Mancala*

Daniel Alfredo Vidal De Leon[1]

[a] *Pontificia Universidad Javeriana, Bogotá, Colombia*

Abstract

En este documento se presenta la escritura formal del problema "calcular la secuencia de pasos para resolver el juego de las Torres de Hanoi", junto con un algoritmo de solución.

Keywords: algoritmo, escritura formal, Torres de Hanoi.

1. Análisis del problema

Se desea escribir unos algoritmos para jugar un juego de Mancala (Kalah). Antes de proponer la solucion, primero describire el juego.

Definición 1. Mancala es uno de los juegos de mesa más antiguos, practicado en diversas culturas alrededor del mundo. La variante Kalah se ha popularizado en Occidente por sus reglas simples y la profundidad estratégica que ofrece. El juego se desarrolla en un tablero compuesto por 12 hoyos pequeños, distribuidos en dos filas de 6 hoyos cada una, y dos almacenes (o stores), uno para cada jugador. La mecánica se basa en recoger todas las semillas de un hoyo determinado y sembrarlas en los hoyos siguientes (en sentido antihorario), respetando una serie de reglas especiales que incluyen saltar el almacén del oponente, obtener turnos extras y capturar las semillas del adversario cuando se cumplen determinadas condiciones.

1. Se inicializa el tablero con 4 semillas en cada uno de los 12 hoyos pequeños; los almacenes comienzan en 0.
2. Cada jugador tiene asignados 6 hoyos y su respectivo almacén.
3. En cada turno, el jugador selecciona un hoyo de su lado que contenga semillas, La jugada consiste en retirar todas las semillas del hoyo seleccionado y sembrarlas, una a una, en sentido antihorario. Se omite el almacén del oponente durante esta siembra.
4. Si la última semilla cae en el almacén propio, el jugador obtiene un turno extra.

*Mancala

Email address: vidal.da@javeriana.edu.co (Daniel Alfredo Vidal De Leon)

5. Si la última semilla cae en un hoyo vacío perteneciente al jugador y el hoyo opuesto del adversario contiene semillas, se captura esa semilla más todas las semillas del hoyo opuesto, depositándolas en el almacén del jugador.

6. La partida finaliza cuando uno de los jugadores no tiene semillas en ninguno de sus hoyos; en ese caso, se recogen las semillas restantes del oponente y se determina el ganador según quién tenga más semillas en su almacén.

De esta descripción del juego se puede concluir:

1. Se utilizara un listado o vector de 14 indices.
2. en los cuales todos exceptuando los indices que representaran los almacenes tendran un valor de 4(semillas).
3. y este continuara un ciclo de partidas hasta alguna de las secciones de jugador quede vacia (exceptuando el espacio del almacen).

2. Diseño del problema

1. *Entradas*:
   - *a*) Cantidad de semillas por hoyo (valor por defecto: 4).
   - *b*) Selección de un hoyo (números del 1 al 6) por cada jugador durante su turno.
2. *Salidas*:
   - Visualización en del estado actualizado del tablero.
   - Secuencia de jugadas realizadas y, al finalizar, se determina y muestra el ganador.

3. Algoritmo de solución

```
Procedimiento PROCESAR_JUGADA(tablero, jugador_actual, hoyo_seleccionado)
    Si (tablero[hoyo_seleccionado] == 0) Entonces
        Retornar (tablero, Falso)
    FinSi

    semillas ← tablero[hoyo_seleccionado]
    tablero[hoyo_seleccionado] ← 0
    indice_actual ← hoyo_seleccionado

    Mientras (semillas > 0) Hacer
        indice_actual ← indice_actual + 1
        Si (indice_actual == tamaño_del_tablero) Entonces
            indice_actual ← 0
        FinSi

        Si (indice_actual == almacen_del_oponente(jugador_actual)) Entonces
            Continuar  // Saltar este índice y pasar a la siguiente iteración
        FinSi

        tablero[indice_actual] ← tablero[indice_actual] + 1
        semillas ← semillas - 1
    FinMientras

    Si (ultima_semilla_en_mi_lado(indice_actual, jugador_actual)
        Y tablero[indice_actual] == 1
        Y tablero[hoyo_opuesto(indice_actual)] > 0) Entonces
        tablero[almacen_del_jugador(jugador_actual)] ← tablero[almacen_del_jugador(jugador_actual)]
                                                     + tablero[indice_actual]
                                                     + tablero[hoyo_opuesto(indice_actual)]
        tablero[indice_actual] ← 0
        tablero[hoyo_opuesto(indice_actual)] ← 0
    FinSi

    turno_extra ← (indice_actual == almacen_del_jugador(jugador_actual))

    Retornar (tablero, turno_extra)
FinProcedimiento
```

3.1. Invariante

La suma total de semillas en el tablero (más las que se encuentran en los almacenes) se mantiene constante (valor inicial = 4×12 semillas).

Cada operación (siembra, turno extra y captura) asegura que la única modificación se realice siguiendo las reglas del juego y el movimiento realizado corresponde a la distribución secuencial de las semillas en sentido antihorario.

3.2. Análisis de complejidad

La mayor parte del trabajo se concentra en el bucle de siembra, cuyas iteraciones dependen directamente del número de semillas en el hoyo seleccionado. Por lo tanto, si consideramos s como el número de semillas que se deben distribuir, la complejidad del algoritmo es: O(s)