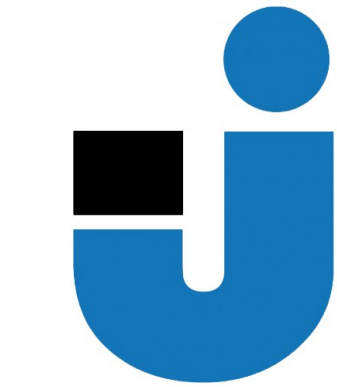


Juego de cartas, Inteligencia Artificial vs Humano

Complejidad Temporal, Estructuras de datos y algoritmos
Trabajo Práctico Final

Julio 2020

***Autor**: Maximiliano Gabriel Ortiz*
***Dni:** 38950523*
***Profesor:** Dr. Ing. Leonardo Javier Amet*

Índice

1. ***Introducción*** *3*

2. ***Diagrama UML*** *3*

3. ***Detalles de implementación*** *4*
   - ***3.1.*** *Condiciones de ejecución* *4*
   - ***3.2.*** *Problemas encontrados* *4*
   - ***3.3.*** *Herramientas utilizadas* *9*

4. ***Descripción de uso del sistema*** *10*

5. ***Sugerencias*** *14*

6. ***Conclusión*** *14*

1. *Introducción*

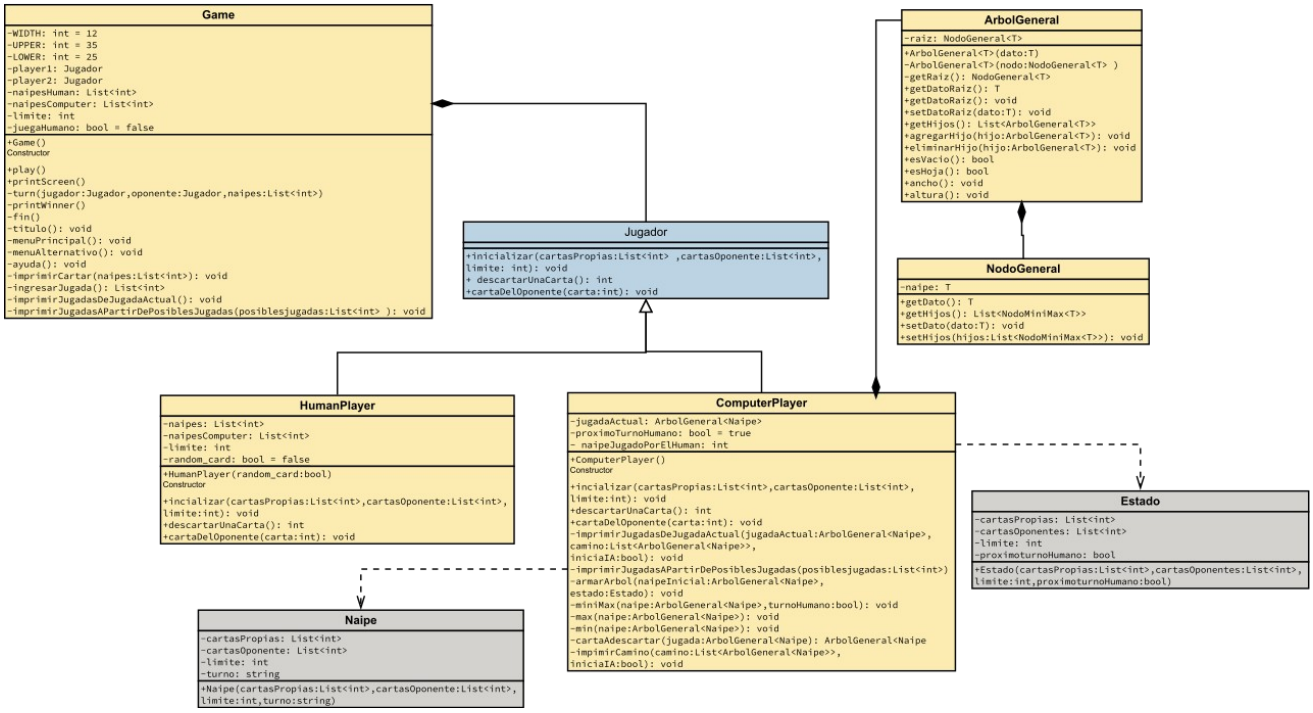
En el marco de la materia , se pide el desarrollo de un juego de computadora compuesto por dos jugadores que tendrán en su poder 6 cartas repartidas de forma aleatoria de un mazo con 12 cartas enumeradas del 1 al 12; a través de turnos alternos, "ahora tu", "ahora yo" deberán ir descartándolas, formando un montículo donde su valor es la suma total de todas las cartas descartadas, el mismo tendrá previamente asignado un valor limite que hará perdedor al jugador que primero lo supere.

