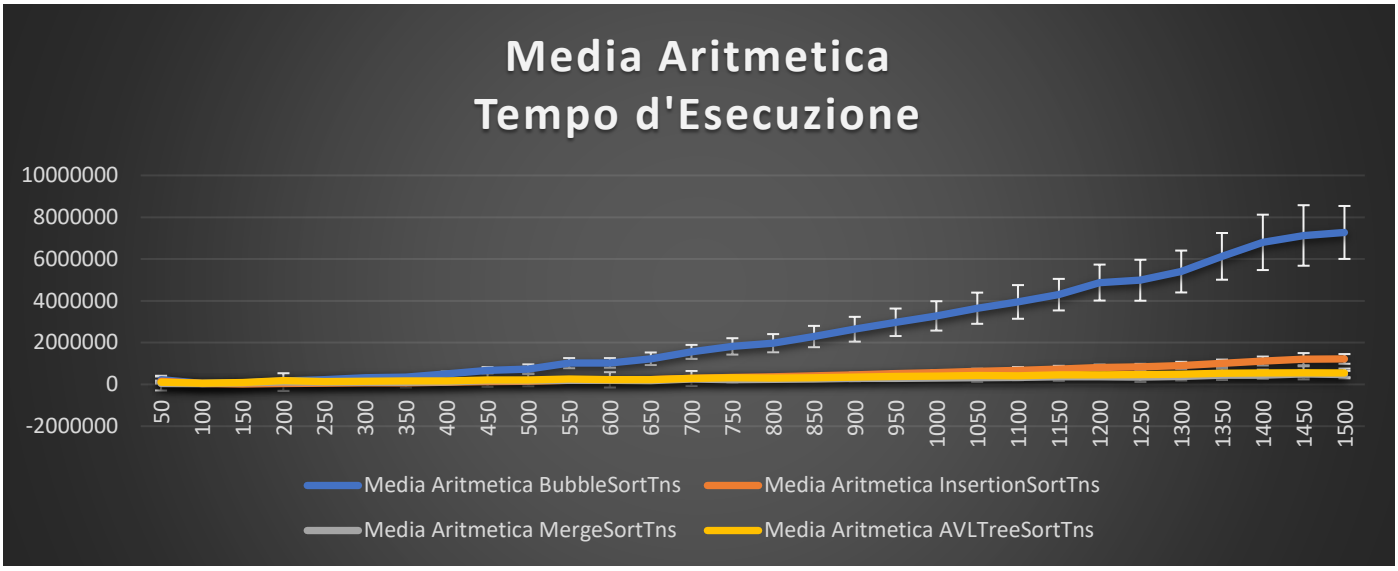
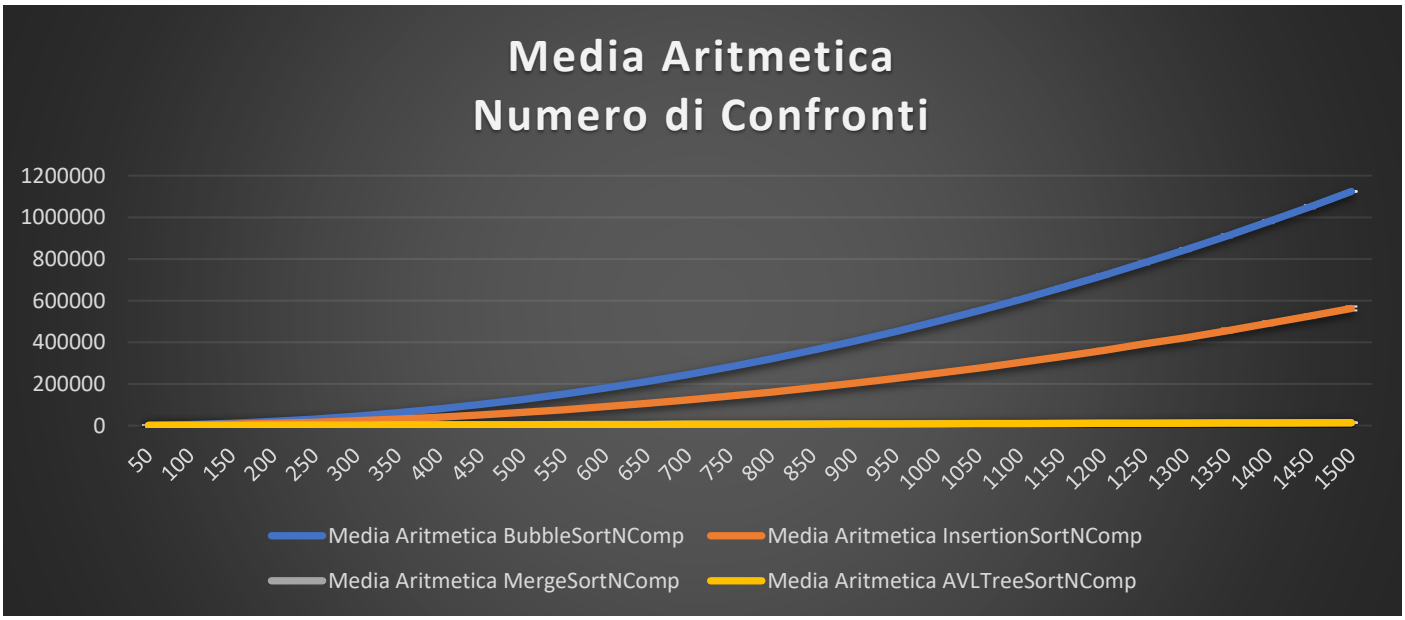
RELAZIONE DI ALEX CITERONI

