UNLP. Facultad de Informática.

Algoritmos y Estructuras de Datos **Trabajo Práctico 5**
Cursada 2014

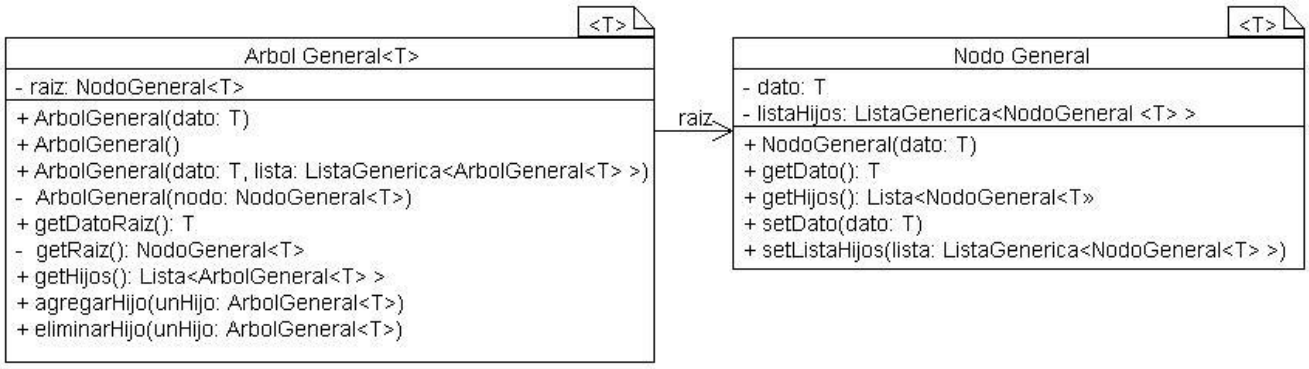
Árboles Generales

Objetivos

- Modelar un árbol n-ario
- Realizar recorridos e implementar aplicaciones varias sobre el árbol.

Ejercicio 1

Considere la siguiente especificación de la clase **ArbolGeneral** (con la representación de **Lista de Hijos**)



Nota: la clase **Lista** es la utilizada en la práctica 3 . Tenga en cuenta que el árbol general también puede implementarse con la representación de **Hijo más izquierdo – Hermano derecho**, sin embargo en la práctica sólo vamos a utilizar la representación **lista de hijos**.

El constructor **ArbolGeneral()** inicializa un árbol vacío, es decir, la raíz en null.

El constructor **ArbolGeneral(T dato)** inicializa un árbol que tiene como raíz un nodo general. Este nodo tiene el dato pasado como parámetro y una lista vacía.

El constructor **ArbolGeneral(T dato, Lista<ArbolGeneral<T>> hijos)** inicializa un árbol que tiene como raíz un nodo general. Este nodo tiene el dato pasado como parámetro y tiene como hijos una copia de la lista pasada como parámetro. Tenga presente que la Lista pasada como parámetro es una lista de árboles generales, mientras que la lista que se debe guardar en el nodo general es una lista de nodos generales. Por lo cuál, de la lista de árboles generales debe extraer la raíz (un Nodo General) y guardar solamente este objeto.

El constructor **ArbolGeneral(NodoGeneral<T> nodo)** inicializa un árbol donde el nodo pasado como parámetro es la raíz. Notar que este constructor NO es público.

El método **getRaiz():NodoGeneral<T>** retorna el nodo ubicado en la raíz del árbol. Notar que NO es un método público.

El método **getDatoRaiz():T** retorna el dato almacenado en la raíz del árbol (NodoGeneral).

El método **getHijos():Lista<ArbolGeneral<T>>**, retorna la lista de hijos de la raíz del árbol. Tenga presente que la lista almacenada en la raíz es una lista de nodos generales, mientras que debe devolver una lista de árboles generales. Para ello, por cada hijo, debe crear un árbol general que tenga como raíz el Nodo General hijo.

