Algoritmos y estructuras de datos

Árboles binarios de búsqueda

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Posgrado en Optimización

19 de mayo de 2021

Citas

Miguel de Unamuno

Hubo árboles antes de que hubiera libros, y acaso cuando acaben los libros continúen los árboles. Y acaso llegue la humanidad a un grado de cultura tal que no necesite ya de libros, pero siempre necesitará de árboles, y entonces abonará los árboles con libros.

Jacinto Benavente

En cuestión de árboles genealógicos es más seguro andarse por las ramas que buscar las raíces.

Árboles binarios enraizados

Definición recursiva

Un árbol binario enraizado A puede:

1. estar vacío o
2. consistir de un nodo r, llamado la raíz de A, y dos árboles binarios enraizados izquierdo A_1 y derecho A_2.

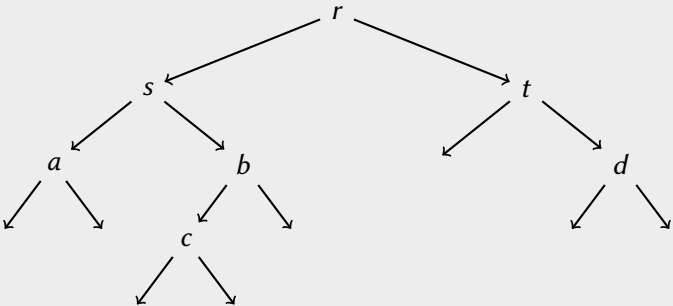
Nodos precursores y sucesores

1. El orden de A es la cantidad n de nodos que contiene. Sólo nos interesan los árboles enraizados de orden finito.
2. Un nodo s es el precursor de las raíces de sus árboles binarios enraizados no vacíos. Estas raíces son los sucesores izquierdo y derecho de s, respectivamente.
3. Dos nodos que tengan el mismo precursor se dicen semejantes.
4. Una hoja de A es un nodo de A sin sucesores.

Árboles binarios enraizados

Ejemplo

El árbol binario enraizado A de orden 7 tiene raíz r con sucesor izquierdo s y sucesor derecho t. El nodo s es el precursor de a y b, por lo que a y b son semejantes. El nodo b sólo tiene sucesor izquierdo c y el nodo t sólo tiene sucesor derecho d. Los nodos a, c, d son las hojas de A.



Árboles binarios enraizados

Grado, altura y profundidad de un nodo

Grado

El grado de un nodo s es la cantidad $d_s \leq 2$ de sucesores no vacíos que tiene.

Altura

La altura h_s de un nodo s se define de forma recursiva:

1. Si s es una hoja, entonces $h_s = 1$.
2. En caso contrario $h_s = 1 + \max_{1 \leq i \leq d_s} h_{s_i}$.

Profundidad

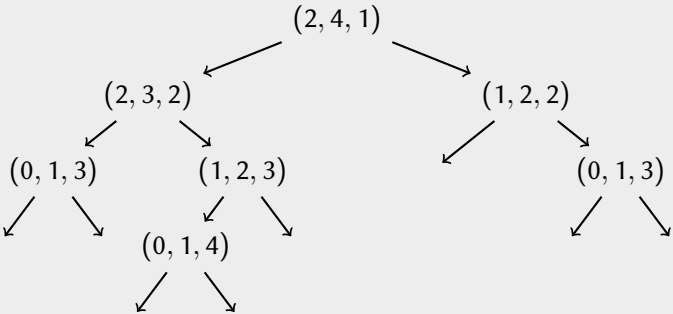
La profundidad p_s de un nodo s se define de forma recursiva:

1. Si s es la raíz, entonces $p_s = 1$.
2. En caso contrario s tiene precursor s' y $p_s = 1 + p_{s'}$.

Árboles binarios enraizados

Un árbol binario enraizado de altura 4

En cada nodo s anotamos (d_s, h_s, p_s).



Árboles binarios enraizados

Grado, altura y profundidad de un árbol

En un árbol binario enraizado

El grado, la altura y la profundidad de A se definen como el máximo grado, la máxima altura y la máxima profundidad de sus nodos, respectivamente.

Altura y profundidad

La altura de un nodo s es la máxima cantidad de nodos en un camino de s a alguna hoja. La profundidad de un nodo s es la cantidad de nodos en el camino de la raíz a s. La altura y la profundidad de un árbol coinciden.