Caso Medio con Deviazione Standard

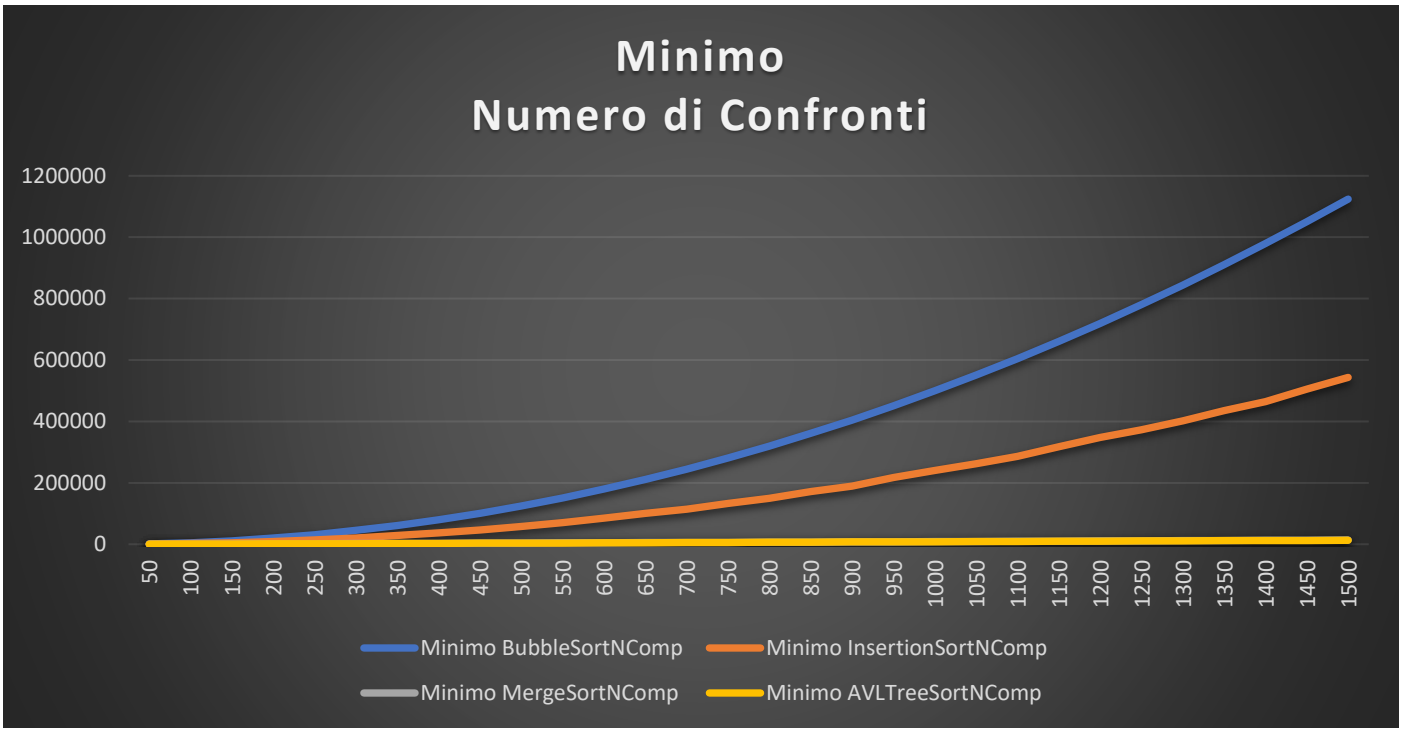


Questo grafico rappresenta il caso medio e possiamo osservare l'algoritmo di ordinamento Bubble Sort (con complessità nel caso medio $O(n^2)$) che ha tra l'altro anche un alto margine d'errore (deviazione standard) e che impiega più tempo rispetto agli altri algoritmi di ordinamento, troviamo l'Insertion Sort subito sotto (anche qui con complessità $O(n^2)$) con minor margine d'errore; infine abbiamo il Merge Sort (con complessità, in questo caso, $O(n\log_2 n)$) e l'AVL Tree Sort allo stesso livello, con un basso margine d'errore in entrambi. Quindi possiamo dire che i più performanti sono stati, in questo caso, gli ultimi due algoritmi di ordinamento cioè il Merge Sort e l'AVL Tree Node.

