

Trabajo Final de Complejidad Temporal, Estructuras de Datos y Algoritmos.

“iNVASIÓN PLANETARIA”

Mariano Alejo Carrizo | Comisión 5 | 30/11/2020

Profesor : Leonardo Amett

Contenido

[Implementación 2](#_Toc58171137)

[Diagrama UML 10](#_Toc58171138)

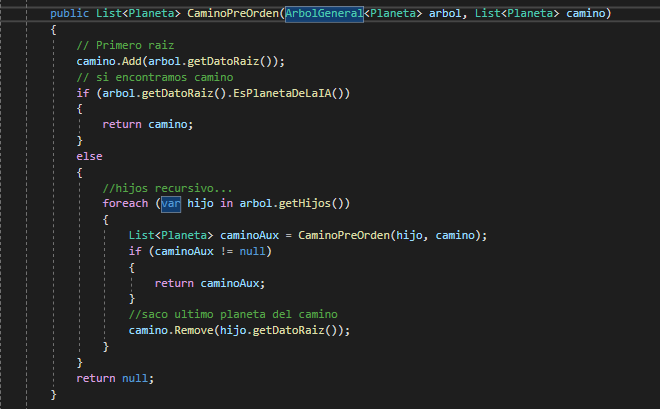
[Sugerencias 10](#_Toc58171139)

[Conclusión 11](#_Toc58171140)

# Implementación

***Consulta 1:***

La primer consulta consistía en retornar un texto con la distancia entre la raiz y el nodo del bot, Al principio tuve que interorizarme en como estaba acomodado el arbol completo en el juego, así que se hizo varias depuraciónes con recorrido preorden, y un contador que contaria desde la raiz, hasta que encuentra el nodo donde estaba la IA, fue mal implementado y daba resultados erroneos ( un ejemplo sería que la distancia entre raiz y el bot sea 8 ) y se decidió dejarlo para lo ultimo.

Se creo una lista llamada CaminoPreOrden (que fue creada inicialmente para utilizarse con CalcularMovimiento, a esta implementarse se notó que pudo reciclarse para la consulta 1)

Esta lista pedia un ArbolGeneral<Planeta> “ arbol” y una List<Planeta>” camino”, para poder funcionar.

Se meterá el primer elemento de arbolGeneral<Planeta> dentro de la lista de planeta camino, y se verifica sí el primer elemento ( el planeta) es perteneciente a la Inteligencia Artificial, si esto es verdad, devolverá la lista que se llenó hasta ese punto, siendo esta un quiebre del bucle recursivo.

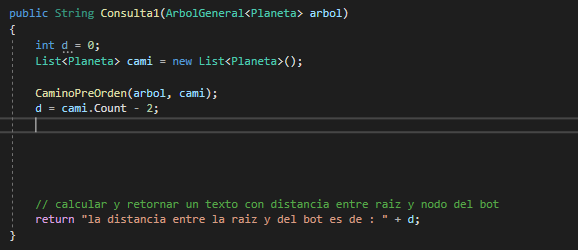
Si no es así, se hará lo siguiente a cada hijo del primer elemento de arbolGeneral<planeta>: se crea una lista de planetas auxiliar, el cual será utilizado para llamar de nuevo a la misma función de lista, esta vez con el hijo del primer elemento de arbolGeneral<Planeta> y camino, luego, se verificará si la lista caminoaux ( lista de planetas auxiliar) no es nula,si es así devolverá la lista de camino auxiliar para cortar el ciclo recursivo.

a lo ultimo, se remueve el ultimo planeta del camino,esto no tiene mucha importancia para la consulta 1 ya que solo será llamada al principio para verificar que todo este en orden, en caso de CalcularMovimiento es de suma importancia que esto se cumpla ya que le brinda el camino a la Inteligencia artificial que debe recorrer.

Lo interesante de CaminoPreorden es que es utilizado para llenar la list<Planeta> con el camino a recorrer, así que no hace falta definir una lista para que sea caminoPreorden, pero si se necesita una list<planeta> y un arbolGeneral<Planeta> para funcionar.

CaminoPreorden se llamará asi misma recursivamente, empezando desde la raiz, hasta que encuentre un nodo sea de la inteligencia Artifical.