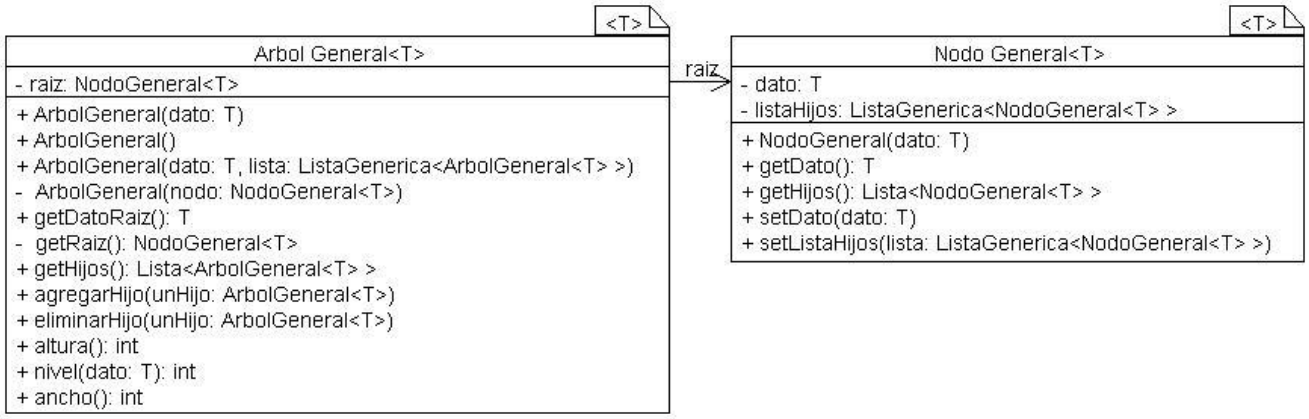
El método **agregarHijo(ArbolGeneral<T> unHijo)** agrega unHijo a la lista de hijos del árbol. Es decir, se le agrega a la lista de hijos de la raíz del objeto receptor del mensaje el Nodo General raíz de unHijo.

El método **eliminarHijo(ArbolGeneral<T> unHijo)** elimina unHijo del árbol. Con la misma salvedad indicada en el método anterior.

a) Analice la implementación en JAVA de las clases **ArbolGeneral** y **NodoGeneral** brindadas por la cátedra.

Ejercicio 2

Agregue a la clase **ArbolGeneral** los siguientes métodos:



a) **altura(): int** devuelve la altura del árbol, es decir, la longitud del camino más largo desde el nodo raíz hasta una hoja. Pista: el mensaje altura debe chequear si el árbol es una sola hoja o no. Si el árbol es una sola hoja, se devuelve 0. Si no, se utiliza el mensaje getHijos() para obtener la lista de hijos (recuerde que devuelve una lista de árboles hijos). Luego, debe iterar por cada uno de los hijos, guardando la máxima altura. A este valor se le debe sumar 1 y retornarlo.
b) **nivel(T dato): int** devuelve la profundidad o nivel del dato en el árbol. El nivel de un nodo es la longitud del único camino de la raíz al nodo. Pista: si el nodo raíz posee el mismo dato que pasado como parámetro, se retorna 0. En caso contrario, se debe buscar en cuales de los subárboles hijos se encuentra el dato (implemente el mensaje include (T dato) en la clase Arbol General) y se debe retornar 1 más el nivel que arroje enviar el mensaje nivel() al subárbol que incluye el dato.
c) **ancho(): int** la amplitud (ancho) de un árbol se define como la cantidad de nodos que se encuentran en el nivel que posee la mayor cantidad de nodos. Pista: realice un recorrido por niveles. Encole inicialmente la raíz del árbol y luego una marca null (o el número de nivel) para indicar el fin de nivel. Mientras la cola no se vacía, itere. En cada iteración extraiga el tope de la cola, y con la operación getHijos() encole los mismos. Cuando encuentra la marca de fin de nivel cuente si los elementos del nivel es mayor a la máxima cantidad que poseía.

Nota: tenga presente que no importaría qué representación interna utilizaría (lista de hijos o hijo más izquierdo – hermano derecho), estas operaciones deberían estar implementadas a partir de las operaciones especificadas en el ejercicio 1.
El fin de este ejercicio es implementar operaciones utilizando las operaciones básicas definidas anteriormente.

UNLP. Facultad de Informática.

