Nombre y apellido: Matías Miguel Silva

Carrera: Ingeniería en Informática

Materia: Complejidad Temporal, Estructuras de Datos y Algoritmos

Año: 2021 1° Cuatrimestre

INTRODUCCIÓN

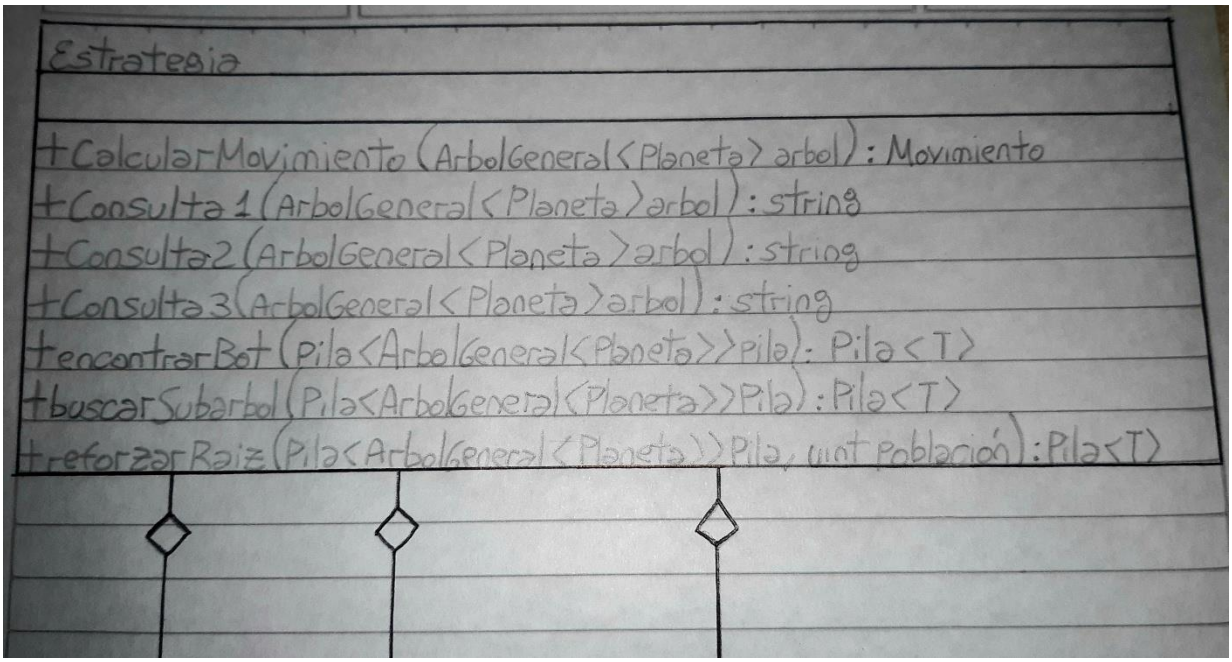
Este trabajo consiste en la programación de algoritmos para un proyecto en lenguaje C#. Tomando como base un programa ya realizado, que consiste en un juego de informática, se busca agregar nuevos métodos para el funcionamiento del mismo.

