**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

(Universidad del Perú, DECANA DE AMÉRICA)

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Logotipo

Descripción generada automáticamente

**“Los Tres Mosqueteros”**

**Asignatura: Inteligencia Artificial**

**GRUPO 05**

**Integrantes:**

Bernabel Mancilla Axell Hernan 19200003

Puyen Huapaya Jafet Raul 19200039

Gutierrez Caruajulca Javier Antonio 19200020

**Docente**

Maguiña Pérez, Rolando Alberto

**Sección**

S1

**Semestre 2022-1**

**Lima, Perú**

**ÍNDICE**

[CAPÍTULO I: LOS TRES MOSQUETEROS 3](#_Toc108387959)

[1.1 Origen 3](#_Toc108387960)

[1.2 Descripción del juego 4](#_Toc108387961)

[1.2.1 Diseño 4](#_Toc108387962)

[1.2.2 Reglas del juego 6](#_Toc108387963)

[1.2.3 Objetivo 9](#_Toc108387964)

[CAPÍTULO II: ESPACIO DE ESTADOS 11](#_Toc108387965)

[2.1 Definición 11](#_Toc108387966)

[2.2 Espacio de estados del juego 12](#_Toc108387967)

[CAPÍTULO III: DISEÑO E IMPLEMENTACIÓN DEL JUEGO 15](#_Toc108387968)

[3.1 Software Utilizado 15](#_Toc108387969)

[3.2 Desarrollo del juego 15](#_Toc108387970)

[3.3.1 Implementación con el algoritmo Minimax 15](#_Toc108387971)

[3.3.2 Implementación con el algoritmo Primero el Mejor 20](#_Toc108387972)

[CAPÍTULO IV: TEST COMPUTACIONAL 24](#_Toc108387973)

[4. 1 Test de Juego con el algoritmo Minimax 24](#_Toc108387974)

[4.2 Análisis y Resultado 28](#_Toc108387975)

[CAPITULO V: CONCLUSIONES 30](#_Toc108387976)

[5.1 Conclusiones 30](#_Toc108387977)

[BIBLIOGRAFÍA 31](#_Toc108387978)

# CAPÍTULO I: LOS TRES MOSQUETEROS

## 1.1 Origen

El Juego de Los Tres Mosqueteros (*Three Musketeers*) es un juego de estrategia abstracta concebido para dos jugadores y disputado sobre un tablero de 5x5 casillas. El juego fue originalmente concebido y diseñado por el israelí Haar Hoolim en la década de los años 50. No obstante, no sería publicado hasta el año 1969, en el libro A Gamut of Games, escrito por el diseñador de juegos norteamericano Sidney Sackson (1920-2002).

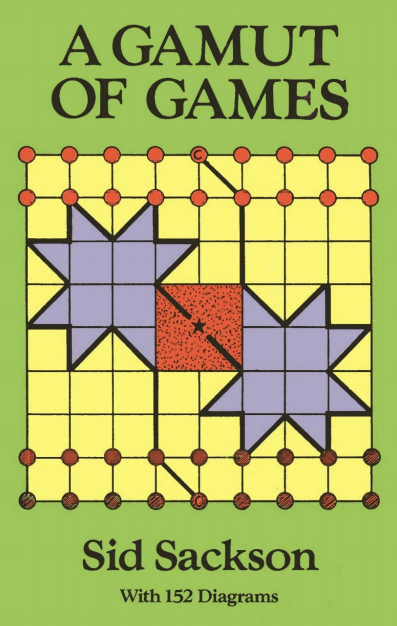


Figura 1  
Portada del libro "A Gamut of Games"

El Juego de Los Tres Mosqueteros se apoya en la conocida novela histórica Los Tres Mosqueteros escrita en el año 1844 por Alejandro Dumas. Basándose en dicha historia, el juego representa el enfrentamiento entre Los Tres Mosqueteros y La Guardia de su principal enemigo, el tirano Cardenal Richelieu, que pretende capturarlos. Siguiendo esta alegoría, el objetivo del juego para Los Tres Mosqueteros es aniquilar a los guardias del Cardenal, mientras que estos pretenden acorralarlos para capturarlos.

## 1.2 Descripción del juego

Una vez que ya conocemos el origen del juego de los “Tres Mosqueteros”, ahora podemos explicar el diseño, sus respectivas reglas y el objetivo del juego.

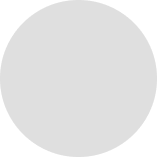
### 1.2.1 Diseño

El Juego de los Tres Mosqueteros es un juego de estrategia en el que participan dos jugadores. Al jugador que utiliza las piezas correspondientes a los Mosqueteros se le denomina “Mosquetero”, mientras al jugador que maneja las piezas correspondientes a La Guardia del Cardenal Richelieu se le denomina “Enemigo”. En la siguiente figura se muestra la representación gráfica de cada tipo de piezas.

Figura 2  
La Guardia



Figura 3  
Los Mosqueteros



Como se explicó previamente, el juego consta de dos jugadores:

* La pieza blanca, la cual representa a “La Guardia”. (Cantidad: 22)
* La pieza negra, la cual representa a “Los Mosqueteros”. (Cantidad: 3)

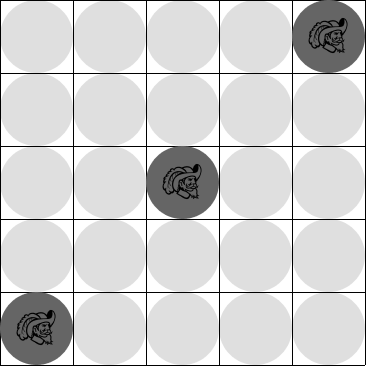
Además, cabe resaltar que existen múltiples variaciones en el diseño de las piezas, en la imagen 3 se puede observar que la pieza negra contiene la imagen de un mosquetero, pero esto se puede excluir, ya que podemos hacer uso de los iniciales de las letras para poder diferenciar a nuestras fichas, por ejemplo:



En este caso la ficha roja contiene la letra “M” haciendo referencia a los mosqueteros, mientras que la ficha azul contiene la letra “G” haciendo referencia a la guardia.