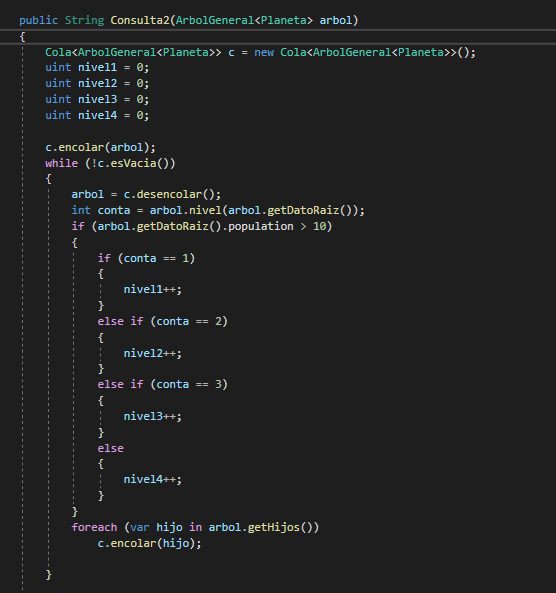
CaminoPreorden nos devuelve una lista con el primer planeta que es la raiz, y el ultimo planeta que es el nodo del bot.



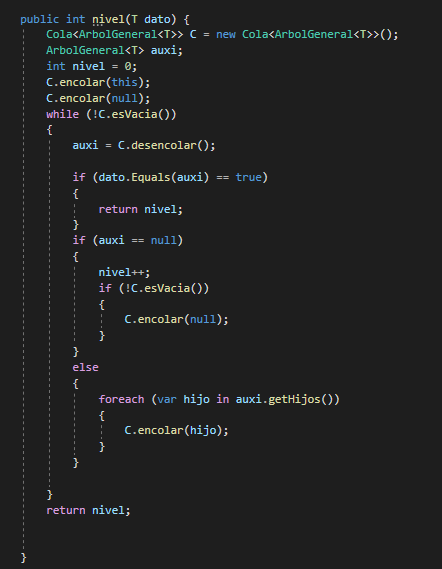
Se crea una lista vacia de planeta para utilizar CaminoPreOrden, y se define un int (entero) en cero, cuando CaminoPreorden termine de utilizarse, la lista cami, que estaba vacia, estará llena con el camino a recorrer.

Lo importante para esta consulta no es el contenido de la lista, que serian los planetas, si no la cantidad de planetas a recorrer, por eso se define que el entero D será la cantidad de la lista Cami menos 2.

Se restó menos 2 ya que si se utilizaba la cantidad total de la lista, no daba resultados acordes a lo pedido, se debia restar de la cantidad total al primer elemento de la lista (la raíz) y al ultimo elemento de la lista ( Nodo del bot).

***Consulta 2***

En esta consulta se utilizó el recorrido por niveles, y se implementó el metodo nivel, al principio se entendió mal la consigna y se implemento un recorrido por nivel donde habia un contador que sumaba el total de planetas con población de 10 sin discriminar por nivel ,pero el resultado final fue el presentado en la imagen.



La función nivel devuelve un entero, y para ser usado necesita un Dato ( este sería el dato de Arbol.getdatoRaiz())) que seríael ato en el cual nosotros estamos parados.

Aquí nivel hará un recorrido por niveles usando una cola<arbolgeneral<T>> y un arbolgeneral<T> auxiliar, se define un entero como cero y se comienza a encolar la lista de arbol., junto a null en cola<arbolgeneral<T>>, este null es para dividir y reconocer luego el nivel del arbol.

Mientras que la cola tenga un elemento en su lista, se verifica que el elemento desencolado es igual al dato que dimos, en ese caso terminaria y devolveria el entero que fue definido anteriormente, en el caso de que lo que desencolamos fuera un null, se incrementa el entero nivel en 1, ya que este es el divisor de nivel a nivel.

