Documento de diseño taller 3

Nelson Mosquera
Andrés Torres

En el desarrollo del taller se trabajaron X funciones. Las funciones y su información de entrada y salida, si las hay, son:

- uploadFile: Almacena la información del archivo de texto en una lista de cadenas de caracteres linea por linea.
  - información de entrada: nombre del archivo a cargar
  - información de salida: N/A
- fillTree: Almacena la información de la lista en un árbol AVL siguiendo las operaciones especificadas al comienzo de cada linea(adición y eliminación)
  - información de entrada: N/A
  - información de salida: N/A
- fillSet: Almacena la información de la lista en un árbol RN(utilizando std::set<T> siguiendo las operaciones especificadas al comienzo de cada linea(adición y eliminación)
  - información de entrada: N/A
  - información de salida: N/A
- fillVec: Almacena la información de la lista en una estructura montículo siguiendo las operaciones especificadas al comienzo de cada linea(adición y eliminación)
  - información de entrada: N/A
  - información de salida: N/A
- fillInOrderList: Si el árbol no esta vacio llama a la función de inOrden del árbol
  - Información de entrada: N/A
  - información de salida: N/A
- isEqual: Dice si el árbol, el vec y el set contienen la misma información
  - Información de entrada: N/A
  - Información de salida: true si contienen la misma información, false en caso contrario

Los TADs creados para la solución del problema son los siguientes:
TAD NodoAvl
Conjunto mínimo de datos:
    dato, plantilla, dato almacenado en el nodo
    Izq, plantilla, hijo menor
    Der, plantilla, hijo mayor
Comportamientos(operaciones)
    NodoAVL(dato)
    NodoAVL()
    obtenerDato()
    fijarDato(val)
    obtenerHijoIzq()
    obtenerHijoDer()
    fijarHijoIzq(izq)
    fijarHijoDer(der)
TAD ArbolAvl
Conjunto mínimo de datos:
    raíz, apuntador a nodo, nodo raíz del árbol

Comportamientos(operaciones)

ArbolAVL()
getRaiz()
esVacio()
datoRaiz()
altura()
tamano()
insertar(val)
altura(inicio)
tamano(inicio)
insertar(val, padre)
eliminarf(val, padre)
eliminar(val)
buscar(val)
preOrden(inicio)
inOrden(inicio)
posOrden(inicio)
nivelOrden(inicio)
preOrden()
inOrden()
posOrden()
nivelOrden()
diff(temp)
rotacionDerecha(parent)
rotacionIzquierda(parent)
rotacionDI(parent)
rotacionID(parent)
balance(temp)
inOrdenL(nicio, l)
inOrdenL(l)

