Complejidad Temporal, Estructuras de Datos y Algoritmos

Enunciado Trabajo Final

Consideraciones Generales

El objetivo de este trabajo es integrar los contenidos vistos en la materia. El trabajo deberá realizarse en forma individual.

La fecha límite para su defensa se encuentra publicada oportunamente.

Se debe crear un repositorio en **github.com** llamado ***TPF_su_apellido*** en el cual el profesor debe participar como colaborador en el mismo. El repositorio git registrará todo el proceso de codificación y la entrega del trabajo final se realizará a través de dicho repositorio.

El trabajo se presentará junto con un informe que debe incluir:

- Datos del autor del trabajo final.
- Un diagrama de clases UML describiendo la estructura del sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre los objetos.
- Detalles de implementación, como por ejemplo, problemas encontrados y como fueron solucionados, condiciones de ejecución, etc.
- Las imágenes de todas las pantallas que componen el sistema codificado junto con una descripción completa de las mismas.
- Ideas o sugerencias para mejorarlo o realizar una versión avanzada del mismo.
- Una breve conclusión o reflexión de la experiencia adquirida a partir de la realización del trabajo final.

Este informe se evaluará tanto en su contenido como en la forma en el que es presentado y tendrá una nota que afectará la nota final del trabajo.

El trabajo deberá defenderse en un coloquio presencial.

Enunciado

En 1926 John von Neumann creo el teorema **MiniMax** que establece que en los juegos de 2 adversarios de turnos alternos (“ahora tú, ahora yo”), donde cada jugador conoce de antemano los movimientos posibles de su oponente y sus consecuencias, existe una estrategia que permite minimizar la pérdida máxima esperada. Así al analizar cada posible escenario los jugadores consideran todas las posibles jugadas del adversario enfocándose en cuál sería la pérdida máxima que cada una puede acarrear. Entonces el jugador utiliza la estrategia que resulta de la minimización de su máxima pérdida.

El funcionamiento de **MiniMax** puede resumirse como elegir el mejor movimiento suponiendo que el contrincante escogerá el peor para ti. Para ello deberá existir una función heurística que devuelva valores elevados para indicar buenas situaciones, y valores bajos para indicar situaciones desfavorables.

Todos los estados posibles del juego se representan mediante el uso de un árbol general, donde los nodos representan una situación del juego y un nivel contiene todas las situaciones posibles para uno de los jugadores.

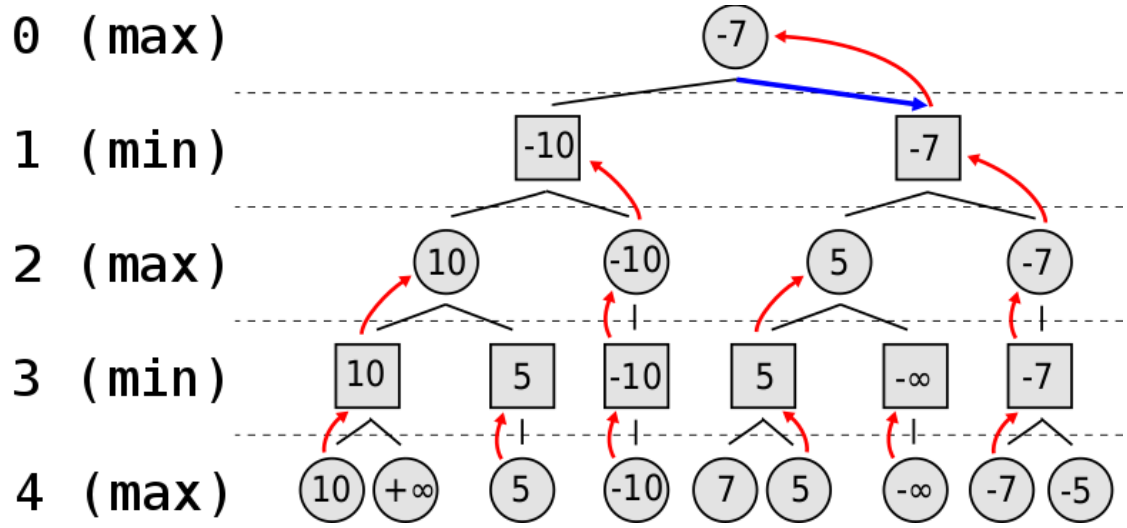

Figura 1: Árbol Minimax que contiene todos los estados posibles para un juego dado.