Algoritmos y Estructuras de Datos **Trabajo Práctico 5**
Cursada 2014

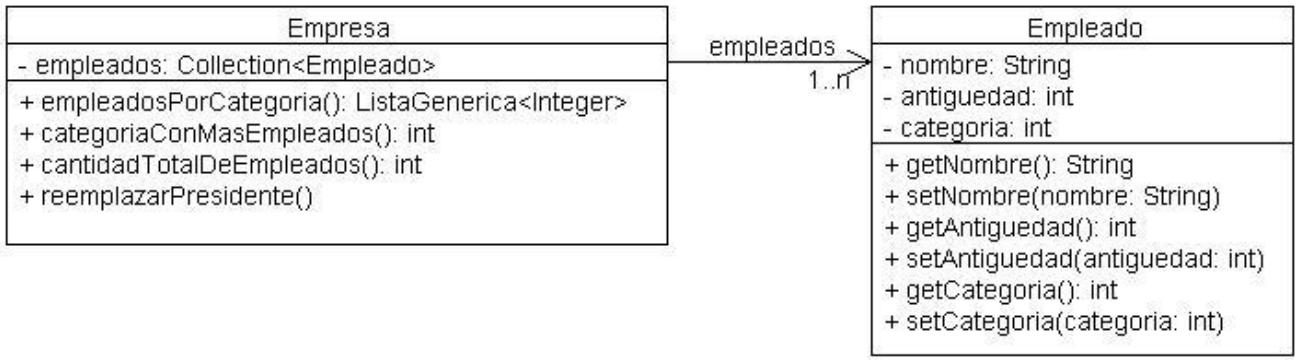
Los siguientes ejercicios son aplicaciones que "usan" el árbol general.

Ejercicio 3

Modelizar e implementar en JAVA la siguiente situación. En una empresa los empleados están categorizados con un número en el rango de 1 a 10, siendo el 1 el presidente, el 2 el vicepresidente, 3 los gerentes y así siguiendo. Los mismos también poseen una antigüedad. Esta información está dispuesta en un árbol general. Para este ejercicio realice métodos que al menos resuelvan los siguientes problemas:

a) devolver la cantidad de empleados por categoría.
b) determinar la categoría que cuenta con la mayor cantidad de empleados.
c) determinar la cantidad total de empleados.
d) sea la situación en donde el presidente deja su función, reemplazarlo por la persona más antigua de sus subordinados, quién a su vez es reemplazada de la misma forma.

Pista:



Debe modificar el árbol para que el nodo represente a cada empleado. Luego, los hijos del nodo representan la relación "es subordinado de"

a) Debe implementar un recorrido por niveles como lo hizo en 2.c. Debe totalizar para cada uno de los niveles.
b) A partir del punto anterior, se debe quedar con la mayor cantidad.
c) A partir de a, debe realizar la suma
d) Debe tomar los hijos de la raíz, y buscar el de mayor antigüedad de los hijos. Sin modificar la estructura, pase el mayor de los hijos a la raíz, y se envía el mensaje al hijo promovido. Cuando el hijo promovido no tenga hijos, se lo debe eliminar.

Ejercicio 4

Implemente una clase en Java llamada **Imagen** que permita crear imágenes en blanco y negro que utilice como representación interna una matriz.
Una forma de comprimir una imagen es transformarla a un árbol 4-ario. El algoritmo es el siguiente. Si toda la matriz tiene un mismo color, se debe definir un nodo con ese color. En caso contrario, se divide la matriz en cuatro partes, se define un nodo con 4 hijos, y cada hijo es la conversión de cada una de las partes de la matriz.
Realice un método de instancia llamado **imagenComprimida()** que devuelva la representación de la imagen en su árbol correspondiente.

Ejemplo:

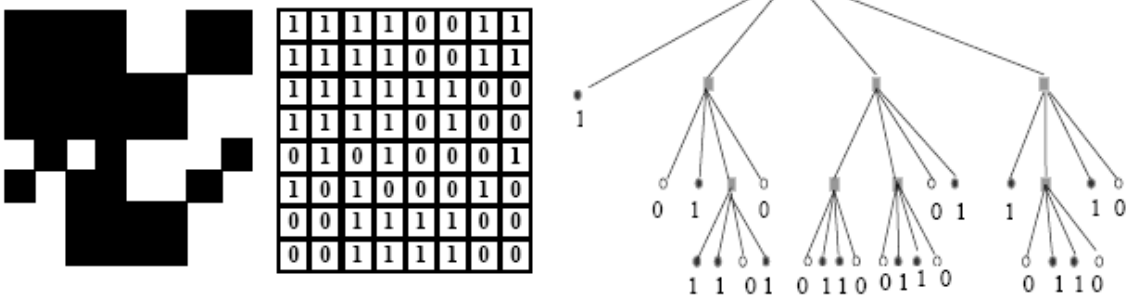


imagen matriz árbol 4-ario

Pista:

