad como en espacio para garantizar la calidad en el software que se está realizando.

Las estructuras de datos empleadas basadas en arreglos pueden tornarse dispendiosas debido a la cantidad de comparaciones que se deben realizar para llegar a una solución correcta de acuerdo a las validaciones del juego.