



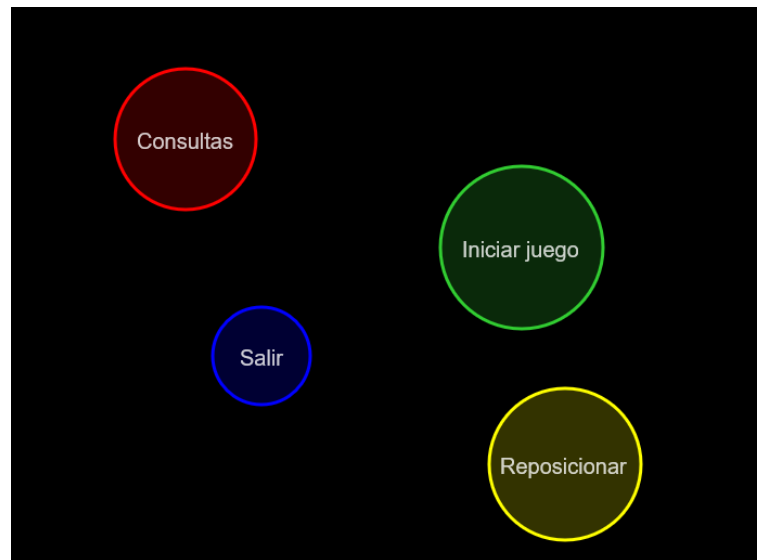
# Conquista Planetaria

- Materia: Complejidad Temporal, Estructuras de Datos y Algoritmos
- Comisión: 1
- Profesor: Pablo José Iuliano
- Autor: Corsiglia Gonzalo

Para dar comienzo a este recorrido, primero quiero contarte que “Conquista Planetaria” es un juego en el cual una serie de planetas se ven envueltos en una contienda sin fin entre dos planetas que buscan dominar la mayor cantidad de civilizaciones posibles. Para apoderarse de estos cuerpos celestes existe la dinámica de ataque o invasión si lo contextualizamos. De esta forma resulta un videojuego divertido y didáctico en torno a la materia que nos incumbe.

En un principio, nos encontramos con un menú en el cual contaremos con una serie de opciones, entre ellas algunas primordiales como lo son “Iniciar Juego”, “Reposicionar” los planetas para que estos se distribuyan de una manera diferente, y “Salir”.

Pero, además tendremos la opción “Consultas” que nos permitirá obtener algunos datos más que curiosos sobre el planeta de la IA.



Para llevar a cabo la implementación de tales consultas, contamos con la ayuda de otras clases con sus respectivos métodos que nos facilitaran mucho la resolución:

**ArbolGeneral:** Esta clase se encarga de la implementación de la estructura de datos en la cual se basa el videojuego, una forma inteligente de organizar la información y que nos permite obtener algoritmos eficientes. Dentro de esta clase encontramos los métodos respectivos a un árbol general.

**Cola:** Esta clase implementa las colas, estructuras de datos que permiten trabajar con un grupo de elementos que serán procesados según su orden de llegada. Contamos con los métodos pertinentes para su aplicación.

**Planeta:** Esta clase se encarga de darle vida y forma a los planetas. Nos proporciona métodos para dilucidar si un planeta es del Jugador, del Bot o Neutral. Además, nos permite saber la población de los planetas, entre otras funcionalidades.

**Movimiento:** De esta clase nacen las invasiones, esta nos permite crear movimientos de tropas de un planeta origen a un destino con el fin de conquistar planetas.

**Estrategia:** En esta clase es donde trabajaremos. Aquí, llevaremos a cabo las consultas, y le daremos vida al bot para que este se defienda de los ataques del jugador.