TAD Sistema

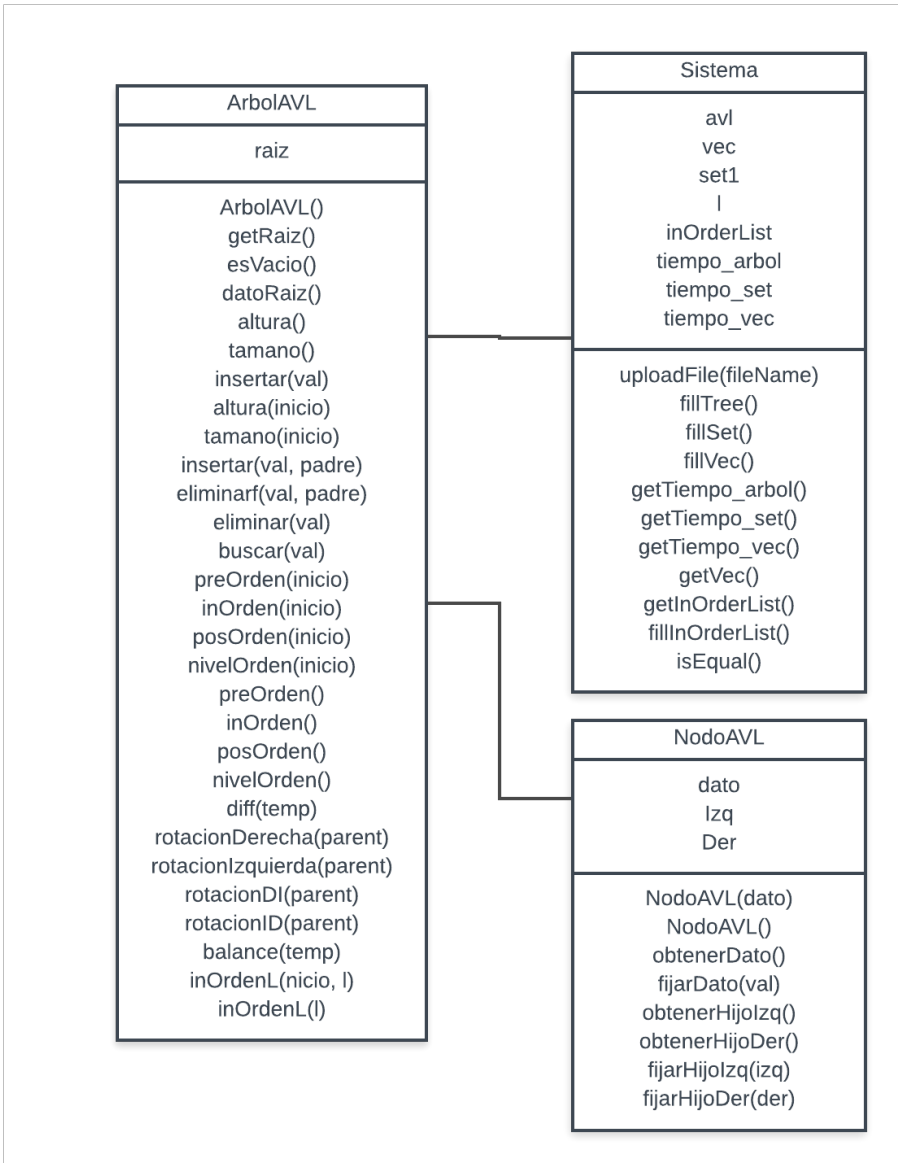
Conjunto mínimo de datos

avl, árbol AVL de enteros, árbol en el que se almacenan los datos
vec, vector de enteros, vector donde se almacenan los datos
set1, conjunto de enteros, conjunto donde se almacenan los datos
l, lista de cadenas de caracteres, lista donde se almacena el contenido del archivo
inOrderList, lista de enteros, lista que almacena el recorrido en inorden del arbol
cod, entero,
tiempo_arbol, numero real, tiempo que el programa se tarda en realizar el proceso usando el árbol
tiempo_set, numero real, tiempo que el programa se tarda en realizar el proceso usando conjunto
tiempo_vec, numero real, tiempo que el programa se tarda en realizar el proceso usando el vector

