Algoritmi di Ordinamento

Mario Ambrosino

27 Giugno 2018

Dato il carattere algoritmico (più che numerico) di questa esercitazione, riporto semplicemente l'indirizzo della pagina GitHub per evitare di dilungarmi.

`https://github.com/mario-ambrosino/sort`

Questa volta ho lavorato con un solo sorgente, `sort.c`, linkato alla libreria statica `lib_sort.c`, entrambi commentati adeguatamente e basati sulle librerie personali scritte per le passate relazioni. Gli algoritmi di ordinamento sono presenti all'interno della libreria. Ho scelto i seguenti algoritmi di ordinamento:

1. `Bubble-Sort`
2. `Insertion-Sort`
3. `Shell-Sort`
4. `Quick-Sort`
5. `Merge-Sort`

Da notare che i numeri identificano in modo univoco la modalità di funzionamento all'interno del codice della funzione `sort`, che ordina seguendo uno degli algoritmi presentati.

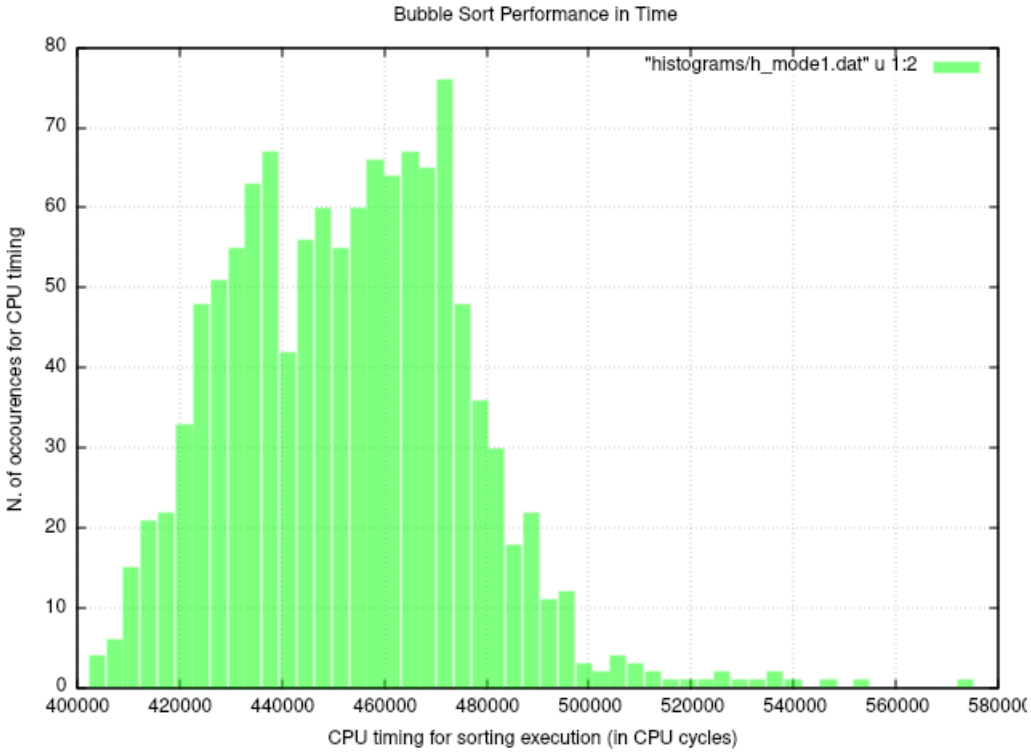
Il programma fornisce in output la sequenza ordinata.