**Juego:** Implementa método main que inicia el juego

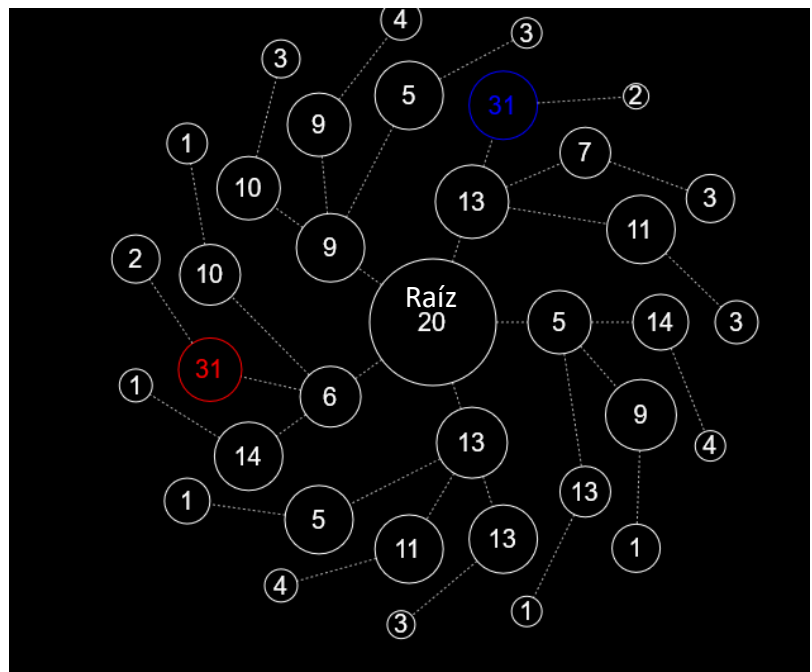
## Consultas

A continuación, detallaremos las consultas.

### Consulta 1:

Nos informa la cantidad de planetas que hay entre un planeta denominado Raíz, y el planeta de la IA.

Como podemos observar, los planetas están estructurados en un árbol general. Esta estructura de datos es conformada por un nodo raíz, y subárboles con nodos padres conectados por aristas.



Un concepto primordial de los árboles, es la “Profundidad”. Esta es la longitud desde la raíz hasta n. La raíz tiene profundidad 0.

Para este entonces te habras dado cuenta, que la Consulta 1 se refiere a la profundidad del bot dentro del arbol general de planetas.

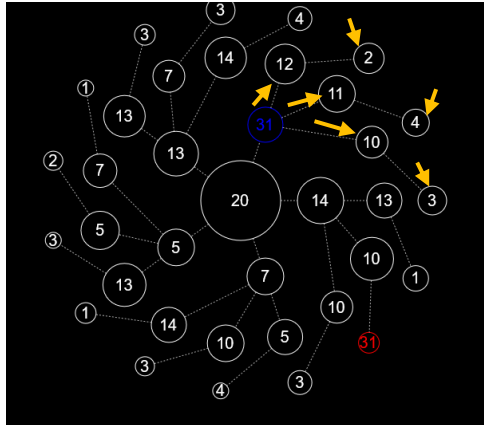
La resolución es sencilla, primero creamos un método privado en el cual se calcula la profundidad a través de una resolución que implementa recursividad. En este caso, utilizamos una adaptación del preorden, donde se analizan los hijos de la raíz en busca de un camino que llegue hasta el bot. Mientras no se encuentre al bot, no pasará nada, pero si se encuentra al bot el valor irá incrementándose en 1 desde el bot, hasta el hijo de la raíz. De esta forma este método privado devuelve la profundidad del bot.

La distancia entre el Planeta Bot y la Raiz es de: 3 planetas



## Consulta 2:

En contexto de la estructura de datos Árbol General, esta consulta pretende retornar un texto que informe cuales son los descendientes del planeta bot.



Para que te des una idea, en el caso hipotético de tener este árbol general de planetas, el resultado de la Consulta 2 sería un texto con los planetas 10 11 12 3 4 2

¿Cómo lo resolvimos?

A la hora de comenzar la resolución, entendí que necesitaba dos cosas:

- 1) Encontrar el bot
- 2) Retornar todos los descendientes del bot

Entonces, a partir de estos dos pasos es que decidí crear un método privado que retorné una lista con los descendientes del bot.

```
private ArrayList SubArbolIA(ArbolGeneral<Planeta> PlanetaIA, ArrayList lista)
{
    if(PlanetaIA != null)
    {
        Cola<ArbolGeneral<Planeta>> cola = new Cola<ArbolGeneral<Planeta>>();
        cola.encolar(PlanetaIA);
        ArbolGeneral<Planeta> aux;
        while(!cola.esVacia())
        {
            aux = cola.desencolar();
            lista.Add(aux.getDatosRaiz().population);
            if(aux.getHijos() != null)
            {
                foreach(var hi in aux.getHijos())
                {
                    cola.encolar(hi);
                }
            }
        }
        lista.RemoveAt(0);
        return lista;
    }
}
```

Este método recibe al Planeta del Bot y una lista por parámetro, luego guarda cada uno de los descendientes del bot en formato de un recorrido por niveles.

