Control 1
INFO08 - Taller Estructuras de Datos y Algoritmos

Académico: Héctor Ferrada.
Instituto de Informática, Universidad Austral de Chile.
Octubre 28, 2021

Ejecución: ./problema n k

En este problema usted debe ordenar descendentemente un arreglo A de n enteros por la distancia que cada elemento tiene con la media aritmética del arreglo. Luego, mediante búsquedas binarias en A, debe buscar si existe algún elemento cuya distancia a la media sea un valor ingresado por el usuario. Para esto, debe leer k flotantes desde el teclado y realizar una búsqueda para cada distancia ingresada. Para cada búsqueda exitosa almacene en un std::queue, de tipo **dato**, el dato encontrado y su posición en A, donde el tipo de datos **dato** está definido en el fuente como:

```
struct myStruct {
    float dist;                ▷ distancia entre val y la media de A
    int val;                   ▷ valor en A con distancia dist de la media
    int pos;                   ▷ posición del valor val encontrado en A con distancia dist
}; typedef struct myStruct dato;
```

Se pide:

1. **[1.0 Pto.]** En el **main**, cree el arreglo A de tamaño n e invoque a la función **generaEnteros**(A, n), la cual debe poblar el arreglo y devolver su media aritmética med, e imprima el arreglo. Luego, invloque al método **insertionSortMed**(A, n, med) y vuelva a imprimir el arreglo.
   A continuación cree un std::queue q de tipo **dato**, dónde guardará las k distancias ingresadas por el usuario (ingresando k datos a q). Para esto, guarde cada distancia leída en el campo $dist$, dejando los campos val y pos con los valores $M + 1$ y -1 respectivamente. Luego, invloque al método **binarySearchMed**(A, n, med, q) y finalice vaciando la cola e imprimiendo los resultados encontrados por las búsquedas.

2. **[0.5 Ptos.]** Cree la función **float generaEnteros**(int *A, int n), en la cual debe generar n enteros aleatorios en el rango $[-M \dots M]$ guardándolos en A. Calcule y retorne la media aritmética de estos valores.

3. **[2.0 Ptos.]** Cree el método **void insertionSortMed**(int* A, int n, float med). En el cual debe ordenar A descendentemente por la distancia de cada elemento $A[i], 0 \leq i < n$, con la media med. Adapte insertionSort para establecer el orden.

4. **[2.5 Ptos.]** Cree el método **void binarySearchMed**(int* A, int n, float med, queue<dato> &q). Por cada elemento en q, debe realizar una búsqueda biaria para buscar en A algún elemento que esté exactamente a la distancia pedida por el usuario (campo $dist$ del nodo extraido de la cola). Para esto, elemine cada uno de los k nodos que la cola posee y cada vez que la búsqueda tenga éxito reingrese el nodo a la cola seteando el campo val con el elemento encontrado en A y el campo pos con la posición de este valor en A.

Ejemplo de ejecución para $n = 20$ y $k = 3$:

```
$ ./problema 20 3
Arreglo original...
A = -9 -6 -1 9 -2 0 0 -1 5 0 -8 9 10 -6 10 -3 -7 5 6 6
media de A, med = 0.85
Arreglo ordenado por distancias a la media...
A = -9 10 10 -8 9 9 -7 -6 -6 6 6 5 5 -3 -2 -1 -1 0 0 0
Ingrese las k distancias de la media que desea buscar en A...
Distancia 1 : 7
Distancia 2 : 1.85
Distancia 3 : 9.15

Resultado de las búsquedas...
Distancia 1.85 encontrada en pos = 15, para val = -1
Distancia 9.15 encontrada en pos = 1, para val = 10
### Fin Control 1 ###
```

En el fuente dispone de los métodos **insertionSort**() y **binarySearch**() estudiados en clases, los cuales puede utilizar para guiarse en la codificación de los métodos pedidos. Además posee el método **printArray**(int* A, int n), el cual puede utilizar para imprimir su arreglo.
Como dato por si lo requiere, en la <cmath> de la STL está declarada la función **abs**(x) la cuál devuelve el valor absoluto de x.