Además, el juego se desarrolla en un tablero de 5 x 5 casillas. Inicialmente, el Mosquetero dispone de 3 piezas situadas en las casillas correspondientes al vértice superior derecho, al vértice inferior izquierdo y al escaque central del tablero. Por su parte, el Enemigo dispone al comenzar la partida de una pieza en las cada una de las restantes casillas del tablero. En la figura siguiente se muestra cuál es la disposición inicial de las piezas sobre el tablero.

Figura 4  
Tablero 5x5: Los Tres Mosqueteros



### 1.2.2 Reglas del juego

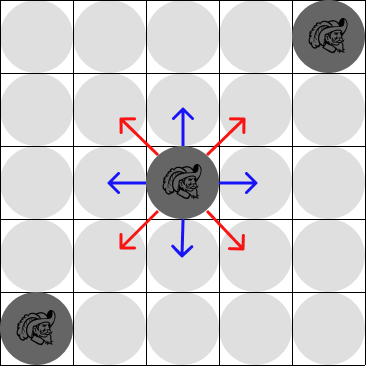
Los distintos movimientos que puede realizar cada jugador dependen del tipo de piezas que manejen.

**Mosquetero:**

Para el jugador Mosquetero, sus piezas sólo pueden ser movidas en dirección vertical u horizontal una sola casilla, siempre y cuando la casilla destino a la que se desea mover la pieza se encuentre ocupada por una pieza del Enemigo. Por lo tanto, no se permiten movimientos en diagonal.

Caso 1.- Existencia de guardias alrededor del mosquetero.

Figura 5  
Existencia de guardias alrededor del mosquetero



Mosquetero Ficha 2

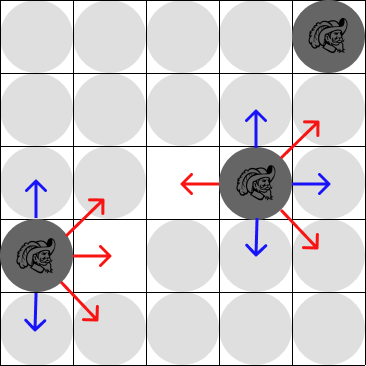
Mosquetero Ficha 3

Mosquetero Ficha 1

Como se puede observar en el “Caso 1”, los movimientos permitidos del mosquetero están definidos por las flechas azules, mientras que los movimientos no permitidos están definidos por las flechas rojas.

Caso 2.- Alrededor del mosquetero no están todos los guardias.

Figura 6  
Alrededor del mosquetero no están todos los guardias



Mosquetero Ficha 2

Mosquetero Ficha 3

Mosquetero Ficha 1

Como se puede observar en el “Caso 2”, los movimientos permitidos del mosquetero están definidos por las flechas azules, mientras que los movimientos no permitidos están definidos por las flechas rojas. Además, este es un caso particular, ya que al realizarse jugadas previas ahora existen casillas las cuales se encuentran vacías, por tal motivo el mosquetero ahora tendrá menos movimientos a realizar.

Finalmente, la pieza del jugador Enemigo que se encuentre situada en la casilla destino del movimiento realizado por el Mosquetero será capturada y eliminada de la partida. Se dice por lo tanto, que el Mosquetero realiza *movimientos de captura* sobre las piezas del adversario.

**Guardia:**

En el caso del Enemigo o Guardia, sus piezas sólo pueden ser movidas en dirección vertical u horizontal una sola posición, de tal manera que la casilla a la que se desea mover la pieza ha de estar vacía. Por lo tanto, las piezas del Enemigo *no realizan nunca capturas sobre las piezas del adversario*. Al igual que ocurría con el otro jugador, no están permitidos los movimientos en diagonal.

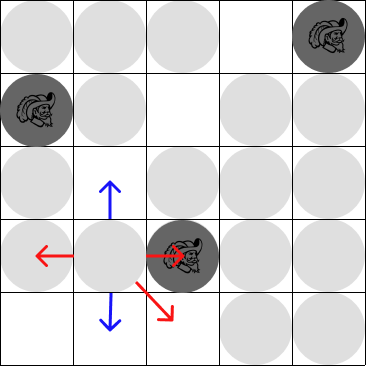


Figura 7  
Movimientos del guardia

Guardia

**Primer movimiento:**

Los movimientos de ambos jugadores se llevan a cabo en turnos alternados. Es decir, en primer lugar mueve un jugador, a continuación el otro y así sucesivamente hasta el final de la partida. Inicialmente, el primer turno corresponde siempre al Mosquetero, ya que al tener el tablero todas las casillas ocupadas, es el único jugador que puede realizar un movimiento.

Imagen que contiene firmar

Descripción generada automáticamente

### 1.2.3 Objetivo

Los objetivos perseguidos por ambos jugadores son contrapuestos.

**Mosquetero:**

Para el caso del jugador Mosquetero, su objetivo es huir de las piezas del Enemigo. Existen dos maneras en las que el Mosquetero puede conseguirlo: capturando todas las piezas sobre el tablero pertenecientes al enemigo; o bien alcanzando un estado en el que, siendo su turno, no pueda realizar ningún movimiento porque no existe ninguna pieza del adversario adyacente a las suyas, es decir, en las casillas a las que podría mover sus piezas.

