Ejercicio 2: Backtracking - Posicionamiento de las m piezas en un tablero de nxn

(revisado Mayo 2018)

Muñoz Plaza. Alejandro

**Abstract**—En este ejercicio se pone en uso la técnica de backtracking en el caso práctico de colocar varias piezas de ajedrez en un tablero de dimension n. Al iniciar el programa se nos pide la dimension del tablero y el número de piezas a colocar, a contuniación pulsamos iniciar y si encuentra una solución nos la muestra en una nueva ventana y el tiempo que ha tardado en encontrarla o nos informa de que esa combinación que se ha introducido no tiene solución posible.

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

————————————————

1. F.A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305. E-mail: author@ boulder.nist.gov.

**\*\*\*Please provide a complete mailing address for each author, as this is the address the 10 complimentary reprints of your paper will be sent**

*Please note that all acknowledgments should be placed at the end of the paper,*

—————————— ◆ ——————————

# 1 Introduction

E



Fig. 1. Ventana principal del programa.

ste problema deriva del problema de las 8 damas planteado por el alemán “Max Bezzel” en la revista “Berliner Schachzeitung” en 1848. Consiste en colocar 8 damas en un tablero de ajedrez de tamaño 8x8. Al analizar este programa se descubrío que existían 92 soluciones posibles a este problema que fueron encontradas por primera vez por “Franz Nauck” en 1850. La diferencia con el problema que se nos plantea es que además de damas podemos colocar otras piezas, en este caso alfiles, torres y dos piezas inventadas, y la dimensión del tablero puede ser definida.

Para resolver este problema usarémos la técnica de programación backtracking reduciendo el tiempo máximo de ejecución del problema de a .

# 2 Guía de Usuario

## 2.1 Campos de datos

Descripción de la Fig.1:

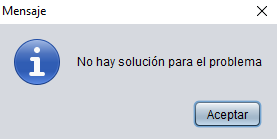


Fig. 2. Ventana que aparece al no encontrar una solución.

El primer campo indica el tamaño del tablero, el tamaño del tablero tiene un mínimo de cuatro y no tiene límite superior.

Los siguientes indica la cantidad de piezas de ese tipo que se van a intentar colocar en el tablero.

Al pulsar el botón aparecerá una ventana con la primera solución encontrada y encima del botón el tiempo que ha tardado en encontrar la solución. Sino se ha encontrado aparecerá la ventana informando de ello.



Fig. 3. Ventana que aparece cuando se ha encontrado una solución.

# 3 coste computacional

Partimos de la solución en la cual se implementa con un bucle dentro de otro. Con la idea de recursividad el algoritmo funciona de la siguiente manera:

1. Elegimos una de las piezas.
2. La intentamos colocar en el tablero.
3. Comprobamos si hay solución:

Si no se ha encontrado vamos al punto “1.” A colocar la siguiente pieza.

Si al volver de la recurrencia no se ha encontrado la solución quitamos la pieza colocada y saltamos esa posición libre, ya que no nos lleva a una solución.

1. Miramos si todas las piezas estan colodas.

La función recursiva a acaba devolviendo un valor “boolean” con “True” cuando ha encontrado una solución o “False” en el caso que no exista.

El coste de este algoritmo reduce el coste máximo de recorrer el árbol de posibilidades a donde en el peor de los casos volvemos al caso inicial de .

Para reducir el tiempo en casos donde la “m” es pequeña se ha implementado la idea de las posiciones libres donde antes de elegir una pieza se revisa el estado del tablero y se obtienen las posiciones del tablero donde se podría colocar la pieza.

# Conclusiones

Para este problema el uso backtracking permite reducir el tiempo considerablemente en la mayoría de casos.

El uso de backtracking para reducir el coste de ejecución no es una reducción directa del coste, ya que, depende de la disposicion inicial en la que se encuentre el problema y de las decisiones que tome el programador para recorrer el árbol de posibilidades. Por lo tanto, podemos decir que el backtracking es una implementación de un problema y dependiendo del análisis del problema funcionara mejor o peor.

**References**

1. <https://elpais.com/elpais/2016/03/10/ciencia/1457629302_411493.html>