HEURÍSTICA:

ONITAMA**:**

- Para el desarrollo del código del juego Onitama, hemos utilizado el algoritmo poda alfa-beta. El algoritmo anterior es una modificación optimizada del algoritmo MINIMAX en el que, a partir de un estado, que en este caso es un tablero y unas cartas, se elige el mejor movimiento teniendo en cuenta los posibles movimientos del jugador contrario hasta alcanzar el estado final o el fin del juego.
- Se utiliza una función de evaluación llamada “eval”, con la que a partir de un nodo como argumento de entrada se devuelve cuánto de bueno es ese nodo o, mejor dicho, cuánta probabilidad tienes de ganar a partir de él.
- La heurística utilizada es la siguiente:

100x(maestroJugador – maestroEnemigo) + 10x(peonesJugador - peonesEnemigo)

- EXPLICACIÓN VARIABLES:
- *maestroJugador (0-1): Representación de la existencia del maestro del jugador en el tablero.*
- *maestroEnemigo (0-1): Representación de la existencia del maestro del enemigo en el tablero.*
- *peonesJugador: Número de peones del jugador en el tablero.*
- *peonesEnemigo: Número de peones del enemigo en el tablero.*

Si se analiza de una forma más detallada la heurística, se puede observar cómo cuanto mayor sea la heurística, mayor será la probabilidad de vencer al jugador contrario debido a que tendrá una mayor diferencia de piezas en el tablero. Además, se le da una mayor importancia al maestro (100) ya que en el caso de que uno de los dos maestros no esté en el tablero significará que dicho jugador pierda o gane la partida en función de si es su maestro o el del adversario.

Para finalizar, cabe destacar que el juego Onitama es un juego de suma cero. Con lo cual, a partir de la heurística anterior, el valor del nodo será el mismo, pero de signo contrario en el caso de que con el mismo tablero (nodo) utilicen la función “eval” ambos jugadores.

```
def eval (self):
    contadorJ = contadorE = 0
    mJ = mE = 0
    if self.jugador == 0:
      peonJugador = 'w'
      peonEnemigo = 'b'
      maestroJugador = 'W'
      maestroEnemigo = 'B'
    else:
      peonJugador = 'b'
      peonEnemigo = 'w'
      maestroJugador = 'B'
      maestroEnemigo = 'W'
    for i in range (len(self.estado)):
        for j in range(len(self.estado[i])):
            if self.estado[i][j] == peonJugador:
                contadorJ = contadorJ + 1
            elif self.estado[i][j] == maestroJugador:
                mJ = 1
            elif self.estado[i][j] == maestroEnemigo:
                mE = 1
            elif self.estado[i][j] == peonEnemigo:
                contadorE = contadorE + 1

    return 100*(mJ-mE) + 10*(contadorJ - contadorE)
```