INFORME TRABAJO FINAL

Materia: Complejidad Temporal, Estructuras de Datos y Algoritmos

Profesor/a: Leonardo Amet

Alumno: Yanequine,Lautaro Martin

Dni: 43.901.862

Legajo:55.838

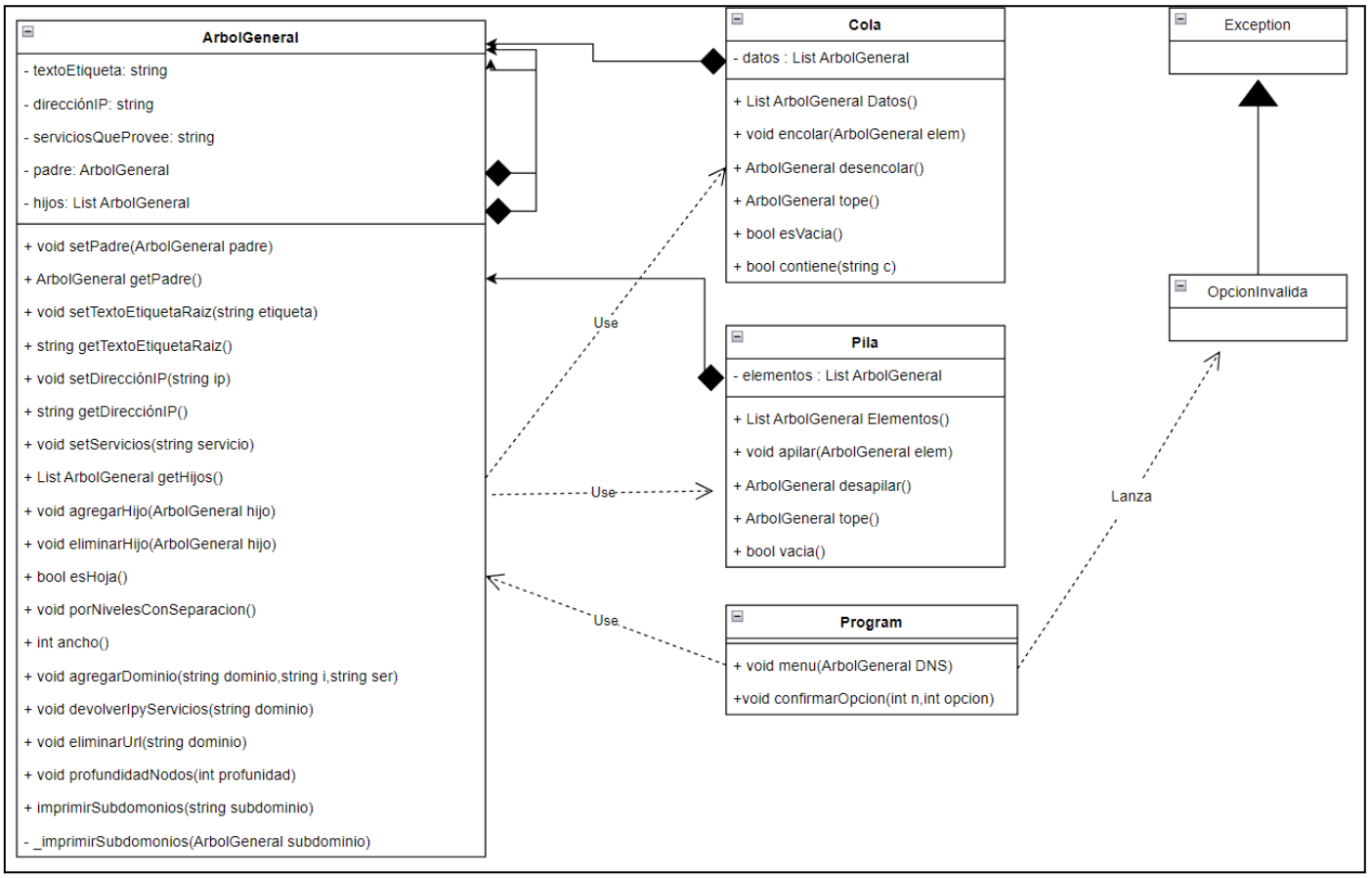
INTRODUCCIÓN

.

El presente trabajo forma parte de la materia Complejidad Temporal, Estructuras de Datos y Algoritmos. Tiene como objetivo general el desarrollo de un sistema que utilice conceptos vistos en la materia.

DESARROLLO

DIAGRAMA UML



1.Diagrama UML del sistema

La clase ArbolGeneral es la principal del sistema. Es similar a la clase que utilizamos durante la cursada,solamente que va a almacenar en el nodo datos de tipo string y se agregaron distintas implementaciones para que funcione correctamente lo pedido. Un ArbolGeneral está compuesto recursivamente por ArbolesGenerales en una lista de hijos y también está compuesto por un ArbolGeneral que sería el padre.

Como la mayoría de problemas están resueltos con recorrido por niveles,hay varios métodos que hacen usa de la clase Cola,la cual tiene una lista de ArbolesGenerales.Los métodos que utilizan la clase Cola son :

- porNivelesConSeparacion
- ancho
- agregarDominio
- devolverIpYServicios
- eliminarUrl
- imprimirSubdominio
    - _imprimirSubdominio
- profundidadNodos

El método _imprimirSubdomonios,particularmente además utiliza la clase Pila,que también tiene una lista de ArbolesGenerales.

La clase Program tiene el método menú,que es el encargado de interactuar con el usuario,dependiendo de qué opciones elija el usuario,menú va a ir usando diferentes comportamientos de la clase ArbolGeneral. También tiene el método confirmarOpcion el cual si el usuario elige una opción incorrecta lanza una excepción para avisarle al usuario y que vuelva a elegir una opción correcta.

