Proyecto
Analisis y Diseño de Algoritmos

Miguel Bayona Rivera
Julián Rodríguez Céspedes
Esteban Altamiranda Julio

Pontificia Universidad Javeriana

Abril de 2025

Bogotá

1 Introducción

Mancala es un juego tradicional con raíces africanas y del Medio Oriente, que ha sido transmitido por generaciones como un ejercicio de lógica, estrategia y anticipación. Su mecánica sencilla y estructura secuencial lo hacen ideal para representar problemas computacionales de forma didáctica, permitiendo explorar recorridos cíclicos y decisiones condicionadas.

El proyecto corresponde al desarrollo de este juego, siendo este un juego de estrategias por turnos basado en la distribución de "semillas" entre una seria de "agujeros" y "graneros", convirtiéndolo así en un caso para aplicar conceptos como descomposición de problemas y análisis de eficiencia.

El sistema contempla la implementación completa del juego, incluyendo lógica de movimientos, reglas de captura, validación de jugadas. e incorporación de un jugador sintético.

Desde la perspectiva algorítmica, el enfoque utilizado promueve la separación de responsabilidades y la reutilización del código permitiendo evaluar, escalar, y probar cada componente de manera individual. La construcción de la interfaz es el primer paso para demostrar la forma en la que el usuario interactuará con el sistema

2 Especificaciones del sistema

A continuación, se mostrarán ciertas especificaciones del sistema que serán esenciales para el correcto funcionamiento del sistema

2.1 Requerimientos funcionales

- El sistema debe mostrar el estado actual del tablero, incluyendo agujeros y graneros de los jugadores que estén en el juego.
- El sistema debe verificar de quien es el turno y mostrarlo en la interfaz
- El sistema debe permitir que cada jugador pueda seleccionar uno de sus propios agujeros para realizar un movimiento
- El sistema debe distribuir correctamente las semillas según las reglas de Mancala y cambiar de turno automáticamente
- El sistema debe verificar el estado del juego, mostrando al jugador ganador
- El sistema debe permitir la incorporación de un jugador sintético

2.2 Requerimientos no funcionales

- El sistema debe ejecutarse completamente en consola.
- El tiempo de respuesta del sistema debe ser inmediato después de cada acción del jugador o del jugador sintético.

- El código debe ser legible y mantenible, incluyendo comentarios relevantes que expliquen la lógica de las funciones principales.
- El sistema debe permitir facilidad de pruebas unitarias y de integración para verificar la lógica del juego.
- El jugador debe tomar decisiones en un tiempo aceptable (menos de 5 segundos)
- El juego debe poder reiniciarse sin necesidad de reiniciar el programa.
- El sistema debe manejar errores de entrada sin interrupciones.

3 Diseño de la Interfaz del usuario

La interfaz de usuario fue implementada en Python, empleando una representación en consola. EL tablero se muestra con dos filas de "agujeros" y dos graneros, representando el lado de cada jugador. El diseño de esta interfaz permite su futura integración a la lógica completa del sistema

3.1 Entrada del usuario

Los jugadores interactuarán directamente con la interfaz, que como se mencionó anteriormente, es por medio de la consola, en esta, el juego solicitará seleccionar uno de los hoyos para realizar un movimiento; algunos componentes que conforman la entrada del usuario son:

- Selección de Agujero: En cada turno el sistema se encargará de solicitar al jugador que seleccione una posición del tablero, pidiendo que ingrese un numero que corresponda al agujero de su lado del tablero, este numero representará la posición desde la cual desea distribuir las semillas
- Validación de jugada: En cada turno, el sistema verificará que el agujero seleccionado contenga al menos una semilla y que pertenezca al jugador en turno, si esto no se cumple, se solicitará la entrada nuevamente
- Control de Flujo: En el momento en el que el jugador realiza su jugada correctamente, se alterna el flujo, dándole el control del sistema al otro jugador.
- Jugador Sintético: El sistema tomará decisiones empleando una lógica simple, escogiendo uno de los agujeros que le corresponden de manera aleatoria, claramente, respetando las reglas del juego.