Imagen
- imagen: boolean[ ][ ]
+ Imagen(dim: int) - Imagen(original: Imagen, sector: int) + igualColor(): boolean + color(): boolean + dividirEnSubImagenes(): ListaGenerica<Imagen> + imagenComprimida(): ArbolGeneral<Boolean> - dimension(): int - set(fila: int, columna: int) - get(fila: int, columna: int): boolean

Teniendo en cuenta que el color blanco lo representamos con el valor falso y el negro con verdadero, implemente los métodos *igualColor*(), *color*() y *dividirEnSubimagenes*() en la clase Imagen. El método *dividirEnSubimagenes*() devuelve una Lista con 4 Imágenes. Para comprimir la imagen, debe verificar si la misma posee igual color, en ese caso debe crear y devolver un Arbol General con un único nodo. En cambio, si la imagen no posee el mismo color, debe *dividirEnSubImagenes* y enviar el mensaje *imagenComprimida*() a cada una.

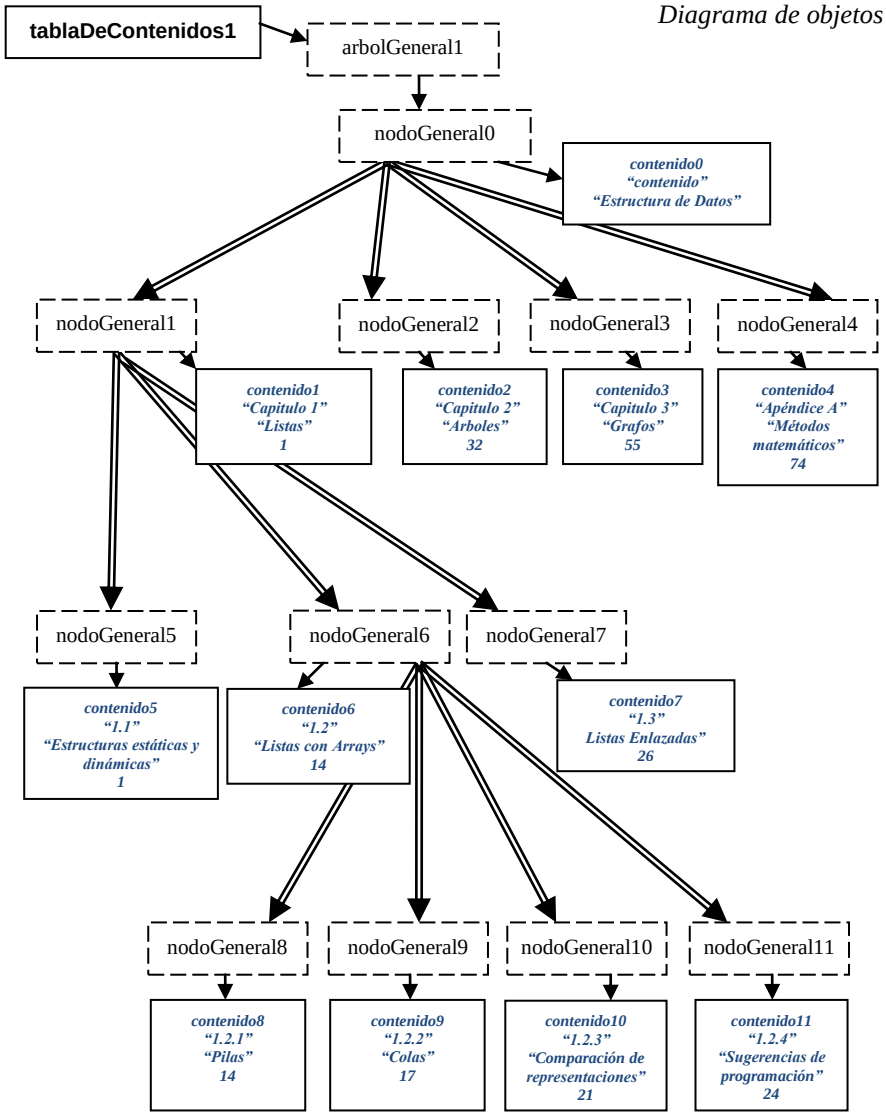
Ejercicio 5

Modelizar e implementar en JAVA una tabla de Contenidos de un libro. Un Libro está compuesto por capítulos y apéndices. Los mismos además del nombre tienen un tema y pagina inicial. A su vez cada uno de estos puede estar compuesto por secciones, que cuentan con la misma información: nombre, tema y página inicial; y cada sección podría contener más secciones. La tabla de contenidos comienza con el contenido destacado que es el nombre del libro.
Para este ejercicio codifique métodos para resolver los siguientes problemas:

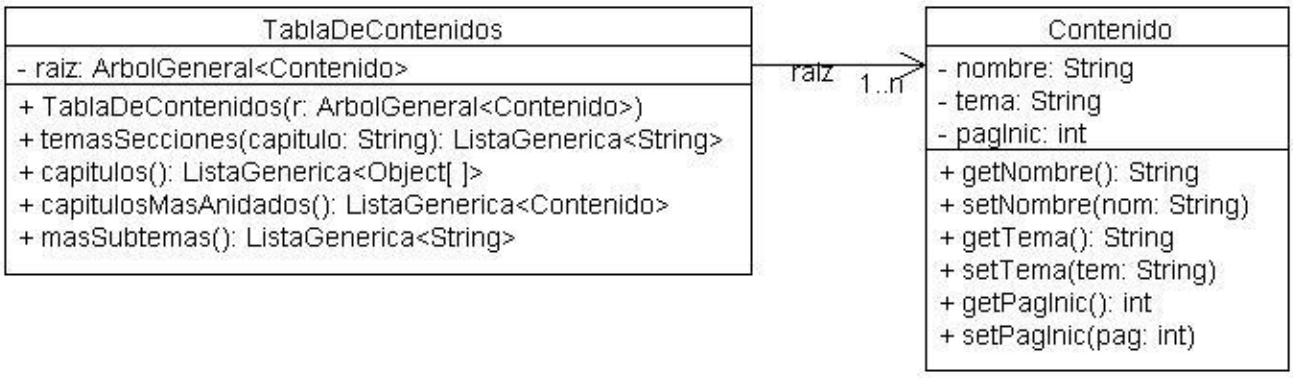
a) devolver para cada capítulo que conforma el libro su tema, pagina inicial y pagina final.
b) retornar el/los capítulos que están compuestos por el mayor numero de anidamientos de secciones.
c) retornar el/los temas que están compuestos por el mayor numero de subtemas.

Ejemplo:

Diagrama de objetos



Pistas:



a) Debe tomar los hijos de la raíz, que son los capítulos y apéndices del libro y de cada capítulo obtener el nombre, tema, página inicial y página final. Tener en cuenta que la página final de un capítulo es la anterior a la página inicial del siguiente capítulo. Se debe devolver la información de los capítulos únicamente.
En el ejemplo debería devolver: **"Capítulo 1, Listas, 1, 31", "Capítulo 2, Árboles, 32, 54", "Capítulo3, Grafos, 55, 73"**

b) Debe calcular la altura de cada capítulo utilizando el método implementado en 2ª y quedarse con todos los capítulos que igualen a la mayor altura.

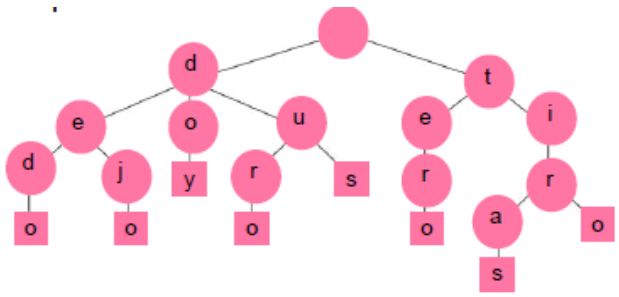
En el ejemplo el árbol general cuyo nodo raíz es capitulo 1 tiene una altura de 2, mientras que capitulo 2 y capitulo 3 tienen ambos una altura 0, por lo tanto la lista contendrá en este caso solo "Capitulo 1"

c) Se debe recorrer toda la tabla de contenidos en profundidad o por niveles. Recorrer cada uno de los temas contabilizando para cada uno la cantidad de hijos que tiene, y manteniendo solo los que igualan al mayor.