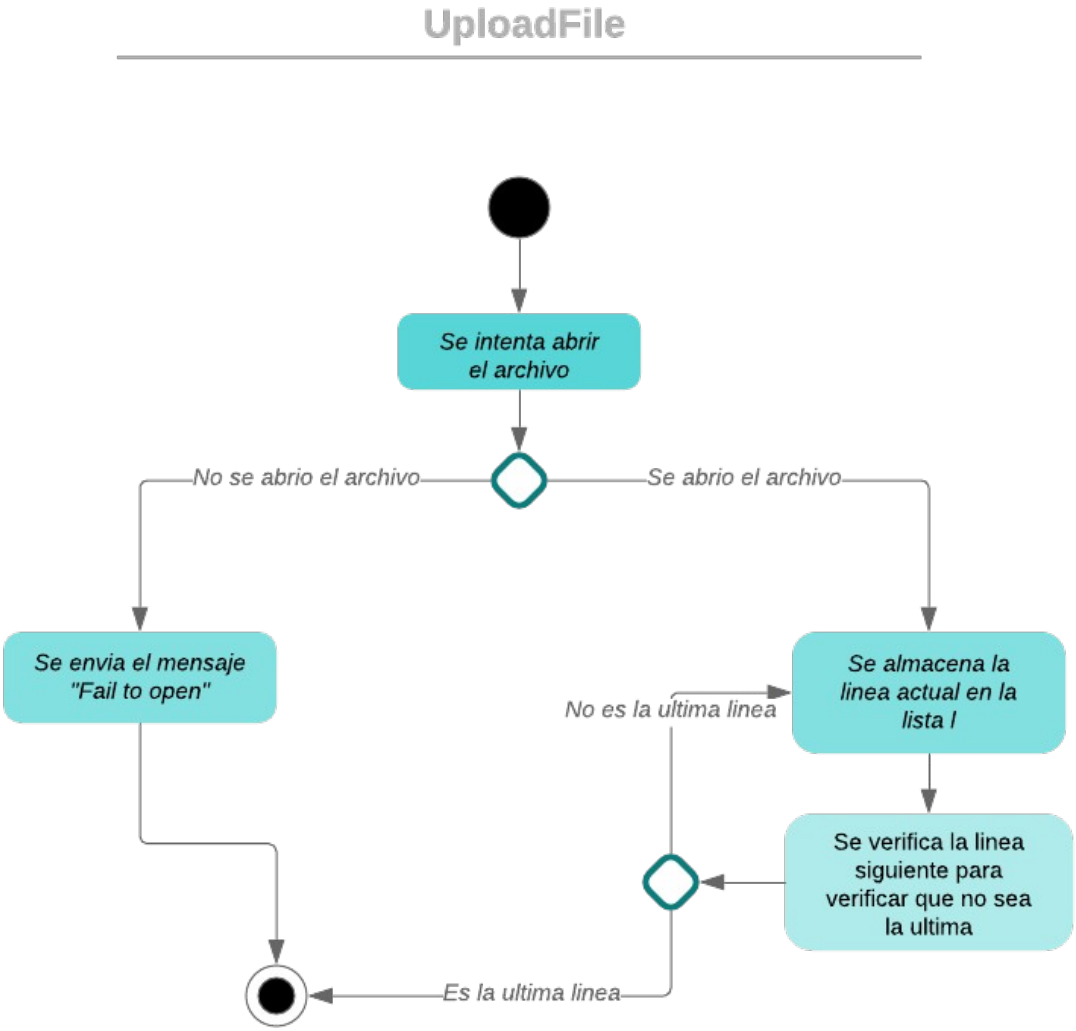
Comportamientos(operaciones)