3.2 Salida de Información

La salida de información le permitirá al jugador comprender claramente la situación actual del tablero y tomar decisiones informadas acorde al contexto.

- Visualización del tablero: Se mostrará el tablero con los valores acorde a las jugadas realizadas durante el juego.
- Estado de los graneros: El sistema se encargará de indicar cuantas semillas ha acumulado cada jugador en sus graneros, actualizando esta información en cada movimiento
- Indicación de turno: En cada momento durante le juego, se indicará cual de los dos jugadores debe realizar la jugada
- Mensaje de error: En caso de que el jugador digite el numero de un agujero que no corresponde, se le indicará y se le pedirá que vuelva a ingresar un numero entre los rangos de sus agujeros del lado de su tablero.
- Mensaje final: Cuando el sistema valide que se finalizó el juego, se mostrará un mensaje indicando que jugador ganó.

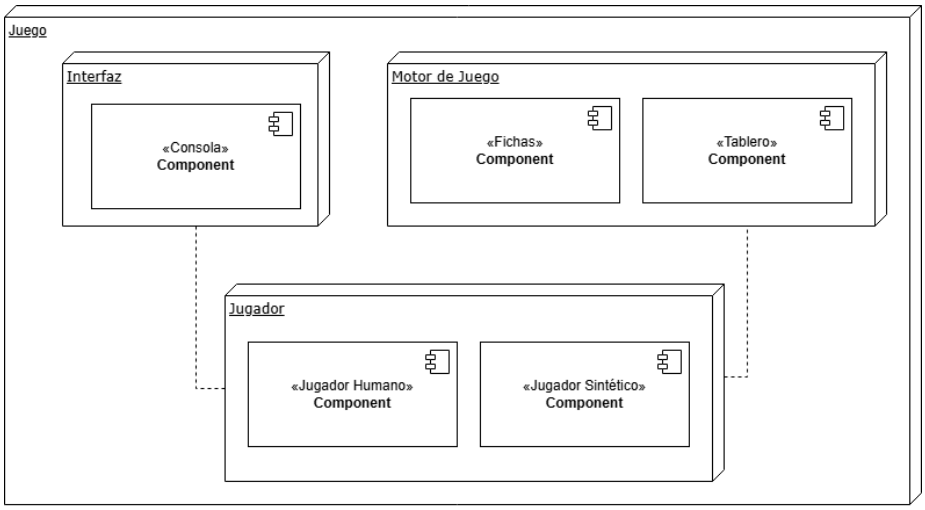
4 Arquitectura del sistema

La arquitectura del sistema del juego Mancala ha sido diseñada bajo un enfoque modular, buscando una clara separación de responsabilidades entre los distintos elementos que componen la aplicación. Esta estructura facilita el mantenimiento, la comprensión del flujo del juego y la posibilidad de realizar pruebas unitarias sobre componentes específicos sin afectar el comportamiento global del sistema.

El sistema está dividido en tres grandes bloques funcionales: la interfaz de usuario, el motor de juego y los módulos de jugador. Cada uno de estos bloques contiene componentes específicos que interactúan entre sí a través de interfaces bien definidas, permitiendo la reutilización del código y una fácil ampliación o modificación futura.

A continuación, se presentan los diagramas fundamentales que representan esta arquitectura: el diagrama de componentes y el diagrama de clases. El primero detalla los módulos principales y sus relaciones, mientras que el segundo profundiza en la estructura interna de cada uno, mostrando atributos y métodos clave para su funcionamiento.

4.1 Diagrama de componentes



El diagrama de componentes presenta la organización lógica del sistema para el juego Mancala, dividiendo la aplicación en módulos funcionales que representan componentes independientes. Esta estructura permite una mejor mantenibilidad, reutilización del código y comprensión de las dependencias entre subsistemas. A continuación, se describen los principales paquetes y componentes del sistema:

Componente: Interfaz

- **Consola:** Este componente representa la interfaz de interacción con el usuario, la cual opera desde la línea de comandos. Se encarga de mostrar el estado del tablero, los mensajes de error, el turno actual y solicitar las jugadas al jugador humano.