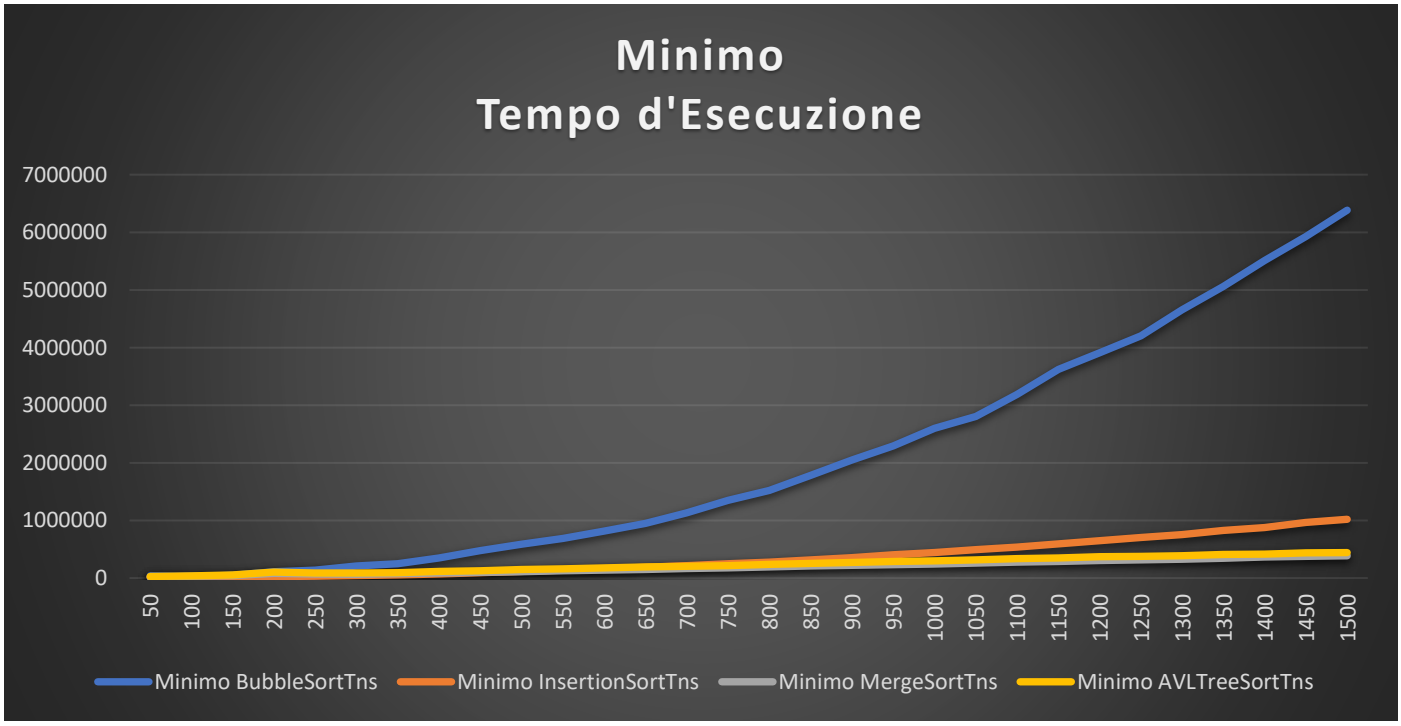


Anche qui possiamo trovare delle similitudini al grafico di sopra (per esempio, l'andamento molto lineare di tutti gli algoritmi), il Bubble Sort fa più confronti rispetto a tutti gli altri algoritmi, troviamo l'Insertion Sort subito sotto e in fondo troviamo il Merge Sort e AVL Tree Sort che vanno a pari passo. Tutti hanno più o meno gli stessi margini di errore.

Caso Ottimo



