RELAZIONE ES 1

Laboratorio di Algoritmi e Strutture Dati

[Group collaborator] [redacted IDs]
Guiot Paolo [redacted IDs]

L'esercizio 1 del progetto di laboratorio di Algoritmi e Strutture Dati richiede di verificare come si comportano gli algoritmi di Binary Insertion Sort e Quicksort in vari casi. Abbiamo eseguito test su vari tipi di dati e, nel caso del Quicksort, abbiamo anche sperimentato con alcuni tipi di pivot differenti.

Per il Binary Insertion Sort, che ha complessità $O(n^2)$, ci aspettavamo che l'algoritmo non sarebbe terminato in meno di 10 minuti utilizzando il file fornito con 20 milioni di righe.
Abbiamo deciso di eseguirlo su un numero molto ridotto di righe del file originale (solo le prime 100.000 del file originale) per mostrare che termina con dei buoni tempi. Ecco i risultati per l'ultima run dell'algoritmo.

int	**double**	**string**
8.97s	8.47s	8.53s

Durante la fase di testing abbiamo provato ad eseguire stampe a schermo per tenere traccia dello stato dell'algoritmo di sorting. È stato interessante notare quanto fosse effettivamente lenta la sua esecuzione, che ha reso ancora più chiaro il significato della complessità dell'algoritmo.

Per il Quicksort, che ha complessità O(n log(n)), siamo riusciti ad ottenere dei buoni risultati anche utilizzando il file originale con 20 milioni di righe.
Inizialmente abbiamo trovato difficoltà a gestire le stringhe, dato che andavano a riempire lo stack, ma siamo riusciti a risolvere questo problema utilizzando il partition di Hoare, che è molto più leggero e veloce.
Ci aspettavamo di trovare il pivot centrale come migliore.
Di seguito mostriamo una tabella che contiene i tempi ottenuti durante l'ultima run del codice in base a pivot e tipo di dato da ordinare.

Pivot\Tipo	int	double	string
sinistro	4.54s	5.23s	18.31s
centrale	4.38s	5.15s	17.73s
destro	4.56s	5.22s	18.19s
random	22.26s	22.69s	22.27s

Pivot sinistro, centrale e destro non mostrano grandi differenze fra loro, ma notiamo che il pivot centrale sembra effettivamente essere il migliore in tutte e tre le categorie.
Notiamo anche che il pivot random è significativamente peggiore rispetto agli altri; ciò che peggiora maggiormente quel modo di partizionare i dati è però probabilmente anche dovuto al fatto che vengono eseguite alcune operazioni lente in più, come il calcolo della posizione casuale, rallentando quindi significativamente l'esecuzione dell'algoritmo nel complesso.