Niveles

Todos los nodos a la misma profundidad conforman un nivel. La raíz está a nivel 1, sus sucesores a nivel 2 y así sucesivamente.

Árboles de recursión binaria

La llamada inicial es la raíz, las llamadas recursivas son sus sucesores, los casos base son las hojas. El tiempo de ejecución es proporcional al orden.

Listas enlazadas

Una lista enlazada con $n \geq 1$ nodos es un árbol binario enraizado de grado 1, altura n y una hoja (el único nodo sin nodo siguiente).

Montículos

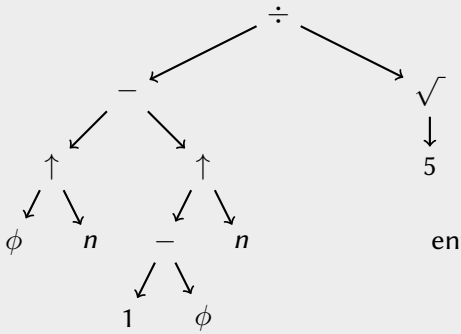
Un montículo con $n \geq 1$ nodos es un árbol binario enraizado con altura $1 + \lfloor \log_2 n \rfloor$ y $\lceil \frac{n+1}{2} \rceil$ hojas. Cada precursor tiene un dato mayor o igual al de sus sucesores. Además, los primeros $\lfloor \log_2 n \rfloor$ niveles están llenos.

Árboles binarios enraizados

Ejemplo de árbol de expresión aritmética

Una expresión aritmética consta de una constante (hoja) o bien de un operador junto con sus 1 o 2 operandos (sucesores), que a su vez son expresiones aritméticas.

Ejemplo (Una fórmula para los números de Fibonacci)



Si $\phi = \frac{1}{2}(1+\sqrt{5})$,

entonces $F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}$.

Árboles binarios de búsqueda

Sea s un nodo de un árbol binario enraizado A.

1. Los ascendientes de s son su precursor y sus ascendientes.
2. Los descendientes de s son sus sucesores y sus descendientes.

La raíz no tiene ascendientes y las hojas no tienen descendientes.

Definición

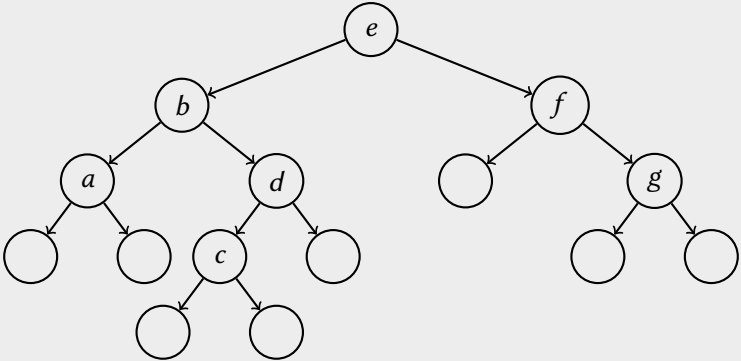
Un árbol binario de búsqueda A es un árbol binario enraizado donde cada nodo s contiene un dato a_s. Los datos en los nodos satisfacen:

1. Si s tiene sucesor izquierdo no vacío s_1, entonces $a_s > a_t$ para cualquier nodo t que sea s_1 o descendiente de s_1.
2. Si s tiene sucesor derecho no vacío s_2, entonces $a_s < a_t$ para cualquier nodo t que sea s_2 o descendiente de s_2.

Árboles binarios de búsqueda

Ejemplo

En cada nodo mostramos el dato que contiene (letra en orden alfabético).



En el futuro nos ahorraremos dibujar los sucesores vacíos.

Árboles binarios de búsqueda

Tipos asociados a un árbol binario de búsqueda

Definiremos un tipo estructurado nodoABB para representar un nodo y un tipo abb para representar un árbol binario de búsqueda. El tipo nodoABB consiste de un dato a y dos apuntadores izq y der a sus sucesores izquierdo y derecho (valdrán NULL si son vacíos).

```
typedef struct nodoABB {
  int a;                // dato almacenado
  struct nodoABB *izq; // enlace al izquierdo
  struct nodoABB *der; // enlace al derecho
} nodoABB;
```

Por otro lado, el tipo abb es un apuntador a nodoABB.

```
typedef nodoABB *abb;
```

Note que los tipos nodoABB *, struct nodoABB * y abb son equivalentes.