Componente: Motor de Juego

- **Fichas:** Este componente representa las unidades básicas del tablero, es decir, las casillas que contienen las semillas. Encapsula la lógica para agregar, remover y consultar la cantidad de semillas que contiene cada agujero.
- **Tablero:** Encargado de modelar el estado general del juego. Contiene los agujeros y graneros, así como los métodos para inicializar, actualizar y mostrar el estado del tablero.

Componente: Jugador

- **Jugador Humano:** Representa a un jugador controlado por una persona. Este componente interactúa directamente con la Consola y contiene la lógica para validar y ejecutar la jugada seleccionada por el usuario.
- **Jugador Sintético:** Representa al jugador automático del sistema. Emplea una lógica básica (como selección aleatoria) para tomar decisiones válidas de juego, respetando las reglas establecidas por el motor.

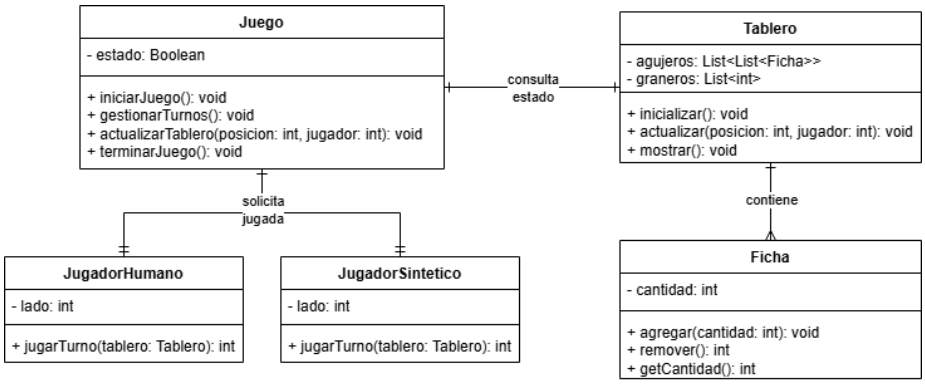
Interacciones entre componentes

- La **Consola** se comunica con los componentes **Jugador Humano** y **Tablero** para gestionar la interacción y mostrar resultados.
- El **Tablero** se apoya en el componente **Fichas** para representar el estado de cada casilla del juego.
- Tanto el **Jugador Humano** como el **Jugador Sintético** consultan el estado del **Tablero** para tomar decisiones en sus turnos.

4.2 Descripción de módulos

- **Consola**: Componente responsable de interactuar directamente con el jugador. Solicita las jugadas y muestra el estado del tablero en cada turno. Representa la capa de entrada/salida del sistema.
- **Fichas**: Encargado de controlar la lógica relacionada con la distribución de semillas (fichas), incluyendo reglas de captura y cambios de turno.
- **Tablero**: Representa la estructura del juego, incluyendo los agujeros y graneros. Permite consultar y modificar el estado del tablero.
- **Jugador Humano**: Componente que procesa las jugadas del jugador. Verifica que las entradas sean válidas y dentro de su rango correspondiente.
- **Jugador Sintético**: Representa al jugador controlado por la máquina. Su objetivo es automatizar jugadas aplicando una estrategia programada. En esta versión inicial, el jugador sintético emplea una cierta lógica a partir de la dificultad seleccionada.

4.3 Diagrama de clases



Este modelo permite visualizar las interacciones entre las clases y cómo cada una contribuye al funcionamiento del sistema:

Clase `Juego`

Esta clase actúa como el controlador principal del sistema. Coordina el flujo del juego, gestiona los turnos y se encarga de actualizar el tablero de acuerdo con las jugadas realizadas.