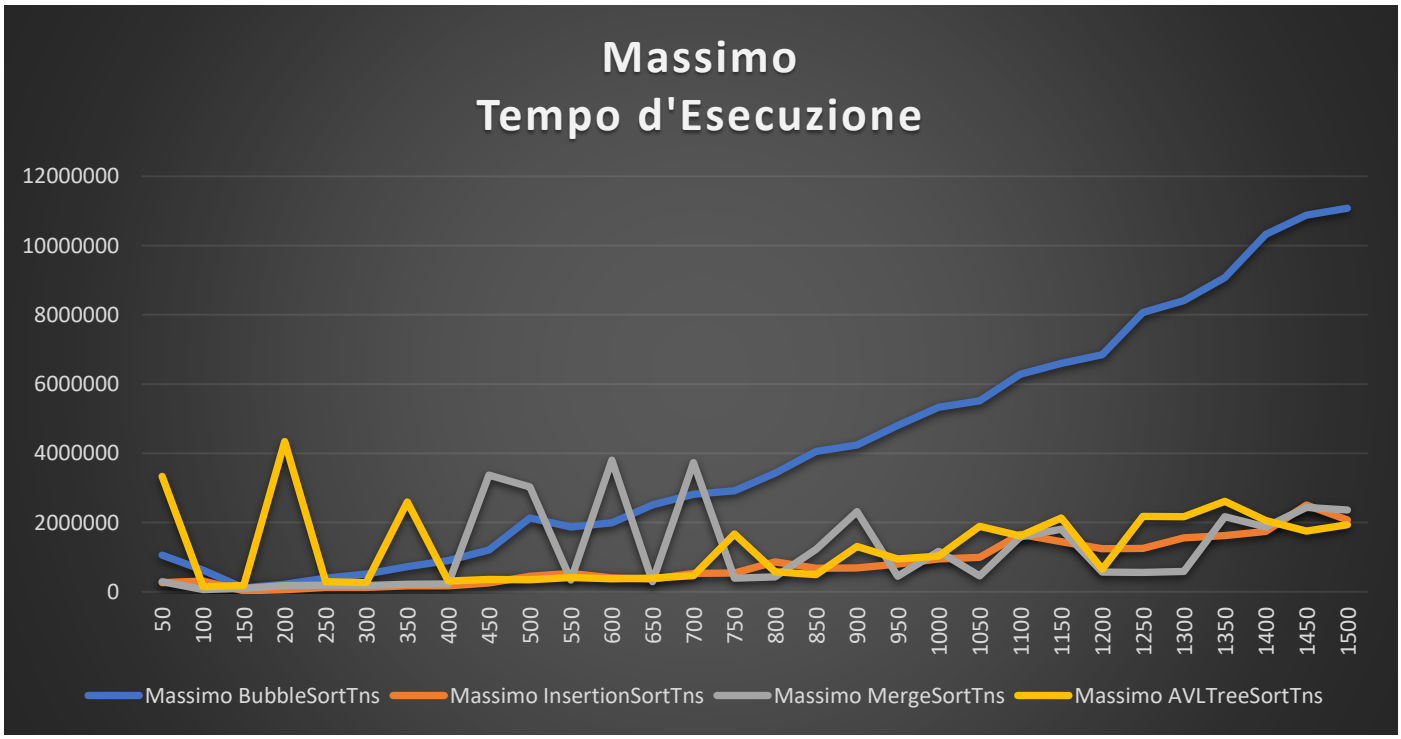
Nel caso ottimo possiamo notare che il Bubble Sort (con complessità computazionale O(n)) è l'algoritmo che fa più confronti tra tutti, il secondo algoritmo che fa più confronti è l'Insertion Sort (con complessità computazionale O(n)), in fondo troviamo di nuovo il Merge Sort (con complessità computazionale $O(n\log_2 n)$) e l'AVL Tree Sort sempre sullo stesso livello.