Árboles binarios de búsqueda

Creación de un nodo

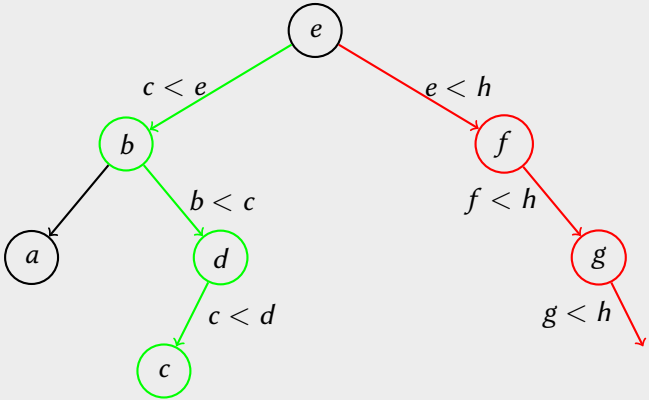
Esta función pide la memoria para un nodo y llena sus campos.

```
nodoABB *creaNodoABB(int x) {
  nodoABB *t = (nodoABB *) malloc(sizeof(nodoABB));
  t->a = x;
  t->izq = NULL;
  t->der = NULL;
  return t;
}
```

Árboles binarios de búsqueda

Ejemplo de búsqueda en un árbol binario de búsqueda

Buscar la c, que sí está, y buscar la h, que no está.



Árboles binarios de búsqueda

Búsqueda de un nodo

Como esto no modifica un árbol `s`, lo pasamos por valor. Si el dato `x` no está en el árbol, regresa `NULL`.

```
nodo *buscaABB(abb s, int x) {
  while (s != NULL) {          // arbol no vacio
    if (s->a == x) break;      // dato encontrado
    s = (s->a > x) ? s->izq : s->der; // avanza
  }
  return s;
}
```

Árboles binarios de búsqueda

Búsqueda de la referencia a un nodo

Aunque no modifica al árbol, lo pasamos por referencia. Si el dato x no está en el árbol, regresa una referencia a donde debería estar.

```
abb *buscarABB(abb *s, int x) {
  while (*s != NULL) {            // arbol no vacio
    if ((*s)->a == x) break;      // dato encontrado
    s = ((*s)->a > x) ? &((*s)->izq) : &((*s)->der);
  }
  return s;
}
```

Árboles binarios de búsqueda

Recorrido en orden

La definición recursiva de un árbol binario de búsqueda nos da un algoritmo recursivo para recorrer sus nodos en el orden de los datos. Esta es una primera versión ineficiente del algoritmo de ordenamiento por árbol.

```
void ordenABB(abb s) {
  if (s != NULL) {     // si no esta vacio
    ordenABB(s->izq); // recorre arbol izquierdo
    procesa(s);       // procesa nodo
    ordenABB(s->der); // recorre arbol derecho
  }
}
```

La función procesa debe hacer lo que nos interese.

Árboles binarios de búsqueda

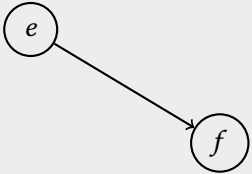
Ejemplo de inserción

Queremos insertar datos en el orden e, f, b, d, g, a, c.

Árboles binarios de búsqueda

Ejemplo de inserción

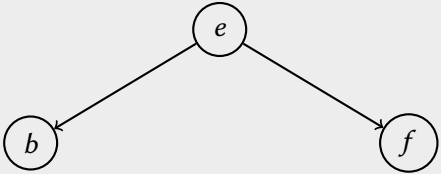
Queremos insertar datos en el orden e, f, b, d, g, a, c.



Árboles binarios de búsqueda

Ejemplo de inserción

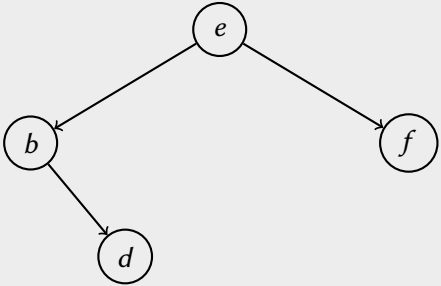
Queremos insertar datos en el orden e, f, b, d, g, a, c.