Por último, borramos el bot de la lista, y retornamos la lista de descendientes.

Luego en un método público:

```
public string Consulta2( ArbolGeneral<Planeta> arbol)
{
    ArrayList planetas = new ArrayList();
    Cola<ArbolGeneral<Planeta>> cola = new Cola<ArbolGeneral<Planeta>>();
    cola.encolar(arbol);
    while(!cola.esVacia())
    {
        ArbolGeneral<Planeta> actual = cola.desencolar();
        if(actual.getDatoRaiz().EsPlanetaDeLaIA())
        {
            string res = "Planetas descendientes del Bot: ";
            if(actual.esHoja())
            {
                return "El planeta del Bot no tiene descendientes";
            }
            foreach(var planeta in SubArbolIA(actual, planetas))
            {
                res += planeta + " ";
            }
            return res;
        }
        else
        {
            foreach(var hijo in actual.getHijos())
            {
                cola.encolar(hijo);
            }
        }
    }
    return "";
}
```

Nos encargamos de encontrar el bot con un recorrido por niveles, una vez encontrado si el bot tiene descendientes recorremos la lista retornada por el método privado anteriormente mencionado, y creamos una respuesta acorde a lo pedido.

Esta consulta no conlleva grandes problemas.

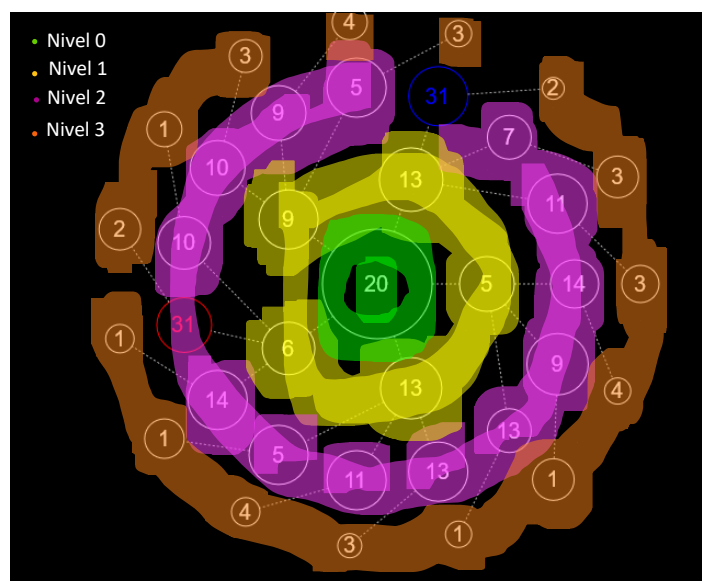
Y llegamos a la respuesta buscada.

Planetas descendientes del Bot: 10 11 12 3 4 2

### Consulta 3:

En este caso deberemos retornar un texto con la población de cada nivel y su promedio por nivel.

Sabiendo esto, ya nos hacemos la idea de que trabajaremos con un recorrido por niveles.



```

public String Consulta3( ArbolGeneral<Planeta> arbol)
{
    Cola<ArbolGeneral<Planeta>> cola = new Cola<ArbolGeneral<Planeta>>();

    ArbolGeneral<Planeta> actual;
    cola.encolar(arbol);
    cola.encolar(null);

    int cantPlanetasXNivel = 0;
    int cantPobXNivel = 0;
    int cantPobTotal = 0;
    int nivel = 0;
    int PromPobXNivel = 0;
    string res = "";

    while (!cola.esVacia())
    {
        actual = cola.desencolar();

        if (actual != null)
        {
            cantPlanetasXNivel++;
            cantPobXNivel += actual.getDatoRaiz().Poblacion();
            cantPobTotal += actual.getDatoRaiz().Poblacion();
            PromPobXNivel = cantPobXNivel / cantPlanetasXNivel;
            foreach (var hijo in actual.getHijos())
            {
                cola.encolar(hijo);
            }
        }
        else
        {
            res += "Nivel " + nivel + ". Poblacion: " + cantPobXNivel + " Promedio de poblacion " + " : " + PromPobXNivel + "\n";
            cantPlanetasXNivel = 0;
            cantPobXNivel = 0;
            nivel++;
            PromPobXNivel = 0;

            if (!cola.esVacia())
            {
                cola.encolar(null);
            }
        }
    }

    return "Poblacion y Promedio por nivel:\n" + res + "Poblacion Total: " + cantPobTotal;
}