El juego deberá ser construido de tal manera que uno de los jugadores posea Inteligencia Artificial (IA).

El problema será abordado a partir de teorema MiniMax creado en 1926 por el matemático John von Neuman.

2. *Diagrama UML*

A continuación se provee un diagrama UML donde se describen las clases participantes y sus relaciones.



3. *Detalles de Implementación*

Para el desarrollo del proyecto la cátedra de la materia provee un conjunto de clases para facilitar la construcción del mismo y que el alumno se centre en el desarrollo de las tareas solicitadas: Implementar los métodos de la Clase ComputerPlayer que será el jugador con inteligencia artificial, permitir al usuario iniciar una nueva partida siempre que él lo desee, permitir que el sistema sea capaz de responder a un conjunto de consultas que le servirán de ayuda en cualquier punto de su ejecución.

Se utilizo las clase de la practica Arboles Generales como estructura de datos para la implementación de un árbol Minimax que contenga todas las posibles jugadas.

3.1 *Condiciones de ejecución*

Al inicializar el juego el primer turno corresponde al humano, las consultas tomaran e iniciaran sus impresiones a partir de la última carta tirada, es decir , la IA.

3.2 *Problemas encontrados*

A continuación se detallara los problemas que fueron surgiendo en el desarrollo del proyecto:

En Primer lugar al momento de implementar el método armarArbol (figura 1) de ComputerPlayer surgió un problema con la actualización del estado que recibía cada jugada correspondiente al árbol, es decir, en primer lugar se definió las variables del estado a actualizar dentro del mismo método, permitiendo completar un camino solo, ya que al volver de la recursión el estado quedaba incorrecto.

Solución: Se creó una Clase Estado(figura 1.a) con toda la información de un nuevo estado, y cada vez que se necesitaba pasar el nuevo estado a su jugada correspondiente se le pasaba como parámetro al método armarArbol una instancia del nuevo estado correctamente actualizado.

```
public class Estado
{
    List<int> cartasPropias;
    List<int> cartasOponente;
    int limite;
    bool proximoturnoHumano;

    3 referencias
    public Estado(List<int> cartasPropias, List<int> cartasOponente, int limite, bool turnoHumano) {
        this.cartasPropias = cartasPropias;
        this.cartasOponente = cartasOponente;
        this.limite = limite;
        this.proximoturnoHumano = turnoHumano;
    }
```