En el ejemplo **"Estructura de Datos"** y **"Listas con Arrays"** tienen 4 subtemas; **"Listas"** tiene 3 y los restantes temas no están compuestos por subtemas; con lo cuál la lista a retornar tendría **"Estructura de Datos, Listas con Arrays "**

Ejercicio 6 - Implementacion de un TRIE (TRIEs == re-trie-val trees)

Definición de un TRIE
- Es una estructura de datos que permite representar conjuntos de cadenas de caracteres.
- Cada nodo de T, excepto la raíz, está etiquetado con un símbolo del alfabeto
- Los hijos de un nodo interno de T están ordenados según el ordenamiento en el alfabeto
- Cada hoja marca el final de una cadena



Una aplicación frecuente de los TRIES es el almacenamiento de diccionarios, como los que se encuentran en los teléfonos móviles.

a) Implemente un TRIE con la siguiente operación:
public void agregarPalabra(String palabra)

Dada una palabra ingresada como parámetro, el método **agregarPalabra** deberá agregar cada uno de los caracteres que la conforman de manera de representar dicha palabra en el TRIE

b) Agregue a la implementación del TRIE la siguiente operación:
public List<StringBuffer> palabrasQueEmpiezanCon(String prefijo)

Devuelve una lista con todas las posibles palabras que comienzan con la cadena de caracteres ingresada como parámetro

UNLP. Facultad de Informática.

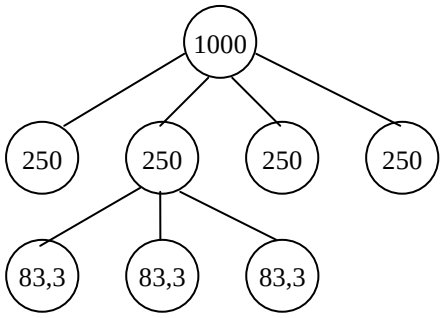
Algoritmos y Estructuras de Datos **Trabajo Práctico 5**

Cursada 2014

Anexo – Ejercicios Parciales

Ejercicio 1

Sea una red de agua potable, la cual comienza en un caño maestro y el mismo se va dividiendo sucesivamente hasta llegar a cada una de las casas. Por el caño maestro ingresan 1000 litros y en la medida que el caño se divide, el caudal se divide en partes iguales en cada una de las divisiones. Es decir, si el caño maestro se divide en 4 partes, cada división tiene un caudal de 250 litros. Luego, si una de esas divisiones se vuelve a dividir en 3 partes, cada una tendrá un caudal de 83,3. La situación descripta se puede modelar de la siguiente forma a través del siguiente árbol general:



Usted debe implementar un método en la clase árbol general, que considerando que ingresan n litros por el caño maestro, calcule cual es el mínimo caudal que recibe una hoja.

Ejercicio 2

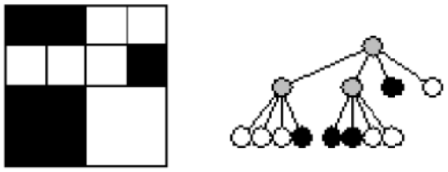
Un *quadtree* es una representación usada para cubrir un espacio cuadrado en dos dimensiones y posteriormente utilizado para determinar ciertas condiciones entre objetos en el mismo.

Un artista moderno trabaja con imágenes codificadas en *quadtrees*. El *quadtree* es un árbol 4-ario que codifica a una imagen con el siguiente criterio:

- Si toda la imagen tiene un mismo color, la misma es representada por un único nodo que almacene un dato que represente a ese color.

- En caso contrario, se divide la imagen en cuatro cuadrantes que se representan en el árbol como un nodo con 4 hijos, y cada hijo es la conversión de cada una de las partes de la imagen.

El artista desea saber cuántos pixeles de color negro posee una imagen dada. Usted debe implementar un método, que dado un *quadtree* y una cantidad total de pixeles, cuente cuantos pixeles de color negro contiene la imagen codificada en él.



Para el quadtree de la Figura, la salida del método sería 448

La figura muestra un ejemplo del árbol *quadTree* correspondiente a la imagen de la izquierda de 32 x 32 pixeles (1024 pixeles en total). Cada nodo en un *quadtree* es una hoja o tiene 4 hijos. Los nodos se recorren en sentido contrario a las agujas del reloj, desde la esquina superior derecha.

UNLP. Facultad de Informática.

