Relazione sull'Algoritmo di Ordinamento Binary Merge Insertion Sort

Introduzione

L'obiettivo di questa relazione è valutare l'algoritmo di ordinamento Binary Merge Insertion Sort, analizzando le sue prestazioni in relazione al parametro `k`, che rappresenta la dimensione dell'array per cui viene utilizzato il Binary Insertion Sort invece del Merge Sort. Il dataset di input è composto da un file contenente 20 milioni di record, ciascuno con tre campi: una stringa, un intero e un numero in virgola mobile (float). Questa relazione esamina i risultati ottenuti al variare di `k`, con un'attenzione particolare al confronto delle complessità algoritmiche dei due algoritmi, Binary Merge Insertion Sort e Merge Sort.

Risultati dei Test

I risultati dei test sono i seguenti:

- Con `k = 0`, cioè senza l'utilizzo del Binary Insertion Sort, i tempi di esecuzione sono ragionevolmente veloci, nell'ordine dei 20-30 secondi.
- Aumentando `k` a 5000, i tempi di esecuzione rimangono accettabili, oscillando tra i 30-70 secondi. In questo caso, poiché gli array sono relativamente piccoli rispetto al totale, l'effetto sull'ordinamento dei campi interi e in virgola mobile è trascurabile, con solo piccole variazioni nei tempi di chiusura.
- Con `k = 50000`, i tempi di esecuzione crescono significativamente, superando i 2-5 minuti. Inizia a emergere il peso del Binary Insertion Sort nell'algoritmo.
- A `k = 150000`, si registra un notevole aumento dei tempi di esecuzione, specialmente per il campo stringa, con tempi superiori ai 10 minuti.
- Con `k = 300000`, il Binary Merge Insertion Sort riesce a completare l'ordinamento solo per i campi interi e in virgola mobile, mentre il campo stringa richiede notevolmente più tempo, ben oltre i 10 minuti.

Commento e Analisi

I risultati ottenuti sono in linea con le aspettative. L'algoritmo Binary Merge Insertion Sort combina l'Insertion Sort (complessità O(n^2)) e il Merge Sort (complessità O(n log n)). Di conseguenza, i tempi di esecuzione aumentano in modo esponenziale all'aumentare di `k`, ma rimangono generalmente accettabili per valori di `k` fino a 5000. Il Merge Sort, con complessità O(n log n), dimostra una maggiore efficienza in termini di tempo di esecuzione, soprattutto per dataset di grandi dimensioni.

Il Binary Merge Insertion Sort è particolarmente vantaggioso quando `k` è abbastanza piccolo, poiché il costo aggiuntivo del Merge Sort è trascurabile rispetto all'efficienza dell'Insertion Sort per array di piccole dimensioni. Con `k` più elevati, il Merge Sort diventa preferibile poiché il suo costo è ammortizzato dalla riduzione delle iterazioni dell'Insertion Sort.

In termini di campo utilizzato come chiave di ordinamento, il tempo di esecuzione è influenzato principalmente dalla complessità delle operazioni di confronto tra gli elementi. Gli interi e i numeri in virgola mobile richiedono meno confronti rispetto alle stringhe, il che spiega il tempo aggiuntivo necessario per il campo stringa.

Scelta di `k` nella Pratica

La scelta di `k` dovrebbe essere basata sulle dimensioni effettive dei dati da ordinare e sulle prestazioni desiderate. Per dataset di piccole dimensioni, `k` può essere più grande, poiché il costo aggiuntivo dell'Insertion Sort è trascurabile. Per dataset più grandi, è preferibile utilizzare `k` più piccoli o passare direttamente al Merge Sort.

In conclusione, la scelta di `k` deve bilanciare l'efficienza dell'ordinamento con la quantità di tempo disponibile per l'operazione. I risultati mostrano che il Binary Merge Insertion Sort è una soluzione flessibile che consente di adattare l'approccio di ordinamento alle dimensioni del dataset.