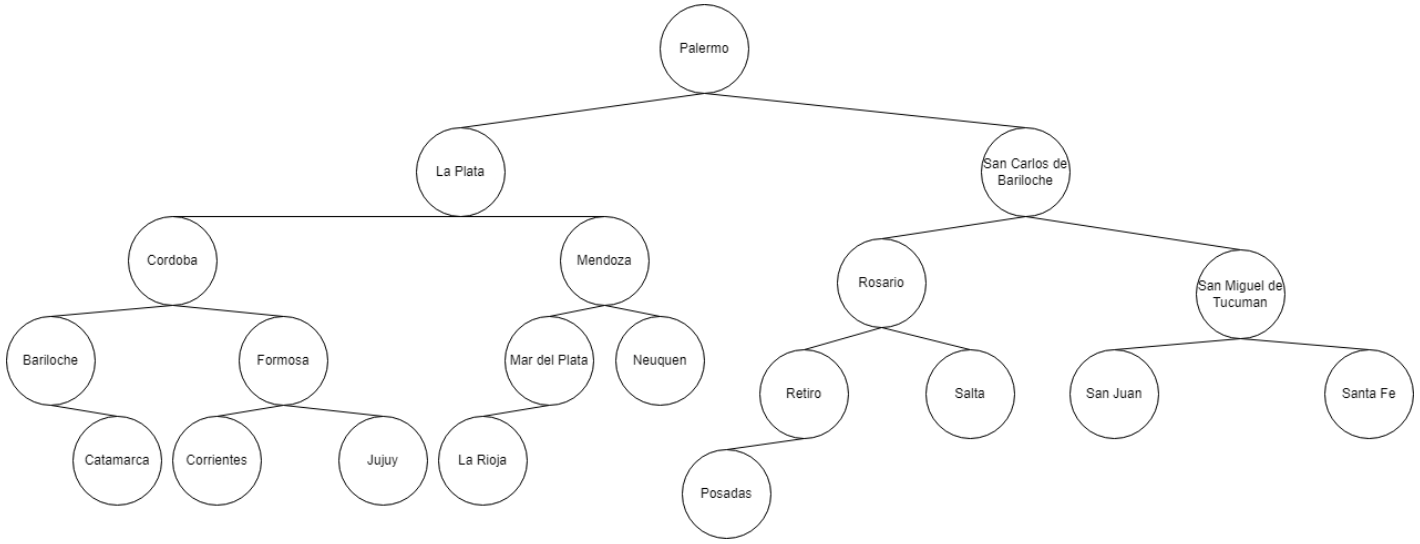
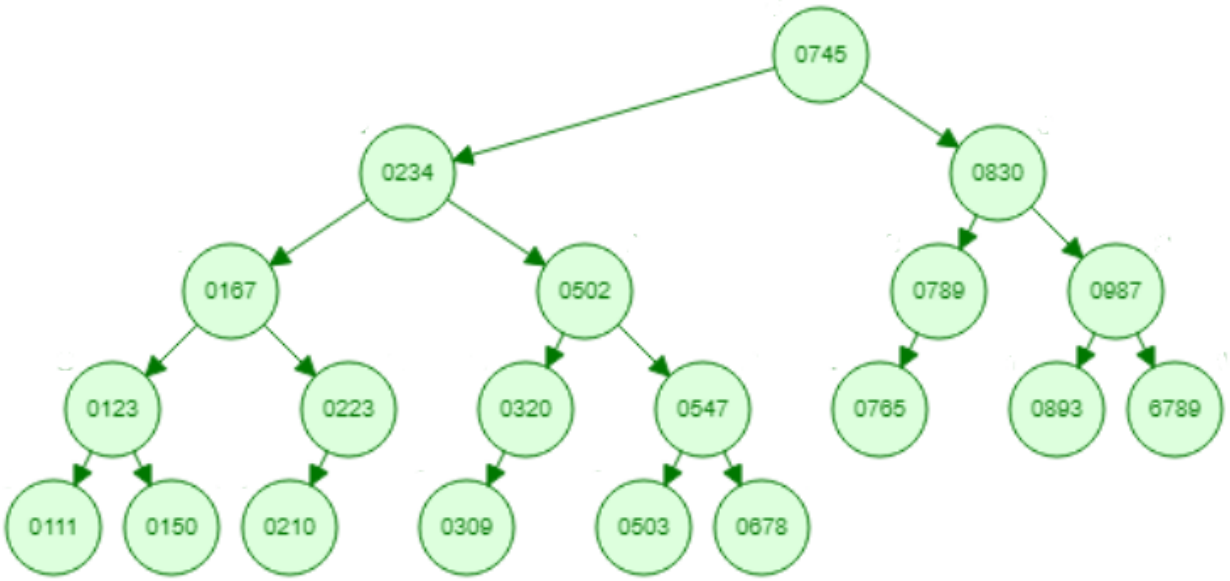
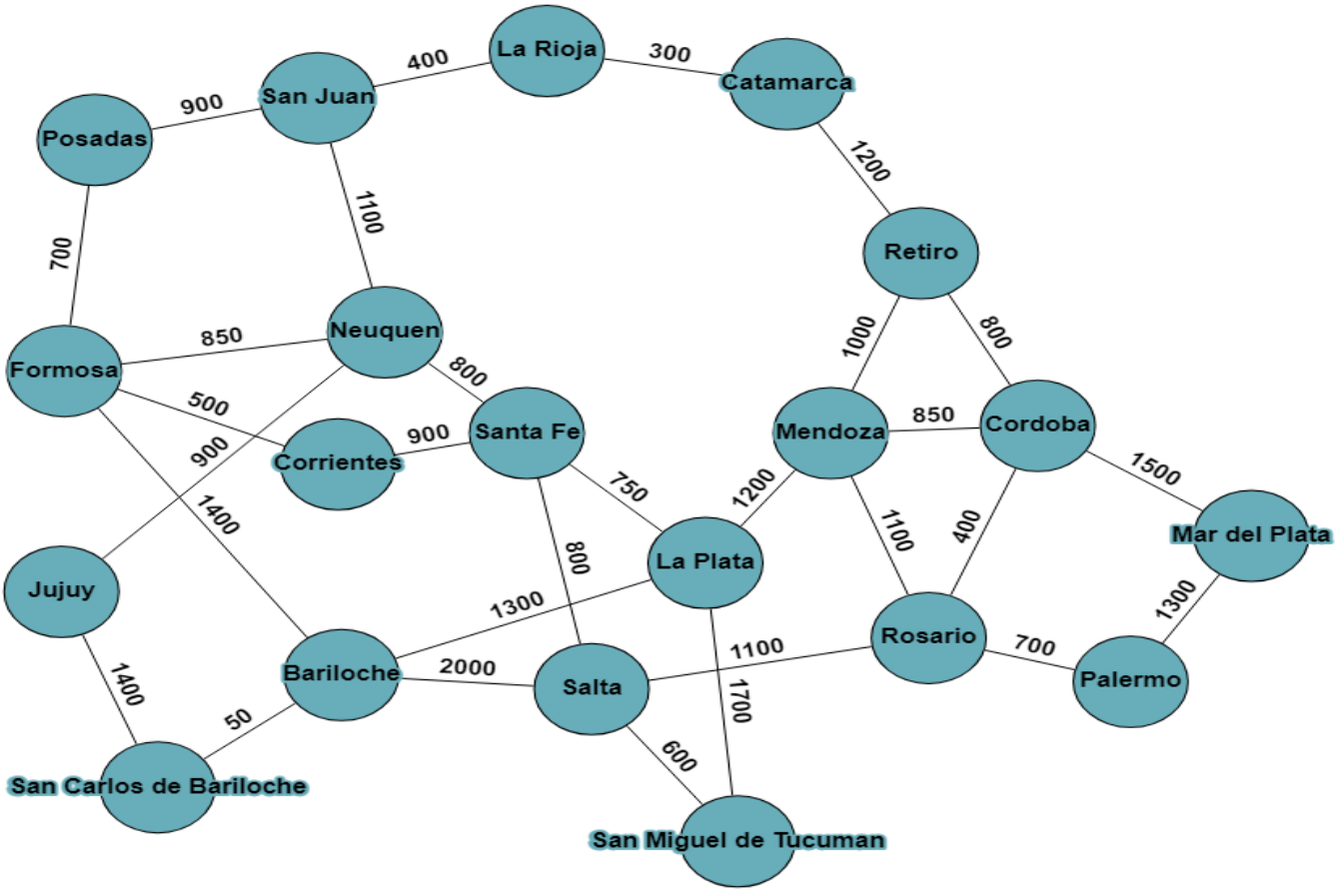
Estaciones
ESTACION Retiro CARGADO. ESTACION Cordoba CARGADO. ESTACION Mar del Plata CARGADO. ESTACION Palermo CARGADO. ESTACION Rosario CARGADO. ESTACION Mendoza CARGADO. ESTACION La Plata CARGADO. ESTACION San Miguel de Tucuman CARGADO. ESTACION Salta CARGADO. ESTACION Bariloche CARGADO. ESTACION San Carlos de Bariloche CARGADO. ESTACION Jujuy CARGADO. ESTACION Neuquen CARGADO. ESTACION Santa Fe CARGADO. ESTACION Corrientes CARGADO. ESTACION Formosa CARGADO. ESTACION Posadas CARGADO. ESTACION San Juan CARGADO. ESTACION La Rioja CARGADO. ESTACION Catamarca CARGADO.


