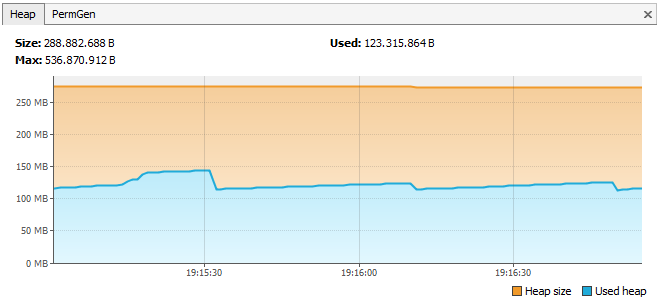
**CAMBIOS EN LA IMPLEMENTACIÓN DEL ALGORÍTMO MINIMAX**

**Versión 1:**

Primero se calculaban todas las posibles jugadas para todos los estados posibles, dada una profundidad o tiempo. Con ellas, se armaba el árbol de jugadas y recién después se buscaba la mejor jugada por profundidad.

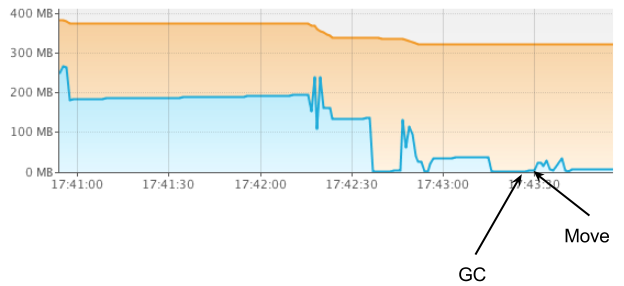
CONTRA: La complejidad del algoritmo era muy alta, teníamos demasiadas instancias de la clase Punto (Posición del tablero) lo que generaba que la memoria se sature para profundidad mayor que dos.



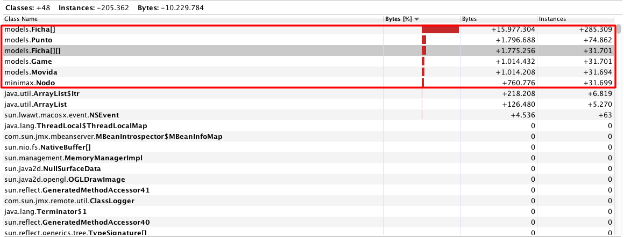
**Versión 2:**

Cambiamos el algoritmo para no tener todo el árbol en memoria cargado. Se calculan todas las posibles movidas para un nodo y se itera sobre ellas. Se crea un nodo para la primera movida y se llama de manera recursiva a la misma función hasta que devuelve el mejorHijo. Ese nodo se guarda en el nodo padre que generó la recursión y se sigue iterando sobre las movidas comparando contra este mejorHijo. A lo último se devuelve el mejorHijo con la mejor jugada.

De esta manera evitamos mantener todo el árbol de jugadas en memoria. Se puede observar en las capturas como el uso de la memoria bajó considerablemente, de la misma forma que lo hicieron las instancias de la clase Punto.



En la imágen se observa un llano ya que el Garbage Collector estaba actuando sobre las instancias. Entre los dos llanos se observan tres picos que están mostrando el desarrollo de toda una jugada (Move): ejecuta movida del jugador y luego la movida de la máquina.



**CAMBIOS EN LA HEURÍSTICA**

**Prueba 1 (Versión 1 de Minimax):**

El valor heurístico es calculado restándole la cantidad de guardias a la cantidad de enemigos.

PRO: De ésta forma se priorizan los estados de tablero en los que se contribuye a la eliminación de guardias, dado que la resta entre enemigos y guardias da mayor.

CONTRA: El problema es que ante la posibilidad de eliminar al rey, la computadora no toma ninguna decisión al respecto.

**Prueba 2 (Versión 1 de Minimax):**

Se intenta mejorar el valor calculado en la prueba anterior. Se cuenta la cantidad de fichas enemigas que hay en un radio de dos casilleros alrededor del rey, llamamos a esta cantidad "bloqueos". Se le suma al anterior valor heurístico los bloqueos multiplicados por dos.

PRO: el valor heurístico es una combinación lineal que da más peso a los tableros que posee mayor cantidad de fichas enemigas cerca del rey. Es decir que se priorizan los tableros que "persiguen" al rey.

CONTRA: Ante la posibilidad de matar al rey cuando queda una posición libre en un radio de un casillero alrededor del rey, la computadora no toma ninguna decisión.

**Prueba 3 (Versión 1 de Minimax):**

Se mejora la heurística anterior contando la cantidad de enemigos posicionados en un radio de 1 casillero alrededor del rey. Llamamos a esta cantidad "cuadranteMenor".

PRO: Ayuda a encerrar al rey.

CONTRA: Ante la posibilidad de matar al rey, en algunos casos elige estados de tablero que acercan al primer cuadrante, dado que los dos estados tienen igual valor heurístico. El problema es que no le se está dando mayor peso a las posiciones que permiten matar al rey.

**Prueba 4 (Versión 1 de Minimax):**

Se mejora la heurística anterior contando la cantidad de enemigos posicionados en los casilleros adyacentes horizontal y verticalmente al rey. Llamamos a esta cantidad "matarAlRey". Sumamos al valor heurístico que multiplica por 6 la cantidad de matarAlRey.

PRO: Las fichas enemigas persiguen al rey y lo matan de presentarse la oportunidad.

CONTRA: Hay situaciones en las que queda un casillero habilitado para que una filla se pueda acercar al rey (por mas que no lo pueda comer), pero no lo hace.

**Prueba 5 (Versión 2 de Minimax):**

Al cambiar el algoritmo minimax, comienza a priorizar matar guardias, que perseguir al rey. No podemos encontrar explicación a este comportamiento. Es confuso ya que el valor heurístico es una combinación lineal.