```

En este método hacemos un recorrido por niveles, pero como debemos poder diferenciar cada nivel del otro, utilizamos la táctica de encolar null para determinar el pasaje de un nivel al otro.

Por cada nivel, vamos calculando cada uno de los ítems necesarios para la resolución, y los agregamos al texto final cuando nos encontramos en un pasaje de nivel.

Esta consulta es bastante sencilla, solo hay que tener en cuenta las variables necesarias para la respuesta y manejarlas para que se devuelva la información por cada nivel del árbol. Para esto último el encolar los null resulta ser la clave del ejercicio.

Respuesta:

```

Poblacion y Promedio por nivel:
Nivel 0. Poblacion: 20 Promedio de poblacion : 20
Nivel 1. Poblacion: 71 Promedio de poblacion : 14
Nivel 2. Poblacion: 154 Promedio de poblacion : 10
Nivel 3. Poblacion: 69 Promedio de poblacion : 4
Poblacion Total: 314

```

Por último, debemos darle vida al bot de tal forma que este ataque al planeta del jugador.

¿Cómo lo haremos?

A grandes rasgos, necesitamos un camino que vaya desde el Planeta Bot al Planeta del Jugador, este camino será la ruta de ataque de la IA. Parece sencillo, pero no lo es.

A continuación, analizaremos un árbol para entrar en contexto:



Si reflexionamos sobre estos distintos caminos, podemos darnos cuenta que el camino de ataque está conformado por una combinación entre el camino de la raíz al bot, y el camino de la raíz al jugador.

Entonces, el primer paso es crear un método que nos devuelva estos caminos.

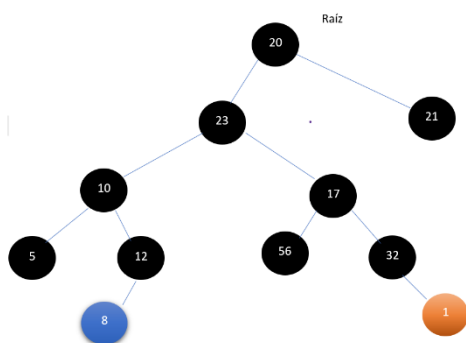
Una adaptación del recorrido preorden para búsqueda de caminos es una buena opción, de esta forma vamos buscando caminos y descartándolos, hasta dar con uno que encuentre al bot o al jugador según el caso que estemos analizando.

Bien una vez tengamos ambos caminos, aparece la siguiente pregunta:

¿Necesariamente necesitamos atacar la raíz? La respuesta es, depende.

Si el ancestro en común entre ambos caminos es la raíz, deberemos atacarla, caso contrario debemos calcular este ancestro en común para que el ataque tenga la mayor precisión posible.

Okay, tenemos ambos caminos y tenemos el LCA. Todos los ingredientes para el camino de ataque están listos.




Raíz a bot = 20 23 10 12 8

Raíz a jugador = 20 23 17 32 1

LCA = 23

Ataque = 8 12 10 23 17 32 1





A través de un método encargado de armar el camino de ataque, recorreremos el camino del bot inversamente añadiendo a la lista de ataque los planetas hasta toparnos con el LCA, luego recorreremos la lista del jugador y una vez encontrado el LCA comenzamos a añadir a la lista de ataque.

...

Bien, ya tenemos la lista de ataque solo resta hacer al ataque realidad.

