Ampala Alessandro
Borrelli Mattia

Relazione su esercizio 1

In riferimento all'applicazione sviluppata per l'utilizzo ed il confronto tra i due algoritmi di ordinamento, Insertion e Quick Sort, sono stati osservati dei risultati tutto sommato prevedibili.

L'esperimento si è svolto utilizzando come input un file contenente 20 milioni di record, ciascuno composto da un identificatore univoco e tre campi contenenti dati di tipo diverso. L'applicazione prevede di ordinare i record secondo il contenuto di uno di questi tre campi, utilizzando uno degli algoritmi proposti. Tale scelta va specificata all'avvio del programma. Viste le scelte implementative prese, a ciascun ordinamento corrisponde dunque un'esecuzione dell'applicazione. Ai fini della valutazione, un ordinamento è da considerarsi fallito nel caso in cui siano superati i 10 minuti di esecuzione.

I risultati ottenuti sono riportati nella seguente tabella:

	Insertion Sort	**Quick Sort**
Char (field1)	Fallito	23 secondi
Int (field2)	Fallito	9 secondi
Float (field3)	Fallito	10 secondi

Dalla tabella si evince subito come l'ordinamento tramite Insertion Sort non è andato a buon fine in nessun dei casi previsti. Tuttavia, si tratta di un fallimento abbastanza prevedibile, vista la complessità dell'algoritmo preso in esame. Infatti, considerando n come il numero degli elementi (in questo caso n = 20 000 000) Insertion Sort prevede un costo pari a $O(n^2)$, sia nel caso peggiore che nel caso medio, dimostrandosi estremamente inefficiente per ordinare un input composto da così tanti elementi.

D'altro canto, il Quick Sort ha complessità variabile che dipende dalla scelta del pivot, cioè l'elemento intorno al quale si esegue il partizionamento, a sua volta influenzato dall'ordinamento relativo degli elementi, piuttosto che dai particolari valori di questi. Il partizionamento avviene scegliendo il primo elemento come pivot e spostandolo in modo tale che a sinistra ci siano gli elementi minori di quest'ultimo ed a destra quelli maggiori. Ciò viene fatto effettuando opportuni scambi e tramite l'uso di due indici i = 2 e j = length(Array) che, quando assumono lo stesso valore, identificano la posizione corretta del pivot. In questi termini, il posizionamento del pivot richiede quindi una scansione dell'array, che rende il partizionamento di complessità lineare O(n). A determinare la complessità totale sarà l'albero generato dalle chiamate ricorsive, poiché la sua altezza rappresenta quante volte è stata effettuata una ripartizione. Il caso migliore, corrispondente al bilanciamento massimo (n/2) per ogni partizionamento, prevede complessità pari a O(nlogn). Si può dimostrare che tale stima è valida anche per qualunque altra ripartizione con proporzionalità costante, poiché la profondità dell'albero è $\Theta(\log n)$ (dove il costo di ogni livello è quello del partizionamento, cioè O(n)). Il caso peggiore prevede invece uno sbilanciamento massimo del

partizionamento ad ogni chiamata ricorsiva, il cui tempo di esecuzione è $\Theta(n^2)$, uguale ad Insertion Sort. Tuttavia, in una casistica media, è lecito pensare che si avranno sia partizioni bilanciate che sbilanciate. Possiamo aspettarci che tali partizioni appariranno casualmente all'interno dell'albero di ricorsione, ma supponendo di alternare partizionamenti migliori e peggiori, intuitivamente si può dire che il costo del partizionamento migliore bilancerà quello peggiore, generando comunque un tempo di esecuzione pari a come se ci fossero solo partizioni buone, a meno di una costante un po' più grande (l'albero è più profondo del caso bilanciato) nascosta però dalla O. Inoltre, tale ordine di grandezza è anche da considerarsi ottimo per il problema dell'ordinamento. Il caso medio del Quick Sort è quindi molto più vicino a quello migliore, pari a O(nlog n), e ciò permette di ottenere risultati di gran lunga migliori rispetto ad una soluzione dalla complessità quadratica.

Un'altra ipotesi fatta a monte e poi confermata dall'esecuzione riguarda invece l'ordinamento basato sul field 1, cioè quello contenente stringhe. Era atteso infatti un tempo leggermente superiore rispetto ad altri campi, da attribuirsi al costo maggiore della funzione di comparazione tra due stringhe rispetto a quelle per numeri interi o floating point. Tale ipotesi è stata comprovata durante l'esecuzione del Quick Sort, come riportato in tabella.