IMPLEMENTACIÓN

Clase Arbol General

En primer lugar, se mostrará la implementación de la clase ArbolGeneral

```
namespace TPF
{
    public class ArbolGeneral
    {


        private string textoEtiqueta,direccionIP,serviciosQueProvee;
        private ArbolGeneral padre;

        private List<ArbolGeneral> hijos = new List<ArbolGeneral>();

        public ArbolGeneral(string textoEtiqueta,ArbolGeneral padre) {
            this.textoEtiqueta = textoEtiqueta;
            this.padre=padre;
        }

        public void setPadre(ArbolGeneral padre){this.padre=padre;}
        public ArbolGeneral getPadre(){return this.padre;}
        public void setTextoEtiquetaRaiz(string etiqueta){this.textoEtiqueta = etiqueta;}
        public string getTextoEtiquetaRaiz() {return this.textoEtiqueta;}
        public void setDireccionIP(string ip){this.direccionIP = ip;}
        public string getDireccionIP() {return this.direccionIP;}
        public void setServicios(string servicio){this.serviciosQueProvee = servicio;}
        public string getServicios() {return this.serviciosQueProvee;}
        public List<ArbolGeneral> getHijos() {return hijos;}
        public void eliminarHijo(ArbolGeneral hijo) {this.getHijos().Remove(hijo);}
        public bool esHoja() {return this.getHijos().Count == 0;}
        public void agregarHijo(ArbolGeneral hijo) {
            this.getHijos().Add(hijo);
            hijo.setPadre(this);
        }
```

2. Implementación clase Arbol General

Se observa los atributos de la clase,el constructor,que recibe solamente el texto de la etiqueta(subdominio) y la referencia al árbol padre.También los diferentes setters y getters a los atributos que se utilizaran en la implementación del sistema. El método eliminar Hijo elimina el árbol pasado por parámetro de la lista de hijos,el método esHoja devuelve true si el árbol no tiene hijos y el método agregar Hijo,agrega al árbol pasado por parámetro a la lista de hijos del árbol y al árbol agregado se le setea el padre como el árbol que lo agrega.

```
public void porNivelesConSeparacion(){
    Cola c = new Cola();
    ArbolGeneral arbolAux;
    int nivel = 0;
    c.encolar(this);
    c.encolar(null);
    Console.Write("Nivel " + nivel + ": ");
    while(!c.esVacia()){
        arbolAux = c.desencolar();
        if(arbolAux == null){
            if(!c.esVacia()){
                nivel++;
                Console.Write("\nNivel " + nivel + ": ");
                c.encolar(null);
            }
        }
        else{
            Console.Write(arbolAux.textoEtiqueta + " ");

            foreach(var hijo in arbolAux.hijos)
                c.encolar(hijo);
        }
    }
}
```

3. Implementación método por Niveles Con Separación de clase Arbol General

El método porNivelesConSeparacion es la “plantilla” que se utiliza para recorrer el árbol.Utiliza la clase Cola.

Método ancho

```
public int ancho(){
    Cola c = new Cola();
    ArbolGeneral arbolAux;
    int anc= 0; //Lleva la cuenta del ancho del arbol
    int contNodos= 0 ; //Cuenta los Arboles_Generales porNiveles
    c.encolar(this);
    c.encolar(null);
    while(!c.esVacia()){
        arbolAux = c.desencolar();
        if(arbolAux == null){

            if(contNodos>anc)                    //Me voy quedando con el ancho
                anc=contNodos;
            contNodos=0;                        //reseteo contador
            if(!c.esVacia()){
                c.encolar(null);
            }
        }
        else{
            contNodos++;
            foreach(var hijo in arbolAux.hijos)
                c.encolar(hijo);
        }
    }
    return anc;
}
```

4. Implementación método ancho de clase Arbol General

El método ancho utiliza el recorrido de niveles,para devolver el grado del árbol.Este método se utiliza en el método _imprimirSubdominios.

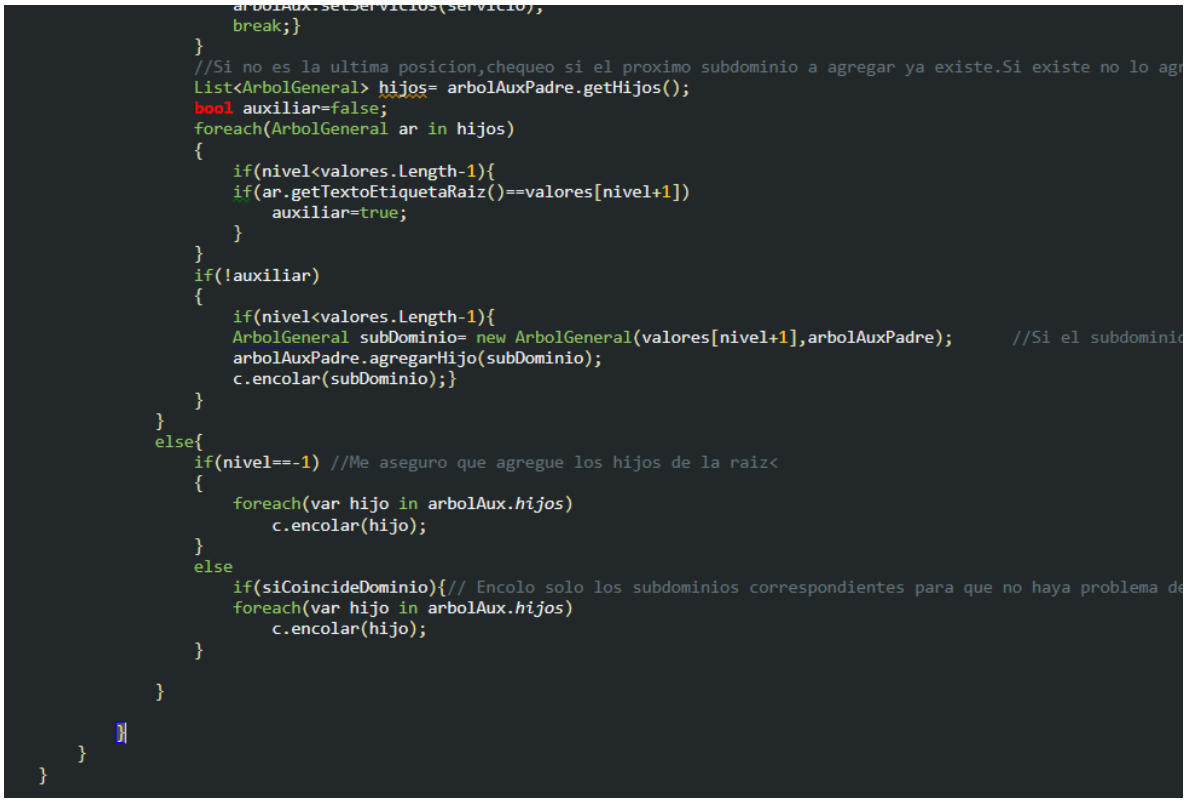
Metodo agregar Dominio

```
public void agregarDominio(string dominio,string i,string ser){
    |
    string ip= i,servicio=ser;
    string[] valores= dominio.Split('.'); //Separo el string en partes por el "."
    Array.Reverse(valores); //Doy vuelta los valores del array para que coincidan los niveles con los indices.
    bool siCoincideDominio=false;//me indica si el dominio a agregar ya existe
    Cola c = new Cola();
    ArbolGeneral arbolAux,arbolAuxPadre=this; //Se tiene una referencia al arbolpadre para agregarle el subdominio
    int nivel = -1; //Inicializo el nivel en -1 ,para que coincida posiciones del nivel con el array de strings
    c.encolar(this);
    c.encolar(null);

    while(!c.esVacia()){
        arbolAux = c.desencolar();
        if(arbolAux == null){
            if(!c.esVacia()){
                nivel++;
                if(nivel>=0) //Si no es la raiz
                {
                    if(!c.contiene(valores[nivel]) ){ //Chequeo si en el nivel existe el subdominio a agregar si no exist
                        ArbolGeneral subDominio= new ArbolGeneral(valores[nivel],arbolAuxPadre);
                        arbolAuxPadre.agregarHijo(subDominio);
                        c.encolar(subDominio);
                    }
                }
                c.encolar(null);
            }
        }
        else{
            if(nivel>=0 && nivel< valores.Length) // Caso  si ya procese la raiz y que no haya procesado los valores a a
            {
                if(string.Compare( arbolAux.textoEtiqueta,valores[nivel]) == 0  ) //Si en el nivel existe el subdominio
                {
                    siCoincideDominio=true;
                    arbolAuxPadre=arbolAux; //Pasa a ser el padre,ya que el proximo subdominio se le asignara a este
                }
                else
                    siCoincideDominio=false;
            }
```

