Evaluación Formativa 1
INFO08 - Taller Estructuras de Datos y Algoritmos

Académico: Héctor Ferrada.
Instituto de Informática, Universidad Austral de Chile.
Septiembre 16, 2021

Ejecución: ./problema n

En clases se estudiaron los algoritmos insertionSort y quickSort para ordenar un arreglo. El pseudocódigo de quickSort para ordenar ascendendemente $A[\ell \dots r]$ es el siguiente:

```
quickSort(A, ℓ, r){
    if(ℓ < r){
        p =partition(A, ℓ, r)
        quickSort(A, ℓ, p − 1)
        quickSort(A, p + 1, r)
    }
}
```

Dónde partition(A, ℓ, r) realiza la partición de $A[\ell \dots r]$ tomando como pivote al primer elemento $A[\ell]$. Ahora usted debe crear un método híbrido, llamado **quickInsert**(), que mezcla el esquema de particiones de quickSort con insertionSort, de la siguiente manera:
Su método debe recibir un parámetro, *umbral*, que le servirá para determinar si $A[\ell \dots r]$ se debe ordenar con insertionSort, siempre y cuando el largo de $A[\ell \dots r]$ sea menor o igual al *umbral*; ó, en caso contrario, deberá particionar $A[\ell \dots r]$ y continuará recursivamente tal como lo hace quicsort. Al comienzo de su método recursivo debe validar que $A[\ell \dots r]$ contenga más de un elemento, tal como lo hace quickSort.

Teniendo en cuenta lo anterior, se pide:

1. **[1 Pto.]** En el **main**, cree un arreglo A de capacidad n, para el n leído desde los argumentos del programa. Invoque a la función llenaArray(A, n) e imprima el arreglo generado por la función invocando al método listaArray(). Luego, pida el ingreso del valor del umbral como se introdujo anteriormente y, a continuación, invoque a quickInsert$(A, 0, n-1, umbral)$ e imprima otra vez el arreglo.

2. **[1 Pto.]** Cree la función **llenaArray**(int *A, int n), en la cual debe llenar el arreglo $A[]$. Para esto, pida el ingreso de dos valores: $minVal$ y $maxVal$, validando y forzando que $minVal <= maxVal$, y utilice estos valores para crear los n enteros aleatorios en el intervalo $[minVal \dots maxVal]$ que ingresará al arreglo.

3. **[1.5 Ptos.]** Adapte el método insertionSort() para que ahora ordene el segmento de arreglo $A[\ell \dots r]$. La declaración del método debe ser ahora: **insertionSort**(int *A, int ℓ, int r).

4. **[2.5 Ptos.]** Cree el método **quickInsert**(int *A, int ℓ, int r, int *umbral*) que implementa el algoritmo híbrido descrito anteriormente.

En el fuente dispone de los médodos **partition**() e **insertionSort**() estudiados en clases. Además, dispone del método **listaArray**(int* A, int ℓ, int r) que imprime el segmento de arreglo $A[\ell \dots r]$.