Trenes
TREN 234 CARGADO A LINEA Mitre TREN 6789 CARGADO A LINEA Mar del Plata TREN 893 CARGADO SIN LINEA TREN 745 CARGADO A LINEA Salta Express TREN 123 CARGADO A LINEA Norte TREN 547 CARGADO SIN LINEA TREN 987 CARGADO A LINEA Patagonia TREN 678 CARGADO A LINEA Andina TREN 223 CARGADO A LINEA Mitre TREN 167 CARGADO SIN LINEA TREN 789 CARGADO A LINEA Mesopotamia TREN 320 CARGADO A LINEA Patagonia TREN 502 CARGADO A LINEA Norte TREN 830 CARGADO SIN LINEA TREN 111 CARGADO A LINEA Andina TREN 765 CARGADO A LINEA Mar del Plata TREN 210 CARGADO A LINEA Mitre TREN 309 CARGADO SIN LINEA TREN 503 CARGADO A LINEA Salta Express TREN 150 CARGADO A LINEA Norte


RIELES

RIEL CARGADO: (Retiro<--800 KM-->Cordoba).
RIEL CARGADO: (Cordoba<--1500 KM-->Mar del Plata).
RIEL CARGADO: (Mar del Plata<--1300 KM-->Palermo).
RIEL CARGADO: (Palermo<--700 KM-->Rosario).
RIEL CARGADO: (Rosario<--1100 KM-->Mendoza).
RIEL CARGADO: (Mendoza<--1200 KM-->La Plata).
RIEL CARGADO: (La Plata<--1700 KM-->San Miguel de Tucuman).
RIEL CARGADO: (San Miguel de Tucuman<--600 KM-->Salta).
RIEL CARGADO: (Salta<--2000 KM-->Bariloche).
RIEL CARGADO: (Bariloche<--50 KM-->San Carlos de Bariloche).
RIEL CARGADO: (San Carlos de Bariloche<--1400 KM-->Jujuy).
RIEL CARGADO: (Jujuy<--900 KM-->Neuquen).
RIEL CARGADO: (Neuquen<--800 KM-->Santa Fe).
RIEL CARGADO: (Santa Fe<--900 KM-->Corrientes).
RIEL CARGADO: (Corrientes<--500 KM-->Formosa).
RIEL CARGADO: (Formosa<--700 KM-->Posadas).
RIEL CARGADO: (Posadas<--900 KM-->San Juan).
RIEL CARGADO: (San Juan<--400 KM-->La Rioja).
RIEL CARGADO: (La Rioja<--300 KM-->Catamarca).
RIEL CARGADO: (Catamarca<--1200 KM-->Retiro).
RIEL CARGADO: (Retiro<--1000 KM-->Mendoza).
RIEL CARGADO: (Mendoza<--850 KM-->Cordoba).
RIEL CARGADO: (Cordoba<--400 KM-->Rosario).
RIEL CARGADO: (Rosario<--1100 KM-->Salta).
RIEL CARGADO: (Salta<--800 KM-->Santa Fe).
RIEL CARGADO: (Santa Fe<--750 KM-->La Plata).
RIEL CARGADO: (La Plata<--1300 KM-->Bariloche).
RIEL CARGADO: (Bariloche<--1400 KM-->Formosa).
RIEL CARGADO: (Formosa<--850 KM-->Neuquen).
RIEL CARGADO: (Neuquen<--1100 KM-->San Juan).



