Informe final de ejecución del proyecto Innovación Docente realizado durante el curso académico 2012-2013 en el Departamento de Informática de la Facultad de Ciencias financiado mediante la convocatoria de 2012 de Ayudas de la Universidad de Salamanca para los Proyecto Estratégicos de Formación y Mejora Docente.

ID2012/193

Creación de material para prácticas de Estructuras de Datos Complejas

Miembros del Equipo de Trabajo:

María José Polo Martín

Ángeles María Moreno Montero

Iván Álvarez Navia

Susana Álvarez Rosado

Belén Curto Diego

Vidal Moreno Rodilla

José Rafael García Bermejo Giner

Francisco Javier Blanco Rodríguez

Departamento de Informática y Automática.
Facultad de Ciencias de la Universidad de Salamanca.
Plaza de los Caídos S/N, 37008

El objetivo principal del proyecto es el diseño de una metodología que permita al estudiante afianzar los contenidos teóricos de las asignaturas Estructuras de Datos y Algoritmos a través de la realización de las prácticas. Estas asignaturas se enmarcan dentro de la materia de Programación en el Grado de Ingeniería Informática. Complementan la formación obtenida en las asignaturas de primer curso y proporcionan una base para asignaturas posteriores, como Redes de Computadores.

Introducción

En este proyecto se han diseñado una serie de guiones para prácticas de Estructuras de Datos que permiten reforzar los conocimientos adquiridos en las clases teóricas mediante técnicas de implementación en C. Se han completado con una serie de ejercicios o pruebas que hacen posible evaluar el nivel de conocimiento adquirido por el estudiante. Se ha trabajado con todas las estructuras de datos y los algoritmos de manipulación que se tratan en la parte teórica de la asignatura Estructuras de Datos II.

Contexto de desarrollo

El aprendizaje de los algoritmos que manipulan las estructuras de datos plantea ciertas dificultades a los estudiantes de Ingeniería Informática. En esta materia se presentan al estudiante gran cantidad de algoritmos que manejan estructuras de datos, [7], [2], [1], [8], como árboles y gafos. La manera más habitual de explicarlos es mediante ejemplos gráficos sencillos que se proporcionan como material de apoyo a la asignatura. Se dispone además de una herramienta visual elaborada en el curso académico anterior "Herramienta para el aprendizaje visual y dinámico de estructuras de datos" fruto del proyecto de innovación de la convocatoria anterior (ID11/150). Esta herramienta permite al estudiante aplicar los algoritmos sobre nuevos ejemplos, comprobar el resultado cuando se aplica a casos más complejos y ver, de forma dinámica, la evolución del algoritmo aplicado.

En las sesiones prácticas hay que pasar del diseño y análisis de los algoritmos, a su implementación en un lenguaje de programación y, a su posterior ejecución y prueba. Es necesario, por tanto, elaborar unos guiones de prácticas que sirvan al estudiante como directriz y le permitan una comprensión escalonada de los conceptos estudiados en teoría.

Método de trabajo

El proyecto se ha realizado en cuatro fases:

Fase 1. Definición del proyecto. Fase inicial en la que se han definido los objetivos y se decide que algoritmos son los más adecuados para la realización del estudio empírico. En esta fase, desarrollada durante el mes de Noviembre de 2012, se decidió elaborar un guión por cada tema de la asignatura. Sin embargo, teniendo en cuenta la cantidad y dificultad de los conceptos de cada uno de ellos, se estimó que unos temas necesitarían más sesiones prácticas que otros. Se hizo una primera estimación que en el momento de llevarlo a la práctica tuvo algunos cambios.

Fase 2. Elaboración de material didáctico. Partiendo de la fase anterior se han creado seis guiones de prácticas que intentan cubrir todo el temario. Esta fase se ha desarrollado entre Diciembre de 2012 y Enero de 2013. En estos guiones se han planteado una serie de enunciados cuyo objetivo principal es permitir al estudiante asimilar los conceptos presentados en teoría. Se ha intentado plantear problemas que inciten a la búsqueda de soluciones, de forma, que el estudiante no se limite a transcribir

lo que el profesor explica en clase, sino que intente buscar una solución entre las diferentes estrategias que se le han presentado.