figura 1.a

```
private void armarArbol(ArbolGeneral<Naipe> naipeInicial, Estado estado) {

    //turno Humano
    if (estado.getProximoturnoHumano()) {
        //Si el humano tiene cartas para jugar, se las agrego al naipeInicial
        if (estado.getCartasOponente().Count != 0) {
            foreach (int carta in estado.getCartasOponente())
                naipeInicial.agregarHijo(new ArbolGeneral<Naipe>(new Naipe(carta, 0)));
        }
        //Para cada naipe correspondiente a la jugada del humano,
        foreach (ArbolGeneral<Naipe> naipeActual in naipeInicial.getHijos()) {
            int nuevoLimite = estado.getLimite() - naipeActual.getDatoRaiz().getCarta();
            //verifico
            //si el nuevo limite no es caso base
            if (nuevoLimite >= 0) {
                //  actualizo su estado e inicio una llamada recursiva armarArbol, con el naipeActual y su estado actualizado
                //  al volvel de la recursion asigno el valor de la funcion heuristica a los nodos intermedios hasta su raiz.
                List<int> nuevasCartasOponente = new List<int>();
                foreach (var carta in estado.getCartasOponente()) {
                    if (carta != naipeActual.getDatoRaiz().getCarta())
                        nuevasCartasOponente.Add(carta);
                }
                Estado nuevoEstado = new Estado(estado.getCartasPropias(), nuevasCartasOponente, nuevoLimite, false);
                armarArbol(naipeActual, nuevoEstado);
                miniMax(naipeActual, estado.getProximoturnoHumano());
            }
            //si es caso base
            else
                //setea el valor heuristico de las jugadas terminales
                naipeActual.getDatoRaiz().setValorFuncionHeuristica(+1);
        }
    }
    //turno IA
    //El algoritmo es lo mismo que en el turno del humano, con la diferencia que utiliza las cartas de la IA
```

figura 1

En segundo lugar, el problema que surgió corresponde a un caso especifico de uso, es decir, cuando el humano tira una carta donde su valor heurístico es igual a -1 significa que todos sus hijos tendrán heurística -1, ósea, todas las posibles jugadas de la IA favorecen al humano, siempre y cuando el mismo calcule bien sus próximas jugadas.

Al principio el criterio de selección de la IA, bajo ese contexto eran nulas, simplemente se tiraba la primera carta que tenía en su lista de naipes.

Solución: Se implemento un método llamado cartaADescartar (figura 2.a) , que es invocado por descartarUnaCarta (figura 2) cuando se daba la condición descripta anteriormente, este método lo que hace es recorrer las posibles cartas a tirar por la IA en su próximo turno y por cada una de ellas recorre sus hijos (posibles jugadas del humano) analizando y retornando la jugada IA que tenga la mayor cantidad de posible jugadas del humano que le favorezcan. A continuación de mostrara un ejemplo de uso del sistema donde se evidencie lo descripto

```
public override int descartarUnaCarta() {

    Console.ForegroundColor = ConsoleColor.Yellow;
    int naipeADescartar = 0;
    List<ArbolGeneral<Naipe>> jugadas = jugadaActual.getHijos();

    //Para cada jugada correspondiente a las posibles jugadas que puede tirar la IA
    foreach (var jugada in jugadas) {
        //si la carta tirada por el humano es igual a la de la jugada y ademas tiene valor heuristico -1
        //muestro naipes, e inicio una busqueda de la carta que mejor le convenga a la IA actualizando la jugada actual para el proximo turno
        if (jugada.getDatoRaiz().getCarta() == naipeJugadoPorElHumano && jugada.getDatoRaiz().getValorFuncionHeuristica() == -1) {
            Console.WriteLine("");
            Console.Write("Naipes disponibles (IA):      ");
            foreach (var carta in jugada.getHijos()) {
                Console.Write(carta.getDatoRaiz().getCarta() + ", ");
            }
            ArbolGeneral<Naipe> naipeAuxADescartar = cartaAdescartar(jugada);
            this.jugadaActual = naipeAuxADescartar;
            naipeADescartar = naipeAuxADescartar.getDatoRaiz().getCarta();
            Console.Write("     jugada IA: " + naipeADescartar);
            Console.ForegroundColor = ConsoleColor.White;
            return naipeADescartar;
        }
        //si no,
        else {
            //verifico que la carta sea igual a la que tiro el humano, y tiro la carta que mejor le convenga a la IA,
            //actualizando la jugada actual para el proximo turno
            if (jugada.getDatoRaiz().getCarta() == naipeJugadoPorElHumano) {
                foreach (var naipeAJugar in jugada.getHijos()) {
                    if (naipeAJugar.getDatoRaiz().getValorFuncionHeuristica() == 1) {
                        naipeADescartar = naipeAJugar.getDatoRaiz().getCarta();
                        Console.Write("     jugada IA:   " + naipeADescartar);
                        this.jugadaActual = naipeAJugar;
                        Console.ForegroundColor = ConsoleColor.White;
                        return naipeADescartar;
                    }
                }
            }
        }
    }
    Console.ForegroundColor = ConsoleColor.White;
    return naipeADescartar;
}
```