Il Bubble Sort è sempre in vetta anche nel tempo d'esecuzione del caso ottimo, quindi anche qui è l'algoritmo che richiede più tempo di tutti per terminare. Sotto possiamo trovare quasi a pari livello gli altri tre algoritmi di ordinamento, tra qui l'Insertion Sort che ha bisogno di un po' più di tempo rispetto al Merge Sort e all'AVL Tree Sort.

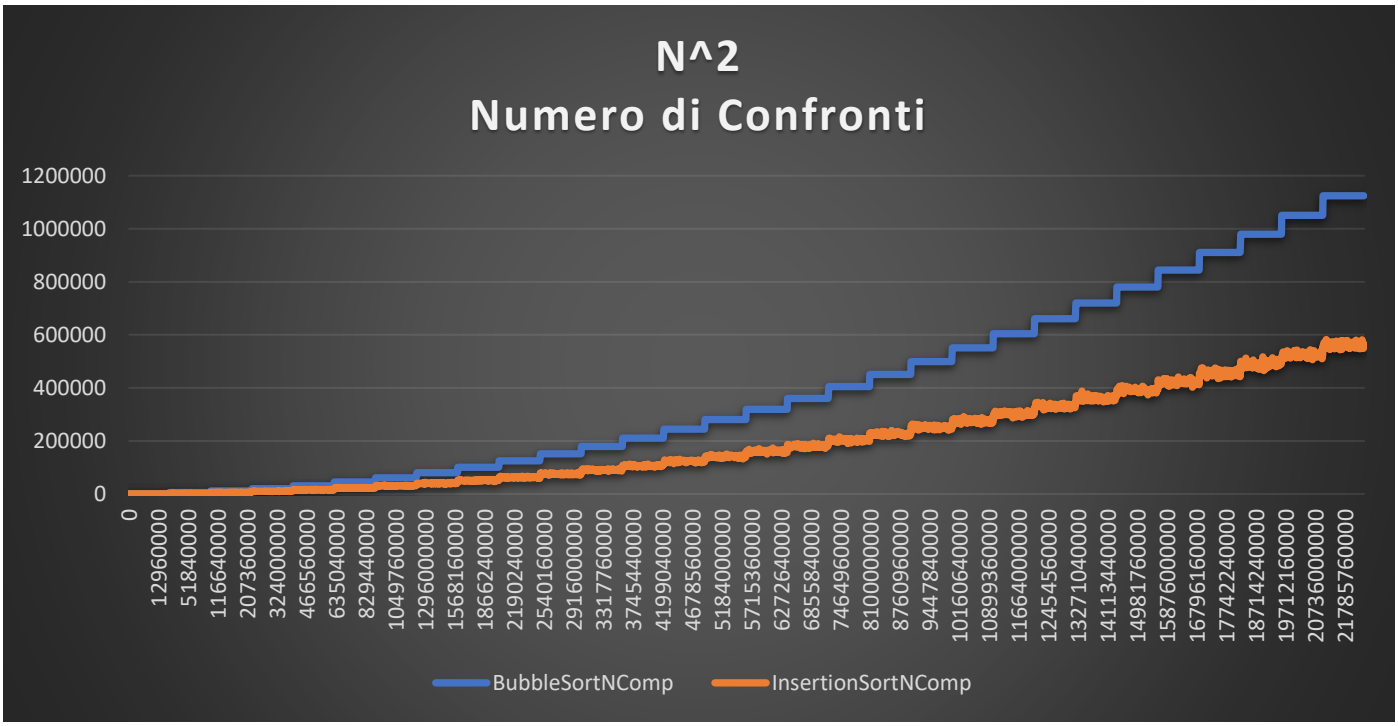
Caso Pessimo

Anche nei confronti del caso pessimo, il Bubble Sort (con complessità computazionale $O(n^2)$) resta in testa con il numero più alto di confronti, seguito dall'Insertion Sort (con complessità computazionale $O(n^2)$). Infine, abbiamo sullo stesso livello il Merge Sort (con complessità computazionale $\mathrm{O}(\mathrm{nlog}_2\, n)$) e l'AVL Tree Sort.