- Figura 1 -

La realización de los enunciados que se presentan en cada guión, como se ha mencionado antes, puede requerir varias sesiones de prácticas. En estas sesiones el estudiante cuenta con el apoyo del profesor para resolver sus dudas, pero hay que destacar que es muy importante también que el estudiante aporte su trabajo y estudio personal fuera de estas sesiones prácticas.

Fase 3. Aplicación. Puesto que se trata de una asignatura de segundo cuatrimestre, es durante los meses de Febrero a Mayo cuando se ejecuta esta fase. En ella se ha ido poniendo a disposición de los estudiantes todo el material de forma paulatina. Es decir, a medida que se iba avanzando en las clases teóricas se comenzaba en las sesiones prácticas con la implementación de los ejercicios planteado en los guiones elaborados. En la figura 1 se muestra la página de la asignatura en la plataforma STUDIUM. En ella puede observarse una práctica inicial, con su guión correspondiente, que se ha elaborado para no demorar el comienzo de las clases prácticas mientras se avanza en las clases teóricas. Este guión se corresponde con el último tema de Estructuras de Datos y Algoritmos I, asignatura que se imparte en el primer cuatrimestre y que, obviamente, está estrechamente relacionada con la asignatura Estructuras de Datos y Algoritmos II.

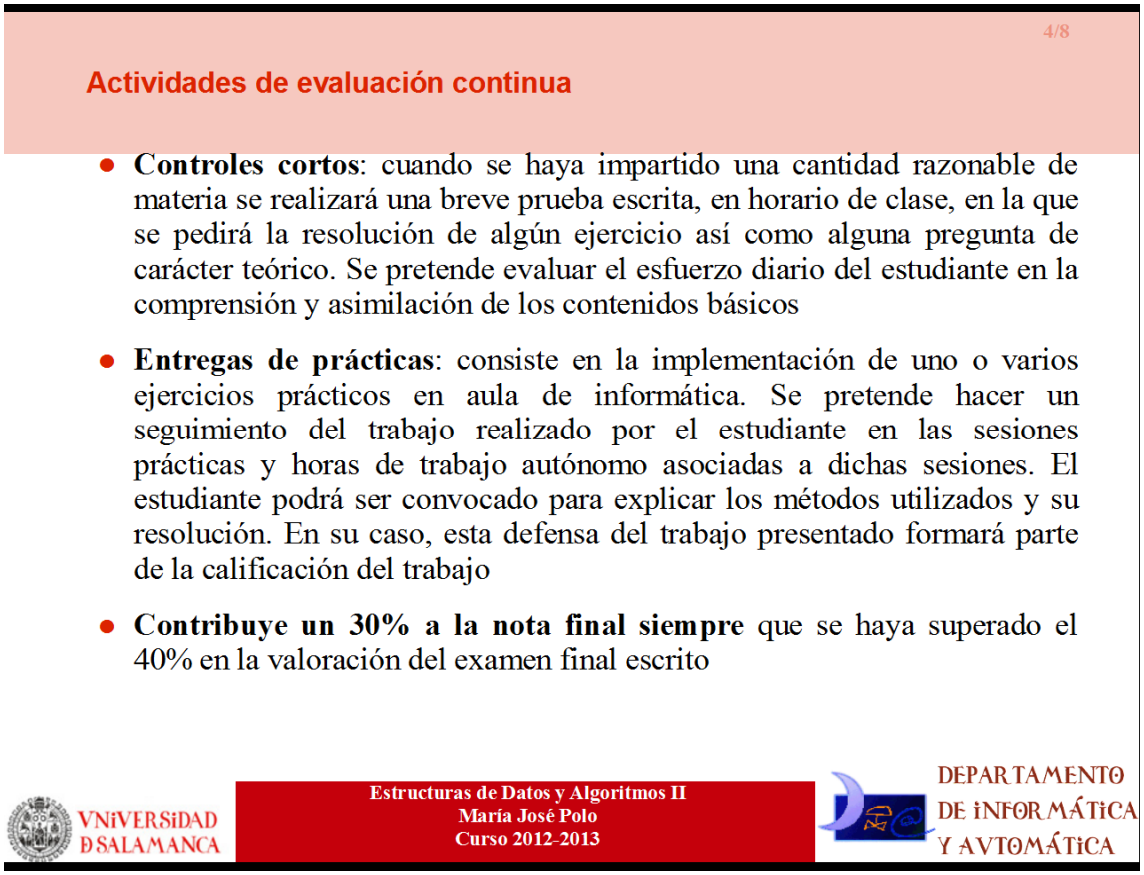
Se ha intentado conseguir una participación más activa de los estudiantes en el aprendizaje supervisando el trabajo que el alumno va realizando. De nuevo, se pone de