Árboles binarios de búsqueda

Ejemplo de inserción

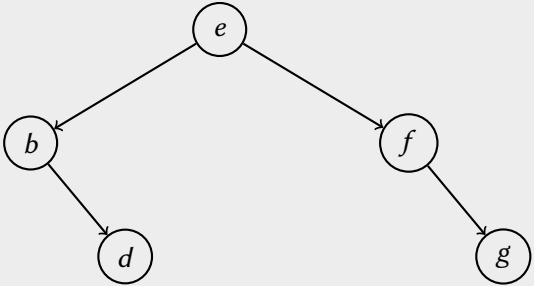
Queremos insertar datos en el orden e, f, b, d, g, a, c.



Árboles binarios de búsqueda

Ejemplo de inserción

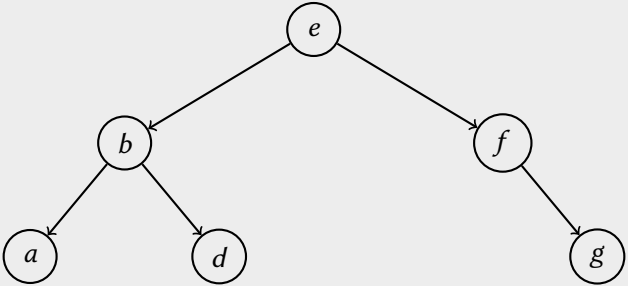
Queremos insertar datos en el orden e, f, b, d, g, a, c.



Árboles binarios de búsqueda

Ejemplo de inserción

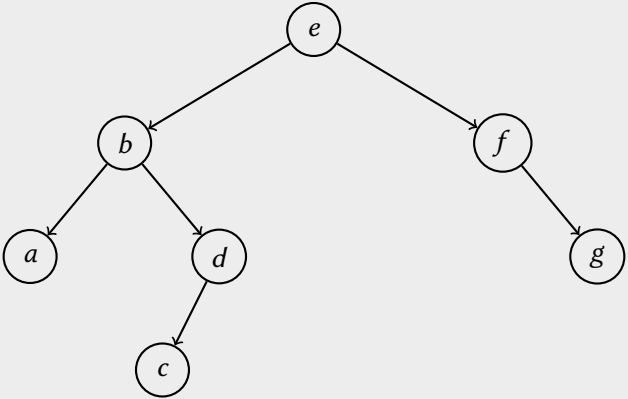
Queremos insertar datos en el orden e, f, b, d, g, a, c.



Árboles binarios de búsqueda

Ejemplo de inserción

Queremos insertar datos en el orden e, f, b, d, g, a, c.



Árboles binarios de búsqueda

Inserción de un nodo

Como esto implica modificar un árbol s, requerimos de una referencia p al árbol en donde queremos insertar el nodo. Por ejemplo, si queremos insertar en la raíz, esa referencia sería p = &s.

```
void insertaABB(abb *p, int x) {
  p = buscarABB(p, x);
  if (*p == NULL)          // no se encuentra dato x
    *p = creaNodoABB(x); // crea nodo nuevo con x
}
```

Borrar una hoja

Se libera el nodo y su referencia se hace `NULL`.

Borrar un nodo con un sucesor

Se libera el nodo y su referencia apunta a su único sucesor.

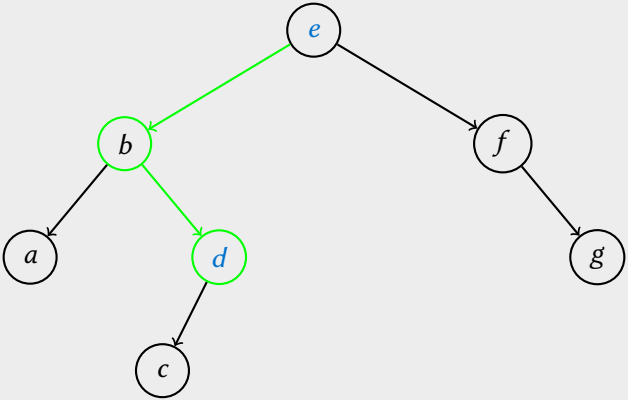
Borrar un nodo con dos sucesores

Se busca el nodo que contenga el mayor dato que sea menor al que se quiera borrar (es decir, el dato anterior). El dato anterior sobreescribe al que se quería borrar y su nodo se borra como en el caso anterior.