Figura 2

```
private ArbolGeneral<Naipe> cartaAdescartar(ArbolGeneral<Naipe> jugada) {
    int positivo = 0, negativo, positivoMaximo = 0;
    ArbolGeneral<Naipe> naipeADescartar = jugada.getHijos()[0];
    foreach (var naipe in jugada.getHijos()) {
        positivo = 0; negativo = 0;
        foreach (var n in naipe.getHijos()) {
            if (n.getDatoRaiz().getValorFuncionHeuristica() == 1)
                positivo++;
            else
                negativo++;
        }
        if (positivo > negativo) {
            if (positivo > positivoMaximo) {
                positivoMaximo = positivo;
                naipeADescartar = naipe;
            }
            if (positivoMaximo == naipe.getHijos().Count() - 1) {
                return naipeADescartar;
            }
        }
    }
    return naipeADescartar;
}
```

Figura 2.a

Caso de uso del problema:

- La figura 3 muestra las posibles jugadas que puede realizar el humano , evidenciando que si opta por una carta con heuristica -1 existe posibilidad de que gane.
- En la figura 3.a el humano selecciona la carta 7, por su parte la IA al evidenciar el caso de uso, inicia su criterio de selección y opta por la carta 8; mostrándose en la figura 3.b como la jugada actual.
- La figura 3.c contiene las jugadas que el humano podría realizar, haciendo evidente que la selección de la IA entorpece la decisión del humano, ya que, si el mismo no calcula bien o simplemente opta por el azar, existe una mayor probabilidad de que con su jugada favorezca a su oponente.