Correcciones

Grafo etiquetado

Antes | Despues

```java
public boolean eliminarVertice(Object eliminar) {
    boolean exito = false;
    boolean caso = false;
    NodoVert anterior = null;
    NodoVert actual = this.inicio;

    while(actual != null && !exito) {
        if (actual.getElem().equals(eliminar)) {
            if (anterior == null) {
                caso = true;
            }

            exito = true;
        } else {
            anterior = actual;
            actual = actual.getSigVertice();
        }
    }

    if (exito) {
        this.eliminarArcosConVertice(eliminar);
        if (caso) {
            this.inicio = actual.getSigVertice();
        } else {
            anterior.setSigVertice(actual.getSigVertice());
        }
    }

    return exito;
}
```

```java
public boolean eliminarVertice(Object eliminar) {
    boolean exito = false, caso = false;
    NodoVert anterior = null;
    NodoVert actual = this.inicio;
    NodoVert nodoEliminar=null;
    while (actual != null && !exito) {
        if (actual.getElem().equals(eliminar)) {
            if (anterior == null) {
                // El vértice a eliminar es el nodo inicio
                caso = true;
            }
            exito = true;
            //encontre el nodo que quiero eliminar asi que guardo
            //la Ref
            nodoEliminar=actual;
        } else {
            anterior = actual;
            actual = actual.getSigVertice();
        }
    }
    if (exito) {
        //'nodoEliminar' no va a ser nunca null porque si
        //exito es true es porque lo encontro
        eliminarArcosConVertice(nodoEliminar);
        if (caso) {
            this.inicio = actual.getSigVertice();
        } else {
            anterior.setSigVertice(actual.getSigVertice());
        }
    }
    return exito;
}
```

*Se almacena la referencia del nodo a eliminar para luego acceder a sus adyacentes mediante el método 'eliminarArcosConVertice'.

```java
private void eliminarArcosConVertice(NodoVert eliminar) {
    NodoAdy aux = eliminar.getPrimerAdy();
    while (aux != null) {
            eliminarArcosConVerticeAux(aux.getVertice(), eliminar.getElem());
        aux = aux.getSigAdy();
    }
}
```

(codigo corregido)
Se utiliza el método 'eliminarArcosConVerticeAux(...)' que recibe el elemento del NodoVert a eliminar y la variable 'aux', que representa el adyacente. Se emplea el método 'getVertice()' para acceder al elemento y obtener la lista de adyacencia asociada a este nodo. Posteriormente, se eliminan los enlaces que tienen con el NodoVert a eliminar.

```java
private boolean eliminarArcosConVerticeAux(NodoVert nodo, Object eliminar) {

    boolean exito = false;
    NodoAdy actual = nodo.getPrimerAdy();
    NodoAdy anterior = null;

    while (actual != null && !exito) {

        if (actual.getVertice().getElem().equals(eliminar)) {
            // el vertice es el encontrado

            if (anterior == null) {
                // caso primer Ady
                nodo.setPrimerAdy(actual.getSigAdy());
            } else {
                anterior.setSigAdy(actual.getSigAdy());
            }
            exito = true;
        } else {
            anterior = actual;
            actual = actual.getSigAdy();
        }
    }
    return exito;
}
```

Diccionario
Corrección respecto al método eliminar(..): Se modifico todo el metodo respecto al codigo anterior:

```java
private boolean eliminarAux(NodoAVLDicc n, Comparable buscar, NodoAVLDicc padre) {
    boolean retornar = false;
    if (n != null) {
        if (n.getClave().compareTo(buscar) == 0) {
            // encontrado
            int caso = casoHijo(n);
            switch (caso) {
                case 1:
                    eliminarCaso1(n, padre);
                    break;
                case 2:
                    eliminarCaso2(n, padre);
                    break;
                case 3:
                    eliminarCaso3(n.getIzquierdo(), n);
                    break;
            }
            retornar = true;
            n.recalcularAltura();
            int bal = this.calcularBalance(n);
            if (bal == -2 || bal == 2) {
                this.realizarRotacion(n, bal == 2, padre);
            }
        } else {
            if (n.getClave().compareTo(buscar) < 0) {
                // derecha
                retornar = eliminarAux(n.getDerecho(), buscar, n);
            } else {
                retornar = eliminarAux(n.getIzquierdo(), buscar, n);
            }
            if (retornar) {
                // solo voy a haber o recalcular la altura solamente si hubo una modificacion en
                // el arbol
                n.recalcularAltura();
                int bal = this.calcularBalance(n);
                if (bal == -2 || bal == 2) {
                    this.realizarRotacion(n, bal == 2, padre);
                }
            }
        }
    }
    return retornar;
}
```

