Simón Alonso Gutiérrez, 821038
Diego Domingo Ralla, 818637

Práctica 2: Divide y vencerás

Para este problema se dejan caer *n* bolas, una a una, desde la raíz de un árbol binario completo. Todos los nodos están numerados comenzando en 1 con la raíz. Los nodos de un mismo nivel están numerados de izquierda a derecha.

Cada vez que se deja caer una bola, esta pasa por un nodo no terminal, y continúa hacia abajo, bien por el subárbol izquierdo o por el derecho, hasta llegar a un nodo hoja. Para determinar la dirección, se simula que cada nodo tiene asociado un flag booleano: falso o verdadero. Al inicio, todos los nodos tendrían este flag a falso, y cada vez que una bola pasa por un nodo, cambia el valor de su flag. Si el flag es falso, la bola caerá por el lado izquierdo, y si es verdadero caerá por el lado derecho.

Se ha diseñado un algoritmo divide y vencerás que toma como argumentos la profundidad *p* del árbol y el índice *i* de la bola que se deja caer. Este algoritmo devuelve la posición de parada de la bola, es decir, el número del nodo hoja donde cae la bola. Por tanto, *i* debe ser menor o igual que el número total de hojas del árbol, que es igual a 2^p - 1.

Dado que realmente no se ha implementado un árbol, y por lo tanto no se tiene cada uno de los nodos para almacenar si su flag es falso o verdadero, se ha seguido la siguiente estrategia, basada en la recursividad por inmersión:

1. Inicialmente la función *búsqueda* invoca a *busquedaInm*, que tiene además como argumento el número de nodo del árbol completo en el que se encuentra la bola (inicialmente 1).
2. Si *i* es impar, la bola caerá por el subárbol izquierdo, y si es par por el derecho. Se invoca recursivamente a *busquedaInm* con *p* - 1 y el próximo nodo a visitar. Además, también se recalcula *i*, ya que realmente representa el índice de la bola para el subárbol en el que se encuentra la búsqueda. Por tanto, cada vez que se baja un nivel, *i* se divide entre 2 (o *i* + 1 si el nodo era impar).
3. Finalmente, cuando *p* = 1, se devuelve el nodo.

Con esta estrategia basada en calcular el nuevo índice de la bola basado en el subárbol por el que se va a continuar con la búsqueda, se puede saber únicamente con el valor de *i* la dirección por la que va a caer la bola. Esto sucede porque como cada nodo tiene un flag booleano que va alternando cada vez que pasa una nueva bola, e inicialmente su valor es falso, todas las bolas que para un nodo sean impares caerán por el lado izquierdo, mientras que las pares caerán por el derecho.

Utilizando este algoritmo, se ha desarrollado un programa que permite calcular la posición de las primeras *n* bolas dejadas caer por un árbol con profundidad *p*, por lo que se pueden tirar tantas bolas como nodos hoja tenga el árbol.

Ejecución de pruebas

Se ha realizado un script *ejecutar.sh* que automatiza la ejecución del programa con varios casos de prueba, que se incluyen en el fichero *pruebas.txt*. En cada línea se encuentra cada una de las configuraciones a probar, donde el primer número es la profundidad *p* del árbol y el segundo es el número *n* de bolas a tirar.

Cabe destacar que se ha utilizado unsigned long long para poder alcanzar profundidades mayores y, por tanto, nodos más profundos. Esto permite probar una configuración con una profundidad de 64.

Dado que el tiempo de búsqueda de cada bola para una misma configuración es similar, el mayor número de bolas que se ha tirado es de 100 para los árboles de profundidad 10, 32 y 64. No se ha considerado necesario utilizar valores mayores de *n*, puesto que el coste computacional depende de *p*.

Análisis de las pruebas realizadas

Coste espacial

O(*p*), es lineal a la profundidad del árbol.

Explicación

En cada llamada recursiva se van almacenando más valores en la pila. Por lo que la cantidad de memoria necesaria depende de la profundidad del árbol.

Coste temporal

O(*p*), es lineal a la profundidad del árbol

Explicación

La profundidad del árbol viene dada por el parámetro *p*. En cada llamada recursiva de la función, se disminuye en uno el valor de *p*, y se finaliza cuando *p* = 1. Por lo tanto, el número de llamadas recursivas que se hacen es igual a la profundidad del árbol. Además, para cualquier bola en un mismo árbol, el tiempo de búsqueda va a ser similar, puesto que todas caen a un nodo hoja, y como todos se encuentran en el mismo nivel, requieren del mismo número de llamadas recursivas.

Profundidad	**Tiempo medio de búsqueda de cada bola**
3	5931,25 ns
4	6837,00 ns
5	7709,12 ns
6	8258,28 ns
10	13725,00 ns
32	25871,80 ns
64	55474,60 ns

Como se puede observar en los resultados obtenidos, efectivamente el coste temporal es lineal en función de la profundidad del árbol, puesto que este aumenta conforme lo hace el valor de *p*. Además, si se observa el tiempo de búsqueda de cada una de las bolas tiradas en un mismo árbol, este es similar en todas ellas.