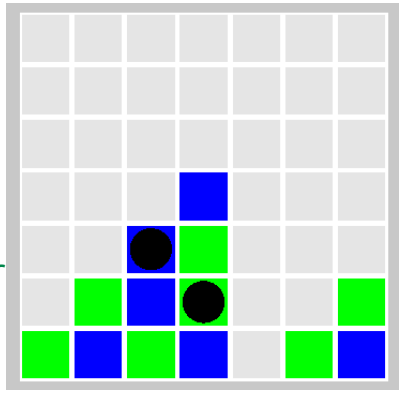
> 1. Entendiendo el problema. Objetivos.

En esta práctica se propone hacer una implementación de un agente capaz de jugar al DesConecta4-Boom, una variante de Conecta-4 cuyo objetivo es exactamente el contrario al mencionado: Evitar a toda costa conectar 4. Para ayudar con el proceso, existen las fichas bomba, capaces de destruir una fila de fichas del color del jugador.



El tablero se compone de una matriz de 7 filas y 7 columnas, y los jugadores proceden a colocar sus fichas (siendo el Jugador 1 en Verde y el 2 en Azul) en sus turnos, siendo primero el Jugador 1 y luego el Jugador 2. Cada 5 movimientos los contendientes tendrán en su poder una ficha bomba (si no la tienen todavía, bien por ser la primera vez o porque la han explotado) con las propiedades mencionadas previamente.

> 2. ¿Y cómo resuelvo esto?. Heurísticas.

Al principio me puse a jugar para sacar estrategias de juego. De esas partidas pude sacar una cosa en claro: Intentar no apilar más de dos o tres fichas de forma horizontal , vertical o diagonal. Cuantas más hubiera, más importancia le daba para que el contrincante me ganara.

Entonces, lo más prudente sería analizar cada una de mis fichas para ver mis adyacencias. Con esa idea, decidí implementar varios bucles para ir viendo a los lados y a las diagonales. Quedando al final una estructura de la siguiente forma.

```
Mientras haya filas/columnas
    si fila+1/columna+1/fila y columna +1 tiene una casilla del mismo jugador
        incremento en 1 las fichas unidas horizontal/vertical/diagonalmente.
```

Esto al principio funcionaba… más o menos. Lograba vencer al ninja como Jugador 1, pero no como Jugador 2. Pensaba que al ver solo las fichas que tenia en el tablero podría valorar qué hacer, pero evidentemente no tomaba en cuenta aspectos como ¿y si tengo apiladas varias de estas fichas?¿y si acaso algún cambio en el tablero como podría ser la explosión de una ficha enemiga me haría perder la partida, como iba pasando en algunas ocasiones?
En ese caso, me vi con la idea de tener que dar más valoración a ciertos casos donde iba teniendo fichas cercanas.
Dicho esto, valoraríamos más los casos en los que las fichas unidas son 2 o 3, para ver su determinación de victoria/derrota. La puntuación es la suma de fichas del jugador en el tablero, con salvedades:
- Si hay una unión de 2 vértices, se multiplica la puntuación por 2 veces las fichas en vertical/horizontal/diagonal, según la zona que corresponda.
- Lo mismo aplica para uniones de 3 vértices, multiplicando por 3.

Con esa implementación era capaz de derrotar a los Ninjas, pero el puntaje quedaba muy raro,y además tardaba muchos turnos. Resultaba que era por la valoración de las fichas cercanas, la cual no se reiniciaba cuando el tablero no encontrara una ficha suya, no contando bien. Por ello, se pone que si la cuenta de fichas adyacentes sale mayor de 3 (ya que los atributos son comunes a los bucles), se resetee el contador y que sume lo que ya hay, haciendo que el máximo de puntos por alineamiento de fichas juntas en una linea sea 13 (4 por ser dos alineadas y 9 por ser tres).
De esa forma se ha logrado reducir en parte el número de turnos, pero a costa de que tarde más por turno ya que tener que evaluar 8 bucles por movimiento es costoso en memoria.

3. Algoritmo de poda alfa-beta. Implementación y uso con las heurísticas.

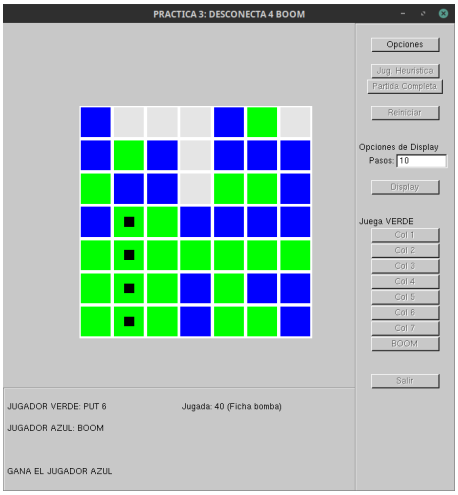
Para saber dónde sacar el valor máximo o mínimo del árbol, primero haremos una valoración de los nodos. Esto podemos verlo en 2 fases :
1. La primera es saber quien es el jugador y quien es el contrincante, teniendo para ello una comprobación previa basada en el parámetro jugador.

2. La segunda es saber en qué situación estamos :
- ¿El jugador gana? Puntuamos a +inf.
-¿El jugador pierde? Puntuamos a -inf.
- ¿No ha pasado nada de lo anterior y hemos rellenado la matriz? Puntuamos 0, empate.
- ¿Nada de lo anterior? Puntuaremos según el método del análisis del Entorno del jugador y del contrincante. Se penalizará el hecho de que el jugador tenga muchas fichas y/o estén juntas en grupos de 2 o 3, y se beneficiará de que el contrincante haga eso mismo.