- **Atributo:** `estado: Boolean`
  Indica si el juego está en curso o ha finalizado.
- **Métodos:**
  - `iniciarJuego(): void` — Inicializa todos los componentes y comienza la partida.
  - `gestionarTurnos(): void` — Alterna entre los jugadores y coordina la ejecución de sus turnos.
  - `actualizarTablero(posicion: int, jugador: int): void` — Realiza una jugada y modifica el tablero.
  - `terminarJuego(): void` — Finaliza el juego y muestra los resultados.

Clase `JugadorHumano` y `JugadorSintético`

Estas clases representan a los jugadores del sistema. Cada una implementa el método `jugarTurno()` que permite realizar una jugada sobre el tablero.

- **Atributo:** `lado: int`
  Determina de qué lado del tablero juega el jugador.
- **Método:** `jugarTurno(tablero: Tablero): int` — Realiza una jugada y devuelve la posición seleccionada.

Clase `Tablero`

Esta clase modela el estado del tablero de Mancala, incluyendo los agujeros con semillas y los graneros de ambos jugadores.

- **Atributos:**
  - `agujeros: List<List<Ficha>>` — Representa los agujeros de cada jugador.
  - `graneros: List<int>` — Representa los graneros de los jugadores donde se acumulan las semillas.
- **Métodos:**
  - `inicializar(): void` — Configura el estado inicial del tablero.
  - `actualizar(posicion: int, jugador: int): void` — Realiza la distribución de semillas desde un agujero seleccionado.
  - `mostrar(): void` — Muestra visualmente el estado actual del tablero.

Clase `Ficha`

Esta clase representa cada casilla del tablero donde se almacenan las semillas.

- **Atributo:** `- cantidad: int`
  Número de semillas almacenadas en una ficha.
- **Métodos:**
  - `agregar(cantidad: int): void` — Agrega semillas a la ficha.
  - `remover(): int` — Retira todas las semillas de la ficha y retorna la cantidad retirada.
  - `getCantidad(): int` — Retorna el número actual de semillas en la ficha.

5 Descripción del jugador

- **Jugador Humano**:

  El jugador humano toma decisiones mediante interacción directa a través de la consola. En cada turno, el sistema solicita al jugador que seleccione uno de sus agujeros válidos para realizar un movimiento. Esta selección depende únicamente de su análisis personal del tablero, sin intervención automática.

  Antes de permitir la jugada, el sistema valida:

  - Que el número ingresado esté dentro del rango permitido.
  - Que el agujero pertenezca al lado del jugador actual.

- Que el agujero contenga al menos una ficha.

En caso de error, se solicita una nueva entrada válida al usuario.

- **Jugador Sintético (IA)**:

  El jugador sintético toma decisiones de forma automática utilizando el algoritmo `Minimax` con una profundidad limitada. Esta estrategia le permite simular posibles jugadas y seleccionar la opción que ofrezca mayor ventaja.

  Durante su turno, el sistema realiza las siguientes acciones:

  - Identifica los agujeros válidos del jugador sintético.
  - Simula cada jugada posible sobre un tablero auxiliar.
  - Evalúa los resultados usando `Minimax`, considerando los posibles movimientos del oponente.
  - Compara los valores obtenidos y selecciona la jugada más favorable.
  - Ejecuta la jugada elegida y actualiza el estado del tablero.

  Este proceso le permite tomar decisiones estratégicas anticipando movimientos futuros, especialmente en niveles de dificultad alta.

5.1 Estrategia de decisiones

El jugador sintético ha sido diseñado con una lógica de toma de decisiones que le permite actuar de forma más inteligente que una simple selección aleatoria. Para ello, se definieron una serie de condiciones jerárquicas que guían su comportamiento durante su turno.

La estrategia se basa en evaluar las posibles jugadas según su conveniencia táctica, priorizando aquellas que maximicen su ventaja o minimicen el riesgo frente al oponente. En orden de prioridad, las jugadas se eligen bajo los siguientes criterios:

1. **Captura de semillas**: Si existe una jugada que le permite capturar semillas del lado del oponente, esta tiene máxima prioridad.

