Gabriele Palazzolo (Mat: 883434) - Michela Rivautella (Mat: 881496)

Laboratorio di Algoritmi e Strutture Dati - Corso B

	Caso peggiore temporalmente	**Caso medio temporalmente**	**Caso ottimo temporalmente**
Insertion sort	$O(n^2)$	$O(n^2)$	$O(n)$
Quick sort	$\Theta(n^2)$	$\Theta(n\log_2 n)$	$\Theta(n\log_2 n)$

MACCHINA UTILIZZATA

Modello: MacBook Pro 13-inch, 2017
Processore: 2,3 GHz Intel core i5 Dual-core
Memoria: 8 GB 2133 MHz LPDDR3

MODALITA' ORDINAMENTO

Per ogni algoritmo di sorted si effettuano tre ordinamenti consecutivi con riferimento allo stesso array di records, ma secondo differente campo. In più, ogni ordinamento si basa su quello precedente, ovvero l'array ordinato secondo il primo campo è l'input di ordinamento per la chiamata alla stessa funzione sul secondo campo.
Quindi i tempi presi in considerazione si riferiscono a chiamate di una stessa funzione per tre volte.

ANALISI DEL QUICK SORT

L'ordinamento di 20 milioni di records presi in input dal file *records.csv*, tramite questo algoritmo, permette di ottenere una risposta dopo 10 minuti.

Questo tempo di risposta è attendibile in quanto si sono effettuati una serie di test ottenendo i seguenti risultati:

Numero records ordinati	**Tempo impiegato**
10000	0.007393 s
100000	0.151794 s
1000000	9.810525 s
10000000	+5 min
20000000	fallimento

Dai risultati riportati in tabella si conferma la veridicità della complessità asintotica di questo algoritmo.

Infatti effettuando il calcolo della complessità asintotica per:

n = 10000:

- caso peggiore: $\Theta(n^2) = \Theta(10000^2) = \Theta(10^7)$
- caso intermedio: $\Theta(n\log_2 n) = \Theta(10000\log_2 10000) = \Theta(132877.12)$

Questi calcoli valgono per l'esecuzione dell'algoritmo di una sola volta, quindi a meno di una costante moltiplicativa $c = 3$. Volendo considerare tale costante, per ottenere la complessità effettiva relativa ai tempi di risposta ottenuti in tabella, vale la proprietà della riflessività, dunque si ha che: $cf(n) = \Theta(n)$ e dunque:

- caso peggiore: $c\Theta(10^7) = \Theta(10^7 c)$
- caso intermedio: $c\Theta(132877.12) = \Theta(132877.12c)$

n = 1000000:

- caso peggiore: $\Theta(n^2) = \Theta(1000000^2) = \Theta(10^{11})$
- caso intermedio: $\Theta(n\log_2 n) = \Theta(1000000\log_2 1000000) = \Theta(19931568.57)$

Considerando la constante c = 3 allora:

- caso peggiore: $c\Theta(10^{11}) = \Theta(10^{11}c)$
- caso intermedio: $c\Theta(19931568.57) = \Theta(19931568.57c)$

Per n = 20000000 considerando il caso intermedio e quello peggiore (a meno della costante moltiplicativa c) si otterrebbe che:

$\Theta(1000000\log_2 1000000) < \Theta(1000000^2) << \Theta(2000000\log_2 2000000) < \Theta(2000000^2)$

Dunque i tempi di risposta crescono, anche se in modo $n\log_2 n$, molto velocemente portandoli per *n* piccolo da pochi secondi a molti minuti per *n* grande.

ANALISI DELL'INSERTION SORT

L'ordinamento di 20 milioni di records presi in input dal file *records.csv*, tramite questo algoritmo, permette di ottenere una risposta dopo 10 minuti.

Questo tempo di risposta è attendibile in quanto si sono effettuati una serie di test ottenendo i seguenti risultati:

Numero records ordinati	Tempo impiegato
10000	0.542238 s
100000	101.647476 s
1000000	+10 min (fallimento)
10000000	+10 min (fallimento)
20000000	fallimento

Dai risultati riportati in tabella si conferma la veridicità della complessità asintotica di questo algoritmo.

Infatti effettuando il calcolo della complessità asintotica per:

n = 10000:

- caso peggiore: $O(n^2) = O(10000^2) = O(10^7)$
- caso intermedio: $O(n^2) = O(10000^2) = O(10^7)$

Questi calcoli valgono per l'esecuzione dell'algoritmo di una sola volta, quindi a meno di una costante moltiplicativa $c = 3$. Volendo considerare tale costante, per ottenere la complessità effettiva relativa ai tempi di risposta ottenuti in tabella, vale la proprietà della riflessività, dunque si ha che: $cf(n) = O(n)$ e dunque:

- caso peggiore: $cO(10^7) = O(10^7c)$
- caso intermedio: $cO(10^7) = O(10^7c)$

n = 1000000:

- caso peggiore: $O(n^2) = O(1000000^2) = O(10^{11})$
- caso intermedio: $O(n^2) = O(1000000^2) = O(10^{11})$

Considerando la constante c = 3 allora:

- caso peggiore: $cO(10^{11}) = O(10^{11}c)$
- caso intermedio: $cO(10^{11}) = O(10^{11}c)$

Per n = 20000000 considerando il caso intermedio equivalente a quello peggiore (a meno della costante moltiplicativa c) si otterrebbe che:

$O(1000000^2) << O(20000000^2)$

Dunque i tempi di risposta crescono esponenzialmente (n^2) e quindi in modo molto veloce portandoli per n piccolo da pochi secondi a n grande a molti minuti.

QUICK SORT VS INSERTION SORT

Considerando una quantità di valori piccola ($n \leq 10000$), vi è una differenza di tempi di risposta nell'ordine dei decimi di secondo e a parità di caso peggiore la differenza è quasi nulla, quindi gli algoritmi si equivalgono.

Nonostante entrambi gli algoritmi abbiamo una complessità asintotica che nei casi intermedi e peggiori per quantità di valori ordinati grandi ($n \geq 1000000$), porta a tempi di risposta che superano i 5-10 minuti, risulta evidente che resta comunque migliore l'algoritmo di *Quick sort*.

Nel caso dell'ordinamento di tutti i records (per 3 volte, secondo uno specifico campo) e quindi di 20 milioni di dati la complessità asintotica risulta:

	Caso peggiore temporalmente	**Caso medio temporalmente**	**Caso ottimo temporalmente**
Insertion sort	$O(20000000^2)$	$O(20000000^2)$	$O(20000000^2)$
Quick sort	$\Theta(20000000^2)$	$\Theta(20000000\log_2 20000000)$	$\Theta(20000000\log_2 20000000)$