5. Implementación método agregaDominio de clase Arbol General

```
if(siCoincideDominio)                   //Si el subdominio ya existe
{
    bool auxiliar=false;
    if(nivel>=valores.Length-1 ) //Si no es la ultima posicion del array,ejemplo si es www,no se le puede agre
    {
        arbolAux.setDireccionIP(ip);
        arbolAux.setServicios(servicio);
        break; //Ya agregue todo el dominio,termina la ejecucion
    }
    //Recorro los subdominios del padre para chequear si ya tiene ese subdominio o no
    List<ArbolGeneral> hijos= arbolAuxPadre.getHijos();
    foreach(ArbolGeneral ar in hijos)
    {
        if(ar.getTextoEtiquetaRaiz()==valores[nivel+1])
            auxiliar=true;
    }
    if(!auxiliar){                          //En caso q no tenga el subdominio lo agrego
        ArbolGeneral subDominio= new ArbolGeneral(valores[nivel+1],arbolAuxPadre);
        arbolAuxPadre.agregarHijo(subDominio);
    }
}

if(arbolAux.esHoja())                   //Caso si el subdominio es hoja
{

    if(nivel>=valores.Length-1 ) //Si es la ultima posicion le agrego los datos IP y servicios
    {
        if(string.Compare(valores[nivel],arbolAux.getTextoEtiquetaRaiz())==0){
        arbolAux.setDireccionIP(ip);
        arbolAux.setServicios(servicio);
        break;}
    }
    //Si no es la ultima posicion,chequeo si el proximo subdominio a agregar ya existe.Si existe no lo agrego
    List<ArbolGeneral> hijos= arbolAuxPadre.getHijos();
    bool auxiliar=false;
    foreach(ArbolGeneral ar in hijos)
    {
        if(nivel<valores.Length-1){
        if(ar.getTextoEtiquetaRaiz()==valores[nivel+1])
```

6. Implementación método agregaDominio de clase Arbol General



```
                    arbolAux.setServicios(servicio);
                    break;}
                }
                //Si no es la ultima posicion,chequeo si el proximo subdominio a agregar ya existe.Si existe no lo agr
                List<ArbolGeneral> hijos= arbolAuxPadre.getHijos();
                bool auxiliar=false;
                foreach(ArbolGeneral ar in hijos)
                {
                    if(nivel<valores.Length-1){
                    if(ar.getTextoEtiquetaRaiz()==valores[nivel+1])
                        auxiliar=true;
                    }
                }
                if(!auxiliar)
                {
                    if(nivel<valores.Length-1){
                    ArbolGeneral subDominio= new ArbolGeneral(valores[nivel+1],arbolAuxPadre);      //Si el subdominio
                    arbolAuxPadre.agregarHijo(subDominio);
                    c.encolar(subDominio);}
                }
            }
            else{
                if(nivel==-1) //Me aseguro que agregue los hijos de la raiz<
                {
                    foreach(var hijo in arbolAux.hijos)
                        c.encolar(hijo);
                }
                else
                    if(siCoincideDominio){// Encolo solo los subdominios correspondientes para que no haya problema de
                    foreach(var hijo in arbolAux.hijos)
                        c.encolar(hijo);
                }

            }

        }
    }
}
```

7. Implementación método agregaDominio de clase Arbol General

El método agregarDominio tiene como objetivo el ingreso y almacenamiento de nombres de dominio correspondientes a equipos conectados a la red, junto con sus direcciones IPs y los servicios que estos proveen.

Recibe el dominio entero que se quiere agregar(ej “www.wikipedia.com”),la dirección IP y servicio que se le va a asignar a la hoja de la url(ej."www).Luego preparó el terreno para posteriormente ejecutar el algoritmo de agregado.Utilizó la función split,separando cada string por el ".",que me devuelve un array de strings,cada posición con cada subdominio, doy vuelta los valores del array para que coincidan los niveles con los índices,declaro una variable booleana que me indica si el dominio a agregar ya existe,se tiene dos referencias de árboles,uno apuntó al árbol desencolado y otro al padre que le tendría que agregar el subdominio ,por último , inicializo el nivel en -1 ,para que coincida posiciones del nivel con el array de strings,había surgido un problema en este punto ya que la raíz del árbol no tiene que ser procesada ya que su etiqueta no tiene valor(“ ”).