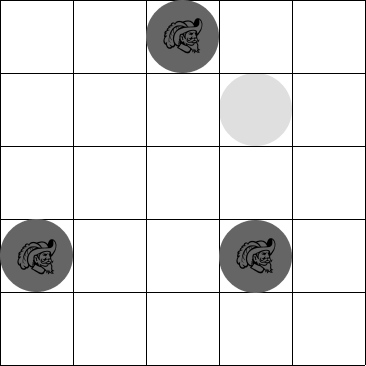


Figura 8  
Victoria del Mosquetero

Por lo tanto, el objetivo del Mosquetero se puede resumir en que persigue alcanzar un posicionamiento de sus piezas durante la partida tal que, siendo su turno, no pueda realizar ningún movimiento.

**Guardia:**

Por su parte, el objetivo que persigue el Enemigo (Guardia) es acorralar y capturar las piezas del Mosquetero, hecho que se produce cuando todas sus piezas quedan situadas sobre una misma columna o fila del tablero de juego. Es decir, si durante el transcurso de la partida se alcanza un estado en el que las 3 piezas del Mosquetero se encuentren situadas sobre una misma columna o fila del tablero, independientemente de a quién corresponda el turno, el Enemigo gana la partida porque es capaz de capturar a los mosqueteros.

Figura 9  
Caso 1: Victoria del Guardia

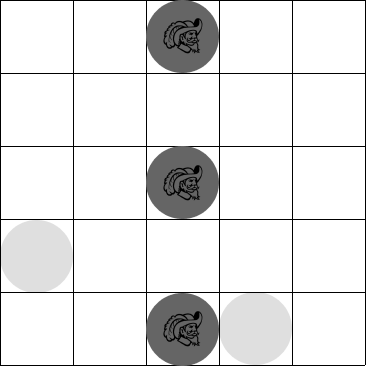
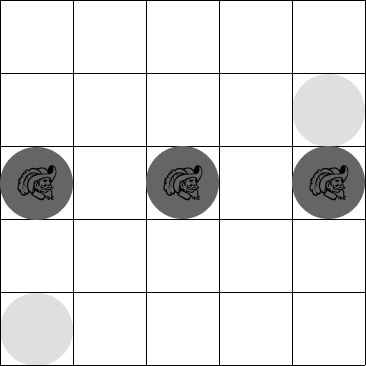


Figura 10  
Caso 2: Victoria del Guardia



# CAPÍTULO II: ESPACIO DE ESTADOS

## 2.1 Definición

Muchos de los problemas que pueden ser resueltos aplicando técnicas de inteligencia artificial se modelan en forma simbólica y discreta definiendo las configuraciones posibles del universo estudiado. El problema se plantea entonces en términos de encontrar una configuración objetivo a partir de una configuración inicial dada, aplicando transformaciones válidas según el modelo del universo. La respuesta es la secuencia de transformaciones cuya aplicación sucesiva lleva a la configuración deseada.

Los ejemplos más característicos de esta categoría de problemas son los juegos (son universos restringidos fáciles de modelar). En un juego, las configuraciones del universo corresponden directamente a las configuraciones del tablero. Cada configuración es un estado que puede ser esquematizado gráficamente y representado en forma simbólica. Las transformaciones permitidas corresponden a las reglas o movidas del juego, formalizadas como transiciones de estado.

Entonces, para plantear formalmente un problema, se requiere precisar una representación simbólica de los estados y definir reglas del tipo condición-acción para cada una de las transiciones válidas dentro del universo modelado. La acción de una regla indica como modificar el estado actual para generar un nuevo estado. La condición impone restricciones sobre la aplicabilidad de la regla según el estado actual, el estado generado o la historia completa del proceso de solución.

El espacio de estados de un juego es un grafo cuyos nodos representan las configuraciones alcanzables (los estados válidos) y cuyos arcos explicitan las movidas posibles (las transiciones de estado). En principio, se puede construir cualquier espacio de estados partiendo del estado inicial, aplicando cada una de las reglas para generar los sucesores inmediatos, y así sucesivamente con cada uno de los nuevos estados generados (en la práctica, los espacios de estados suelen ser demasiado grandes para explicitarlos por completo).

Cuando un problema se puede representar mediante un espacio de estados, la solución computacional corresponde a encontrar un camino desde el estado inicial a un estado objetivo.

## 2.2 Espacio de estados del juego

Como podemos observar en la figura 11, el espacio de estados mostrado solo es una pequeña parte del espacio de estados total, ya que es complicado reducir la ramificación con el uso de la simetría cuando existen múltiples jugadas realizadas por 3 fichas en el caso de los mosqueteros y 22 fichas en el caso de los guardias en un tablero 5x5.

 El factor de ramificación cuando juegan los mosqueteros es 8,6 = 9, es decir, en un turno de los mosqueteros en promedio se generan 9 jugadas posibles.

Figura 11  
Espacio de Estados del Juego de los Tres Mosqueteros

# CAPÍTULO III: DISEÑO E IMPLEMENTACIÓN DEL JUEGO

## 3.1 Software Utilizado

Para el desarrollo de programa se ha utilizado las siguientes herramientas tecnológicas, las cuales son:

* Jupyter Notebook o Jupyter Lab: IDE para la respectiva codificación del código.
* Python 3.x: Lenguaje de programación.
* GitHub: Para el trabajo colaborativo.

## 3.2 Desarrollo del juego

### 3.3.1 Implementación con el algoritmo Minimax

**Algoritmo Minimax:**

El algoritmo Minimax es un algoritmo de búsqueda recursivo para búsqueda en árboles. Este algoritmo consiste en generar un árbol de estados a partir de cierto nodo inicial; este algoritmo analiza los nodos terminales y retornar un valor heurístico de acuerdo con la función definida para el sistema experto.

El pseudocódigo para el algoritmo Minimax es el siguiente:

Texto

Descripción generada automáticamente

Figura 12  
Algoritmo Minimax

Como se observa en el pseudocódigo, el algoritmo devuelve la heurística del nodo hijo cuando la profundidad es la máxima deseada o cuando es un nodo terminal; entiéndase por nodo terminal a aquel estado del tablero de juego el cual no tiene movimientos posibles.