2. **Turno extra**: Si una jugada finaliza en el propio granero del jugador sintético, se considera favorable por otorgar un turno adicional.

3. **Conservación de recursos**: Se prefieren jugadas que no dejen sus propios agujeros más vulnerables, evitando vaciar estratégicamente los que aún pueden ser útiles.

4. **Distribución de semillas**: Si ninguna de las condiciones anteriores se cumple, se escoge el agujero que contenga la mayor cantidad de semillas, buscando generar una distribución más amplia.

5.2 Algoritmo empleado

El algoritmo implementado para el jugador sintético se basa en la técnica de búsqueda `Minimax` con profundidad limitada. Esta estrategia permite analizar distintos escenarios del juego proyectando posibles jugadas del oponente y evaluando sus consecuencias antes de tomar una decisión.

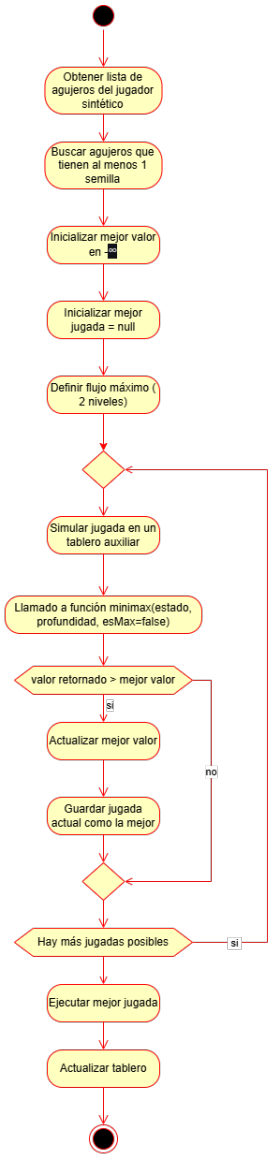
El procedimiento general es el siguiente:

- Se obtiene la lista de agujeros válidos del jugador sintético (que contienen al menos una ficha).
- Para cada jugada posible, se simula el movimiento sobre una copia del tablero.
- Se llama recursivamente a la función `minimax`, alternando entre jugador sintético y oponente, hasta alcanzar una profundidad definida.
- En cada nodo del árbol de decisiones, se calcula un valor que representa qué tan favorable es el estado del tablero.
- Se selecciona la jugada que retorna el valor máximo (en el caso del jugador sintético) o el mínimo (en el caso del oponente).
- Finalmente, se ejecuta la jugada seleccionada y se actualiza el estado real del juego.

Esta estructura permite evaluar estratégicamente cada movimiento posible, priorizando capturas, turnos adicionales y distribución eficiente de fichas. A medida que se incremente la profundidad del análisis o se mejore la función de evaluación, el comportamiento del jugador sintético podrá volverse más complejo y competitivo.

5.3 Diagrama de flujo

En el siguiente diagrama, se presenta el flujo que representa visualmente el proceso de toma de decisiones del jugador sintético. Este diagrama permite entender cómo el sistema analiza las posibles jugadas, en qué orden se evalúan las condiciones y cómo se define finalmente qué movimiento se ejecuta.


Obtener lista de agujeros del jugador sintético
Buscar agujeros que tienen al menos 1 semilla
Inicializar mejor valor en -∞
Inicializar mejor jugada = null
Definir flujo máximo (2 niveles)
Simular jugada en un tablero auxiliar
Llamado a función minimax(estado, profundidad, esMax=false)
valor retornado > mejor valor
si
Actualizar mejor valor
Guardar jugada actual como la mejor
no
Hay más jugadas posibles
si
Ejecutar mejor jugada
Actualizar tablero

6 Pruebas y validación

Para validar el comportamiento del sistema se definieron ciertas características para asegurar que el sistema del juego Mancala funcione satisfactoriamente cumpliendo con los requerimientos funcionales y no funcionales. Para así, identificar y corregir errores que se puedan llegar a presentar, los casos de prueba y estrategia de pruebas que serán implementadas son las siguientes:

