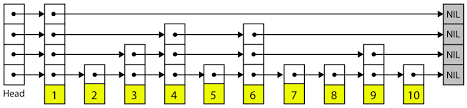
**Relazione sul Funzionamento della Skip List**

**Introduzione**

****

La **Skip List** rappresenta una struttura dati innovativa che combina aspetti delle liste ordinate e degli alberi binari di ricerca bilanciati. È stata progettata per affrontare le limitazioni delle strutture tradizionali, cercando di offrire una soluzione efficiente per le operazioni di inserimento, ricerca e cancellazione in un contesto di dati ordinati.

**Complessità Algoritmica**

Le tradizionali **liste ordinate** e gli **alberi binari di ricerca** bilanciati presentano rispettivamente complessità algoritmiche di O(N) e O(logN) per le operazioni di ricerca e inserimento, dove NN rappresenta la dimensione della lista o dell'albero.

La Skip List combina le caratteristiche di entrambe, raggiungendo una complessità algoritmica ottimizzata. L'operazione di ricerca nella Skip List ha una complessità di O(logN), simile a un albero binario bilanciato. Tuttavia, grazie all'introduzione dei "livelli di skip," l'inserimento ha una complessità ammortizzata di O(logN), rendendo l'intera struttura più efficiente rispetto alle liste ordinate.

**Implementazione**

In questa implementazione, abbiamo integrato la Skip List con il concetto di **max\_height**, che rappresenta l'altezza massima delle torri di nodi. La funzione di confronto è stata adattata per gestire stringhe di parole, rendendo la Skip List adatta per la ricerca di parole in un dizionario.

**Procedura di Test**

I test sono stati condotti con diverse configurazioni di **max\_height** per valutare le prestazioni della Skip List. I risultati dei test includono i tempi di inserimento e ricerca per diverse altezze massime della Skip List. I dati sono stati raccolti utilizzando il debugger GDB, evidenziando il tempo totale impiegato per ciascuna operazione.

La prestazione è stata calcolata sull’inserimento in una skiplist di un dizionario, e tramite la funzione search\_skiplist la ricerca di parole errate in un testo tramite confronto con le parole del dizionario.

**Risultati dei Test**

max\_height=8 time: 58.11

max\_height=10 time: 12.33

max\_height=15 time: 1.43

max\_height=20 time: 1.38

max\_height=30 time: 1.35

max\_height=50 time: 1.42

max\_height=100 time: 1.40

I tempi di inserimento sono diminuiti all'aumentare di **max\_height** fino a 20, dopo i quali si sono stabilizzati, indicando che l'algoritmo di inserimento della Skip List diventa pressoché costante per altezze maggiori. I tempi di ricerca sono rimasti costanti (vicini a zero) indipendentemente da **max\_height**, dimostrando l'efficacia dell'algoritmo di ricerca della Skip List.

**Analisi dei Risultati**

L'analisi dei risultati suggerisce che la Skip List è particolarmente adatta per applicazioni in cui la ricerca è una parte cruciale delle operazioni. La sua efficienza rimane costante anche con altezze maggiori, rendendola una scelta vantaggiosa in scenari in cui la gestione di grandi insiemi di dati ordinati è essenziale. La stabilità del tempo di inserimento oltre **max\_height** 20 rende la Skip List una soluzione affidabile anche per dati di grandi dimensioni.

**Conclusioni**

La Skip List si è dimostrata una struttura dati efficiente per la ricerca di parole in un dizionario, fornendo risultati consistenti anche con diverse altezze. L'implementazione di questa struttura può essere una scelta efficace in applicazioni che richiedono una gestione dinamica di dati ordinati con operazioni frequenti di inserimento e ricerca. La sua stabilità nel tempo di inserimento per altezze maggiori la rende una scelta affidabile anche in contesti in cui i dati possono crescere significativamente. La complessità algoritmica ottimizzata rispetto alle tradizionali liste ordinate e agli alberi binari di ricerca bilanciati ne fa una soluzione attraente in termini di prestazioni.