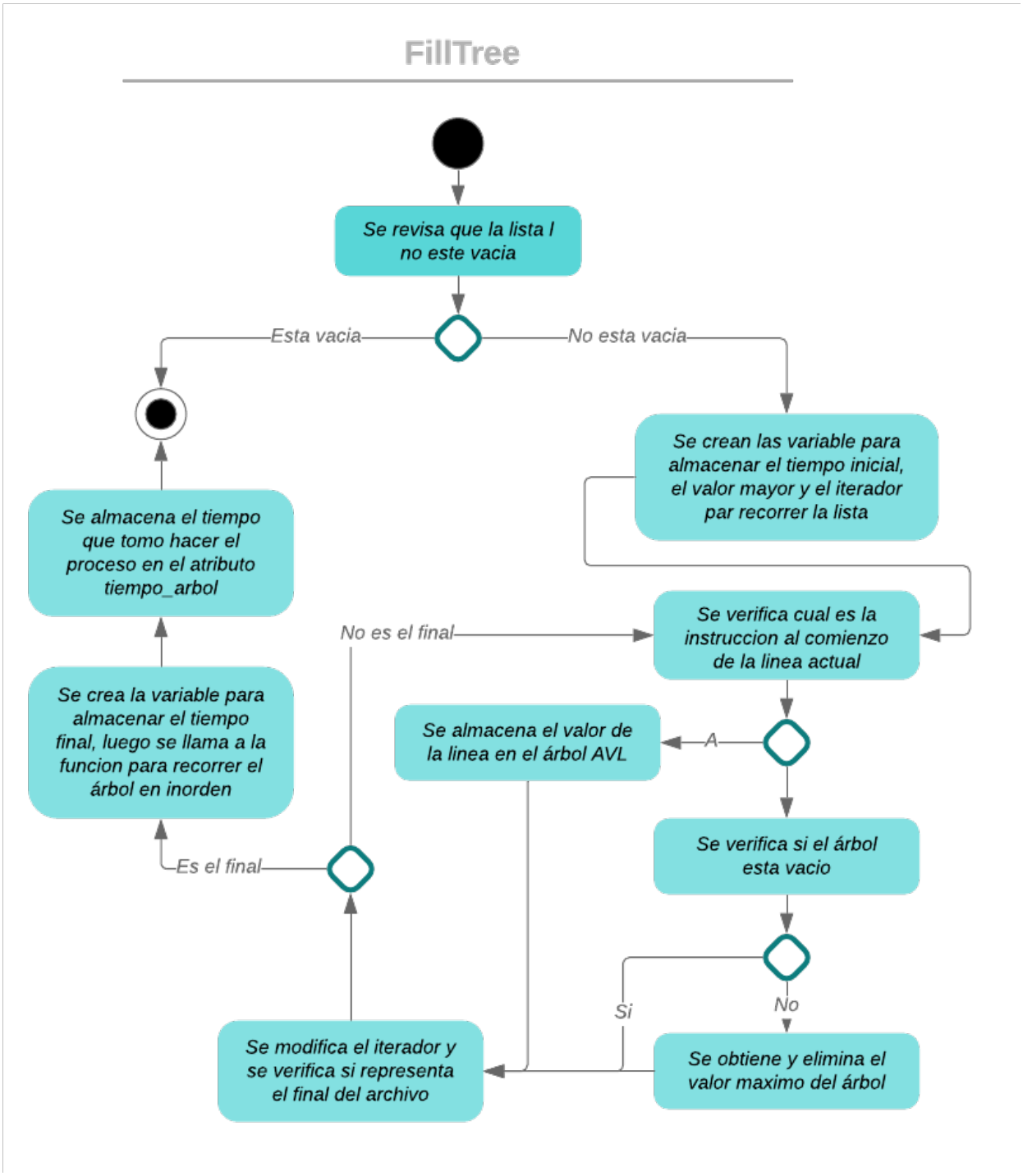
uploadFile(fileName)
fillTree(),
fillSet()
fillVec()
getTiempo_arbol()
getTiempo_set()