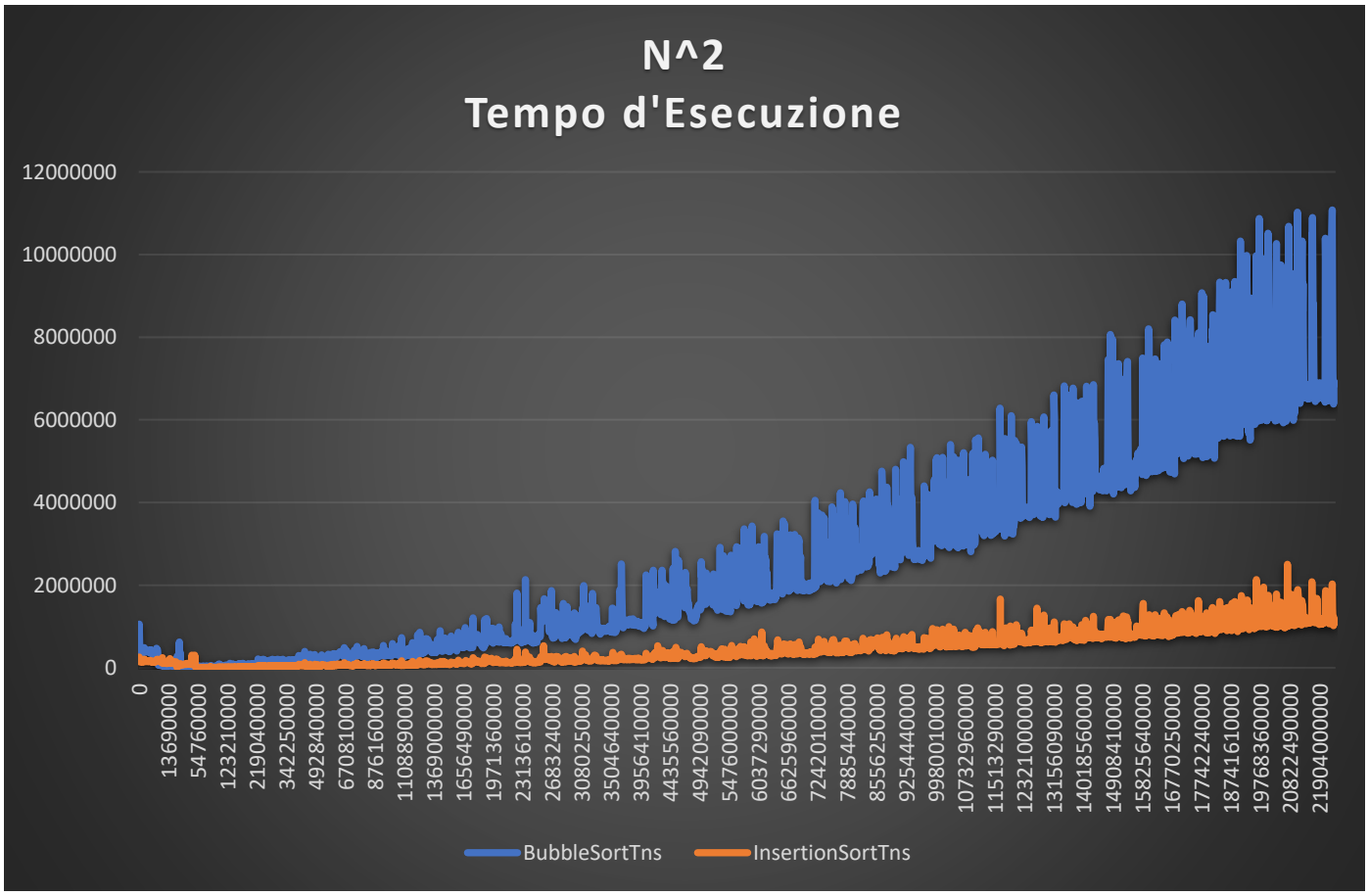


Il tempo d'esecuzione in questo caso è curioso: all'inizio l'AVL Tree Sort subisce diversi picchi di rallentamento, ma poi viene superato di molto dal Bubble Sort, che parte più velocemente, ma poi rallenta di molto. L'Insertion Sort in questo caso è stato il più rapido con nessun picco di rallentamento, mentre il Merge Sort nel mezzo ha avuto dei cali di prestazioni ma poi ha recuperato anche lui velocità.

