David Serrano Gemes

Conclusiones practica 3

Se ha realizado el ejercicio 1, encontrar el maximo y el minimo de un tipo conjunto, con la estrategia divide y venceras.

En primer lugar la realización del tipo conjunto se ha realizado utilizando templates. Al dividir la clase en dos archivos, uno cpp y otro hpp, se generó un problema al no ser capaz la clase de instanciarse a los diferentes tipos. Tras investigar soluciones se encontraron dos:

-Hacer instanciaciones especificas de los tipos que se quieren usar en el hpp.

-Incluir toda la clase en el hpp, usando funciones inline.

Se usó esta ultima para poder instanciar a cualquier tipo que se quisiera.

Centrándonos ya en la parte del algoritmo se implementó y se realizaron pruebas sobre un vector de 10000 elementos, con valores entre 1 y 10000.

Además, se implementó la versión iterativa más simple para hacer comparaciones. Debido a que con solo 10000 elementos el tiempo es casi despreciable se modificó el rango a 0-1000000 con un conjunto de 1000000000 elementos.

Usando la clase tiempo que se dio en la practica 1 realizamos distintas observaciones del tiempo requerido para realizar los cálculos.
Se obtuvieron tres tiempos:

-Generación e inicialización del vector: para saber el tiempo que no debemos contabilizar, pues en un primer momento no se tuvo en cuenta este tiempo y falseaba los tiempos del algoritmo recursivo.

-Algoritmo recursivo: tardaba un tiempo ligeramente inferior a la generación del vector.

-Algoritmo iterativo: tardaba casi una tercera parte del tiempo del algoritmo recursivo.

Comparando el numero de comparaciones podemos observar como el algoritmo recursivo realiza más que el iterativo en su versión más simple.

Ante estos datos sacamos la siguiente conclusión: La necesidad de realizar sucesivas llamadas recursivas ralentiza el sistema, pues el proceso de apilamiento de las llamadas tiene un coste computacional añadido. En este algoritmo en concreto, parece que realizar la estrategia divide y vencerás no es rentable, porque la solución iterativa tiene un complejidad muy baja, y la solución recursiva no consigue reducir dicha complejidad, como si ocurre por ejemplo en ordenación de conjuntos grandes con el método Quicksort, el cual funciona mucho mejor en conjuntos grandes.

Por tanto, antes de decidir que algoritmo usar debemos pensar en las distintas soluciones, y analizar qué solución es mas rentable, pues dependerá de diversos factores. Siguiendo el ejemplo del Quicksort, la mejor solucion sería utilizar los algoritmos basicos en conjuntos pequeños y a partir de un limite, utilizar Quicksort.