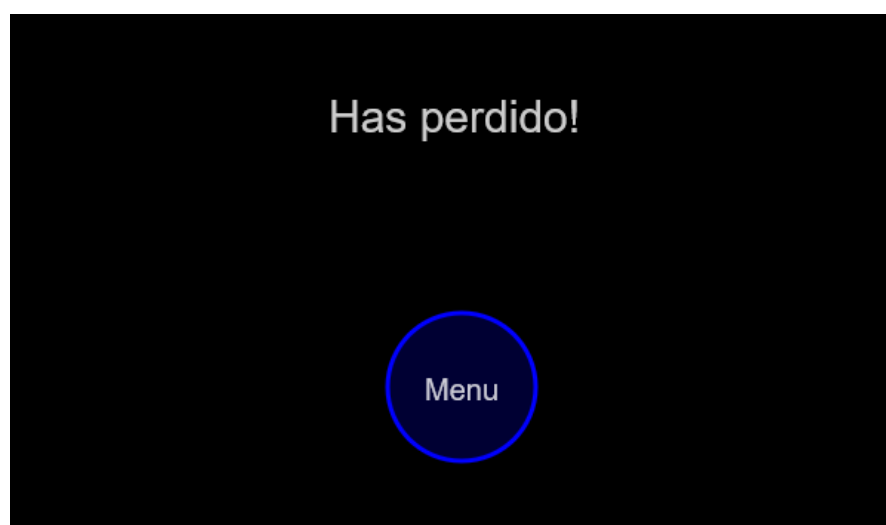
Como dijimos al principio del informe, la clase Movimiento nos otorga las herramientas para realizar el ataque. De esta forma, dentro del método `CalcularMovimiento (ArbolGeneral<Planeta> arbol)` en la clase estrategia, instanciaremos una serie de pautas para que el bot ataque al jugador, pero de una forma inteligente.

El bot además de atacar, podrá pedir ayuda de sus otros planetas con el fin de hacerle la batalla más complicada al jugador.

Entonces atacara a todos los planetas que no sean de su propiedad y que estén en la línea de ataque hacia el jugador. De esta forma, si se ve en aprietos sus planetas aliados irán en su ayuda. Para lograr esto, utilizamos el camino de ataque, y distintas referencias a los hijos del bot para discernir si se puede o no pedir refuerzos.

Todo el desarrollo para llegar a esta funcionalidad, fue sin dudar lo mas complicado del trabajo. En un principio, no se me ocurría el cómo, pero gracias a las variadas ayudas que proporciona la catedra logre orientarme.

Nunca pensé que perder en algo sería tan satisfactorio





# Conclusión

Para ir cerrando con el trabajo, creo que lo mejor sería compartir mi reflexión y valoración sobre este recorrido.

La forma en la que el trabajo está ambientado en un juego sencillo, curioso y como este conecta el entorno con el de la catedra me parece altamente satisfactorio. Permite desarrollar y sobrellevar de una forma didáctica y entretenida algunos de los temas vistos durante la materia. Ponemos en práctica conocimientos sobre Estructuras de Datos y Algoritmos eficientes dentro de un contexto, “Conquista Planetaria”, otorgando un punto de vista que la verdad ayuda mucho.

Sencillo no resulto, pero creo que la esencia del trabajo se encuentra en aprender a manipular las estructuras de datos, que, en combinación con el contexto de los planetas, podemos analizar cada caso para decidir que algoritmo es pertinente utilizar para la correcta devolución de los ejercicios.

Mayoritariamente, las consultas requirieron un buen entendimiento de cómo estaba estructurado todo el programa, mientras que el CalcularMovimiento conllevó una reflexión más profunda para comprender lo que necesitábamos. Los distintos recorridos que se pueden hacer sobre un Arbol General resultaron ser la herramienta más contundente.

Para terminar, creo que el juego cumple con lo buscado y estoy satisfecho con lo que me deja de enseñanza y práctica. Me hubiera gustado crear una inteligencia artificial mejor desarrollada y mas detallada para el Bot, pero creo que cumple con lo pedido.

No aporté ninguna sugerencia para la mejora del juego, ya que creo que cumple con su finalidad perfectamente.

Gracias por la experiencia.



*The End*