La implementación final del algoritmo minimax es el siguiente:

Texto

Descripción generada automáticamente

Figura 13  
Implementación en Python

**Función Heurística:**

La dificultad de implementa el algoritmo Minimax radica en definir una función heurística correcta que determina el valor útil de cierto estado de tablero de acuerdo con el entendimiento real o humano del juego.

No existe un algoritmo para crear una función heurística ni un método determinante. La función heurística que se utilizó fue la siguiente:

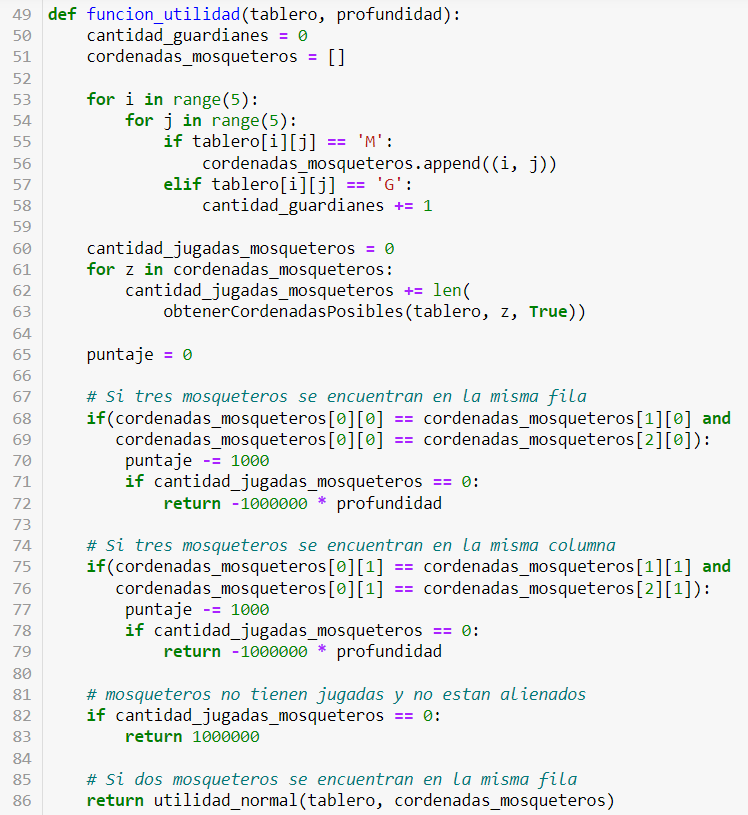


Figura 14  
Implementación en Python de la función heurística

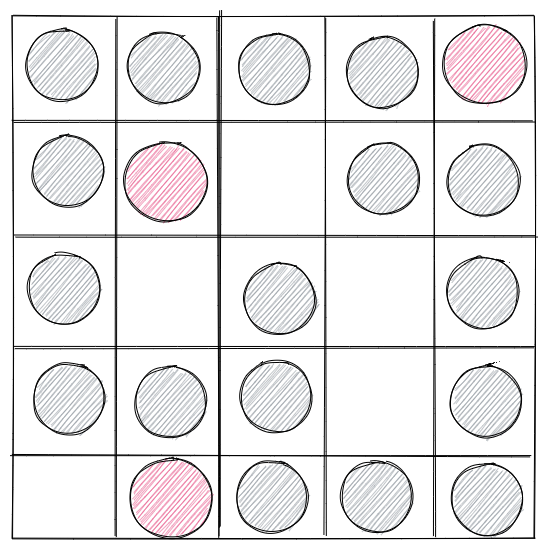
Esta función que se encarga de determinar la heurística a primera impresión es compleja de entender; sin embargo su funcionamiento es muy simple. Si tres mosqueteros se encuentran en una misma fila o columna el puntaje es de -1000, si los mosqueteros, además no tienen movimientos válidos, el puntaje es -1000000 multiplicado por la profundidad de la jugada.

En caso ninguna de estas condiciones se cumpla, se analizar la cantidad de jugadas de los mosqueteros; si la cantidad de jugadas disponibles es 0 significa que los mosqueteros ganan y se devuelve un valor de 1000000. Si todavía no se determina un ganador se analizar un tablero convencional a través de la función **utilidad\_normal**.

Esta función puntúa un tablero de acuerdo con la menor cantidad de movimientos necesarios para alinear a los mosqueteros.

Por ejemplo, observemos el siguiente tablero:

Figura 15  
Ejemplo de jugadas



Mosquetero Ficha 1

Mosquetero Ficha 2

Mosquetero Ficha 3

Como se observa en este tablero existen múltiples opciones para alinear los mosqueteros, por ejemplo, podríamos mover el mosquetero número 1 hacia la columna ya formada y solo necesitaríamos 3 movimientos, mientras que intentar alinear los mosqueteros alrededor de la fila de la ficha número 2, necesitaríamos 5 movimientos, por lo tanto, si jugaran los guardias, estos deberán escoger los tableros que les permitan alinear a los mosqueteros con la menor cantidad de jugadas posibles.

Figura 16  
Movimientos para poner en fila a los mosqueteros

Patrón de fondo

Descripción generada automáticamente

### 3.3.2 Implementación con el algoritmo Primero el Mejor

El algoritmo Primero el Mejor es un método de búsqueda informada que, a partir del establecimiento de una función de evaluación para cada nodo del árbol, selecciona y expande al más cercano a la meta, con lo que, sea el recomendado para comenzar, siempre y cuando, se asuma que sea el más viable respecto de la ordenación en forma descendente de los nodos frontera según la deseabilidad para ser el primer evaluado.