manifiesto, que solo con las sesiones prácticas no es posible la ejecución de todos los problemas planteados y que el alumno debe incluir de su parte algo de trabajo personal. De esta forma, las sesiones prácticas pueden ser mucho mejor aprovechadas por el alumno, pues en ellas puede plantear al profesor las dudas que le van surgiendo.

Fase 4. Resultados. Fase final del proyecto que se efectúa en Junio en el que se desarrolla este informe y se analizan los resultados obtenidos. Un análisis de los resultados obtenidos se aporta en el apartado final de conclusiones de este informe. También allí se aportan algunas mejoras que pueden aplicarse en cursos posteriores y que intentarán subsanar las deficiencias encontradas.

Elaboración del Material

Se han desarrollado una serie de guiones de prácticas teniendo en cuenta los criterios de evaluación y, muy especialmente las actividades de evaluación continua. El estudiante conoce estos criterios a través de la guía académica y de la página de la asignatura. En la figura 2 se muestra un extracto de la presentación de la asignatura referente a la evaluación continua. Los guiones desarrollados se enfocan al epígrafe *Entrega de prácticas*. Para el seguimiento del trabajo realizado por el estudiante se utilizan los guiones de prácticas elaborados. En el anexo I de este informe se muestra uno de estos guiones. En el anexo II se muestra la prueba que posteriormente se solicita al alumno para la evaluación continua de la práctica.



- Figura 2 -

En este contexto las prácticas de estructuras de datos se han enfocado teniendo cuidado de no convertir al estudiante en un mero programador. Estamos tratando con futuros Ingenieros Informáticos y queremos transmitir en la realización de las prácticas la diferencia entre un algoritmo que funciona correctamente y otros que además lo hacen de la forma más eficiente posible. Para ello se ha seguido la siguiente metodología:

- Un análisis previo del problema, este se presenta en las clases teóricas
- Una o varias implementaciones del problema: algunas se presentan en las clases teóricas, otras se mencionan y se proponen como ampliación para las clases prácticas.
- Obtención de resultados con la ayuda de los guiones de prácticas. Esta es la parte en la cual el alumno implementa, ejecuta y prueba los algoritmos sobre las estructuras de datos trabajadas.
- Análisis de resultados y relación de los mismos con el análisis teórico previo. Una vez que el estudiante ha trabajado sobre los algoritmos y estructuras de datos es el momento de analizar y relacionar los resultados obtenidos con el análisis teórico. Si el alumno ha sido capaz de llegar a esta fase se habrá encontrado con una serie de problemas y sus soluciones le proporcionarán una gran ayuda para la comprensión de los algoritmos.

Conclusiones

Con el nuevo de enfoque de evaluación y con la ayuda de los guiones de prácticas se ha conseguido una participación más activa del estudiante en el desarrollo de las clases. Esta participación ha ayudado a detectar las deficiencias que los alumnos encuentran en la implementación de los algoritmos. En este sentido se plantean mejoras para el siguiente curso, en el cual se intentará aportar al alumno material adicional. Principalmente código fuente para que las prácticas se centren en la implementación de los algoritmos que se ha estudiado y analizado. Se intentará proporcionar este material junto con los guiones y de esta manera se evita que el estudiante se desvíe del tema central y pueda desde el primer momento implementar, compilar y probar los algoritmos presentados utilizando este código fuente inicial. Código que en otro caso debería aportar el propio estudiante y que hemos observado que le resta tiempo para dedicar a los objetivos principales de la asignatura.

