Javier Aspiroz de la Calle
21957644

Memoria

Presentación del Problema

La cuestión que se nos propone es la ordenación de una lista cuyo contenido son números. Para ello se nos pide hacerlo mediante el algoritmo de ordenación *mergeSort*. La lista que se nos propone ordenar será de la misma dimensión que nuestro número de expediente, en mi caso 21957644.

Resolución

Como ya hemos mencionado antes, en el problema se nos pide ordenar la lista mediante el algoritmo *mergeSort*. Este algoritmo de ordenamiento externo se basa en la técnica de Divide y Vencerás y tiene una eficiencia promedio de $O(n)=nlog(n)$.

Por otra parte, al ser una lista de un tamaño tan grande la ejecución de la forma habitual del código podría eternizarse. Por esta razón, la mejor forma de agilizar el tiempo de ejecución del programa sería realizarlo paralelamente. De esta forma, el trabajo de ejecución estará repartido por todos los núcleos de nuestro ordenador, en vez de solo por uno como está definido por defecto. Es importante recalcar que, el tiempo de ejecución se verá reducido pero complejidad no cambiaría debido a que el algoritmo a realizar es el mismo y solo se añaden unos bloques de código que se encargan de repartir las tareas por núcleo.

Algoritmo

Siguiendo los pasos del ejercicio realizado en clase sobre la multiplicación de matrices y lo aprendido en la asignatura de Técnicas de Programación Avanzada, donde aprendimos a implementar y usar *mergeSort*. Lo que hacemos para implementar el programa es dividir el programa en tres métodos:

- ***mergeSort y merge***: donde se codifica el algoritmo propiamente dicho y el procesador se hace cargo de ordenar la lista en su rango asignado
- ***mergeSortParalelo***: .donde dividimos y repartimos el trabajo y parte de la lista que deberá ordenar cada procesador.

Finalmente, en el ***main*** se crea una lista de entero entre -9999 y 9999 de tamaño 21957644 y se llama al método ***mergeSort*** y ***mergeSortParalelo*** diciendo el tiempo que tarda en ejecutarse.

Adicionalmente, también incluyo otro código que realiza el mismo trabajo en el archivo ***usingProcesses.py.*** Este código fue adaptado para realizar lo que requería la actividad en la fase de investigación. Su implementación se basa también los mismos métodos que el principal. Difiere principalmente en la forma en que son generados los procesos para la paralelización, ya que ograma lo hacemos mediante el uso de **Pipes**.

Para concluir, quería resaltar que la paralelización de *mergeSort* solo tiene sentido si se va a hacer para lista de un gran tamaño, ya que como muestra la ejecución con tamaños más pequeños, la ejecución en paralelo demora más tiempo que en secuencial. Esto se debe a que la creación de los procesos y asignación de su trabajo consume un pequeño espacio de tiempo.