En la Figura 1 puede verse un árbol **MiniMax** generado para un juego imaginario. Los posibles valores de la función heurística pertenecen al intervalo [-∞,+∞]. De las jugadas posibles del contrincante, éste escogerá las que minimicen nuestra posibilidad de ganar, mientras que de las nuestras escogeremos las que maximicen nuestras oportunidades de alcanzar la victoria. También la Figura 1 explica que el armado del árbol se realiza desde las hojas hacia la raíz, donde primero se calculan los valores para las hojas y luego se va ascendiendo calculando los valores para los ancestros. Los valores de los nodos intermedios se obtienen mediante el cálculo del mínimo o máximo de los valores de sus hijos, dependiendo de si corresponde a una Jugada MIN o a una Jugada MAX respectivamente. Por ultimo en la raíz se ubica la jugada actual.

Una empresa informática está llevando a cabo un juego de computadoras el cuál se desarrolla de la siguiente forma:

1. Se dispone de un mazo con 12 cartas numeradas de 1 a 12 que se reparten en cantidades iguales y de manera aleatoria entre 2 jugadores.

2. El juego fija un límite máximo del cual los jugadores no pueden pasarse.
3. Los jugadores juegan una vez por turno y en cada uno se tiene que descartar una carta.
4. El descarte va formando un montículo cuyo valor es la suma de las cartas que lo integran. El montículo de descarte inicialmente está vacío y su valor es 0.
5. El jugador que incorpore la carta al montículo que haga que el valor del mismo supere el límite fijado es aquel que pierde el juego.

El juego debe estar construido de tal manera que uno de los jugadores posea inteligencia artificial y el otro le solicite la carta a jugar al usuario.

Lo han contratado a Ud. para implementar un **oponente con inteligencia artificial** que esté basado en **árboles MiniMax**.

Los requisitos definidos para el desarrollo de su tarea son los siguientes:

1. Inicialización: Al momento de iniciado el juego se deberá construir el **árbol MiniMax** que contendrá todos los posibles estados del juego.

2. Comenzar un nuevo juego: Se deberá permitir comenzar y desarrollar una nueva partida contra el oponente con inteligencia artificial siempre que el usuario del juego lo desee.

3. Consultas: El sistema debe ser capaz de responder a las siguientes consultas en cualquier punto de su ejecución:

   a. Imprimir todos los posibles resultados desde el punto en que se encuentre el juego.

   b. Dado un conjunto de jugadas imprimir todos los posibles resultados.

   c. Dada una profundidad imprimir las jugadas a dicha profundidad.

Las clases con la implementación del juego provista por la catedra a fin de simplificar la construcción del mismo, deben obtenerse con el comando de git:

```
git clone https://github.com/petinato54/cteda2020_TPF.git
```

A continuación se detallan los métodos de la clase **ComputerPlayer** que Ud. deberá implementar:

Métodos
– **void incializar(List<int> cartasPropias, List<int> cartasOponente, int limite)**: Es invocado por el juego al inicio del mismo y le envía como parámetro las cartas que posee el jugador y las que posee su oponente.
– **int descartarUnaCarta()**: Cuando le llega el turno de jugar una carta a un jugador el juego invoca este método.
- **void cartaDelOponente(int carta):** Avisa al jugador que carta jugó el contrincante.

<u>Aclaración 1:</u> La clase <u>Juego</u> implementa el método main que inicia el juego.

<u>Aclaración 2:</u> Toda entrada/salida debe se realiza a través de la consola.