Luego procedo a ejecutar el algoritmo de agregado,tomando como base el recorrido por niveles con separación

- Por cada árbol desencolado,primero chequeo si es null
    - En el caso que no sea:
        - Me fijo si el nivel es mayor o igual a 0(quiere decir que ya procese la raíz) y que el nivel sea menor a la longitud del array de subdominios a agregar,si es mayor significa que ya lo procese
            - Si esto sucede
                - chequeo si el subdominio a agregar es igual al texto del árbol desencolado(es decir si existe en el nivel)
                    - Si sucede:pongo true en coincide el dominio y la referencia al árbol padre sería el árbol que contiene el subdominio,ya que el próximo elemento se le agrega a este último
                    - Si no sucede: Queda falso el booleano subdominio

- Luego,chequeo si la variable “SiCoincideDominio” es true
    - Si esto sucede,chequeo si el nivel coincide con la última posición del array(seria la hoja)
        - Si esto sucede,comparó el subdominio a agregar con la etiqueta del árbol desencolado.Si esto último es verdadero le asigno la dirección IP y los servicios y termina la ejecución
    - Recorro los subdominios del padre(sería el árbol desencolado) para chequear si ya tiene el subdominio a agregar o no.Si no lo tiene, lo agrego.
- Si el árbol desencolado es hoja
    - Si esto sucede,chequeo si el nivel coincide con la última posición del array(seria la hoja)
        - Si esto sucede,comparó el subdominio a agregar con la etiqueta del árbol desencolado.Si esto último es verdadero le asigno la dirección IP y los servicios y termina la ejecución
    - Recorro los subdominios del padre(padre de la hoja) para chequear si ya tiene el subdominio a agregar o no.Si no lo tiene, lo agrego y lo encolo.
- Si el árbol desencolado no es hoja
    - Si el SiCoindiceDeDominio es true o es la raíz,,encolo a sus hijos.Me aseguro que siempre encolelos hijos de la raíz y que solo se encolen los subdominios correspondientes.

- En el caso que sea null
    - Si la cola no está vacía(Para evitar loop infinito)

- Subo el nivel
- Si no es la raíz
    - me fijo si en el nivel actual está el próximo subdominio a agregar .Si no está lo agrego y lo encolo(se agrega en el mismo nivel,antes que él null)
- Encolo el null

Uno de los principales problemas que tuve con este método era cuando quería agregar un subdominio diferente en el mismo nivel. Por ejemplo, yo tengo en el DNS la dirección "www.facebook.com" y quiero agregar la "www.facebook.org". Esto lo solucione chequeando ,antes de agregar el null de separación de nivel, si estaba encolado(existía) el próximo subdominio(en este caso "org"),si no estaba lo agregaba y encola . Sin esta implementación,iba a recorrer todo el DNS y no iba a encontrar ninguna coincidencia y por ende no podría agregar la nueva url.

Método devolver IP y Servicios

```
public void devolverIpyServicios(string dominio){
    string[] valores= dominio.Split('.');
    Array.Reverse(valores);
    bool siCoincideDominio=false;
    Cola c = new Cola();
    ArbolGeneral arbolAux,arbolAuxPadre=this;
    int nivel = -1;
    c.encolar(this);
    c.encolar(null);
    while(!c.esVacia()){
        arbolAux = c.desencolar();
        if(arbolAux == null){
            if(!c.esVacia()){
                nivel++;
                c.encolar(null);}
        }
        else{
            if(nivel>=0 && nivel< valores.Length)
            {
                if(string.Compare( arbolAux.textoEtiqueta,valores[nivel]) == 0  )
                {
                    siCoincideDominio=true;
                    arbolAuxPadre=arbolAux;
                }
                else
                    siCoincideDominio=false;
            }
            if(nivel==-1)
            {
                foreach(var hijo in arbolAux.hijos)
                    c.encolar(hijo);
            }
            else
                if(siCoincideDominio){
                foreach(var hijo in arbolAux.hijos)
                    c.encolar(hijo);
                }
        }
    }
    if(valores[valores.Length-1]==arbolAuxPadre.getTextoEtiquetaRaiz())
        Console.WriteLine("La dirección IP es : "+arbolAuxPadre.getDireccionIP());
    else
        Console.WriteLine("No se encuentra la dirección IP,ya que  no existe el dominio buscado");

}
```

8. Implementación método devolver IP y Servicios de clase Arbol General

El objetivo de este método es dado un nombre de dominio correspondiente a un equipo imprimir su dirección IP junto con los servicios que provee.

El método devolver IP y servicios,utiliza un recorrido por niveles,solo encolando los hijos de los subdominios que coinciden,de manera que siga un “camino”. Al final,sí se encontró la URL,tiene que coincidir el texto de la hoja con el texto del último elemento del array de valores,si esto ocurre

se imprime la dirección IP y los servicios que provee,de lo contrario se le notifica al usuario que no existe el dominio solicitado.

Método eliminar URL

```
public void eliminarUrl(string dominio){

string[] valores= dominio.Split('.');
Array.Reverse(valores); //Doy vuelta los valores del array para que coincidan los niveles con los i
bool hayEliminar=true;  //Booleano auxiliar para ver si sigo eliminando mismos subdominios .
bool imprimio=false;

while(hayEliminar){  //Se pone todo en un bucle para chequear si hay otro subdominio con la misma e
    ArbolGeneral[] aEliminar= new ArbolGeneral[(valores.Length+1)];  //Array de arboles que van a a
    aEliminar[0]=this;
    Cola c = new Cola();
    ArbolGeneral arbolAux;
    int nivel = -1;
    int aux1=0;//Auxiliar de indice de valores
    c.encolar(this);
    c.encolar(null);
    while(!c.esVacia()){
        arbolAux = c.desencolar();
        if(arbolAux == null){
            if(!c.esVacia()){
                nivel++;
                c.encolar(null);}
        }
        else{
            if(nivel>=0 && aux1< valores.Length)
            {
                if(string.Compare( arbolAux.textoEtiqueta,valores[aux1]) == 0  ) //Si encuentro el
                {aEliminar[aux1+1]=arbolAux;//Lo agrego al array para eliminarlo
                    aux1++; } //Incremento el indice de valores,para buscar el proximo subdominio a
            }
            foreach(var hijo in arbolAux.hijos) //Encolo todos los hijos,pq puede haber el caso que
            c.encolar(hijo);
        }
    }
    Array.Reverse( aEliminar); //Invierto los valores del array por implementacion
    foreach (ArbolGeneral element in aEliminar) {
        if(element==null)  //Si tengo un null en aEliminar significa que una parte no se encontro.
            hayEliminar=false;
    }
    if(hayEliminar){  //Si no hay entonces procedo a eliminar
        ArbolGeneral arbolAEliminar=aEliminar[0];  //Se le asigna el primer elemento a eliminar,lue
        ArbolGeneral padreDeAEliminar=aEliminar[0].getPadre();
        if(padreDeAEliminar != null){
            padreDeAEliminar.eliminarHijo(arbolAEliminar);
            while(padreDeAEliminar.esHoja()){ //Un bucle para ver si mientras elimino subdominio,el
                arbolAEliminar=padreDeAEliminar;
                padreDeAEliminar=padreDeAEliminar.getPadre();
                if(padreDeAEliminar != null)
                    padreDeAEliminar.eliminarHijo(arbolAEliminar);
                if(padreDeAEliminar==null)
```