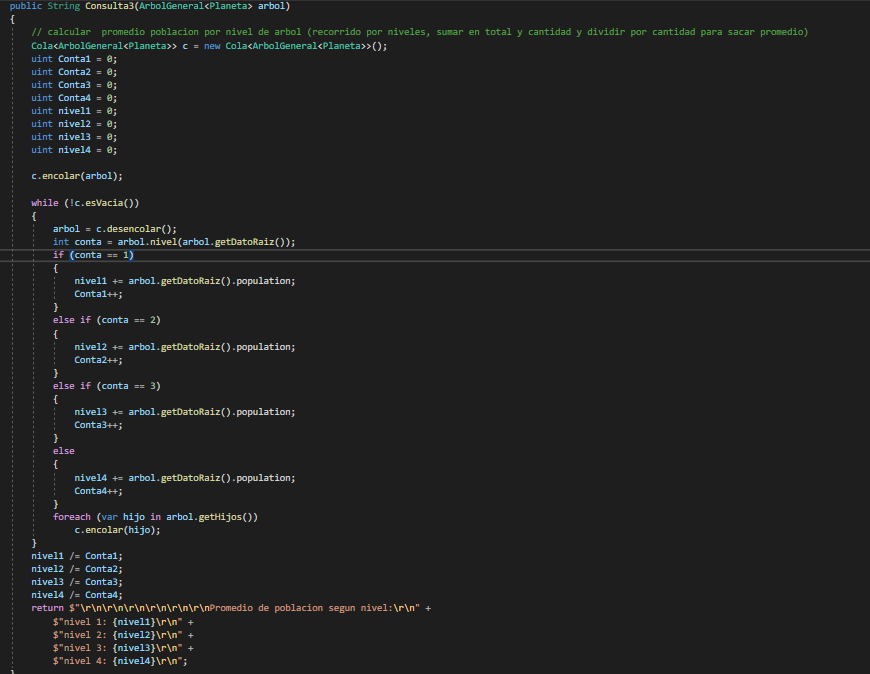
Sí C no es vacia se encolará de vuelta a Null.

Si no, se hará recursivamente los hijos de la lista auxiliar , encolando de vuelta hijo y volviendo al inicio del while, haciendo así el bucle.

Se hicieron varios contadores por niveles, nivel 0 a 4.

Ya con la función nivel implementada solo debia hacerse un recorrido por niveles igual que la funcion nivel, solo que con la diferencia de que creamos un entero que contenga el nivel del dato en el cual nos encontremos, y ahí, nos verificamos si el planeta donde nos encontramos tiene una población mayor de 10, se verificará el nivel del arbol, dependiendo su nivel se agregará 1 al contador correspondiente, luego se carga recursivamente a los hijos de donde nos encontramos (en este caso raiz) y se volverá a encolar, para continuar con el bucle.

**Consulta 3**



Esta es igual que la consulta 2, La unica diferencia que ya no existirá el condicional que discrimina a la población mayor a 10.

La consulta 3 pide calcular el promedio de poblacion POR nivel del arbol, al principio se hizo el promedio total de los 34 planetas.

Se podrá notar la inmensa cantidad de Uint, 4 son contadores de cantidad y los otros 4 son los contadores de la población por cada nivel.

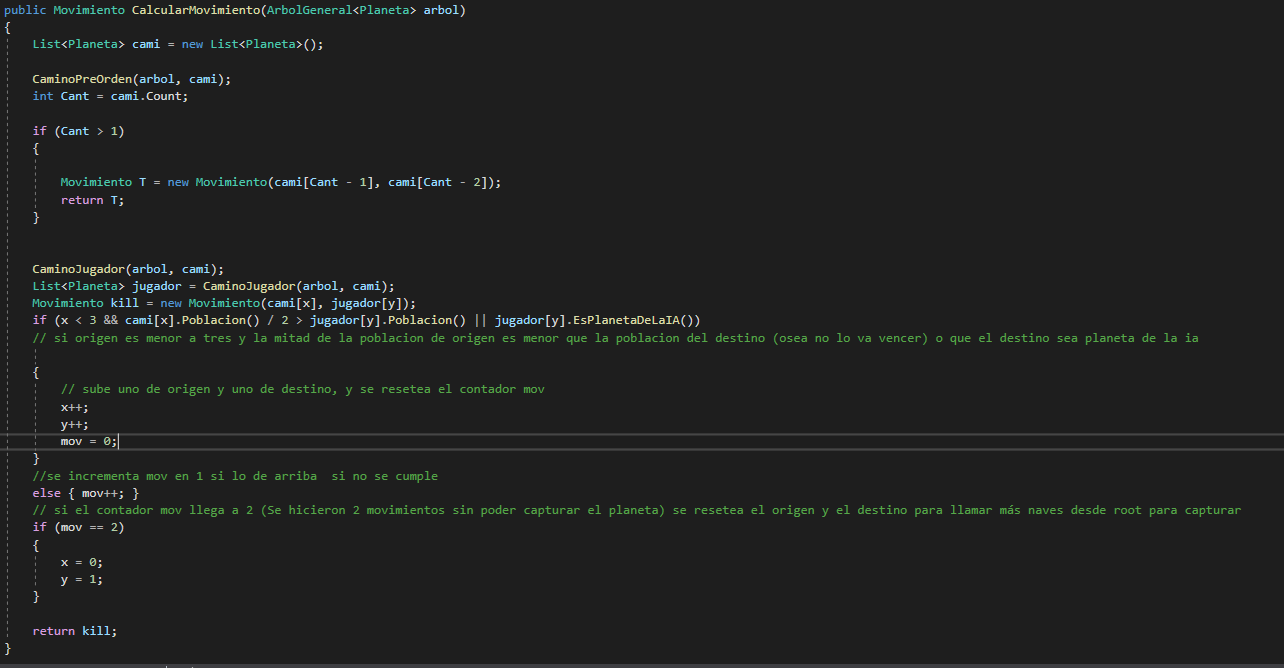
Aquí se verifica el nivel del dato en el que se está poscionado, si por ejemplo el nivel sería 1, el contador nivel1 se le suma la población de ese dato, y se aumenta el conta1 el cual es contador de la cantidad de planetas.