No obstante, dicha afirmación no es concluyente al ser que, no necesariamente, a través de la implementación del algoritmo, se llega a obtener la solución óptima, por lo cual, el mismo, está catalogado dentro del nivel de dificultad medio en el entorno de los juegos inteligentes y es susceptible a su optimización en tanto se defina a una función heurística que contemple una estrategia coherente y expectante a las reglas.

Así, a continuación, se presenta el pseudocódigo de la estrategia “Primero el mejor”.

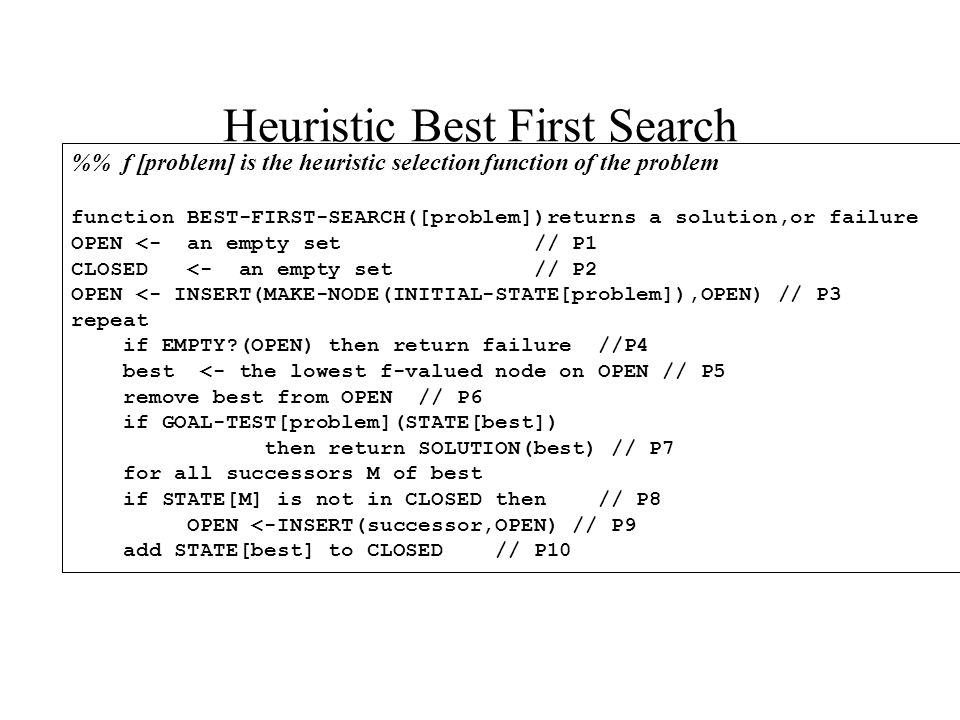
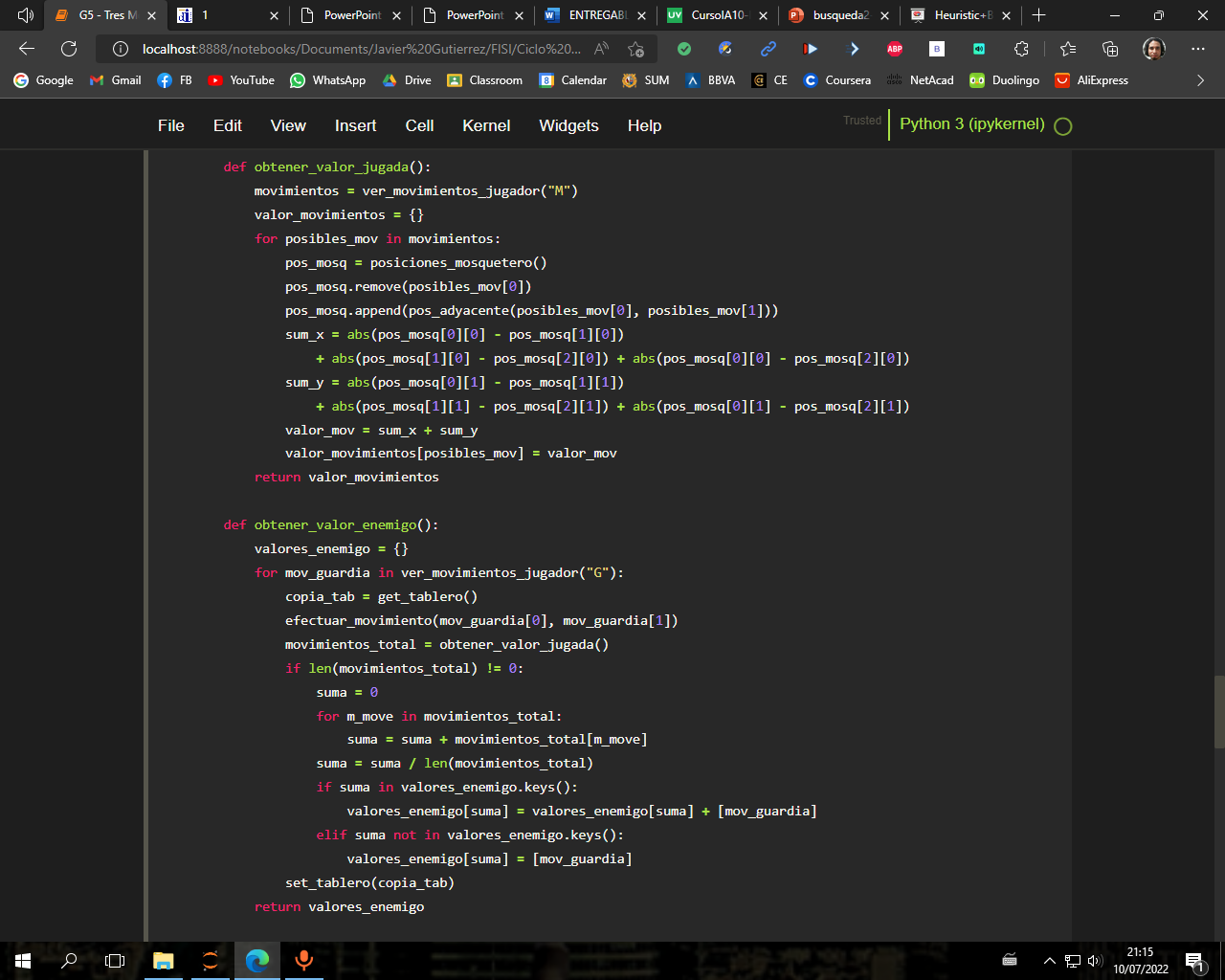