Cree un modulo para cada caso:

- Cuando el nodo a eliminar no tiene hijos.
- Cuando el nodo a eliminar tiene un hijo
- Cuando el nodo a eliminar tiene los dos hijos.

El caso donde fallaba era el caso 3, pero al analizar mas en el codigo anterior hubo errores al momentos de recalcular altura en todos los caso (en el codigo actual fue solucionado).
(Como se ve, hay 2 momentos donde se recalcula la altura pero esto es necesario por esta razon:
Al encontrar el nodo a eliminar, es necesario recalcular la altura del nodo porque la estructura del árbol puede haber cambiado. El nodo eliminado puede haber afectado la altura de su padre o de otros nodos en el camino hacia la raíz debido a rotaciones o eliminaciones posteriores.
y Luego durante el paso recursivo, el parámetro n representa un nodo diferente en cada nivel del árbol. Cuando el proceso recursivo regresa de los subárboles izquierdo o derecho, el nodo n puede haber

cambiado debido a las rotaciones o eliminaciones realizadas.)
CASO 1:

```
private void eliminarCaso1(NodoAVLDicc n, NodoAVLDicc padre) {
    // caso donde no tiene hijos
    if (padre == null) {
        // caso raiz
        this.raiz = null;
    } else {
        // caso no raiz
        if (padre.getClave().compareTo(n.getClave()) < 0) {
            padre.setDerecho(derecho:null);
        } else {
            padre.setIzquierdo(izquierdo:null);
        }

    }
}
```

```java
private void eliminarCaso2(NodoAVLDicc n, NodoAVLDicc padre) {
    // caso donde al menos tiene un hijo (n)

    if (padre == null) {
        if (n.getDerecho() != null) {
            this.raiz = n.getDerecho();
        } else {
            this.raiz = n.getIzquierdo();
        }

    } else {
        int num = padre.getClave().compareTo(n.getClave());
        if (num < 0) {
            // veo el lado donde esta el nodo a eliminar
            if (n.getDerecho() != null) {
                padre.setDerecho(n.getDerecho());

            } else {
                padre.setDerecho(n.getIzquierdo());

            }

        } else {
            if (n.getDerecho() != null) {
                padre.setIzquierdo(n.getDerecho());
            } else {
                padre.setIzquierdo(n.getIzquierdo());
            }

        }

    }

}
```

caso 3:

```java
private void eliminarCaso3(NodoAVLDicc cand, NodoAVLDicc eliminar) {
   //cand 765 eliminar  987
    NodoAVLDicc aux;
    if (cand.getDerecho() == null) {
        // no tiene HD
        eliminar.setClave(cand.getClave());
        eliminar.setDato(cand.getDato());

        if (cand.getIzquierdo() != null) {
            // tiene HI
            eliminar.setIzquierdo(cand.getIzquierdo());
        } else {
            eliminar.setIzquierdo(izquierdo:null);
        }
    } else {
        //cand 765, elim 978
        aux = candidato(cand, eliminar);
        eliminar.setClave(aux.getClave());
        eliminar.setDato(aux.getDato());

    }

}
```

*Primero que nada el método entra si o si con el HI del que se quiere eliminar.