$$O(n^2)$$

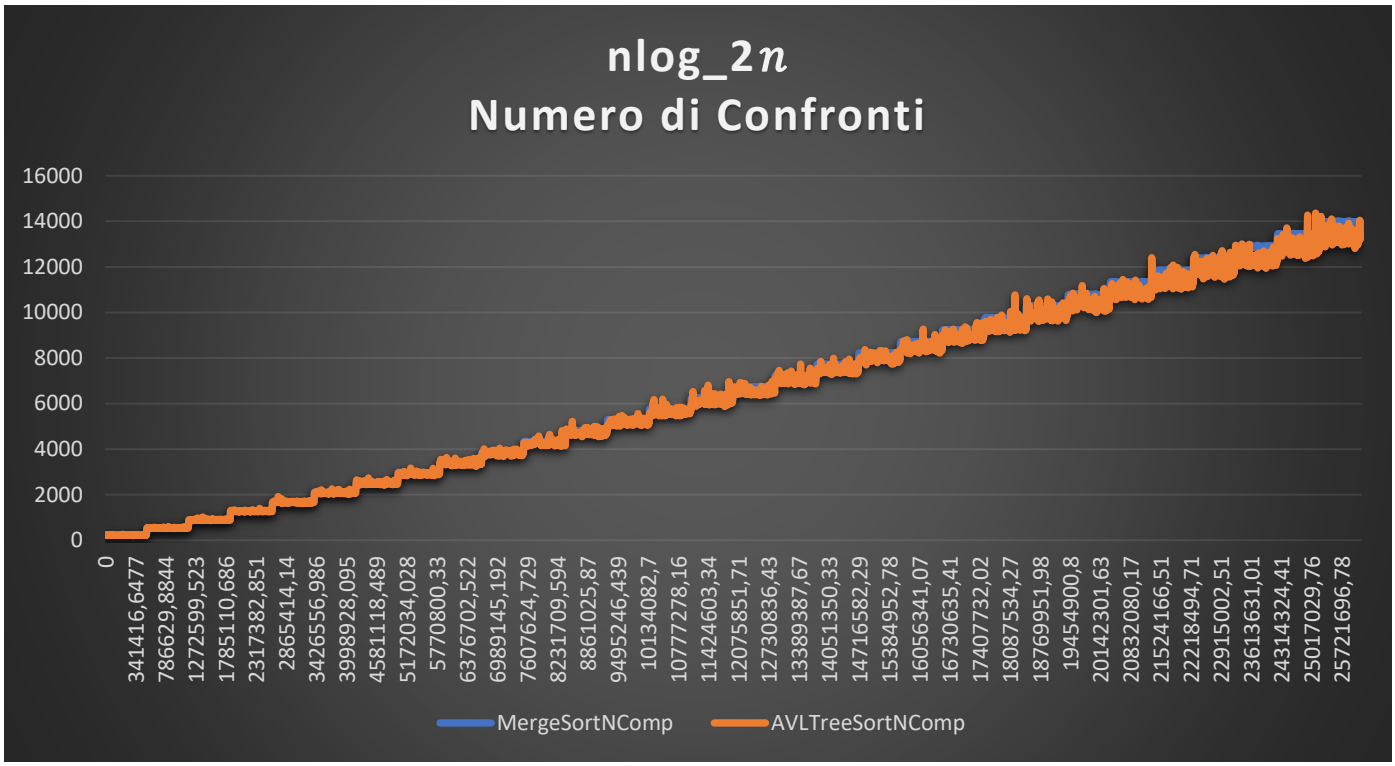


Nel caso medio e pessimo dell'Insertion Sort e del Bubble Sort con il costo computazionale pari a $O(n^2)$ possiamo notare come il numero di confronti nel Bubble Sort sia molto più alto rispetto a quello fatto per l'Insertion Sort.