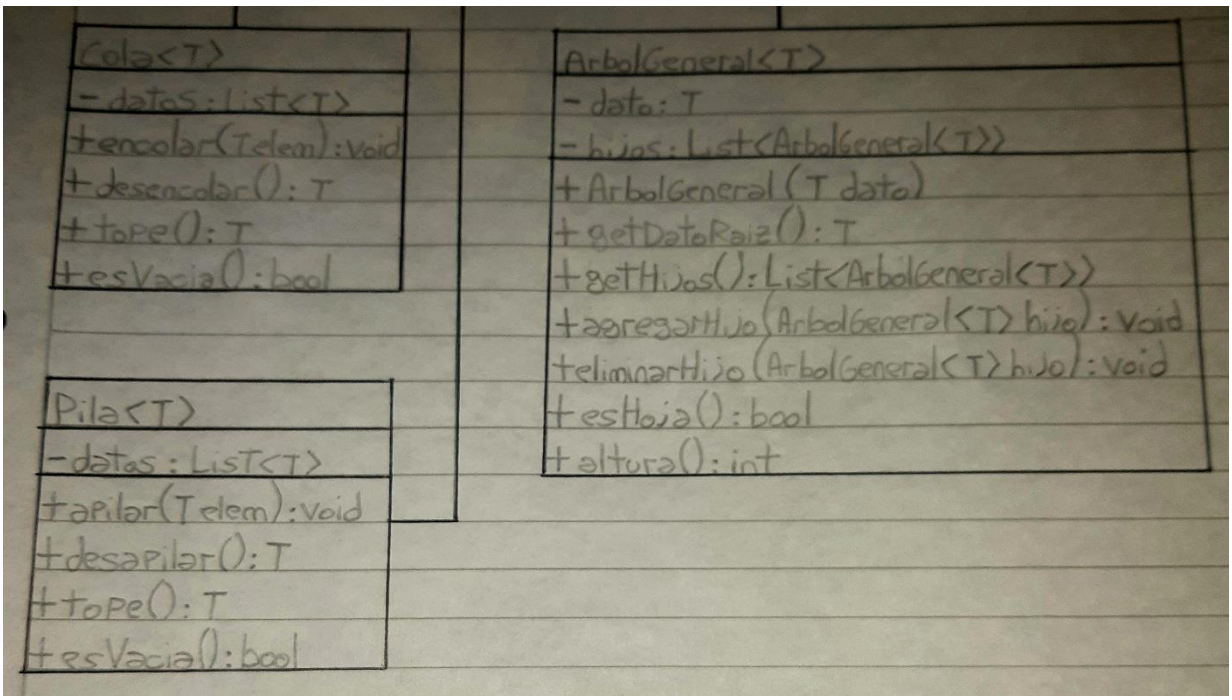
El juego consiste en una conquista planetaria, donde el objetivo consiste en capturar todos los planetas del adversario. Los planetas se organizan en un árbol general, y cada planeta puede ser neutral, del jugador o de la IA.

En base al árbol del juego, se han implementado un método que retorna el movimiento apropiado para la IA según el estado del juego, y tres consultas que retornan datos del árbol actual.

DESARROLLO

Uml de clases:





Calcular Movimiento:

1. Se busca primero el nodo Bot más cercano a la raíz, utilizando la clase Cola para recorrer el árbol.

2. El algoritmo primero verifica si se puede reforzar la raíz, para eso busca en sus subárboles un planeta que tenga mayor población que la raíz. Si lo encuentra, retorna el camino desde dicho planeta hasta la raíz.

3. Si no se refuerza la raíz, se busca en sus subárboles un planeta que no sea de la IA y el método retorna el camino hacia dicho planeta.

4. Si la IA ya ha conquistado todos los planetas del nodo y la raíz del árbol no ha sido conquistada por el Bot, se devuelve el movimiento desde el nodo de la IA hasta la raíz.