Con esto hecho, solo queda implementar el algoritmo de poda alfa-beta, ya estudiada en clase.

4. Poniendo a prueba. Agente vs Ninja.

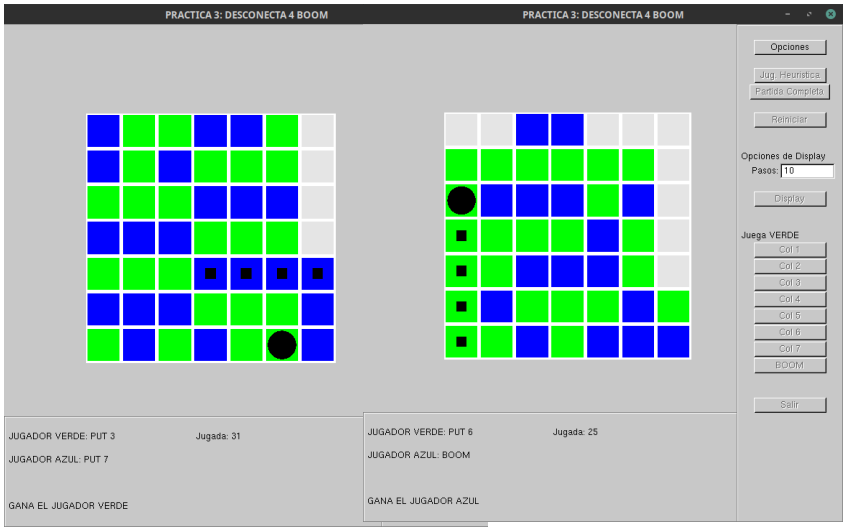
Para probar las habilidades del agente, se ha enfrentado al Ninja habilitado en la sección "vs Ninja" de la práctica; primero como Jugador 1 y luego como Jugador 2.

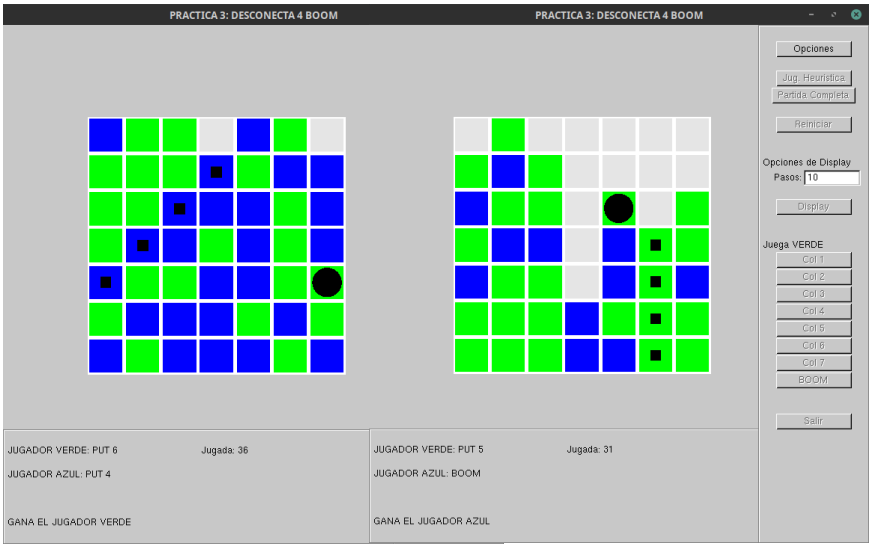


Una cosa que pasó durante las pruebas con la heurística fue el hecho de que salió esta jugada, donde en teoría esta vence al ninja, pero conecta 4 durante la explosión. Al hablarlo con el profesor, se determinó que realmente habría perdido ya que al efectuar el movimiento este causaria su desplazamiento.

Para evitarlo, puse en la función Think el hecho de que nada más saber que tendía a ganar, y si tuviera la bomba, que la explotara. Al parecer hace caso omiso, pero al final se logró paliar el problema. Además de que sube el numero de movimientos , pero su cálculo es más rápido.

Se adjuntan capturas de la resolución de la heurística contra el ninja 1,2 y 3 (de arriba hacia abajo); siendo el Jugador 1 (izquierda) y el Jugador 2 (derecha).


PRACTICA 3: DESCONECTA 4 BOOM
PRACTICA 3: DESCONECTA 4 BOOM
Opciones
Jug. Heuristica
Partida Completa
Reiniciar
Opciones de Display
Pasos: 10
Display
Juega VERDE
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7
BOOM
Salir
JUGADOR VERDE: PUT 3
Jugada: 31
JUGADOR AZUL: PUT 7
GANA EL JUGADOR VERDE
JUGADOR VERDE: PUT 6
Jugada: 25
JUGADOR AZUL: BOOM
GANA EL JUGADOR AZUL


PRACTICA 3: DESCONECTA 4 BOOM
PRACTICA 3: DESCONECTA 4 BOOM
Opciones
Jug. Heuristica
Partida Completa
Reiniciar
Opciones de Display
Pasos: 10
Display
Juega VERDE
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7
BOOM
Salir
JUGADOR VERDE: PUT 6
Jugada: 36
JUGADOR AZUL: PUT 4
GANA EL JUGADOR VERDE
JUGADOR VERDE: PUT 5
Jugada: 31
JUGADOR AZUL: BOOM
GANA EL JUGADOR AZUL