Alberti rosso-neri vs Alberi binari di ricerca

Salvatore Baglieri

Luglio 2020

1 Introduzione

Vogliamo analizzare le differenze tra Alberi Binari di Ricerca e Alberi Rosso-Neri.
Gli alberi binari di ricerca sono strutture dati che gestiscono insiemi dinamici e sono organizzati in una struttura dove ogni nodo contiene un valore detto chiave, un puntatore al nodo di destra e un puntatore al nodo di sinistra, detti figli. Il nodo iniziale è detto radice, mentre i nodi finali senza figli sono detti foglie dell'albero. Gli alberi binari di ricerca godono delle seguenti proprietà: il valore chiave del figlio sinistro di un nodo è sempre minore rispetto a quella del padre e viceversa col destro.
Gli alberi rosso-neri sono un'estensione di quest'ultimi. In particolare si aggiunge un nuovo attributo ai nodi, il colore, e sono definiti come segue:

1. Ogni nodo è rosso o nero
2. La radice è nera
3. Ogni foglia è nera
4. Se un nodo è rosso, allora entrambi i suoi figli sono neri
5. Per ogni nodo, tutti i cammini semplici che vanno dal nodo alle foglie contengono lo stesso numeriodi nodi neri.

In questo modo ottengo un albero bilanciato, a differenza dell'albero binario di ricerca in cui tutto dipende dall'ordine di inserimento delle chiavi.

Gli algoritmi scritti in python che utilizzerò per eseguire i test sugli alberi sono i seguenti:

Insert(key): Inserisce nell'albero il nodo con valore key.
Search(key): Ricerca dell'albero la chiave key, e restituisce FALSE o TRUE a seconda dell'esito dell'operazione.
InorderTreeWalk(x): Stampa i valori dell'albero in ordine crescente, partendo dal nodo x.

2 Aspettative

Dalla teoria so che i tempi di computazione degli algoritmi sopra citati sono differenti per le due strutture dati. In particolare, essendo l'albero rosso-nero bilanciato, si ha che quasi tutte le operazioni su di esso hanno un tempo $\theta(lgn)$, dove n è il numero di nodi dell'albero.
Per quanto riguarda l'insert si ha che l'operazione costa $O(h)$ per l'albero binario di ricerca, dove h è l'altezza dell'albero, poichè scorre i nodi fino ad arrivare alle foglie, mentre per l'albero rosso-nero ho un costo di $\theta(lgn)$.
L'algortimo di ricerca ha un costo stimato di $O(h)$ per gli alberi binari di ricerca e $O(lgn)$ per gli alberi rosso-neri.
L'algoritmo InorderTreeWalk ha invece un costo di $\theta(n)$ per entrambe le strutture dati.

3 Test

I test sono scritti in python, ogni test è il risultato di una media di 200 iterazioni dello stesso problema applicato ad alberi di dimensioni crescenti, da 5 a 1000 nodi, con incrementi di 5.

3.1 Test Insert

Analizziamo la funzione di inserimento di valori casuali.

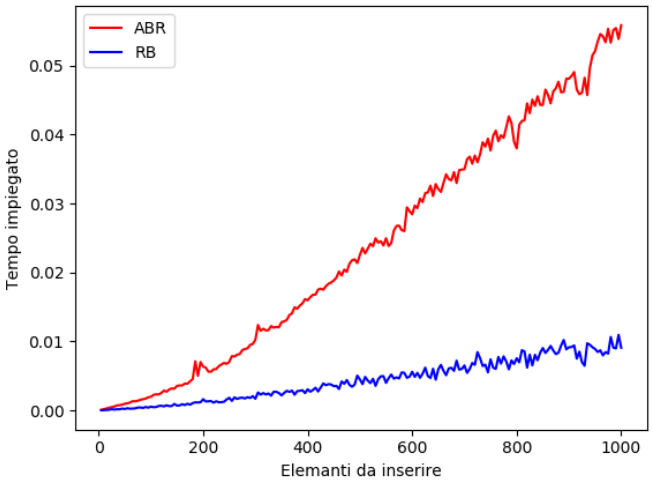

Figure 1: Prestazioni a confronto della funzione Insert

Notiamo che all'aumentare del numero di nodi, le prestazioni dell'albero binario di ricerca peggiorano. Ciò accade perchè l'albero tende a sbilanciarsi e

quindi l'operazione di inserimento diventa più complicata. Nel caso di albero rosso-nero questo non accade, come si nota dal grafico.