Figura 17  
Pseudocódigo "Primero el Mejor"

Véase que, el algoritmo, marca al nodo inicial del planteamiento, y, a partir del mismo, se consideran los diversos caminos a explorar, tal que, cada nodo asociado, tendrá un valor heurístico, con lo cual, se generarán los sucesores y se ordenarán, en este caso, de forma descendente, considerando que se verificará al mejor puntuado y se comprobará si efectivamente es la mejor solución, teniendo en cuenta que, se debe cotejar en cada iteración, a través de un test de parada, si existe solución o si es que finalmente, el nodo marcado es el indicado para definir a la ruta o estado vigente.

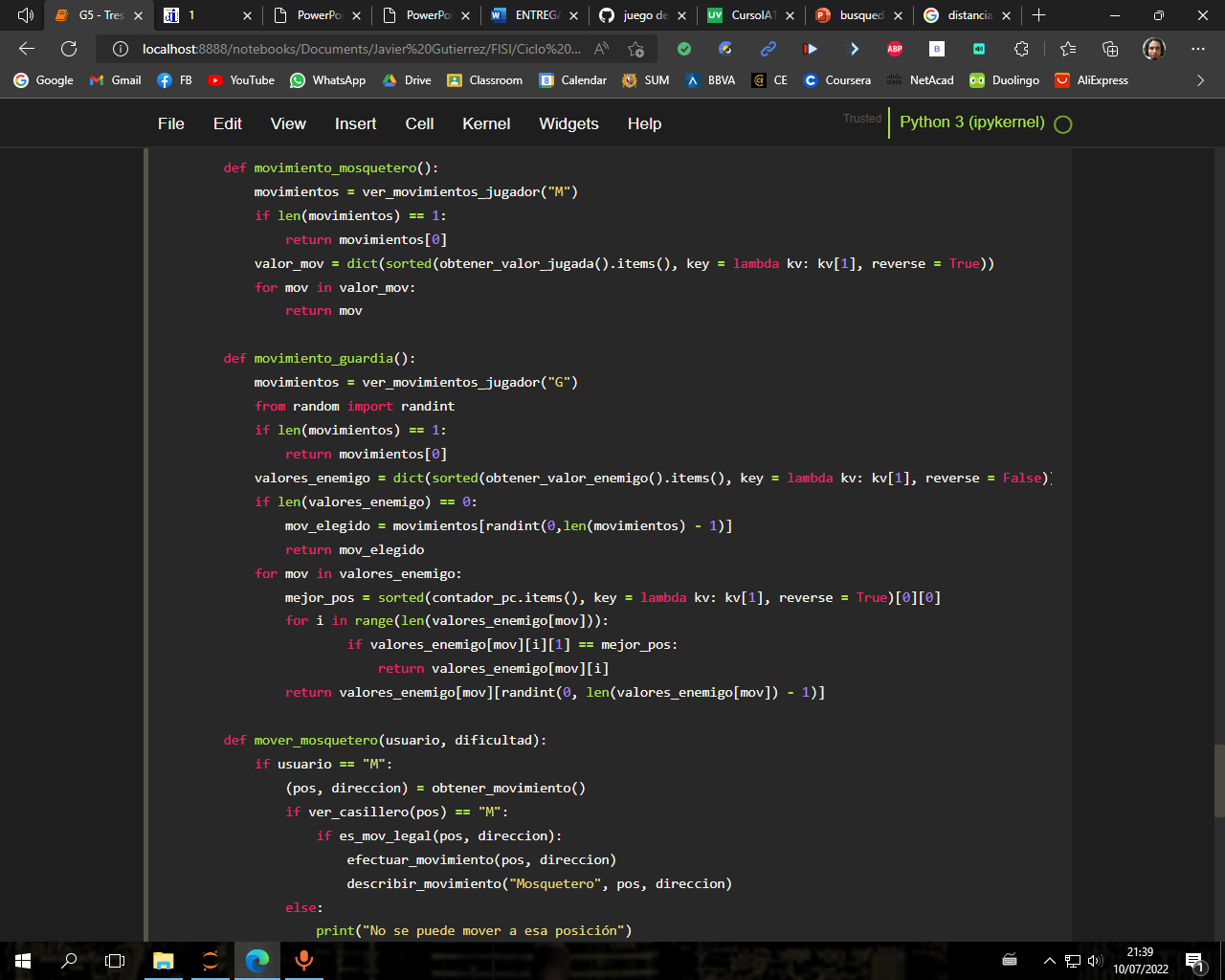
De este modo, considerando lo expuesto, se propone a la implementación en el lenguaje Python de la función que permitirá determinar el valor de evaluación o heurística dependiente del tablero en ejecución (distribución de piezas en el mismo).