6.1 Casos de prueba

1. Se realiza una jugada automática por parte del jugador sintético
    - Objetivo: Verificar el correcto funcionamiento de la IA, al momento de jugar en una partida.
    - Entrada: Turno automático
    - Resultado Esperado: Selección de agujero valido de manera aleatoria
2. Algún jugador escoge un agujero vacío
    - Objetivo: Verificar que el sistema valida el estado de los agujeros del tablero que esta en juego
    - Entrada: Algún hoyo agujero
    - Resultado Esperado: Mensaje en el que se le informa que el agujero esta vació, pidiéndole que digite nuevamente un numero.
3. Selección de agujero fuera de rango
    - Objetivo: Verificar que el sistema restringa los rangos de números que no son validos
    - Entrada: Numero fuera del rango de juego
    - Resultado Esperado: Mensaje de advertencia, en el que se le pide al usuario digitar un numero que corresponda a algún agujero del lado de su tablero
4. Entrada no numérica
    - Objetivo: Verificar que unicamente recibe entradas validas para el juego
    - Entrada: Alguna letra, símbolo, o signo de puntuación.
    - Resultado Esperado: Mensaje de advertencia en el que se solicite que digite una entrada valida.

6.2 Estrategias de prueba

Para asegurar un correcto funcionamiento del sistema y validar que se están cumpliendo satisfactoriamente los requisitos anteriormente definidos, se aplicarán tres estrategias de pruebas:

1. Pruebas Unitarias

   Estas pruebas se aplicarán con el objetivo de validar el comportamiento esperado del sistema de manera individual, para así identificar errores que se puedan llegar a presentar, para esto se tendrá en cuenta:

   - Validación de selección de agujeros correctos teniendo en cuenta el turno del jugador.
   - Detección de jugadas invalidas, como selección de agujeros vacíos o fuera del rango.
   - Calculo correcto de los graneros después de cada jugada.

2. Pruebas de Integración

   Estas pruebas se centrarán en validar la interacción que tendrán las diferentes partes del sistema junto a la interfaz, para esto se tendrá en cuenta:

   - Correcta actualización entre la entrada digitada por el jugador y la actualización del tablero.
   - Gestión automática de los cambios de turnos.
   - Correcta actualización y comunicación entre la jugada realizada y el cambio del estado del juego.

3. Pruebas de Rendimiento

   Estas pruebas se encargarán de evaluar el comportamiento del sistema en diferentes escenarios, para esto se tendrá en cuenta:

   - Simulación de partidas para evaluar la estabilidad del sistema en diferentes contextos
   - Verificación de que el sistema no se ponga mas lento después de varias rondas de juego
   - Medición de tiempo de respuesta, en cada momento que se deba actualizar a interfaz

7 Conclusiones

El documento presenta la arquitectura, diseño e implementación de la primera fase del sistema de juego Mancala, enfocándose principalmente en la construcción de una interfaz funcional en consola. Se definieron los módulos principales del sistema, incluyendo la estructura del tablero, la gestión de turnos y la interacción directa del jugador humano con el entorno del juego.

Uno de los avances más significativos fue la incorporación de un jugador sintético controlado mediante el algoritmo `Minimax` con profundidad limitada. Esta estrategia permite al sistema anticipar escenarios de juego y tomar decisiones óptimas según el contexto, ofreciendo distintos niveles de dificultad que enriquecen la experiencia del usuario.

Se evidencia una clara separación de responsabilidades entre los componentes del sistema, lo cual facilita su comprensión, mantenimiento y futura escalabilidad. Asimismo, se definieron e implementaron estrategias de prueba que garantizan el correcto funcionamiento del sistema en esta etapa inicial.

Con esta entrega, se dispone de una versión funcional que permite jugar partidas entre un humano y un oponente automatizado, estableciendo una base sólida para continuar el desarrollo del proyecto con funcionalidades avanzadas.