9. Implementación método eliminar Url de clase Arbol General

```
            }
        }
        Array.Reverse( aEliminar); //Invierto los valores del array por implementacion
        foreach (ArbolGeneral element in aEliminar) {
            if(element==null)  //Si tengo un null en aEliminar significa que una parte no se
                hayEliminar=false;
        }
        if(hayEliminar){  //Si no hay entonces procedo a eliminar
            ArbolGeneral arbolAEliminar=aEliminar[0];  //Se le asigna el primer elemento a el
            ArbolGeneral padreDeAEliminar=aEliminar[0].getPadre();
            if(padreDeAEliminar != null){
                padreDeAEliminar.eliminarHijo(arbolAEliminar);
                while(padreDeAEliminar.esHoja()){ //Un bucle para ver si mientras elimino sub
                    arbolAEliminar=padreDeAEliminar;
                    padreDeAEliminar=padreDeAEliminar.getPadre();
                    if(padreDeAEliminar != null)
                        padreDeAEliminar.eliminarHijo(arbolAEliminar);
                    if(padreDeAEliminar==null)
                        break;
                }
            }

            if(!imprimio){
                Console.WriteLine("Se elimino correctamente la url");
                imprimio=true;
            }
        }
        else
            if(!imprimio){
            Console.WriteLine("No se encontro la Url a eliminar");
            imprimio=true;
        }


    }
}
```

10. Implementación método eliminar Url de clase Arbol General

El objetivo de este método es la eliminación de nombres de equipos, verificando que no queden subdominios o dominios de nivel superior vacíos (sin hijos) producto de la eliminación. En los casos donde se detecten subdominios o dominios de nivel superior vacíos, estos deberán ser eliminados de la estructura de datos.

Recibe el dominio o subdominio que se quiera eliminar .Luego preparó el terreno para posteriormente ejecutar el algoritmo de agregado.Utilizó la función split,separando cada string por el ".",que me devuelve un array de strings,cada posición con cada subdominio, doy vuelta los valores del array para que coincidan los niveles con los índices,declaro una variable booleana que me indica si hay subdominios a eliminar y otra para chequear si ya se imprimió el mensaje al usuario.Luego ejecutó el algoritmo

- Mientras haya subdominios a eliminar
    - Creo un array de Árboles Generales,en donde cada índice va a apuntar a los árboles a eliminar(aEliminar)
    - Se tiene un entero auxiliar que tiene el índice del array aEliminar
    - Se realiza el recorrido por niveles de manera similar al del agregado.
    - Cuando encuentre el subdominio a eliminar lo agrego al array aEliminar e incrementó el índice del auxiliar.
    - Una vez termina el recorrido invierto el array aEliminar
    - Chequeo si hay elementos a eliminar,recorro el array aEliminar si hay un null significa que no se encontró el subdominio
    - Si hay subdominios a eliminar
        - El árbol a elminar va a ser el primer elemento del array aEliminar
        - El padre del árbol a eliminar va a ser el padre del árbol a eliminar.
        - Si el padre del árbol a eliminar no es nulo
            - Eliminó el árbol a eliminar
            - Mientras el padre del árbol a eliminar sea hoja(puede quedar en hoja producto de la eliminación del arbol)
                - Árbol a eliminar va a ser padre a eliminar
                - Padre a eliminar va a ser padre de árbol a eliminar
                - Si padre a eliminar no es nulo,eliminó árbol a eliminar
                - Si el padre es nulo,significa que ya elimine todos los subdominios correspondientes,término la ejecución
        - Si no se imprimió anteriormente,se imprime que se eliminó el dominio correctamente.
    - Si no hay subdominios a eliminar
        - Si no se imprimio nada,imprimo que no se pudo eliminar el subdominio

El principal problema que tuve en este método fue, que no podía eliminar subdominios que aparecian mas de una vez,por ejemplo si habia variacion de “facebook.com”,”facebook.org”,solo se eliminaba el primer “facebook”. Para solucionar esto se puso todo en un while que se ejecuta mientras haya subdominios a eliminar,de manera que encuentre los subdominios repetidos y pueda eliminarlos.

Método profundidad Nodos

El objetivo de este método es dar una profundidad e imprimir las cantidades de dominios de nivel superior, subdominios y equipos ubicados a dicha profundidad.

```
public void profundidadNodos(int profundidad){
    Cola c = new Cola();
    ArbolGeneral arbolAux;
    int cantAntes=0,cantProfundidad=0,nivel = 0;
    c.encolar(this);
    c.encolar(null);

    while(!c.esVacia() ){
        if(nivel>profundidad)
            break;
        arbolAux = c.desencolar();
        if(arbolAux == null){
            if(!c.esVacia()){
                nivel++;
                c.encolar(null);
            }
        }
        else{
            if(nivel==profundidad)
                cantProfundidad++;
            else{
                if(nivel != 0 )
                    cantAntes++;
            }
            foreach(var hijo in arbolAux.hijos)
                c.encolar(hijo);
        }
    }

    Console.WriteLine("Cantidad de dominios de nivel superior: "+cantAntes);
    Console.WriteLine("Cantidad de dominios de profundidad "+profundidad.ToString()+": "+cantProfundidad);
}
```

11. Implementación método profundidadNodos de clase Arbol General

Se realiza también mediante un recorrido por niveles,contando los nodos que hay antes de llegar a la profundidad pasada por parámetro y luego contando la cantidad de nodos que hay en la profundidad pasada por parámetro.