Hay 2 casos que se puede ver:
Cuando el candidato en primera instancia no tiene HD por lo tanto este pasaria a ser candidato entonces intercambiamos dato y clave con el nodo a eliminar. Luego el HI del nodo que queriamos eliminar seria el HI del candidato si es que lo tuviera.
_ Cuando SI tiene HD, entonces deberemos buscar el candidato mas a la derecha y para eso abra este método:

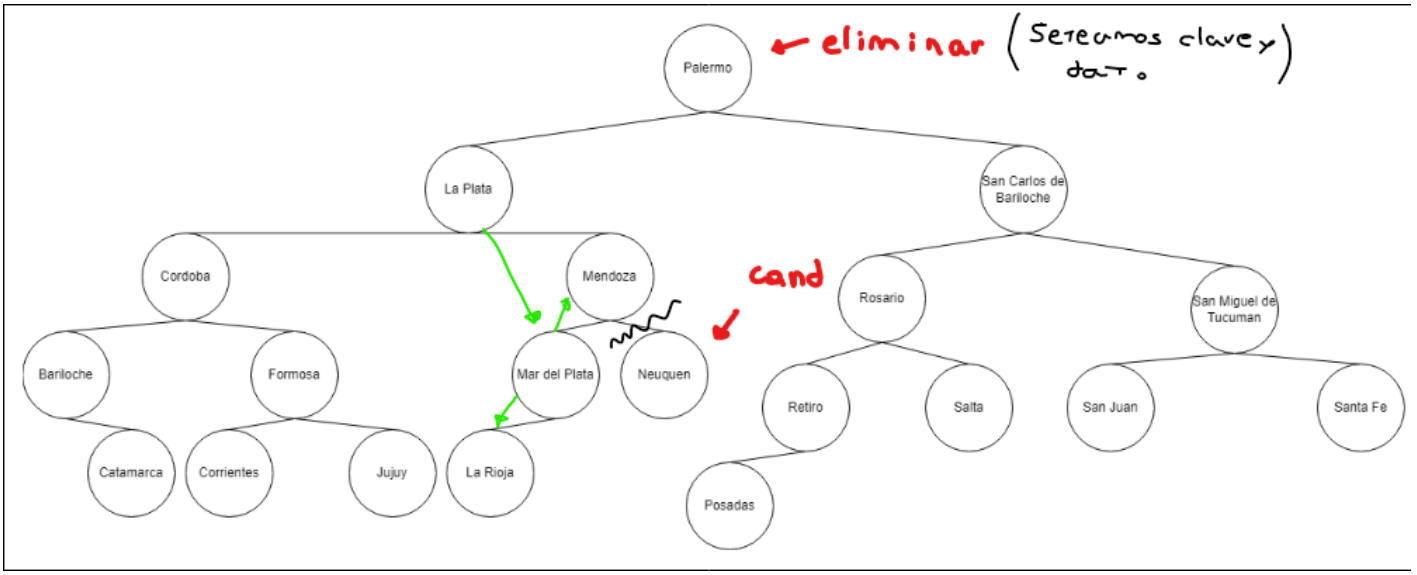
```java
private NodoAVLDicc candidato(NodoAVLDicc n, NodoAVLDicc padreCandidato)
    // caso donde si o si tiene HD
    NodoAVLDicc retornar;
    if (n.getDerecho() == null) {
        // el nodo no tiene a donde ya ir
        retornar = n;
        if (n.getIzquierdo() != null) {
            // si el hijo candidato tiene hijo izquierdo
            padreCandidato.setDerecho(n.getIzquierdo());
        } else {
            padreCandidato.setDerecho(derecho:null);
        }

    } else {
        retornar = candidato(n.getDerecho(), n);
    }
    n.recalcularAltura();
        int bal = calcularBalance(n);
        if (bal == -2 || bal == 2) {
            realizarRotacion(n, bal == 2, padreCandidato);
        }
    return retornar;

}
```

aca hay un caso cuando el candidato mas a la derecha tiene HI entonces el padre candidato tendría otro HD que sería el HI del candidato.

TEST

ELIMINACION DE PALERMO:

