Autore: Daniele La Fauci Matricola 898389

Relazione esercizio 1

La consegna richiedeva di implementare l'algoritmo di Merge-Binary-Insertion-Sort, il quale si occupa di ordinare un array, ottenuto dalla lettura di un file, utilizzando il Merge-Sort o il BinaryInsertionSort in base ad un valore k arbitrario.

Risultati:

K = 0:
Campo String: 25.235 secondi
Campo Int: 20.374 secondi
Campo float: 19.344 secondi

K = 10:
Campo String: 19.828 secondi
Campo Int: 14.969 secondi
Campo float: 16.172 secondi

K = 25:
Campo String: 19.094 secondi
Campo Int: 15.359 secondi
Campo Float: 15.749 secondi

K = 50:
Campo String: 20.609 secondi
Campo Int: 16.514 secondi
Campo Float: 16.671 secondi

K = 100:
Campo String: 19.172 secondi
Campo Int: 15.874 secondi
Campo Float: 16.578 secondi

K = 250:

Campo String: 20.032 secondi
Campo Int: 16.157 secondi
Campo Float: 16.968 secondi

K = 500
Campo String: 22.016 secondi
Campo Int: 17.625 secondi
Campo Float: 19.611 secondi

K = 1000
Campo String: 24.142 secondi
Campo Int: 21.844 secondi
Campo Float: 21.718 secondi

K = 1500:
Campo String: 28.093 secondi
Campo Int: 25.063 Secondi
Campo Float: 26.625 secondi

K = 2500:
Campo String: 39.999 secondi
Campo Int: 35.218 secondi
Campo Float: 35.375 secondi

K = 3000:
Campo String: 40.579 secondi
Campo Int: 35.032 secondi
Campo Float: 36.359 secondi

K = 10000
Campo String: 93.203 secondi
Campo Int: 91.485 secondi
Campo float: 93.594 secondi

Dai risultati ottenuti si può notare come un k ottimale si aggira in un intervallo compreso tra il 25 ed il 100, dunque c'è un'ottima collaborazione tra i due algoritmi. I tempi iniziano ad essere più lunghi con un K > 1500, diventando molto inefficiente con un K >= 10000 dal momento che il Binary Insertion Sort inizia ad essere più svantaggioso e a richiedere tempi più lunghi su array di dimensioni grandi, al contrario del Merge Sort che risulta molto più vantaggioso.