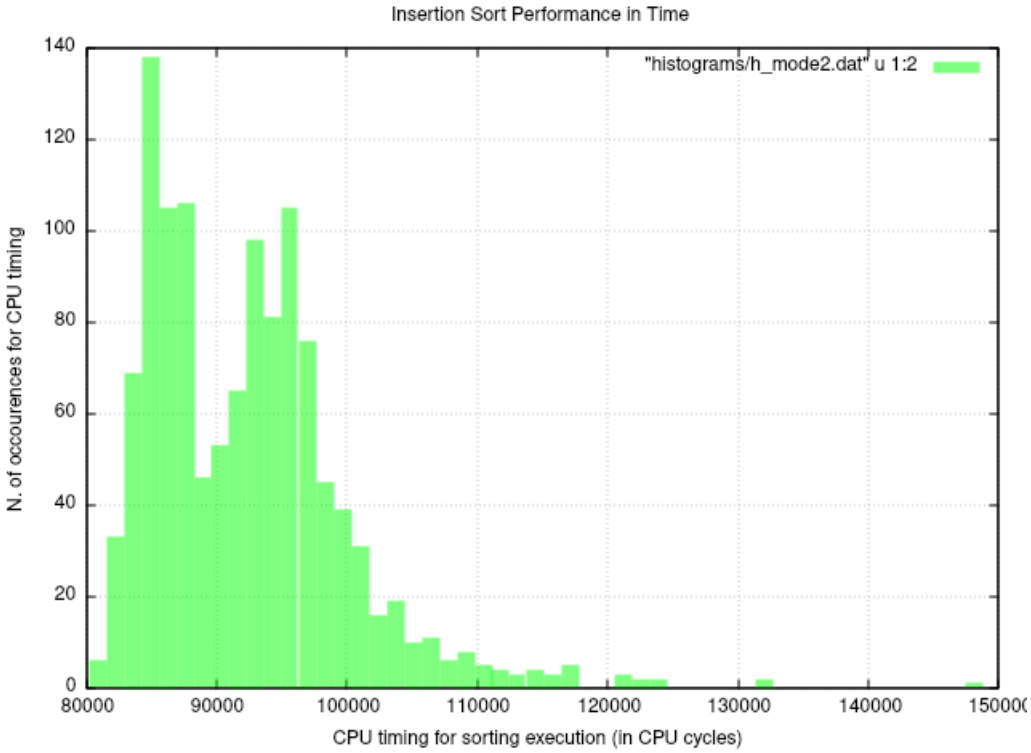
Eseguendo lo script `cfr.sh` si generano 1000 sequenze di 10000 numeri casuali distribuiti in modo uniforme. Lo script poi ordina per ogni algoritmo le sequenze, fornendo nello standard output il modo di funzionamento, il numero di "passi" effettuato e il numero di cicli intercorsi durante l'esecuzione.

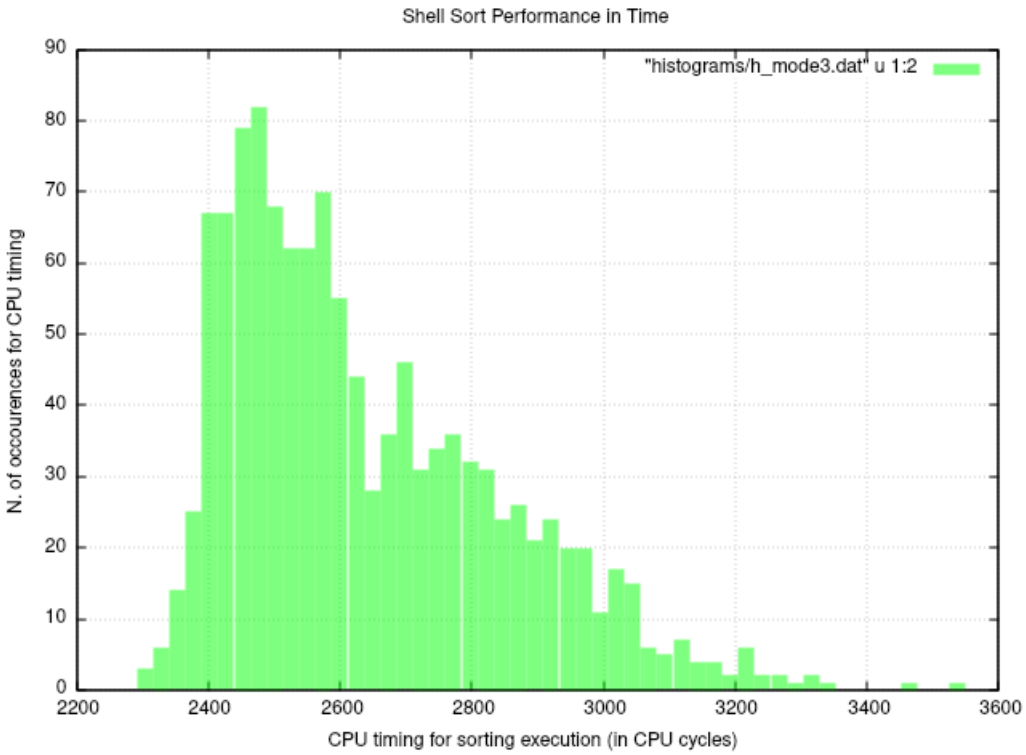
Lo script `second_cfr.sh` invece estrae tramite il comando `gawk` le colonne di "passi" e "cicli" per ogni modo di ordinamento e chiama un'eseguibile (già visto nella seconda tesina) che genera l'istogramma associato.