Para poder sacar el promedio de algo, se debe tener ese algo divido la cantidad total.

En nuestro caso sería población nivel 1 / cantidad de planetas de nivel 1 = promedio nivel1

Y así para cada nivel del arbol.

**Calcular movimiento.**



Esta función para funcionar necesita un arbolgeneral<planeta>, que es el mismo del mapa que se presenta en el juego.

Calcular movimiento se divide en 2 fases:

La fase 1 consiste en que la ia viaja hasta raiz.

La fase 2 consiste en la ia viajando hacia la poscicion del jugador.

Se creó una lista vacia para usar la función CaminoPreorden, el cual fue explicado para la consulta 1, aquí de nuevo se utiliza para llenar la lista vacia con los planetas que se deben recorrer para llegar al destino, en este caso, Raiz.

Se hace un entero con la cantidad de planetas en la lista “cami” para poder hacer una condicional: si la cantidad de planetas es mayor a 1, se hará la

Se hace un contador con la cantidad total de planetas en la lista, y se verifica sí es mayor a 1, si es así la fase uno del movimiento de la ía se ejecuta.

Los movimientos necesitan un planeta origen y un planeta destino, por lo tanto, en la fase uno se inicia un movimiento T donde el planeta origen sea la lista en la poscición del contador total de planetas menos 1, y el planeta destino sea la lista en la poscición del contador total de planetas menos 2.

Esto ultimo es por que la función calcularmovimiento es llamada en intervalos de tiempo, y se va actualizando gracias a medida de que la IA hace movimientos, gracias a la función CAMINOPREORDEN,

El tope es 1 ya que si no se cumple esa condición el programa tendria una excepción out of index.

La fase 2 consiste en llamar la función CAMINOJUGADOR que es igual que caminopreorden, con la diferencia de que la condicional que discrimina los datos es el booleano EsPlanetaDeLaIA())).

Luego se define un movimiento llamado kill, donde planeta origen es cami(entero x= 0) y el palenta destino es cami(entero y = 1)