Algoritmos y Estructuras de Datos **Trabajo Práctico 5**

Cursada 2014

Ejercicio 3

Jack es australiano y está de visita por la Argentina. Como todo turista quiere aprovechar al máximo su estadía. Su vuelo arriba a Buenos Aires y debe decidir que lugares visitar. Lo que sí sabe es que tomará una de las rutas nacionales que parten de Bs.As. y se alejará de la misma hasta llegar a la frontera con algún de los países limítrofes.

Considerando que los caminos posibles tienen forma de árbol en donde cada tramo de ruta tiene una cierta longitud y las ciudades tienen un número de sitios turísticos. Usted debe implementar un algoritmo para que determine qué camino debe recorrer, sabiendo que parte desde la raíz del árbol (ciudad de Buenos Aires) forma de minimizar la distancia recorrida, y si dos caminos tuvieran la misma distancia, deben maximizar los sitios turísticos (suma total de todos los sitios turísticos de las ciudades que atraviesa el camino)

a.- Defina la estructura del árbol en Java

b.- Implemente un método para buscar el mejor recorrido.

Ejercicio 4

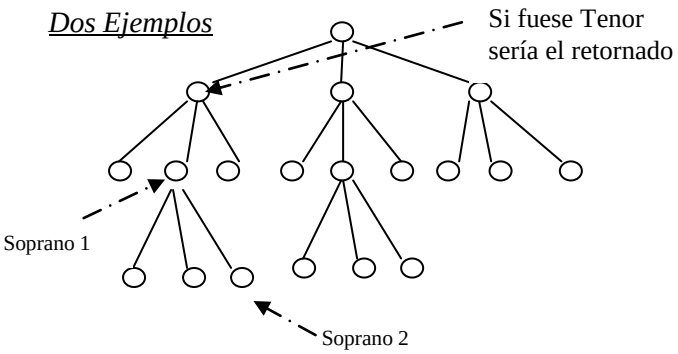
En una iglesia hay un coro conformado por n personas que actuará el próximo domingo en un festival coral en el teatro Coliseo de La Plata. Cada uno de los coristas tiene un tono e intensidad de voz. Entre los hombres del grupo hay Bajos, Barítonos y Tenores; mientras que los tonos de las mujeres son Contraaltos, Mezos y Sopranos.

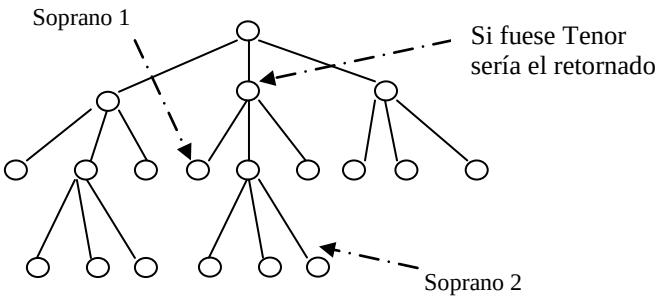
El director del coro encargó a un grupo de ingenieros en sonidos que estudien la acústica de la sala con el objetivo de que lo asesoren cómo disponer a los coristas en el escenario para que las voces de la derecha no se escuchen con más fuerza que las voces de la izquierda. Los ingenieros luego de un laborioso análisis y teniendo en cuenta la cantidad de personas, tonos e intensidades de voces aconsejaron que distribuyan a los cantantes siguiendo la forma de un árbol ternario, donde la raíz del mismo sea el director.

Su tarea será implementar en la clase ArbolGeneral un método Java que retorne el nombre del tenor **común más cercano** a dos sopranos en los caminos que van desde el director a cada una de las mismas. Los nombres completos de las mujeres son pasados como parámetros. En el caso que no existiera el tenor común debe retornar null.

Tenga en cuenta que para realizar la búsqueda los cantantes se identifican con su nombre y apellido.

Dos Ejemplos





Ejercicio 5

Llamaremos a un árbol general creciente si en cada nivel del árbol la cantidad de nodos que hay en ese nivel es igual al número de nivel más 1; es decir, el nivel 0 tiene exactamente un nodo, el nivel 1 tiene exactamente dos nodos, el nivel k tiene exactamente k+1 nodos. Implementar un método que compruebe si un árbol general es creciente y en caso que lo sea retornar el nodo del árbol con mayor cantidad de hijos. Si no lo es retornar null.