Otra aportación importante es la ayuda para la evaluación continua al profesor. Las pruebas se han enfocado de forma que, una vez presentado el tema en las clases teóricas y el guión correspondiente en las clases prácticas, al alumno debe hacer una entrega sobre alguno de los enunciados presentados en el guión. Normalmente se incluye alguna modificación para que el alumno aporte sus propias soluciones al problema que tiene que resolver. Esta entrega tiene como límite de tiempo el comienzo de las sesiones de prácticas correspondientes al siguiente tema y depende de la complejidad del tema

tratado. Hemos observado, sin embargo, que esta forma de evaluar la parte práctica presenta algunos inconvenientes. Por un lado al corregir las prácticas se observa que hay alumnos que presentan trabajos que ellos no ha hecho. Bien porque se detectan copias entre las entregas de diferentes alumno, bien porque debido al seguimiento individual del alumno se intuye que las prácticas no han sido realizadas por él. Por otro lado, una vez que el alumno sabe cuál es la práctica que tiene que entregar se centra en ella y se olvida del resto de los enunciados planteados. Para evitar estos problemas se piensa en modificar este tipo de pruebas y sustituirlas por pequeñas pruebas escritas, basadas en el guión correspondiente que el alumno deberá responder al inicio de la sesión de prácticas del siguiente tema. Esta alternativa además ayudará bastante al profesor, pues la tendencia es que el número de alumnos aumente y es más fácil corregir una pequeña prueba escrita que una entrega codificada que hay que compilar y probar. También beneficiará al alumno, pues será de realización personal con corrección personalizada. En ella el profesor indicará todos los errores encontrados y la forma de resolverlos y serán errores del alumno evaluado no de algún compañero que la haya implementado.

En definitiva, con los guiones de prácticas se ha pretendido mejorar la enseñanza en la materia de Estructuras de Datos y Algoritmos II. La fase de aplicación ha servido de retroalimentación pues se han detectado algunas deficiencias y fallos que se intentarán subsanar el curso que viene.

Referencias

[1] A. V. Aho, J. E. Hopcroft, and J. Ullman. Estructuras de datos y algoritmos. Addison-Wesley Iberoamericana, 1988.
[2] O. Cairó and S. Guardati. Estructuras de datos.McGraw-Hill, 2006.
[3] E. W. Dijkstra. A note on two problems in connection with graphs. Numeriche Mathematics, 1:269-171,October 1959.
[4] I. Jacobson, J. Rumbaugh, and G. Booch. El Proceso Unificado de desarrollo de software. Addison Wesley,2000.
[5] I.J. James Rumbaugh and G. Booch. El Lenguaje Unificado de Modelado. Manual de Referencia. Addison Wesley, 2000.
[6] Polo Martín, M.J.; Moreno Montero, A.M.; Iglesias Alonso, C. Experiencias en el uso de Técnicas de Visualización en el aprendizaje de Estructuras de Datos. Primeras Jornadas de Innovación Docente en la Universidad de Salamanca, pg. 276. Noviembre,2011.
[7] M. Weiss. Estructuras de datos y algoritmos. Pearson Education, 1995.
[8] N. Wirth. Algoritmos y estructuras de datos=Programas. Ediciones del Castillo, 1980.

ANEXO I

Guión de prácticas elaborado para el tema 2. Árboles Binarios de Búsqueda

Prácticas Árboles Binarios de Búsqueda

1. TAD Árbol Binario de Búsqueda

Utilizando la representación de árboles de la práctica anterior, haciendo las modificaciones oportunas para incluir el campo clave que nos permita ordenar los nodos, implementar las operaciones básicas de búsqueda, inserción y eliminación, teniendo en cuenta las siguientes especificaciones y prototipos:

int insertar(tipoClave x, tipoArbol *a)

La operación insertar debe crear un nuevo nodo con clave *x* en el árbol con raíz *a*. Utilizaremos el campo información de cada nodo para almacenar el número de veces que se ha intentado insertar un nodo con clave *x* en el árbol *a*. La operación debe devolver cero si la inserción acaba correctamente y -1 si ha ocurrido algún fallo en el proceso.

int buscar(tipoClave x, tipoArbol a, tipoNodo **nodo)

Esta operación busca en el árbol con raíz *a* un nodo con valor *x* para la clave; si lo encuentra devuelve su dirección en el parámetro nodo. Además devolverá el número de veces que se ha insertado un nodo con clave x, es decir, si el nodo está en el árbol devuelve el contenido de la información del nodo y cero si no lo encuentra. Si ocurre algún error en el proceso devolverá -1.

int eliminar(tipoClave x, tipoArbol *a)