5. Si la raíz es de la IA, se empieza a capturar los subárboles restantes, reforzando la raíz cuando sea necesario.

Para los casos 2,3 y 4, el algoritmo utiliza la clase Pila para recorrer el árbol utilizando el recorrido pre-orden, explorando el primer subárbol del nodo. Cuando se termina de explorar un subárbol, continua con el siguiente.

```
public Movimiento CalcularMovimiento(ArbolGeneral<Planeta> arbol)
{
    // si la raíz no es de la IA
    if(!arbol.getDatoRaiz().EsPlanetaDeLaIA())
    {
        // encontrar Bot
        Pila<ArbolGeneral<Planeta>> raiz = new Pila<ArbolGeneral<Planeta>>();
        raiz.apilar(arbol);
        Pila<ArbolGeneral<Planeta>> caminoBot = encontrarBot(raiz);

        // reforzar raíz
        Pila<ArbolGeneral<Planeta>> subRaiz = new Pila<ArbolGeneral<Planeta>>();
        subRaiz.apilar(caminoBot.tope());
        Pila<ArbolGeneral<Planeta>> caminoR = reforzarRaiz(subRaiz, subRaiz.tope().getDatoRaiz().population);
        if(caminoR.tope() == null)
        {
            caminoR.desapilar();
            Planeta origen = caminoR.desapilar().getDatoRaiz();
            Planeta destino = caminoR.tope().getDatoRaiz();
            Movimiento mov = new Movimiento(origen, destino);
            return mov;
        }



        // buscar subárbol hijo sin planeta IA
        Pila<ArbolGeneral<Planeta>> subraiz = new Pila<ArbolGeneral<Planeta>>();
        subraiz.apilar(caminoBot.tope());
        Pila<ArbolGeneral<Planeta>> subarbol = buscarSubarbol(subraiz);

        // si existe
        if(!subarbol.tope().getDatoRaiz().EsPlanetaDeLaIA())
        {
            // retornar camino al hijo
            Planeta destino = subarbol.desapilar().getDatoRaiz();
            Planeta origen = subarbol.tope().getDatoRaiz();
            Movimiento mov = new Movimiento(origen, destino);
            return mov;
        }

        // si no existe
        else
        {
            // retornar camino a la raíz
            Planeta origen = caminoBot.desapilar().getDatoRaiz();
            Planeta destino = caminoBot.tope().getDatoRaiz();
            Movimiento mov = new Movimiento(origen, destino);
            return mov;
        }
 }
```

```
// si la raíz es de la IA
else
{
    // reforzar raíz
    Pila<ArbolGeneral<Planeta>> raiz = new Pila<ArbolGeneral<Planeta>>();
    raiz.apilar(arbol);
    Pila<ArbolGeneral<Planeta>> CaminoR = reforzarRaiz(raiz, raiz.tope().getDatoRaiz().population);
    if(CaminoR.tope() == null)
    {
        CaminoR.desapilar();
        Planeta ori = CaminoR.desapilar().getDatoRaiz();
        Planeta dest = CaminoR.tope().getDatoRaiz();
        Movimiento m = new Movimiento(ori, dest);
        return m;
    }

    // buscar subárbol hijo
    Pila<ArbolGeneral<Planeta>> subarbol = buscarSubarbol(raiz);

    // retornar camino al hijo
    Planeta destino = subarbol.desapilar().getDatoRaiz();
    Planeta origen = subarbol.tope().getDatoRaiz();
    Movimiento mov = new Movimiento(origen, destino);
    return mov;
}
```

```
public Pila<ArbolGeneral<Planeta>> encontrarBot(Pila<ArbolGeneral<Planeta>> pila)
{
    // si es de la IA, retornar
    if(pila.tope().getDatoRaiz().EsPlanetaDeLaIA())
        return pila;

    // si no es de la IA, recursión por cada hijo
    foreach(ArbolGeneral<Planeta> hijo in pila.tope().getHijos())
    {
        pila.apilar(hijo);
        encontrarBot(pila);
        if(pila.tope().getDatoRaiz().EsPlanetaDeLaIA())
            return pila;
        pila.desapilar();
    }
    return pila;
}
```

```
public Pila<ArbolGeneral<Planeta>> buscarSubarbol(Pila<ArbolGeneral<Planeta>> pila)
{
    // si no es de la IA, retornar
    if(!pila.tope().getDatoRaiz().EsPlanetaDeLaIA())
        return pila;

    // si es de la IA, recursión por cada hijo
    foreach(ArbolGeneral<Planeta> hijo in pila.tope().getHijos())
    {
        pila.apilar(hijo);
        buscarSubarbol(pila);
        if(!pila.tope().getDatoRaiz().EsPlanetaDeLaIA())
            return pila;
        pila.desapilar();
    }
    return pila;
}
```

```
public Pila<ArbolGeneral<Planeta>> reforzarRaiz(Pila<ArbolGeneral<Planeta>> pila, uint poblacion)
{
    // si el planeta no es de la IA, retornar
    if(!pila.tope().getDatoRaiz().EsPlanetaDeLaIA())
        return pila;

    // si el planeta tiene más población que la raíz, marcar terminado(null), y retornar
    if(pila.tope().getDatoRaiz().population > poblacion)
    {
        pila.apilar(null);
        return pila;
    }

    // si no, recursión por cada hijo
    foreach(ArbolGeneral<Planeta> hijo in pila.tope().getHijos())
    {
        pila.apilar(hijo);
        reforzarRaiz(pila, poblacion);
        if(pila.tope() == null)
            return pila;
        pila.desapilar();
    }
    return pila;
}
```

Consulta 1:

```
public String Consulta1( ArbolGeneral<Planeta> arbol)
{
    Cola<ArbolGeneral<Planeta>> c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> aux = null;
    bool check = false;
    c.encolar(arbol);

    // busco raíz de la IA
    // si check = true, encontré la raíz

    while(!check)
    {
        aux = c.desencolar();
        if(aux.getDatoRaiz().EsPlanetaDeLaIA())
            check = true;
        else
        {
            foreach(ArbolGeneral<Planeta> hijo in aux.getHijos())
                c.encolar(hijo);
        }
    }

    // le busco la altura a la raíz
    return "La distancia entre la raíz y la hoja más lejana es: "+aux.altura();
}
```

- Calcular la distancia entre la raíz del bot y la hoja más lejana.