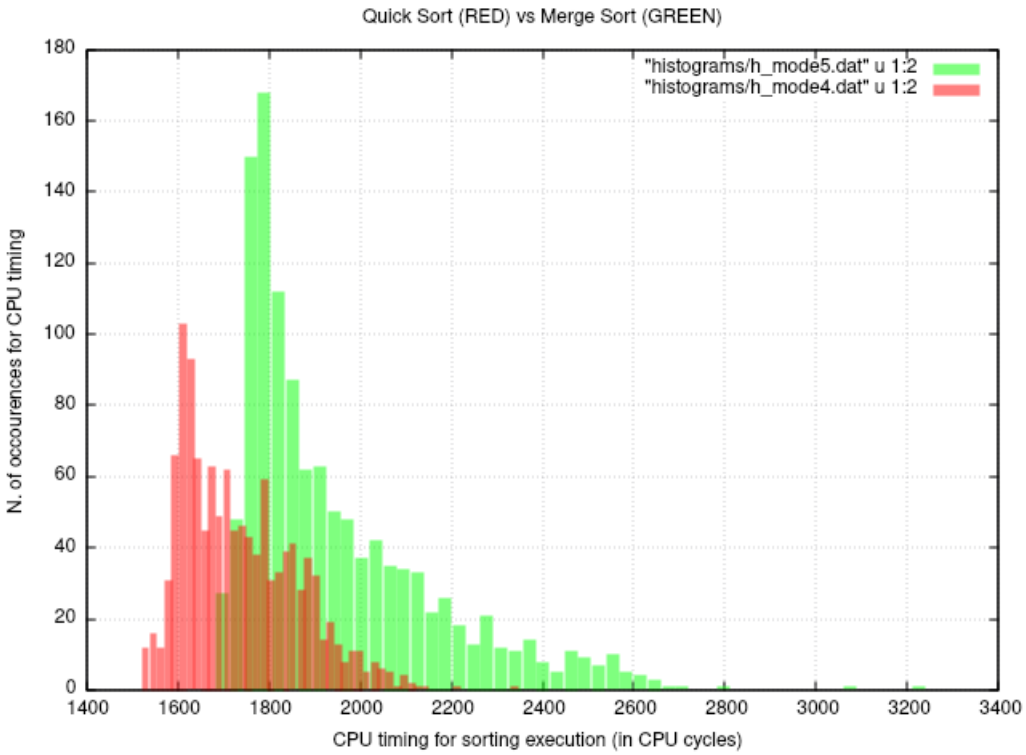
L'istogramma dei tempi per ognuna assume una forma simile ad una distribuzione poissoniana, come c'è da aspettarsi. Notiamo che i tempi di `Quick Sort` e `Merge Sort` si sovrappongono, e sono i migliori per sequenze lunghe. Naturalmente il metodo è piuttosto "quick and dirty", si potrebbe migliorare il conteggio dei passi contando separatamente i confronti e gli accessi alla RAM.

Allego immagini dei tempi.


Bubble Sort Performance in Time
"histograms/h_mode1.dat" u 1:2
N. of occourences for CPU timing
CPU timing for sorting execution (in CPU cycles)
0
10
20
30
40
50
60
70
80
400000
420000
440000
460000
480000
500000
520000
540000
560000


Insertion Sort Performance in Time
"histograms/h_mode2.dat" u 1:2
N. of occourences for CPU timing
CPU timing for sorting execution (in CPU cycles)
0
20
40
60
80
100
120
140
80000
90000
100000
110000
120000
130000
140000


Shell Sort Performance in Time
"histograms/h_mode3.dat" u 1:2
N. of occourences for CPU timing
CPU timing for sorting execution (in CPU cycles)
0
10
20
30
40
50
60
70
80
90
2200
2400
2600
2800
3000
3200
3400
3600


Quick Sort (RED) vs Merge Sort (GREEN)
"histograms/h_mode5.dat" u 1:2
"histograms/h_mode4.dat" u 1:2
N. of occourences for CPU timing
CPU timing for sorting execution (in CPU cycles)
0
20
40
60
80
100
120
140
160
180
1400
1600
1800
2000
2200
2400
2600
2800
3000
3200
3400