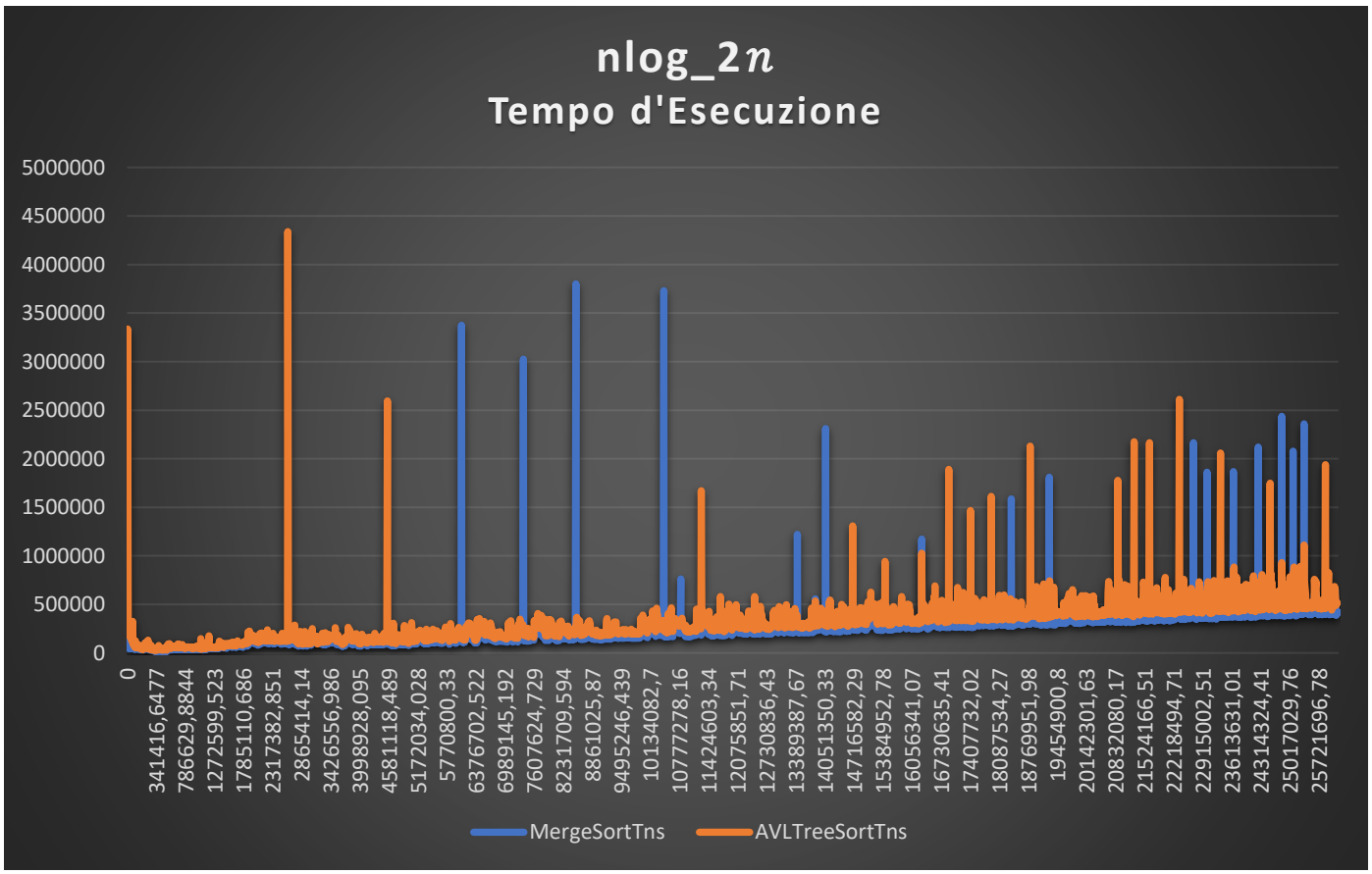


Anche qui possiamo notare che il tempo d'esecuzione del Bubble Sort è superiore a quello dell'Insertion Sort.

$O(n\log_2 n)$



In tutti i casi gli algoritmi Merge Sort e AVL Tree Sort hanno un costo computazionale pari a $O(n\log_2 n)$ e sono costantemente entrambi sullo stesso livello per quanto riguarda il numero di confronti.



Per quanto riguarda il tempo d'esecuzione l'AVL Tree Sort subisce molti più rallentamenti rispetto al Merge Sort.