Figura 18  
Implementación de la Heurística



Ahora, se denota que, a partir de la obtención de todos los movimientos posibles del mosquetero (posición y dirección), se tendrá que, se buscan las distancias euclidianas entre las ubicaciones de los mismos, por lo cual, debe considerarse el objetivo de seguir la estrategia en la que, tanto los mosqueteros como los guardias deben estar lo más separados posible entre sí, por lo cual, se tomará como referencia inmediata al movimiento que denote el mayor valor heurístico, o bien la mayor distancia entre jugadores de la misma índole, tal que, en primera instancia, como fue mencionado, es necesario determinar dichos valores de los movimientos, añadirlos y retornarlos en un diccionario (key: tupla de coordenadas con orientación y value: evaluar heurística)

Figura 19  
Implementación del Movimiento del Mosquetero



Ahora, es en esta función en donde se despliega el movimiento de cada mosquetero, al ser que, se defina lo siguiente: si existe un solo movimiento posible en dicho momento del juego, debe ser el retornado; ahora, si es que no se ha tomado alguno hasta el momento, se debe crear un diccionario de las posibles jugadas con su valor asociado, ordenados de forma descendente y luego, retornar el que tenga la mayor puntuación; es decir, según lo ya indicado, se debe retornar al movimiento que ofrezca la mayor distancia entre mosqueteros, ya que, es la función determinada, al ser que, justamente se desea evitar que estos formen una línea, ya sea, en una fila o columna, por lo cual, es apropiado que se cumpla lo dado, siempre que sea posible.

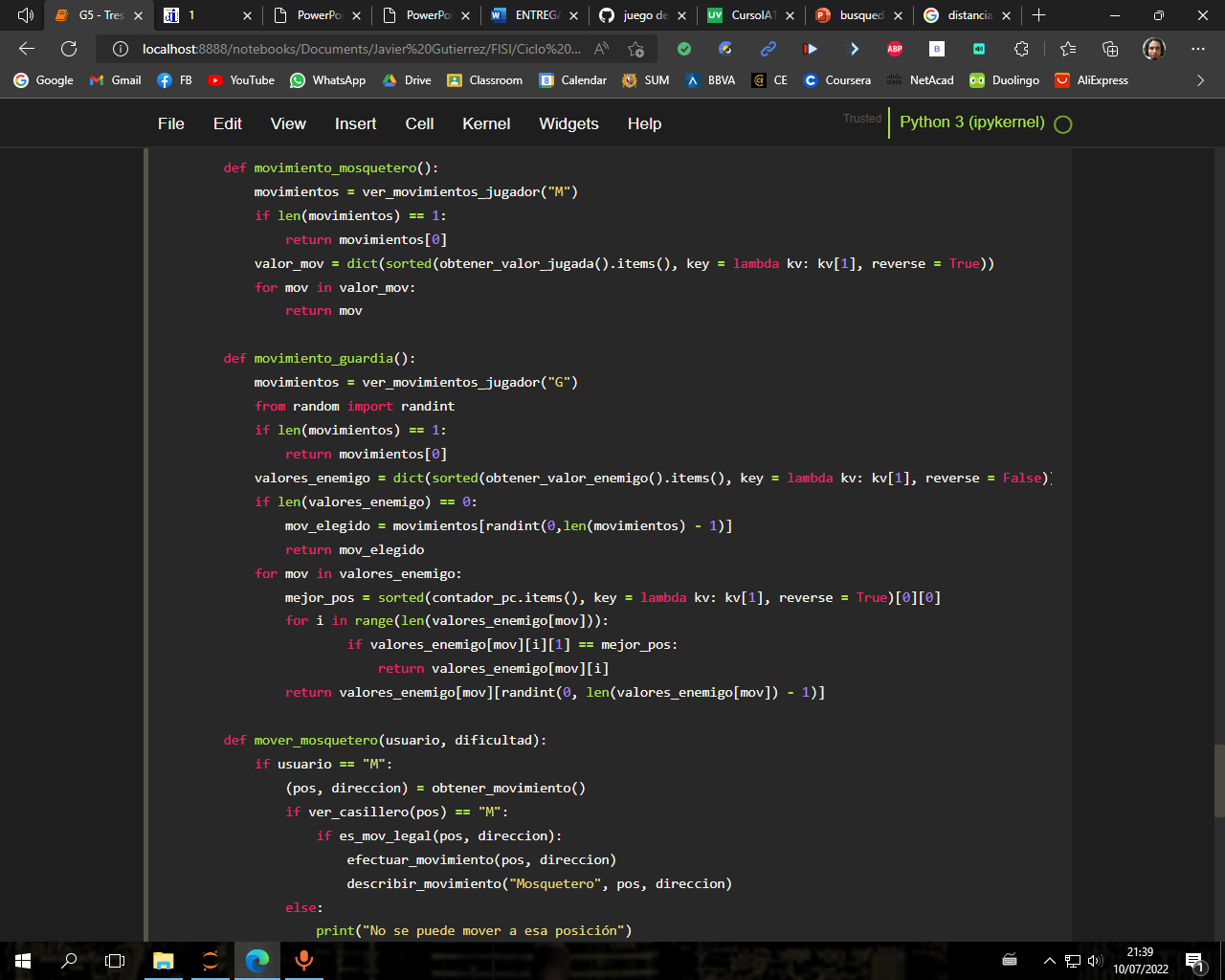


Figura 20  
Implementación de movimiento del guardia

Análogamente al movimiento generado de los mosqueteros, se dispone ahora el de los guardias, al ser que, se sigue una lógica similar, que es que, si solo hay un movimiento posible, se tendrá que tomar dicha jugada; ahora, si no se hubiera dado alguno hasta el momento, se debe elegir uno de forma aleatoria; caso contrario, se determinará al movimiento más realizado hasta el momento y se seguirá dicha dirección, ya que, se puede asumir que si aún el juego se mantiene en actividad con esa forma de proceder, el movimiento puede ser bien valorado por su definición.