Operación que elimina del árbol con raíz *a* el nodo con clave *x* si existe. El algoritmo devolverá un valor mayor o igual que cero si la eliminación acaba correctamente y -1 si ocurre algún error en el proceso. El valor mayor o igual que cero que devuelve la función indicará el número de veces que se intentó insertar un nodo con clave *x* en el árbol *a*. **Modificar** el algoritmo de eliminación en árboles binarios de búsqueda visto en teoría de forma que el **criterio** para eliminar un nodo con dos hijos sea sustituirlo por el **nodo más a la izquierda del subárbol derecho.**

2. Aplicación TAD Árbol Binario de Búsqueda

Implementar un algoritmo que, utilizando el TAD de la práctica anterior, clasifique todas las palabras de un fichero de texto determinando cuantas veces aparece cada palabra. ¿Sería eficiente utilizar listas enlazadas en lugar de a.b.b.?

3. **Inserción y eliminación en Árboles Binarios de Búsqueda con claves duplicadas**

   Utilizando la representación de árboles de la práctica anterior, haciendo las modificaciones oportunas para incluir el campo clave que nos permita ordenar los nodos, implementar las operaciones de inserción y eliminación, teniendo en cuenta los siguientes criterios

   1. Para todo nodo del a.b.b. se cumple que su clave es **mayor o igual** que todas las claves del subárbol izquierdo.
   2. Para todo nodo del a.b.b. se cumple que su clave es menor que todas las claves del subárbol derecho.
   3. Los nodos con claves duplicadas, si existen, deben ser nodos directamente enlazados

   Obsérvese que la definición admite claves duplicadas.

   El prototipo de la función insertar debe ser el siguiente:

   int insertar(ARBOL *raíz, tipoClave clave, tipoInfo info)

   La función debe devolver 0 si la inserción acaba correctamente o -1 si se produce algún tipo de error en el proceso

   El prototipo de la función eliminar debe ser el siguiente:

   int eliminar(ARBOL *raíz, tipoClave clave)

   La función debe devolver un número mayor o igual que cero indicando el número de nodos eliminados (todos los que tuvieran valor x para la clave) y un valor negativo (-1) si se produce algún tipo de error en el proceso

4. **Árboles Balanceados**

   Diseñar un algoritmo que recorra un a.b.b calculando el factor de equilibrio de todos sus nodos (hacer las modificaciones oportunas en tipoNodo) y devuelva verdadero o falso indicando si el árbol está balanceado o no.

Normas generales

En la realización de las prácticas se deben seguir los siguientes criterios:

1. Es **obligatorio** utilizar los tipos y prototipos siempre que se proporcionan en el fichero cabecera. Si no se proporcionan debe crearse el fichero cabecera correspondiente o añadir al proporcionado los prototipos de todas las funciones que se implementen.
2. La codificación de las funciones se realizará en un fichero fuente diferente al del programa principal.
3. Para visualizar el correcto funcionamiento de las operaciones implementadas se debe crear un fichero de prueba desde el cual se llamará a las funciones implementadas.
4. Se debe crear el correspondiente Makefile para la correcta creación del fichero ejecutable.

ANEXO II

Prueba para la evaluación continua elaborada para el tema 2. Árboles Binarios de Búsqueda.

Entrega ÁRBOLES BINARIOS DE BÚSQUEDA con claves repetidas

Definición del TAD Árbol Binario de Búsqueda con claves duplicadas

Utilizando la representación de árboles que se proporciona en el fichero cabecera arbol.h, implementar las operaciones de inserción y eliminación en un árbol binario de búsqueda, que admite claves **duplicadas**, teniendo en cuenta los siguientes criterios de definición (figura 1):

1. Para todo nodo del a.b.b. se cumple que su clave es **mayor o igual** que todas las claves del subárbol izquierdo.
2. Para todo nodo del a.b.b. se cumple que su clave es menor que todas las claves del subárbol derecho.
3. Los nodos con claves duplicadas, si existen, deben ser nodos **directamente enlazados.**