3.2 Test InorderTreeWalk

Analizziamo la funzione di attraversamento in ordine dell'albero.

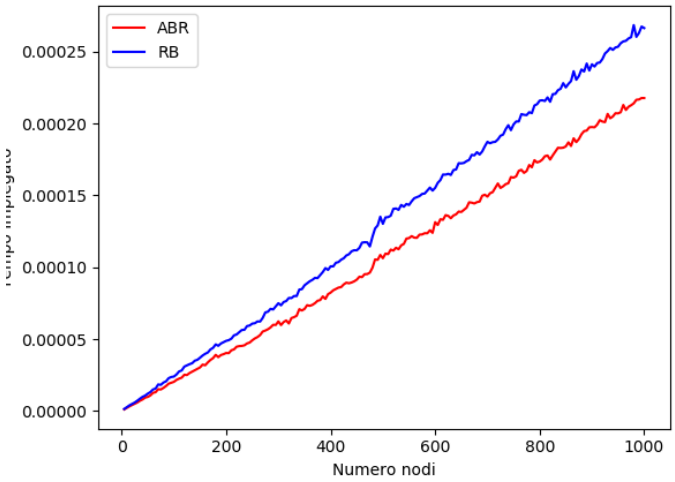

Figure 2: Prestazioni a confronto della funzione InorderTreeWalk

In questo caso entrambe le strutture dati hanno un andamento quasi identico e lineare.

3.3 Test Search

Analizziamo la funzione di ricerca di un elemento nell'albero. In questo test sono stati scelte chiavi che sono sicuramente presenti negli alberi, quindi non ho considerato il caso di ricerca senza successo.

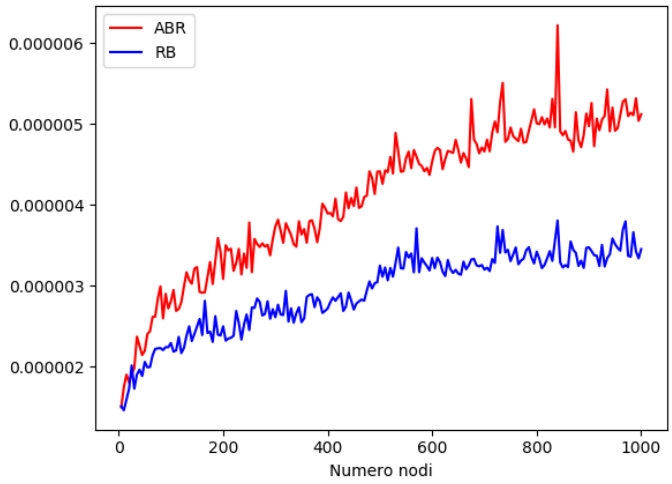

Figure 3: Prestazioni a confronto della funzione TreeSearch

Anche in questo caso abbiamo un andamento simile, anche se l'albero binario di ricerca ha delle prestazioni leggermente migliori.

4 Conclusioni

Abbiamo analizzato le tre funzioni principali sugli alberi binari di ricerca e rosso-neri. Non abbiamo però analizzato la funzione delete.
Abbiamo quindi osservato (figura 1) che per quanto riguarda l'inserimento, le prestazioni dell'albero rosso-nero sono decisamente migliori rispetto all'albero binario di ricerca.
Per quanto riguarda invece l'algoritmo di ricerca e di attraversamento in ordine, non abbiamo notato sostanziali differenze tra le due strutture dati.
I test effettuati quindi rispettano le aspettative discusse precedentemente.