Árboles binarios de búsqueda

Ejemplo de borrado

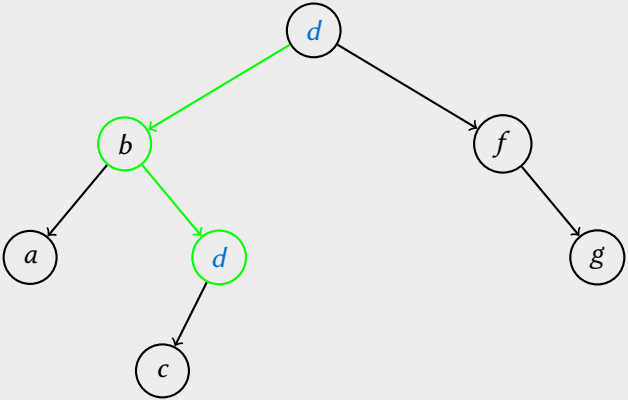
Borrar c o d es sencillo. En lugar de eso, para borrar e, se busca la mayor clave menor que e (es decir d), se copia al lugar de e y se borra su nodo.



Árboles binarios de búsqueda

Ejemplo de borrado

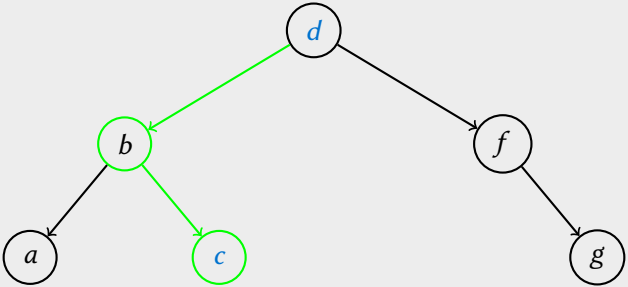
Borrar c o d es sencillo. En lugar de eso, para borrar e, se busca la mayor clave menor que e (es decir d), se copia al lugar de e y se borra su nodo.



Árboles binarios de búsqueda

Ejemplo de borrado

Borrar c o d es sencillo. En lugar de eso, para borrar e, se busca la mayor clave menor que e (es decir d), se copia al lugar de e y se borra su nodo.



Árboles binarios de búsqueda

Borrado de un nodo

Como esto implica modificar un árbol s, requerimos de una referencia p al árbol de donde queremos borrar el nodo. Por ejemplo, si queremos borrar de la raíz, esa referencia sería p = &s.

```
void borrarABB(abb *p, int x) {
  p = buscarABB(p, x);    // busca la referencia
  if (*p != NULL) {       // se encuentra dato x
    if ((*p)->izq != NULL && (*p)->der != NULL)
      p = mueveABB(p);    // caso dos sucesores
    quitaABB(p);          // caso 0 o 1 sucesor
  }
}
```

Árboles binarios de búsqueda

Borrado de un nodo (casos de 0, 1 o 2 sucesores)

```
abb *mueveABB(abb *p) { // caso dos sucesores
  abb *q = &((*p)->izq);
  while ((*q)->der != NULL)
    q = &((*q)->der);
  (*p)->a = (*q)->a;
  return q;
}

void quitaABB(abb *p) { // caso 0 o 1 sucesor
  nodoABB *q = *p;
  *p = (q->izq != NULL) ? (q->izq) : (q->der);
  free(q);
}
```

Seis representaciones de conjuntos

Resumen de resultados

Número de pasos en el peor caso, sobre un conjunto A de n elementos y un elemento x. Abajo $h \leq n$ es la altura del árbol binario de búsqueda.

Operación	Símbolo	Bit	AD	AO	LD	LO	ABB
crear	$\emptyset$	n	1	1	1	1	1
destruir		1	1	1	n	n	n
cardinalidad	$\lvert A \rvert$	1	1	1	1	1	1
pertenencia	$x \in A$	1	n	$\log_2 n$	n	n	h
agregar	$A \cup x$	1	n	n	n	n	h
eliminar	$A \setminus x$	1	n	n	n	n	h

Nuestro objetivo: hacer pertenencia, agregar y eliminar en $\approx \log_2 n$ pasos.