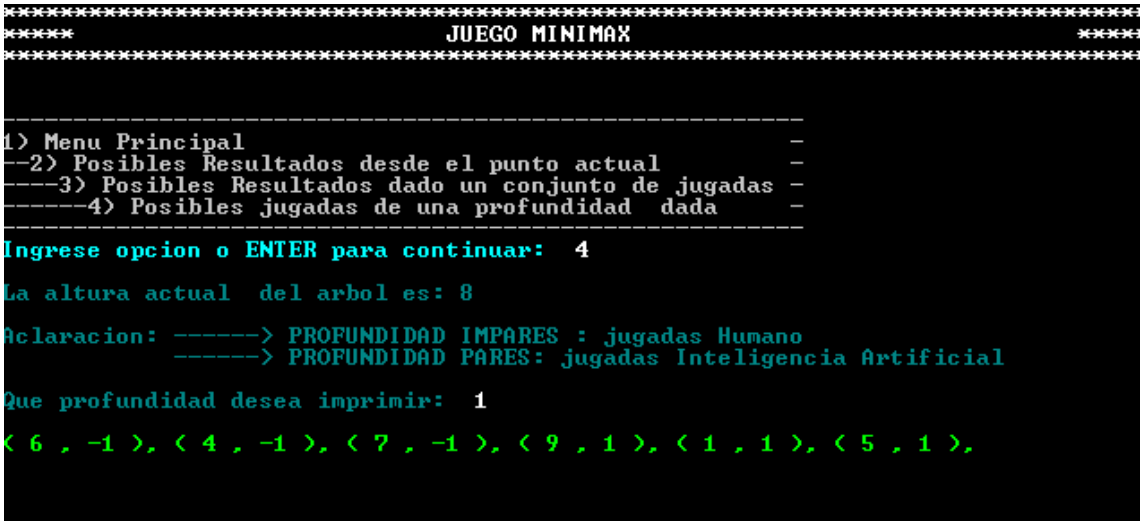


Figura 3

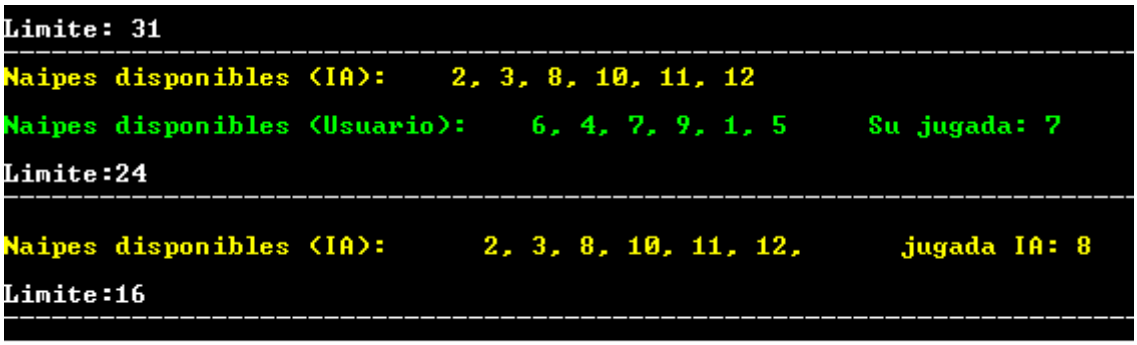


Figura 3.a

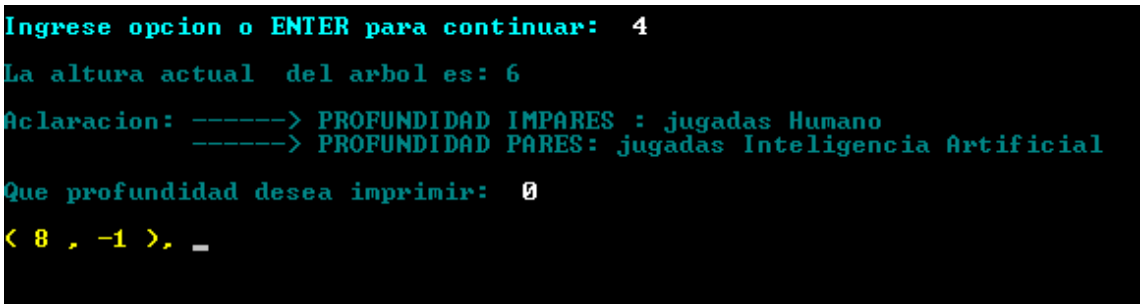


Figura 3.b

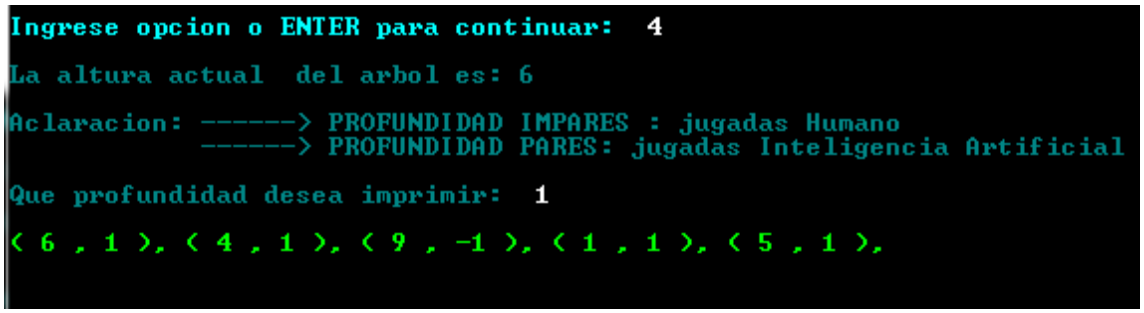


Figura 3.c

3.3 *Herramientas y tecnologías utilizadas*

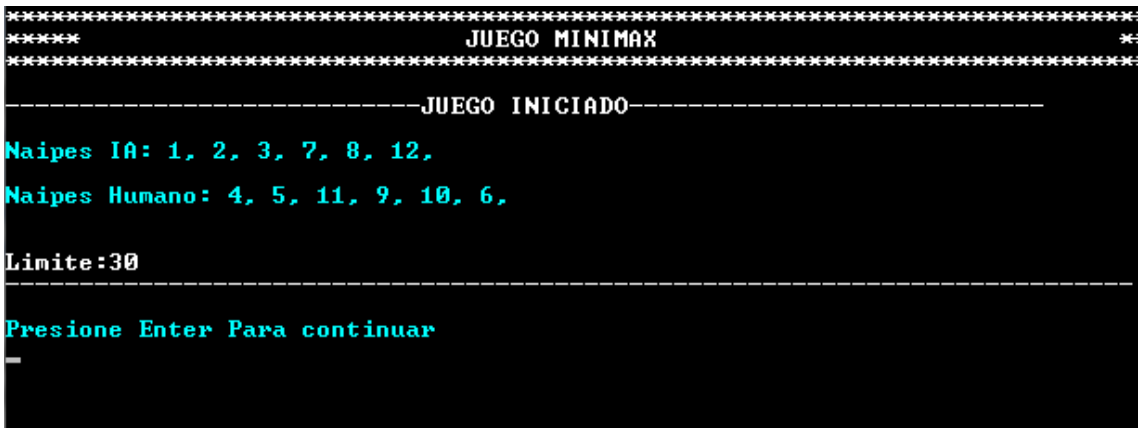
- **Tecnologías*:***
  - IDE Microsoft Visual Studio 2019.
  - Plataforma .NET Core 2.1
  - Lenguaje de programacion C#
  - Git como control de versiones.
- **Herramientas:**
  - GitHub como repositorio remoto
  - DIA para el diseño de UML
  - Microsoft Word para el desarrollo del presente informe

4 *Descripción del sistema en uso.*