getTiempo_vec()
getVec()
getInOrderList()
fillInOrderList()
isEqual()

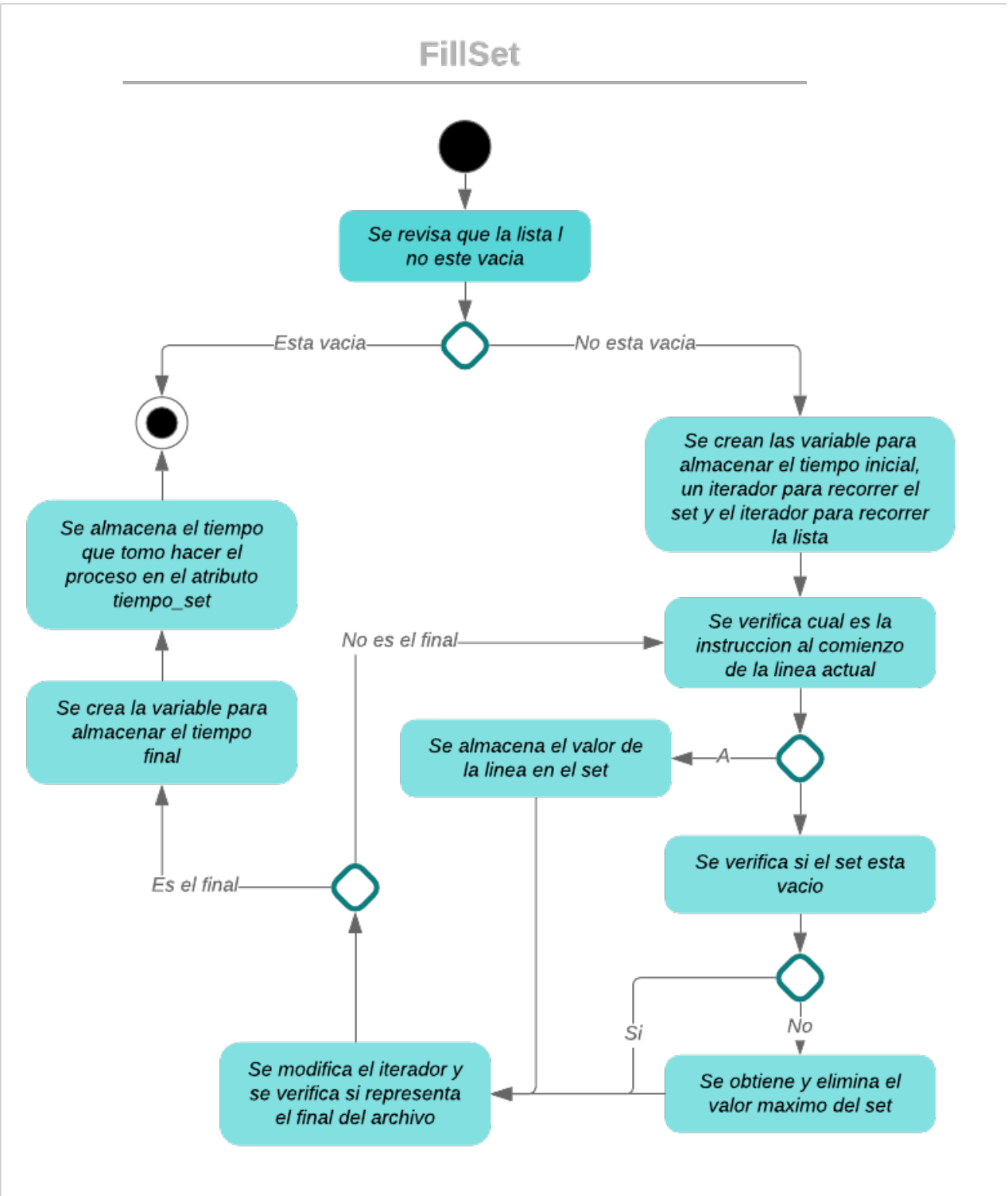
Tabla de relación de los TADs:

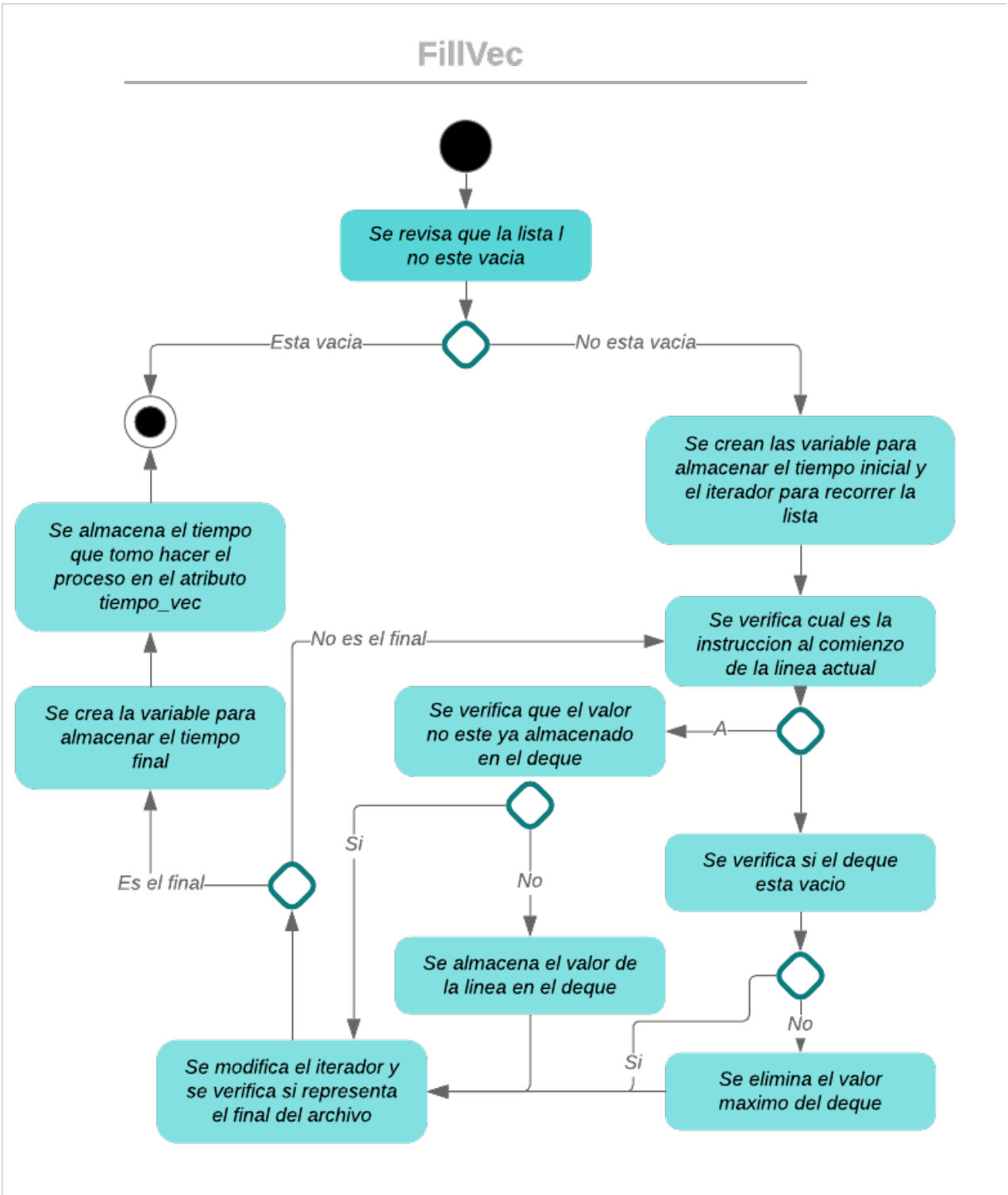


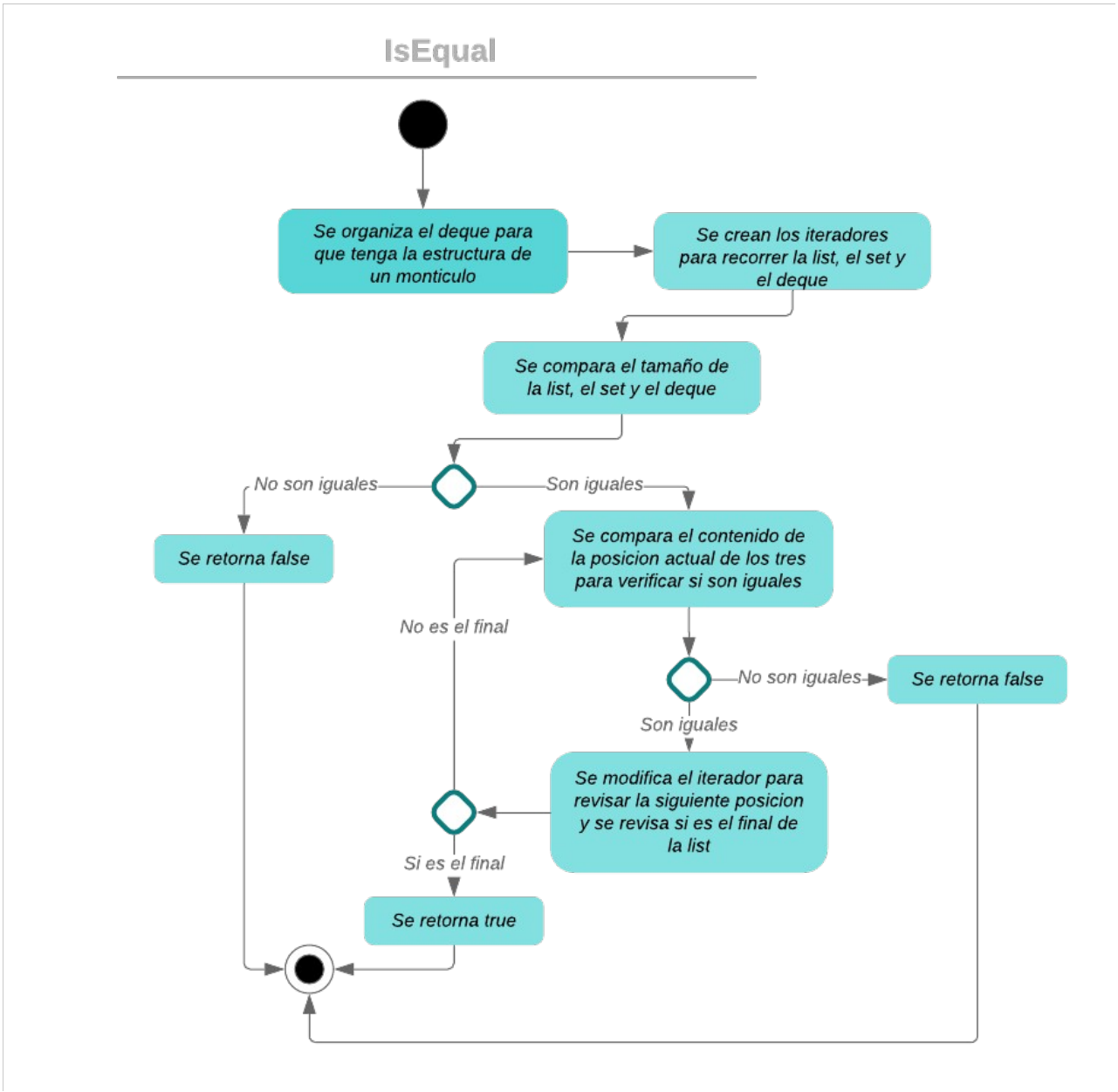
Diagramas de flujo funciones:




FillTree
Se revisa que la lista l no este vacia
Esta vacia
No esta vacia
Se crean las variable para almacenar el tiempo inicial, el valor mayor y el iterador par recorrer la lista
Se verifica cual es la instruccion al comienzo de la linea actual
No es el final
A
Se almacena el valor de la linea en el árbol AVL
Se verifica si el árbol esta vacio
Si
No
Se obtiene y elimina el valor maximo del árbol
Se modifica el iterador y se verifica si representa el final del archivo
Es el final
Se crea la variable para almacenar el tiempo final, luego se llama a la funcion para recorrer el árbol en inorden
Se almacena el tiempo que tomo hacer el proceso en el atributo tiempo_arbol


FillSet
Se revisa que la lista l no este vacia
Esta vacia
No esta vacia
Se crean las variable para almacenar el tiempo inicial, un iterador para recorrer el set y el iterador para recorrer la lista
Se verifica cual es la instruccion al comienzo de la linea actual
No es el final
A
Se almacena el valor de la linea en el set
Se verifica si el set esta vacio
Si
No
Se obtiene y elimina el valor maximo del set
Se modifica el iterador y se verifica si representa el final del archivo
Es el final
Se crea la variable para almacenar el tiempo final
Se almacena el tiempo que tomo hacer el proceso en el atributo tiempo_set


FillVec
Se revisa que la lista l no este vacia
Esta vacia
No esta vacia
Se crean las variable para almacenar el tiempo inicial y el iterador para recorrer la lista
Se verifica cual es la instruccion al comienzo de la linea actual
A
Se verifica que el valor no este ya almacenado en el deque
Si
No
Se verifica si el deque esta vacio
No
Si
Se elimina el valor maximo del deque
Se almacena el valor de la linea en el deque
Se modifica el iterador y se verifica si representa el final del archivo
No es el final
Es el final
Se crea la variable para almacenar el tiempo final
Se almacena el tiempo que tomo hacer el proceso en el atributo tiempo_vec



Comparación de los tres métodos:

Con la información obtenida al momento de llevar a cabo las pruebas con los 3 archivos dados se puede observar como en las dos primeras pruebas, las cuales usan archivos considerablemente más pequeños, las 3 metodologías de almacenamiento son todas igual de eficientes en términos de su tiempo de ejecución. Por otro lado al llevar a cabo la prueba con el tercer archivo se evidencia claramente como el método más adecuado para manejar archivos más pesados el que aplica el std::set<T>(árbol RN), ya que en el ejemplo este fue casi instantáneo, mientras que el montículo(aplicación de las estructuras de la STL) se demoro alrededor de un segundo y el árbol AVL se demoro considerablemente mucho más que el anterior, una posible explicación para que el árbol presentara una velocidad tan reducida en

comparación es que como las otras son librerías estandarizadas estas ya están completamente optimizadas.