Método imprimir subdominios

El objetivo de este método es dar un nombre de dominio correspondiente a un subdominio e imprimir todos los nombres de los equipos que de él dependen. Este método está resuelto con dos métodos.

```
public void imprimirSubdominios(string subdominio)
    {
        List <ArbolGeneral> subdominios= new List<ArbolGeneral>();
        Cola c = new Cola();
        ArbolGeneral arbolAux,arbolAuxPadre=this;
        int nivel = -1;
        c.encolar(this);
        c.encolar(null);
        while(!c.esVacia()){
            arbolAux = c.desencolar();
            if(arbolAux == null){
                if(!c.esVacia()){
                    nivel++;
                    c.encolar(null);
                }
            }
            else{
                if(nivel>=0 )
                {
                    if(string.Compare( arbolAux.textoEtiqueta,subdominio) == 0)
                    {
                        arbolAuxPadre=arbolAux; //Pasa a ser el padre,ya que el proximo subdominio se le asigna
                        _imprimirSubdominios(arbolAuxPadre);  //Cada vez q encuentre el subdominio veo los subd
                    }
                }

                if(nivel==-1)
                {
                    foreach(var hijo in arbolAux.hijos) c.encolar(hijo);
                }
                else
                    foreach(var hijo in arbolAux.hijos)
                        c.encolar(hijo);
            }
        }

    }
```

12. Implementación método imprimir Subdominios de clase Arbol General

El primer método es el método al cual se va a invocar(público). Consiste en hallar el subdominio pasado por parámetro ,para imprimir los subdominios dependientes a él .Una vez que lo encontramos,llamamos a la segunda función que tiene el algoritmo de impresión por pantalla.Esto por cada vez que se encuentre el subdominio a imprimir(puede existir el caso que haya muchos “facebook”).

```
e void _imprimirSubdominios(ArbolGeneral arbol){
      int cantUrls= arbol.ancho();
      Pila[] Urls = new Pila [cantUrls]; //Array de Pilas,cada pila va a ir formando el dominio
      for (int i = 0; i < Urls.Length; i++) {
          //Las inicializo y ya almaceno el primer subdominio
          Urls[i]=new Pila();
          Urls[i].apilar(arbol);
      }
      int contPila=0;
      Cola c = new Cola();
      ArbolGeneral arbolAux;
      int nivel = 0;
      c.encolar(arbol);
      c.encolar(null);

      while(!c.esVacia()){
          arbolAux = c.desencolar();

          if(arbolAux == null){
              if(!c.esVacia()){
                  nivel++;
                  contPila=0;
                  c.encolar(null);
              }
          }
          else{
              if(!arbolAux.esHoja() && nivel !=0){  //Si no es hoja lo almaceno en todos los subdominios q lo comparten,lo comparten
                  int cantHijos= arbolAux.getHijos().Count;
                  for (int i = 0; i < cantHijos; i++) {
                      Urls[contPila].apilar(arbolAux);
                      contPila++;
                  }
              }
              else{
                  if(nivel !=0){
                      while(contPila< Urls.Length)
                      {
                          if(Urls[contPila].tope().getHijos().Contains(arbolAux) ){
                              Urls[contPila].apilar(arbolAux);
                              contPila++;
                              break;
                          } //Si es hoja solo lo almaceno en el indice q corresponde
                          contPila++;
```

13. Implementación método privado imprimir Subdominios de clase Arbol General

```
                              contPila++;


                          }


                      }
                  }
                  foreach(var hijo in arbolAux.hijos)
                      c.encolar(hijo);
              }

          }


          for (int i = 0; i < Urls.Length; i++) {//Imprimo todos los subdominio
              string url="";
              while(!Urls[i].vacia()){
                  ArbolGeneral aux= Urls[i].desapilar();
                  string aux2= aux.getTextoEtiquetaRaiz();
                  if(!Urls[i].vacia())    //Si tiene un proximo elemento
                      url+=aux2+".";
                  else
                      url+=aux2;
              }
              Console.WriteLine(url);
          }
      }
```

14. Implementación método privado imprimir Subdominios de clase Arbol General

Al método privado, le paso como parametro el árbol con el subdominio hallado. Se tiene una variable entera con la cantidad de Urls que van a poder formarse que va a ser el grado del árbol. Se va a utilizar la clase Pila,se crea un array de pilas,en cada una vamos a ir apilando los subdominios dependientes.Inicializo cada Pila y apilo el subdominio pasado por parámetro(“independiente”) ya que todos los urls que se formen van a tener este subdominio como principal. Se va a tener una variable entera que va a indicar en qué pila(índice) apilar los subdominios correspondientes.Luego realizó recorrido por niveles,con nivel a 0,iniciando por el subdominio "independiente".

- Si el árbol desencolado no es null
    - Si el árbol desencolado no es el árbol independiente y tampoco es hoja,quiere decir que ese subdominio lo comparten varios subdominios de niveles inferiores,entonces:
        - Me fijo cuantos subdominios dependientes tiene el árbol desencolado(hijos).
        - Apilo el árbol desencolado por cada subdominio dependiente que tenga e incremento el contador de pila
    - Si el árbol desencolado es hoja
        - Mientras que el índice de la pila sea menor a la longitud del array de pilas
            - Recorro los hijos del tope de todas las pilas y si coincide con el elemento a agregar, lo apilo ,aumentó el índice y salgo de la ejecución. Si es hoja solo lo almaceno en el índice que corresponda.
            - Aumento contador de pila
    - Encolo los hijos del árbol desencolado
- Si el arbol desencolado es null

- Si la cola no está vacía
    - Aumentó el nivel y el contador de pila se pone a cero, ya que al aumentar el nivel,el índice empezaría de 0 .Por último, encolo el null

Una vez terminado el recorrido,desapilo, cada pila del array,aprovechando su funcionamiento(LIFO),sumo cada dominio en un string y luego los imprimo.