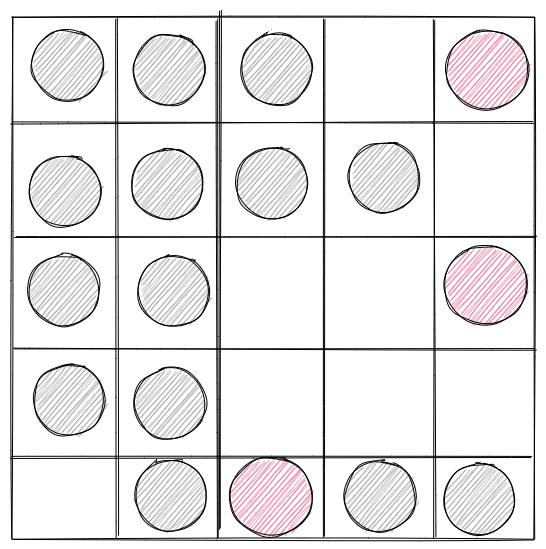
# CAPÍTULO IV: TEST COMPUTACIONAL

## 4. 1 Test de Juego con el algoritmo Minimax

El algoritmo Minimax fue testeado a través de múltiples jugadas; sin embargo, existen jugadas particulares en la cual la IA logra ganar; esto se debe a la elaboración de la función utilidad, pues no es la óptima; sin embargo la IA es capaz de ganar, pues a pesar de que no tiene el conocimiento necesario para jugar como un experto, es capaz de realizar jugadas obviar o de sentido común, como evitar jugadas que conlleven la derrota o acertar en jugadas radicalmente obvias.

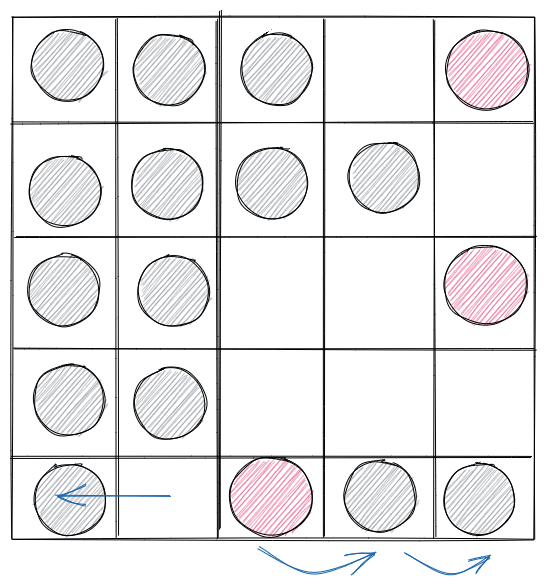
Por ejemplo, obsérvese el siguiente tablero:

Figura 21  
Ejemplo de jugada



En este tablero, es turno de los guardines. La jugada que deben mover los guardianes para obtener la victoria es más que obvia, deben aislar al mosquetero que no se encuentran alineado para que termine alineándose con el resto. La IA pasó esta prueba exitosamente.

Figura 22  
Jugada de la IA



Sin embargo, este es un tablero poco realista ya que fue diseñado arbitrariamente. Partamos ahora de un tablero real, Obsérvese el siguiente tablero.

Figura 23  
Jugada real

Patrón de fondo

Descripción generada automáticamente

En este tablero existen una gran cantidad de movimientos para los mosqueteros; sin embargo, son pocos los movimientos que llevan a la victoria a la IA. Dentro de todos los movimientos que se pueden explorar conocemos 2 que pueden lograr el cometido de la IA.

Dibujo de persona con raqueta de tenis

Descripción generada automáticamente

Figura 24  
Posibles movimientos de la IA

Nuestra IA realizó el movimiento de la flecha roja y finalizó con un tablero de la siguiente forma.

Figura 25  
Finalización de la jugada

Patrón de fondo

Descripción generada automáticamente

No es necesario realzar la última jugada, pues ya podemos determinar quién es el ganador. Para esta prueba la IA pasó exitosamente.

La secuencia de jugadas en nuestro cuaderno jupyter fue de la siguiente forma:

1. **Tablero de partida**

Imagen que contiene teclado, luz, computadora

Descripción generada automáticamente

1. **Jugada Mosquetero**

Imagen que contiene Gráfico

Descripción generada automáticamente

1. **Jugada IA**

Imagen que contiene teclado

Descripción generada automáticamente

1. **Jugada Mosquetero**

Imagen que contiene Texto

Descripción generada automáticamente

1. **Jugada IA**

Imagen que contiene teclado

Descripción generada automáticamente

1. **Jugada Mosquetero**

Imagen que contiene Texto

Descripción generada automáticamente

## 4.2 Análisis y Resultado

Este tablero fue analizado con una profundidad de máximo 7 y vemos la jugada ganadora retorna -2000000; sin embargo, observamos que para varias jugadas retornar 1. Esto es un error de la función heurística, es muy poco probable que en el conocimiento real de un juego dos movimientos generen la misma utilidad. Con una mejor función podríamos lograr que nuestra IA logre mejores jugadas; sin embargo, ahí radica el problema de la construcción de un sistema experto.

Figura 26  
Resultado Obtenido

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# CAPITULO V: CONCLUSIONES

## 5.1 Conclusiones

* El planteamiento de una eficiente función de evaluación es muy importante al momento de diseñar un juego inteligente, ya que esta función contiene el conocimiento del juego expresado de forma matemática. Nosotros como equipo hemos podido identificar que la función de utilidad depende múltiples factores, pero el grupo ha decido resaltar dos, las cuales son:
* Documentación del juego propuesto.
* Pruebas de ensayo y error.

Con respecto a la documentación, consideramos que existe poca información que haga referencia al juego de los tres mosqueteros, sobre todo a técnicas o métodos utilizados para realizar jugadas eficientes.

* Esta investigación se ha centrado en muchos aspectos del juego que redujeron la complejidad computacional y el uso de la memoria. Se ha demostrado que el juego tiene muchos subárboles que aparecen muchas veces en el árbol de búsqueda del juego y se aborda mediante la memorización.

# BIBLIOGRAFÍA

* Sackson, S. (1992). *A gamut of games*. Courier Corporation.