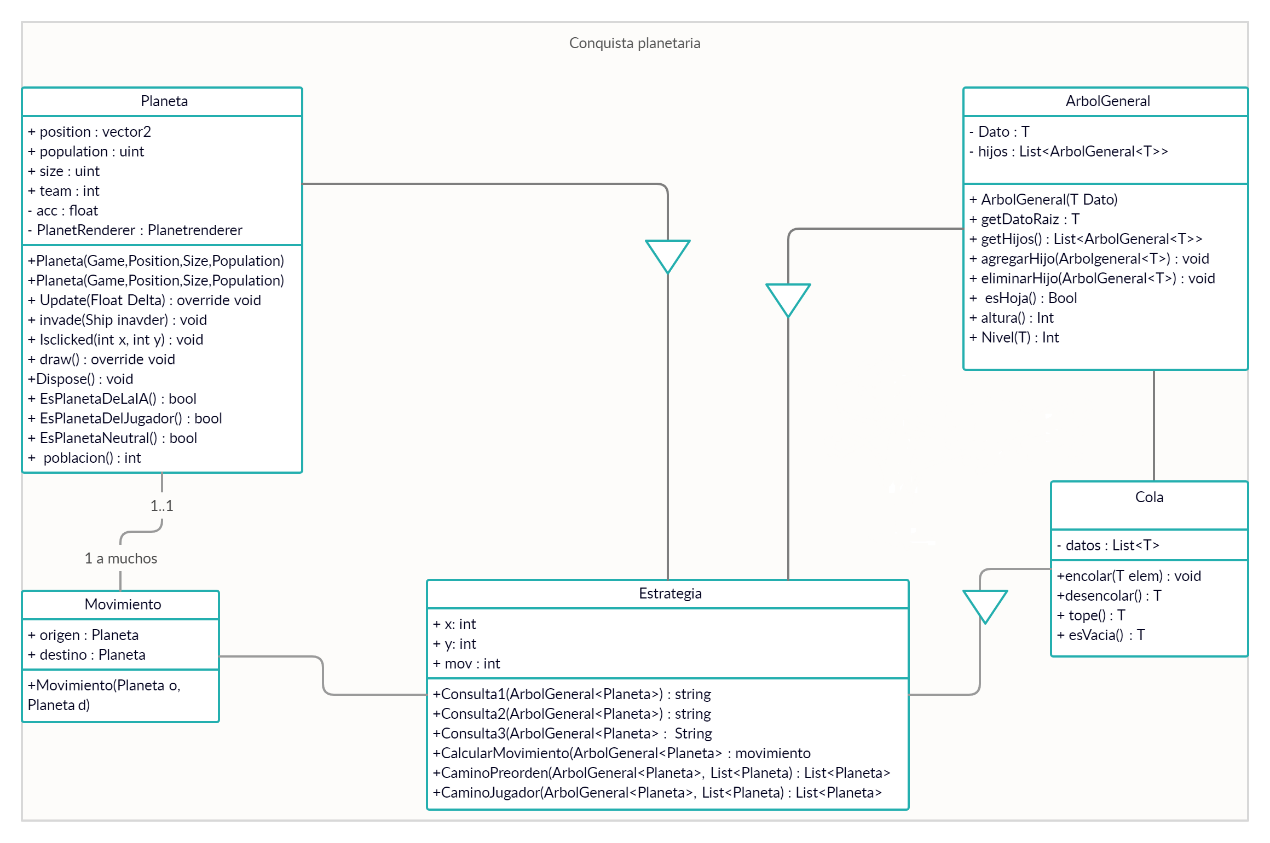
Estos enteros fueron definidos para realizar los condiconales siguientes:

Sí el origen es menor a tres y la población del planeta origen es menor que la población del destino, se incrementa x y Y en 1, y se resetea al contador mov en 0,ya que este caso es cuando la IA venció al planeta destino.

Si no logra vencerlo con un movimiento, se incrementa el contador de mov en 1, y para entonces la IA hizo otro movimiento, y si no lo vence, en el proximo movimiento de la ia se resetea el origen (x ) y el destino (y) para llamár más naves desde la poscición anterior.

Por ultimo retorna el movimiento kill, ya modificado con los if de X y Y.

# Diagrama UML



# Sugerencias

1. La temática de juego por turno NO se cumple en este caso; ya que los movimientos de IA no son en respuesta al movimiento del jugador, si no a un intervalo de tiempo fijo, que puede ser cambiado, además de que el intervalo de crecimiento de la población de los planetas de la IA y jugador son también por Intervalo y no en respuesta al movimiento del jugador ( en mi opinión, la única manera de hacer que este juego sea por turnos, es que el código reaccione a los movimientos (invasiones ) del jugador.
2. Me gustaría haber implementado un método para que la IA no pudiera tocar los planetas neutrales y los de jugador, y viceversa (en ciertos casos, cuando se mantiene seleccionado el planeta del jugador, y es invadido satisfactoriamente por la máquina, también el propio jugador puede mandar tropas de la IA en otro lugar)
3. Una sugerencia sería de movimientos más complejos de la IA, como por ejemplo que mueva varias cosas a la vez, o que busque la población más grande y vaya moviéndola hacia donde la necesite para atacar, así el juego sería más desafiante.

# Conclusión

Se logró crear soluciones Dinámicas y eficientes (en lo posible) al problema dado, usando lo aprendido en clase y se pudo comprender la importancia practica de los árboles binarios, y los métodos de búsqueda para la resolución de problemas.

Sobre Calcular movimiento logré de una manera conseguir el objetivo planteado de la fase 2 de la IA, y logré que sea dinámico (con dinámico me refiero a que el próximo desarrollador quiera ampliar el juego, como un mapa más grande, no tendrá que reformar todo código) pero aun así, siento que no he llegado a aplicar todo lo que la clase tuvo que ofrecer para esta segunda fase.

Descubrí que Programar para dar con el objetivo para solo resolverlo de momento es muy diferente a tratar de hacer el código lo más flexible posible.

Las depuraciones iniciales del trabajo fueron donde me “empapé” con la teoría practica de la materia, haciendo prueba y error para ver que recorrido era el adecuado para cada caso, pero no logro quitarme de la cabeza que quizás pudiera haber hecho el trabajo aún más limpio y eficiente, más simple.