Clase Cola y Pila

```
public class Cola
{
    private List<ArbolGeneral> datos = new List<ArbolGeneral>();

    public List<ArbolGeneral> Datos{get{return datos;}  }
    public void encolar(ArbolGeneral elem) {this.datos.Add(elem);   }

    public ArbolGeneral desencolar() {
        ArbolGeneral temp = this.datos[0];
        this.datos.RemoveAt(0);
        return temp;
    }

    public ArbolGeneral tope() {return this.datos[0]; }

    public bool esVacia() {return this.datos.Count == 0;}

    public bool contiene (string c){
        for(int i = 0; i<datos.Count; i++){
            if(datos[i].getTextoEtiquetaRaiz()==c)
                return true;
        }
        return false;
    }
}
}
```

15. Implementación clase Cola

```
public class Pila
{
    private List<ArbolGeneral> elementos = new List<ArbolGeneral>();

    public List<ArbolGeneral> Elementos{get{return elementos;}}
    public void apilar(ArbolGeneral elem) {elementos.Add(elem);}
    public ArbolGeneral desapilar()
    {
        ArbolGeneral aux;
        int tam=elementos.Count;
        aux=(ArbolGeneral)elementos[tam-1];
        elementos.Remove(aux);
        return aux;
    }
    public bool vacia() {return elementos.Count==0;}
    public ArbolGeneral tope()
    {
        int tam=elementos.Count;
        return (ArbolGeneral) elementos[tam-1];
    }
}
}
```

16. Implementación clase Pila

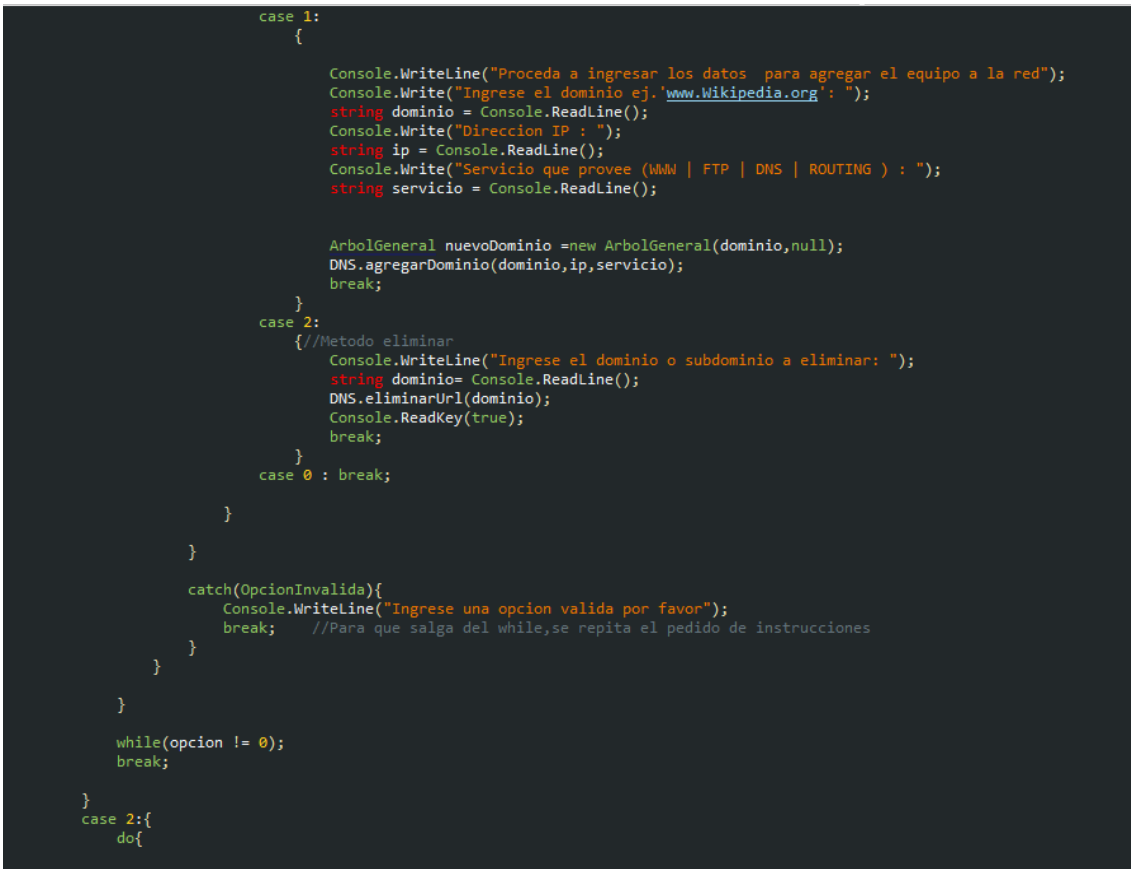
Tanto la clase Pila y Cola,almacenan Árboles Generales. Cada una tiene su funcionalidad. La clase cola tiene el método FIFO “primero en entrar, primero en salir”,la clase pila tiene el método LIFO “último en entrar, primero en salir”.

Clase Program

Método menú

```
public static void Menu(ArbolGeneral DNS)
{
    Console.WriteLine("BIENVENIDOS Al SISTEMA DNS ");
    Console.WriteLine("-----------------------------------------------------------");
    int opcion,opcion2;

    do{

        Console.Write("\n1. Ingresar a Administración " +
                      "\n2. Ingresar a Consultas." +
                      "\n0. Salir\nIngrese una opción: ");

        opcion2 = int.Parse(Console.ReadLine());
        try{
            confirmarOpcion(opcion2,1);
        }
        catch(OpcionInvalida){
                    Console.WriteLine("\nIngrese una opcion valida por favor"); //Para que salga del while,se repita el pedido de instrucciones

        }

        switch(opcion2)
        {
                case 1:{  //Modulo de Administracion
                    do{

                        Console.Clear();
                        Console.Write("\n1. Ingreso y almacenamiento de nombres de dominio correspondientes a equipos conectado a la red " +
                                      "\n2. Eliminación de nombres de equipos." +
                                      "\n0. Volver al menu principal" +
                                      "\nIngrese una opción: ");

                        opcion = int.Parse(Console.ReadLine());
                        bool auxmenu=true;
                        Console.Clear();

                        while(auxmenu){
                            try{
                                confirmarOpcion(opcion,1);
                                auxmenu=false;

                                switch(opcion)
                                {
                                    case 1:
                                        {
```

17. Implementación método Menú



```
                        case 1:
                            {

                                Console.WriteLine("Proceda a ingresar los datos  para agregar el equipo a la red");
                                Console.Write("Ingrese el dominio ej.'www.Wikipedia.org': ");
                                string dominio = Console.ReadLine();
                                Console.Write("Direccion IP : ");
                                string ip = Console.ReadLine();
                                Console.Write("Servicio que provee (WWW | FTP | DNS | ROUTING ) : ");
                                string servicio = Console.ReadLine();


                                ArbolGeneral nuevoDominio =new ArbolGeneral(dominio,null);
                                DNS.agregarDominio(dominio,ip,servicio);
                                break;
                            }
                        case 2:
                            {//Metodo eliminar
                                Console.WriteLine("Ingrese el dominio o subdominio a eliminar: ");
                                string dominio= Console.ReadLine();
                                DNS.eliminarUrl(dominio);
                                Console.ReadKey(true);
                                break;
                            }
                        case 0 : break;

                    }

                }

                catch(OpcionInvalida){
                    Console.WriteLine("Ingrese una opcion valida por favor");
                    break;    //Para que salga del while,se repita el pedido de instrucciones
                }
            }

        }

        while(opcion != 0);
        break;

    }
    case 2:{
        do{
```

