Auxiliar 2 - Programas recursivos y notación Big-O

Profesores: Iván Sipiran
Patricio Poblete
Nelson Baloian

Auxiliares: Gabriel Flores, Gabriel Norambuena
Rodrigo Llull, Lucas Oyarzún

P1. Recursión

En esta pregunta recordaremos lo que es la recursión, contenido que se vio en el curso de Introducción a la Programación. Definiremos un árbol de búsqueda binaria de números en base a nodos. Para ello, necesitamos la clase Nodo, que contenga su número en un atributo llamado `info`, y una referencia a los hijos izquierdo y derecho, en atributos llamados `izq` y `der`, respectivamente.

Una vez definida Nodo, procedemos de la siguiente forma:

(a) Creamos el nodo con valor 1, sin hijos.

(b) Continuamos con el 3, sin hijos.

(c) Creamos el 2, con hijos el 1 y el 3.

(d) Seguimos con el 6, sin hijos.

(e) Ahora con el 5, con hijo derecho el 6 e hijo izquierdo None.

(f) Terminamos creando el 4, con hijos el 2 y el 5. Este nodo será llamado raíz.

Utilice la librería AED-Utilities para dibujar el árbol, y verificar que tiene la forma deseada. Para ello:

(a) Cree una instancia de una clase Arbol (que tiene solo la raiz como variable miembro), entregándole como argumento el nodo con valor 4 de la sección anterior.

(b) Cree una instancia de la clase BinaryTreeDrawer, con los argumentos fieldData="info", fieldLeft="izq" y fieldRight="der".

(c) Dibuje el árbol utilizando el BinaryTreeDrawer creado.

Finalmente, se le pide definir una función que recorra los nodos recursivamente, imprimiendo sus valores en orden ascendente. Por ahora no necesita trabajar con el árbol creado para dibujar.

P2. Subarreglo de suma máxima

Dado un arreglo con enteros positivos y negativos, se quiere conseguir el subarreglo que contiene la suma máxima posible. Un subarreglo es cualquier subconjunto consecutivo del arreglo.

a) Programe una función `segmentoMaximo(arreglo)` que resuelva el problema, sin importar cuan eficiente sea. Debe imprimir la suma y los índices de inicio y fin.

b) ¿Es óptimo el algoritmo creado? ¿Qué optimizaciones se le podrían hacer para que sea más eficiente?

c) ¿Puede la solución llegar a tiempo de ejecución $O(n)$? *Hint: La clave está en los números que son negativos.*