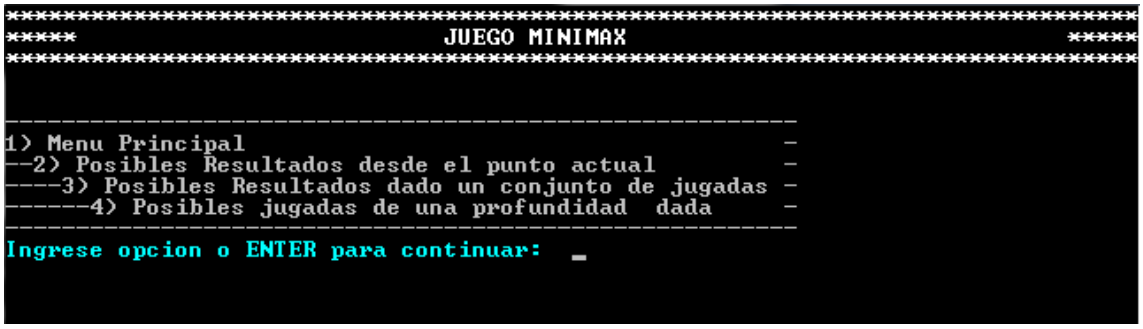
A continuación se presentaran mediantes imágenes el funcionamiento del sistema

- **Menú principal**

- **Opción 1) Nueva Partida**: se muestra los valores iníciales del juego



A continuación se muestra el menú alternativo que el usuario tendrá disponible y podrá seleccionar cualquier opción o simplemente continuar con el juego



- **Desarrollo normal de una partida**

Al decidir continuar, el usuario visualiza toda la información necesaria para decidir qué carta seleccionar. Seguido de su acción, se actualiza el límite y continua la Inteligencia Artificial.