Se inicializa una cola con la raíz, para recorrer el árbol. Si el planeta es de la IA, le calculo la altura a ese planeta y retorno ese valor con el mensaje. Si no es de la IA, agrego sus hijos a la cola y sigo con el siguiente planeta.

Consulta 2:

```
public String Consulta2( ArbolGeneral<Planeta> arbol)
{
    Cola<ArbolGeneral<Planeta>> c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> aux;
    int x = 0; // contador
    c.encolar(arbol);

    // mientras cola no se vacíe
    while(!c.esVacia())
    {
        aux = c.desencolar();
        foreach(ArbolGeneral<Planeta> hijo in aux.getHijos())
            c.encolar(hijo);

        // si es hoja y su población > 3, incremento contador
        if(aux.esHoja() && aux.getDatoRaiz().population > 3)
            x = x + 1;
    }
    return ("El número de planetas hojas con población mayor a 3 es: "+x);
}
```

- Calcula y retorna un texto con la cantidad de planetas ubicados en las hojas que tienen población mayor a 3.

Se recorre el árbol con una cola que contiene la raíz del árbol. Por cada planeta, agrego sus hijos a la cola, y si es una hoja y su población es mayor a 3, incremento el contador. Una vez recorrido todo el árbol (cola vacía), retorno el mensaje con el valor del contador.

Consulta 3:

```
public String Consulta3( ArbolGeneral<Planeta> arbol)
{
    Cola<ArbolGeneral<Planeta>> c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> aux;
    c.encolar(arbol);
    c.encolar(null);
    uint pro = promedio(arbol);
    uint count = 0;
    uint nivel = 0;
    string mensaje = "Número de planetas con población mayor al promedio:\n";

    // mientras cola no se vacíe
    while(!c.esVacia())
    {
        aux = c.desencolar();

        // si no terminé con el nivel
        if(aux != null)
        {
            // si población > promedio, incremento contador
            if(aux.getDatoRaiz().population > pro)
                count = count + 1;
            foreach(ArbolGeneral<Planeta> hijo in aux.getHijos())
                c.encolar(hijo);
        }




        // si ya terminé con el nivel
        else
        {
            // imprimo mensaje del nivel
            mensaje = mensaje + ("Nivel "+nivel+": "+count+"\n");

            // reinicio contador y paso al sig. nivel
            nivel = nivel + 1;
            count = 0;

            // si la cola está vacía, terminé de explorar el árbol
            if(!c.esVacia())
                c.encolar(null);
        }
    }
    return mensaje;
}
```

```
public uint promedio(ArbolGeneral<Planeta> arbol)
{
    Cola<ArbolGeneral<Planeta>> c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> aux;
    c.encolar(arbol);
    uint pro = 0;
    uint count = 0;

    // mientras cola no se vacíe
    while(!c.esVacia())
    {
        aux = c.desencolar();

        // sumo al total la población del planeta e incremento contador
        pro = pro + aux.getDatoRaiz().population;
        count = count + 1;
        foreach(ArbolGeneral<Planeta> hijo in aux.getHijos())
            c.encolar(hijo);
    }
    return (pro / count);
}
```

- Calcula y retorna en un texto el número de planetas por cada nivel que tienen una población mayor a la cantidad promedio del árbol

Primero se calcula el promedio del árbol. La función promedio() recorre el árbol mediante una cola, sumando al total (pro) la población de cada planeta. Luego, se divide esa suma por la cantidad de planetas del árbol.

Una vez calculado el promedio, se recorre el árbol utilizando una cola con un verificador de nivel (null). Si el elemento es un planeta, agrego sus hijos a la cola y si su población es mayor al promedio del árbol, incremento el contador. Si el elemento es un null, añado al mensaje el nivel actual con el contador, reinicio el contador y paso al siguiente nivel. Si no hay más elementos en la cola, apilo el null en la cola y continúo.

FINAL

Este trabajo ha sido útil para mejorar los conocimientos sobre las estructuras de datos, aplicarlos para un proyecto y poder programar en lenguaje C#. Lo más difícil fue implementar la estrategia según el árbol de la raíz, y como reforzar el mismo, ya que hubo que hacer diferentes algoritmos para cada caso.

Se puede mejorar el sistema de captura de planetas, cuando hay que capturar un planeta del jugador y tiene mucha más población, continuar con otro subárbol.