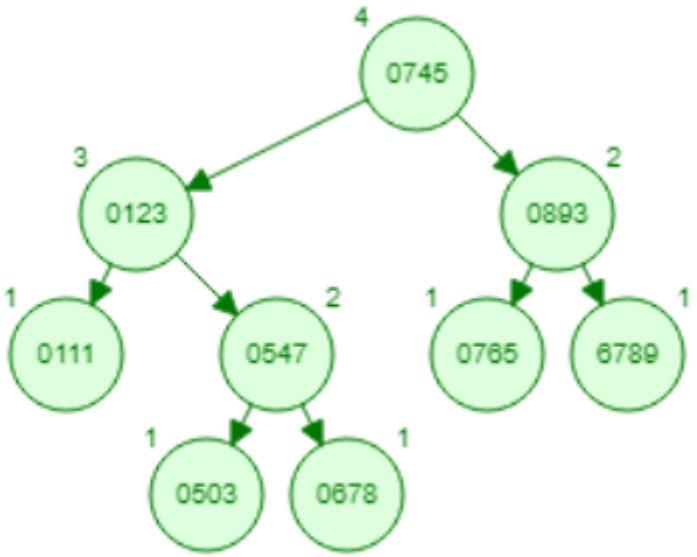
```
Neuquen: ALTURA(4)
        HI=La Plata     HD=San Carlos de Bariloche
La Plata: ALTURA(3)
        HI=Cordoba      HD=Mar del Plata
Cordoba: ALTURA(2)
        HI=Bariloche    HD=Formosa
Bariloche: ALTURA(1)
        HI= -   HD=Catamarca
Catamarca: ALTURA(0)
        HI= -   HD= -
Formosa: ALTURA(1)
        HI=Corrientes   HD=Jujuy
Corrientes: ALTURA(0)
        HI= -   HD= -
Jujuy: ALTURA(0)
        HI= -   HD= -
Mar del Plata: ALTURA(1)
        HI=La Rioja     HD=Mendoza
La Rioja: ALTURA(0)
        HI= -   HD= -
Mendoza: ALTURA(0)
        HI= -   HD= -
San Carlos de Bariloche: ALTURA(3)
        HI=Rosario      HD=San Miguel de Tucuman
Rosario: ALTURA(2)
        HI=Retiro       HD=Salta
Retiro: ALTURA(1)
        HI=Posadas      HD= -
Posadas: ALTURA(0)
        HI= -   HD= -
Salta: ALTURA(0)
        HI= -   HD= -
San Miguel de Tucuman: ALTURA(1)
        HI=San Juan     HD=Santa Fe
San Juan: ALTURA(0)
        HI= -   HD= -
Santa Fe: ALTURA(0)
        HI= -   HD= -
```



TEST

test con mi DICCIONARIO(Altura de un nodo cantidad de arcos)

```
Trenes:
745: ALTURA(3)
        HI=123  HD=893
123: ALTURA(2)
        HI=111  HD=547
111: ALTURA(0)
        HI= -   HD= -
547: ALTURA(1)
        HI=503  HD=678
503: ALTURA(0)
        HI= -   HD= -
678: ALTURA(0)
        HI= -   HD= -
893: ALTURA(1)
        HI=765  HD=6789
765: ALTURA(0)
        HI= -   HD= -
6789: ALTURA(0)
        HI= -   HD= -
```