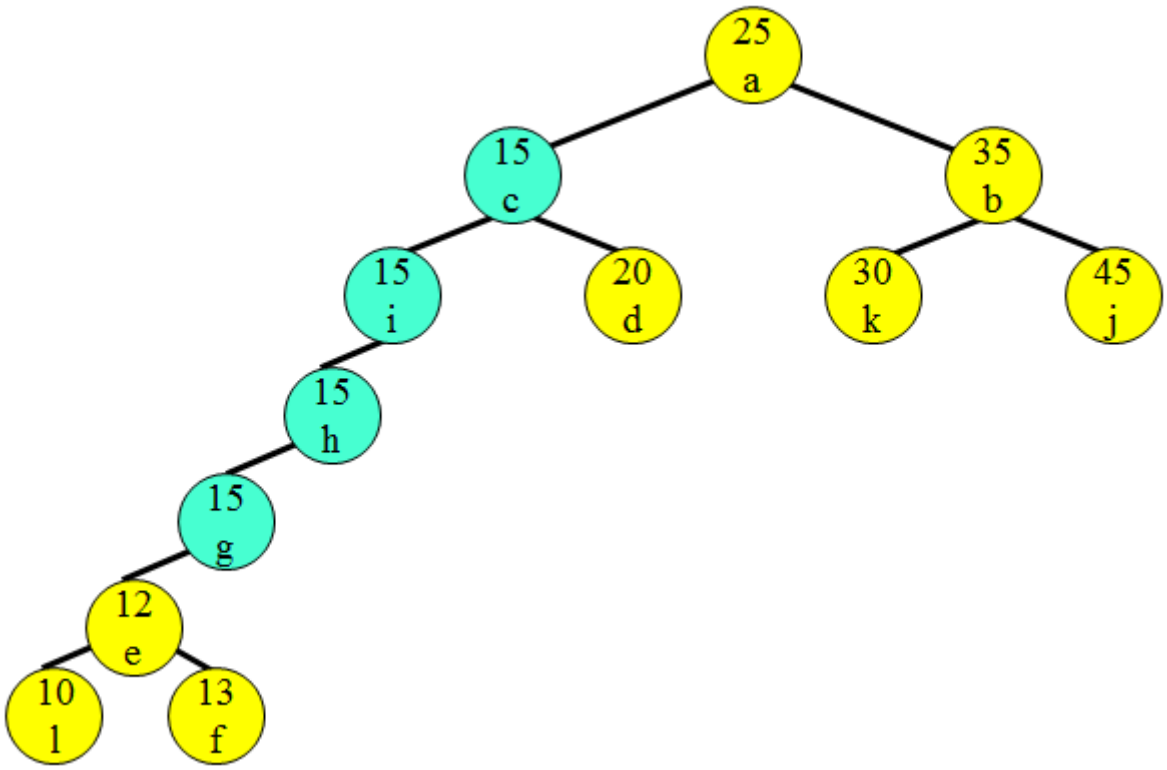


Figura 1. Representación del árbol que genera la secuencia de operaciones insertar de **prueba.c**.

Especificación de las operaciones a implementar

1. **Insertar**. Función que inserta un nuevo nodo en el árbol binario de búsqueda ***raíz,*** **teniendo en cuenta la definición anterior**. El nuevo nodo tendrá clave ***x*** e información ***info***. La función devuelve 0 si la inserción acaba correctamente y -1 si se produce algún tipo de error en el proceso. El prototipo de la función insertar debe ser el siguiente:

   int insertaABB(tipoArbolBB *raíz, tipoClave clave, tipoInfo info)

2. **Eliminar**. Función que elimina del árbol binario de búsqueda **raíz**, todos los nodos que tengan en su campo clave el valor del parámetro ***clave***. La función debe devolver un número mayor o igual que cero indicando el número de nodos eliminados (todos los que tuvieran el mismo valor para la clave) y un valor negativo (-1) si se produce algún tipo de error en el proceso. El prototipo de la función eliminar debe ser el siguiente:

   int eliminaABB(tipoArbolBB *raíz, tipoClave clave)

Ayuda para la realización de la práctica

Se proporciona un fichero **prueba.c** que va insertando claves en un a.b.b. inicialmente vacío. En este fichero (listado 1) se suceden una serie de operaciones de inserción, que generan un árbol como el presentado en la figura 1. También se llama a la operación eliminar con clave 5 (no hay ningún nodo) y con clave 15 (hay cuatro nodos que se deben eliminar), después de la eliminación el árbol resultante es el representado en la figura 2 (siempre que se siga el criterio de buscar y sustituir el nodo mas a la derecha del subárbol izquierdo cuando el nodo a eliminar tiene dos hijos). En el listado 2 se muestra el resultado de la ejecución del fichero **prueba.c**, donde además de las inserciones y eliminaciones, se recorre en orden el árbol generado antes y después de las operaciones de eliminación.