Luego el usuario tendrá la opción de hacer alguna consulta o simplemente seguir jugando.

Si decide continuar tendrá toda la información actualizada.

Desarrollo del submenú (los datos son de una nueva partida)

- **Opción 2**

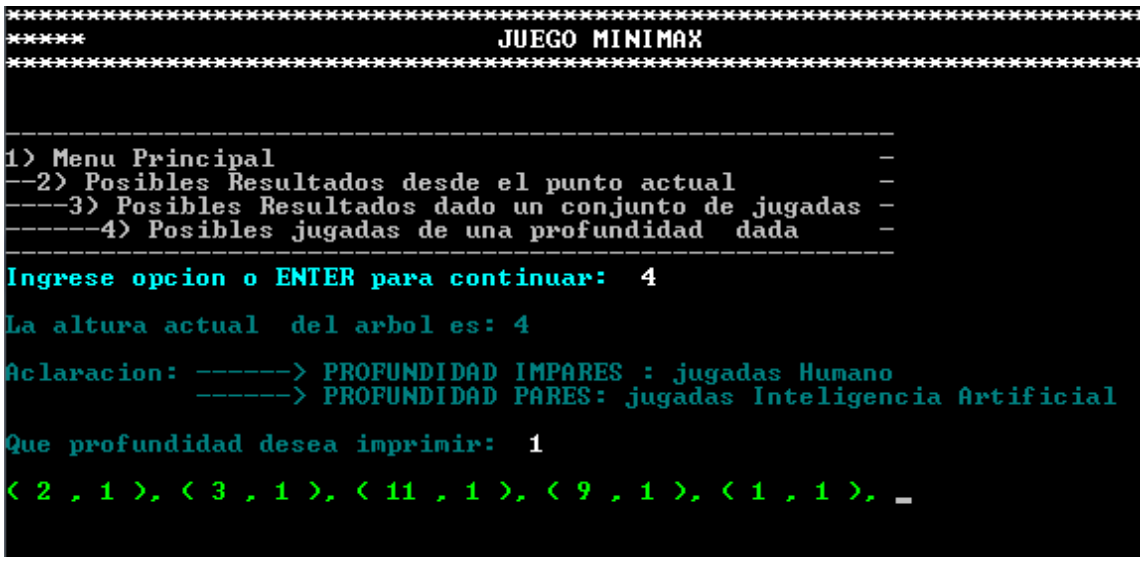
Se imprime las posibles jugadas partiendo desde la jugada que hiso la IA.

- **Opción 3**

Se muestran las cartas de ambos jugadores y se pide que ingrese una secuencia de jugadas en el formato solicitado.

- **Opción 4**

La altura del árbol se calcula desde la última jugada, en este caso se solicito las posibles jugadas del humano.



5. *Sugerencia para una nueva versión*

A continuación se expone una sugerencia con su posible resolución para versiones posteriores.

Sería interesante desacoplar la interfaz de usuario de la lógica del juego y así poder dividir las responsabilidades en dos capas

En primer lugar, se quitaría todo manejo de impresiones en consola, dejando solamente el funcionamiento interno. En segundo lugar, se debería exponer solo aquellos métodos que impliquen funcionalidades concretas con la lógica del juego. Por último, desarrollar una interfaz grafica(web, escritorio, etc.) que consuma esos métodos.

6. Conclusión

Creo que el proyecto nos permite sumar experiencias tanto en programación, como también en el manejo de información atreves de estructuras de datos de una forma completamente distinta a como se venía realizando en materias anteriores. Además nos permite explorar ramas sumamente interesantes de la informática como la Inteligencia Artificial. En lo personal, la materia nos brinda un afán de herramientas que nos permite desarrollar en sentido crítico a la hora de tomar decisiones sobre cómo manejar la información de nuestro desarrollos según el contexto del mismo.