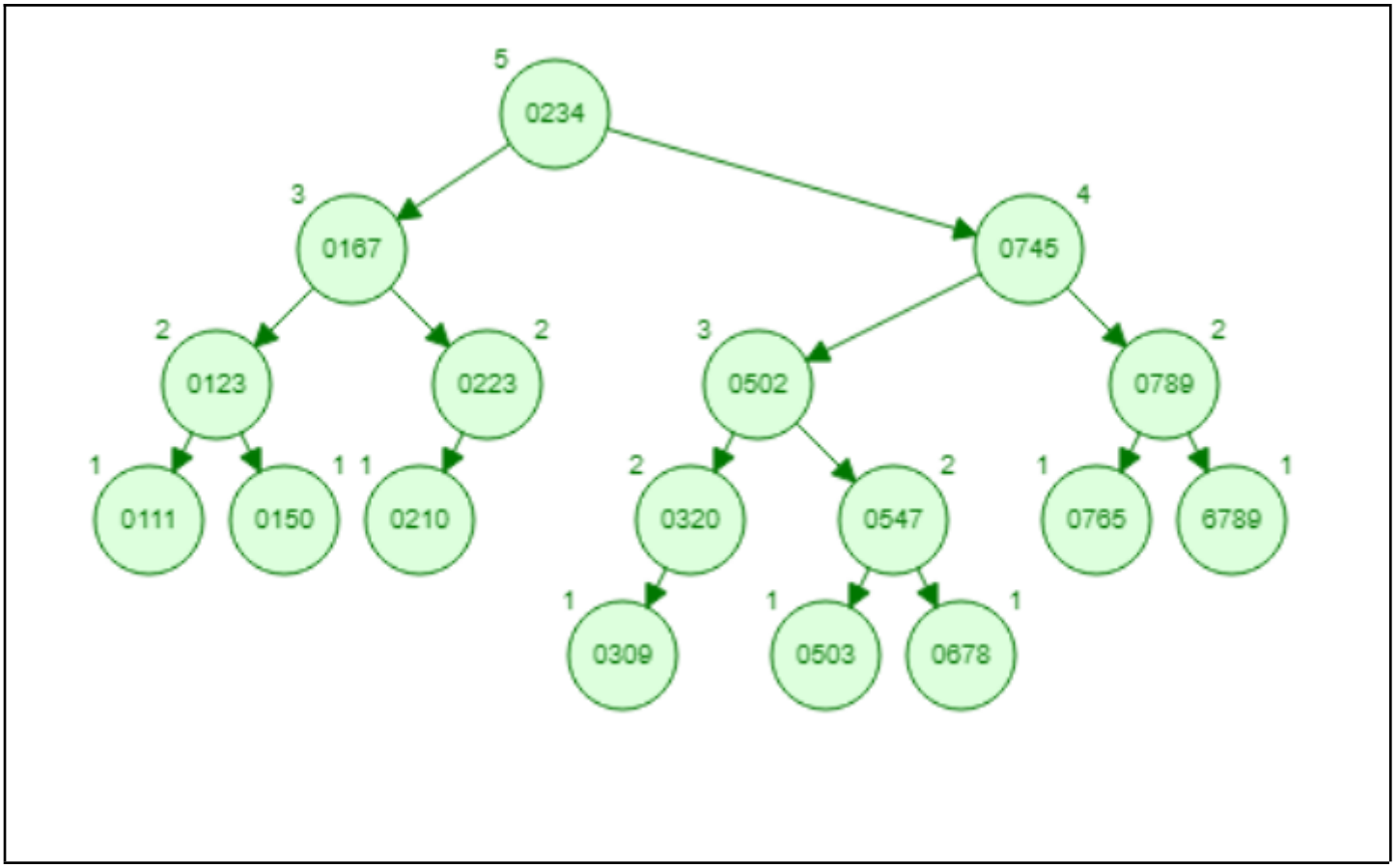


Se testeo en una pagina de AVL(https://www.cs.usfca.edu/~galles/visualization/AVLtree.html)
se elimino trenes de la siguiente manera:
(EN LA PAGINA LA ALTURA LO PONE RESPECTO A LA CANTIDAD DE NODOS mas larga)
Se elimino el Tren 150
Se elimino el Tren 830
Se elimino el Tren 234
Se elimino el Tren 789
Se elimino el Tren 210
Se elimino el Tren 987
Se elimino el Tren 223
Se elimino el Tren 502
Se elimino el Tren 309
Se elimino el Tren 167
Se elimino el Tren 320

se elimino:

Se elimino el Tren 893
Se elimino el Tren 987
Se elimino el Tren 830

ese ejemplo genera mas rotaciones.


5
0234
3
0167
4
0745
2
0123
2
0223
3
0502
2
0789
1
0111
1
0150
1
0210
2
0320
2
0547
1
0765
1
6789
1
0309
1
0503
1
0678

```
    *Trenes:
234: ALTURA(4)
        HI=167  HD=745
167: ALTURA(2)
        HI=123  HD=223
123: ALTURA(1)
        HI=111  HD=150
111: ALTURA(0)
        HI= -   HD= -
150: ALTURA(0)
        HI= -   HD= -
223: ALTURA(1)
        HI=210  HD= -
210: ALTURA(0)
        HI= -   HD= -
745: ALTURA(3)
        HI=502  HD=789
502: ALTURA(2)
        HI=320  HD=547
320: ALTURA(1)
        HI=309  HD= -
309: ALTURA(0)
        HI= -   HD= -
547: ALTURA(1)
        HI=503  HD=678
503: ALTURA(0)
        HI= -   HD= -
678: ALTURA(0)
        HI= -   HD= -
789: ALTURA(1)
        HI=765  HD=6789
765: ALTURA(0)
        HI= -   HD= -
6789: ALTURA(0)
        HI= -   HD= -
```