18. Implementación método Menú

```
case 2:{
    do{

        Console.Clear();
        Console.Write("\n1.Ingreso el nombre del dominio para ver su direccion IP y los servicios que provee. " +
                      "\n2. Ingrese un subdominio para ver todos los equipos que dependen de él" +
                      "\n3. Ingrese una profundidad para ver  las cantidades de dominios de nivel superior, subdominios y equipos ubicados a dicha profundidad."+
                      "\n0. Volver al menu principal" +
                      "\nIngrese una opción: ");


        opcion = int.Parse(Console.ReadLine());
        bool auxmenu=true;
        Console.Clear();

        while(auxmenu){
            try{
                confirmarOpcion(opcion,2);
                auxmenu=false;

                switch(opcion)
                {
                    case 1:
                        { // Metodo ver ip y servicio
                            Console.WriteLine("Ingrese el dominio que necesite ver su direccion IP y los servicios que proveé: ");
                            string dominio= Console.ReadLine();
                            DNS.devolverIpyServicios(dominio);
                            Console.ReadKey(true);
                            break;
                        }
                    case 2:
                        {//Metodo ver subdominios dependientes
                            Console.WriteLine("Ingrese un subdominio para ver todos los equipos que dependen de él: ");
                            string dominio= Console.ReadLine();
                            DNS.imprimirSubdominios(dominio);
                            Console.ReadKey(true);
                            break;
                        }
                    case 3:
                        {
                            //metodo profundidad
                            Console.WriteLine("Ingrese la profundidad para ver las cantidades de dominios de nivel superior,subdominios y equipos ubicados a dicha profundidad: ");
                            int dominio= int.Parse(Console.ReadLine());
                            DNS.profundidadNodos(dominio);
                            Console.ReadKey(true);
```

19. Implementación método Menú

```
                        }
                    case 3:
                        {
                            //metodo profundidad
                            Console.WriteLine("Ingrese la profundidad para ver las cantidades de dominios de nivel superior,subdominios y equipos ubicados a dicha profundidad: ");
                            int dominio= int.Parse(Console.ReadLine());
                            DNS.profundidadNodos(dominio);
                            Console.ReadKey(true);
                            break;
                        }

                    case 0 : break;

                }

            }

            catch(OpcionInvalida){
                Console.WriteLine("Ingrese una opcion valida por favor");
                break;    //Para que salga del while,se repita el pedido de instrucciones
            }
        }

    }

    while(opcion != 0);
    break;
}
case 0:{
    break;
}
}


}
while(opcion2 !=0);      //Tenerlo en un while me obliga a elegir la opcion correcta siempre
Console.WriteLine();
Console.WriteLine("Fin del progrma");

}
```

20. Implementación método Menú

El método menu,recibe como parámetro al árbol raíz del DNS. Este método es el que interactúa con el usuario,ofreciéndole todas las funcionalidades requeridas,las de administración y consultas. Se asegura que cada vez que el usuario elije una opción,elija una opción correcta,esto poniendo la

opción de elegir opcion en un while. En el caso que el usuario elija una opción incorrecta el método confirmar Opción lanzara una excepción.

Método confirmar Opción

```
    public static void confirmarOpcion(int n,int opcion)
    {
        // modificar por si toca cualquier tecla

        if(opcion == 1){ //Caso "menu principal"
            if(n <0 || n >2 ){
                throw new OpcionInvalida();
            }
        }
        else if (opcion==2)
        {
            if(n <0 || n >3 ){
                throw new OpcionInvalida();
            }

        }

    }
}
```

21. Implementación método confirmar Opción

El método confirmar opción verifica si la opción elegida por el usuario es correcta,de lo contrario lanza una excepción

```
public class OpcionInvalida: Exception{};
```

22. Implementación excepción opcion Invalida

El método main

```
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");

        ArbolGeneral DNS = new ArbolGeneral("",null);
        Menu(DNS);



        // TODO: Implement Functionality Here

        Console.Write("Press any key to continue . . . ");
        Console.ReadKey(true);
    }
```

23. Implementación clase Program

El método Main solo instala el dns y luego llama al menú con parámetro al DNS y el sistema ya está listo para usar.

Mejoras

Al sistema se le pueden hacer varias mejoras. El uso de interfaces y patrones de diseño podrían hacer al sistema más escalable y más fácil de mantener. De manera que utilice datos genéricos tanto la clase árbol,como cola y pila,que no haya dependencia a una clase o dato específico sino a una interfaz.

También se podrían hacer mejorar en el metodo agregarDominio para que sea más intuitivo.

Otra mejora que se le puede hacer es a la clase confirmar Opción, haciendo que reciba un string en vez de un número,ya que si el usuario escribe una letra en vez de una opción numérica el sistema lanzaría una excepción no controlada.

Conclusión

Finalizado el desarrollo del trabajo final podemos describir varios conceptos asimilados y utilizados que se vieron en la materia Complejidad temporal,estructuras de datos y algoritmos,principalmente la estructura de datos tipo árbol general y sus recorridos.

Gracias al presente trabajo, se pudo observar el funcionamiento ,utilidad y aplicabilidad de la estructura de datos tipo árbol.También entender en qué contexto es bueno utilizar cierta estructura de datos sobre la otra(grafo,árbol,hash,heaps,etc), el costo y beneficio que tienen cada una de sus implementaciones. A su vez, tratar de implementar algoritmos que no sean costos en términos de temporalidad,para que su ejecución lleve el menor tiempo posible y el sistema sea lo más óptimo.

Como conclusión final, el trabajo fue muy enriquecedor ya que permite al alumno incorporar los conceptos de estructuras de datos,sus características y la complejidad que tiene un algoritmo, lo cual es muy beneficioso,ya que en la vida de un desarrollador va a haber innumerables ocasiones en la que se utilizan estos conceptos.