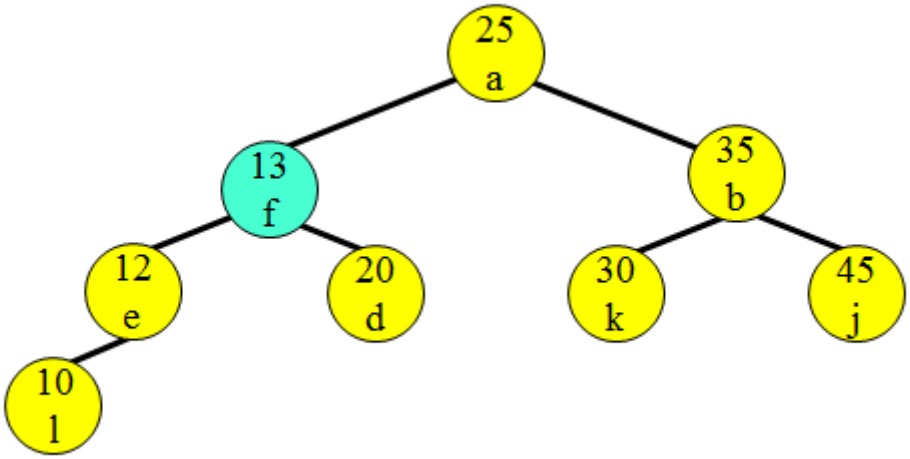


Figura 2. Representación del árbol resultante después de las operaciones de eliminación en **prueba.c**

- **Listado 1.** Extracto del fichero de prueba (**prueba.c**).

```
int main(void)
{ tipoArbolBB a=NULL;

 insertaABB(&a,25,'a');
 insertaABB(&a,35,'b');
 insertaABB(&a,15,'c');
 insertaABB(&a,20,'d');
 insertaABB(&a,12,'e');
 insertaABB(&a,13,'f');
 insertaABB(&a,15,'g');
 insertaABB(&a,15,'h');
 insertaABB(&a,15,'i');
 insertaABB(&a,45,'j');
 insertaABB(&a,30,'k');
 insertaABB(&a,10,'l');
 printf("Secuencia de los nodos en orden: \n");
 enOrden(a);
 printf("Eliminados %d nodos con clave %d\n", eliminaABB(&a,5),5);
 printf("Eliminados %d nodos con clave %d\n", eliminaABB(&a,15),15);
 printf("Secuencia de los nodos en orden: \n");
 enOrden(a);
 return 1;
}
```

- **Listado 2.** Resultado de la ejecución del fichero de prueba (**prueba.c**).

```
Secuencia de los nodos en orden:
10 l
12 e
13 f
15 g
15 h
15 i
15 c
20 d
25 a
30 k
35 b
45 j
Eliminados 0 nodos con clave 5
Eliminados 4 nodos con clave 15
Secuencia de los nodos en orden:
10 l
12 e
13 f
20 d
25 a
30 k
35 b
45 j
```

Normas generales

Para la realización y entrega de práctica se deben seguir los siguientes criterios:

1. Es **obligatorio** utilizar los tipos y prototipos que se proporcionan en el fichero cabecera arbol.h.

2. La codificación de las funciones se realizará en el fichero proporcionado arbol.c, donde se incluye la implementación de algunas funciones de ayuda. Este fichero, modificado con la implementación de las funciones insertaABB y eliminaABB, es el único que hay que subir a STUDIUM.

3. Para la corrección de las funciones implementadas se utilizará el fichero prueba.c que se proporciona como ayuda para la visualización de las operaciones implementadas, es por tanto, muy importante utilizar los tipos y prototipos proporcionados. Pueden utilizarse alternativamente otros fichero de prueba para ver la correcta implementación de las funciones.
4. Para la compilación y gestión de todo el proyecto se utilizará el fichero Makefile que se proporciona, es por tanto, muy importante no cambiar el nombre de los ficheros proporcionados, especialmente **arbol.c** y **arbol.h**.