Alberi binari di ricerca

Cosimo Michelagnoli

June 10, 2020

Abstract

In questa relazione è stato messo a confronto la struttura dati degli alberi binari di ricerca con quella degli alberi binari rosso neri. Questi ultimi sono una variante degli alberi binari di ricerca. I nodi degli alberi rosso neri, contengono un'informazione aggiuntiva, il colore, che permette di definire delle caratteristiche strutturali dell'albero. Nonostante questo rimangono degli alberi binari di ricerca con maggiori vincoli per l'inserimento ma con evidenti miglioramenti nelle performance.

1 Struttura dati

La struttura analizzata è quella degli alberi binari di ricerca, messa a confronto con una sua variante. Gli alberi rosso neri infatti sono alberi binari di ricerca a cui è stato aggiunto un attributo ai nodi dell'albero, il colore, che può essere rosso o nero. La loro struttura è complessa ma rende il tempo delle principali operazioni di modifica uguale a $\theta(lgn)$, dove n numero degli elementi memorizzati nell'albero.

2 Algoritmi

2.1 Inserimento

L'algoritmo di inserimento è uguale per entrambe le strutture dati, ma in quello degli alberi rosso neri si conclude con l'algoritmo di fixup. Quest'ultimo controlla che tutte le proprietà degli alberi rosso neri vengano rispettate dopo l'inserimento dell'elemento, e in caso contrario, modifica la struttura dell'albero.

2.2 Ricerca

L'algoritmo di ricerca dovrebbe avere prestazioni maggiori se eseguito su alberi rosso-neri. Questo perchè l'altezza dell'albero rosso-nero è sempre proporzionata al numero di nodi inseriti e, a differenza dei normali alberi di ricerca, non può verificarsi il caso i nodi vengano aggiunti in modo da sbilanciare nettamente l'albero.

3 Differenze teoriche:

3.1 Sequenze ordinate

La struttura dati degli alberi rosso neri e quella degli standard alberi binari di ricerca risultano apparentemente simili se confrontate su n (numero di elementi) preso piccolo a piacere con ordine sequenziale. Molto differente è il caso in cui n tenda ad infinito. Con sequenze ordinate l'albero rosso nero, grazie alle sue proprietà, garantisce un altezza massima di $2\ln(n+1)$. Con le stesse sequenze l'albero binario di ricerca è soggetto a uno sbilanciamento della sua altezza che raffiunge il valore di n.

3.2 Sequenza randomica

Per quanto riguarda le sequenze randomiche le due strutture dati si comportano in maniera similare in quanto gli alberi binari di ricerca mantengono un altezza vicina a $\ln n$. Avendo entrambe le strutture dati altezze sullo stesso ordine di grandezza l'inserimento e la ricerca avranno esiti pressochè simili.

4 Descrizione degli esperimenti

Per analizzare le due strutture dati (attraverso la funzione *timer()*) verrà calcolato il tempo di esecuzione degli algoritmi di inserimento e di ricerca del massimo implementati opportunamente per le due diverse strutture :

- al variare della dimensione del vettore in input: il test verrà fatto a partire da una dimensione nulla ad una massima (2000 elementi) con un passo di 100. Per ogni dimensione del vettore in input il test verrà ripetuto 100 volte e in seguito verrà calcolato il tempo medio di esecuzione.
- al variare della tipologia del vettore in input. Verranno pertanto utilizzati tre tipi di vettori:
  (a) vettore ordinato in modo crescente;
  (b) vettore orinato al contrario
  (c) vettore composto da numeri generati casualmente in un intervallo (a,b) attraverso *random.randrange(a,b)*. In particolare, è stato scelto (a,b)=(0,n) con n dimensione del vettore.

5 Risultati dei test

Si riportano i risultati dei test eseguiti su i due algoritmi sopra descritti.

5.1 Inserimento sequenza ordinata

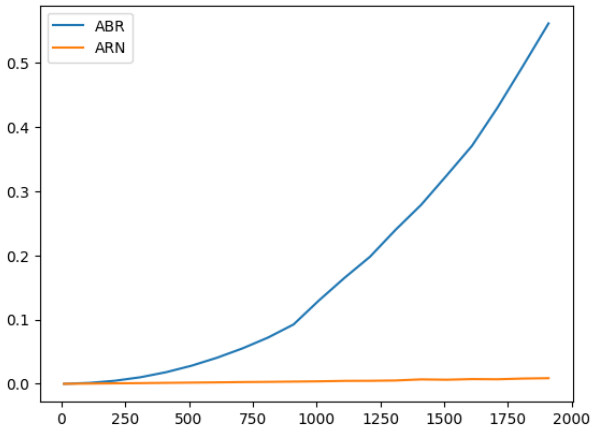

Figure 1: Confronto con sequenza ordinata

Dimensione	ABR	ARN
10	1.681e-05	2.264e-05
110	0.001	0.000
210	0.004	0.000
410	0.018	0.001
510	0.028	0.001
610	0.040	0.002
810	0.072	0.003
910	0.092	0.003
1010	0.130	0.003
1210	0.198	0.004
1310	0.239	0.005
1410	0.278	0.006
1610	0.370	0.007
1710	0.430	0.007
1810	0.495	0.008

5.2 Ricerca con sequenza ordinata

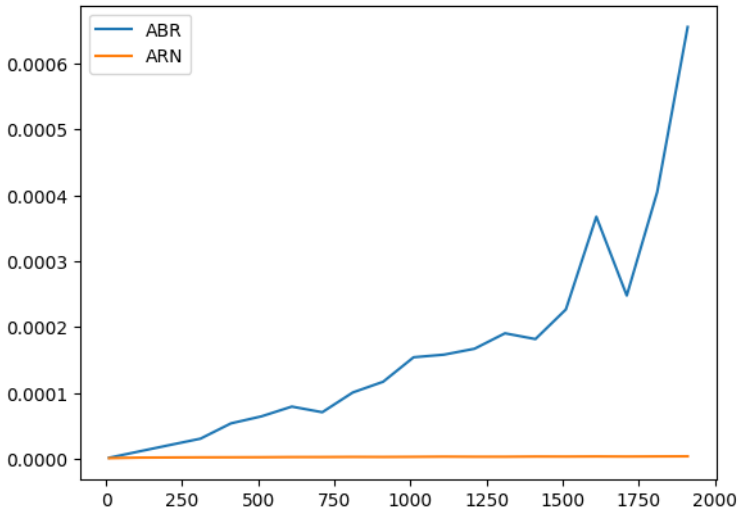

Figure 2: Confronto con sequenza ordinata

Dimensione	ABR	ARN
10	1.7648e-06	1.123e-06
110	1.1479e-05	1.976e-06
210	2.1145e-05	2.177e-06
410	5.3923e-05	2.437e-06
510	6.4561e-05	2.554e-06
610	7.9402e-05	2.799e-06
810	0.0001	3.033e-06
910	0.0001	2.957e-06
1010	0.0001	3.154e-06
1210	0.0001	3.267e-06
1310	0.0001	3.282e-06
1410	0.0001	3.607e-06
1610	0.0003	3.746e-06
1710	0.0002	3.578e-06
1810	0.0004	3.795e-06

5.3 Inserimento sequenza random

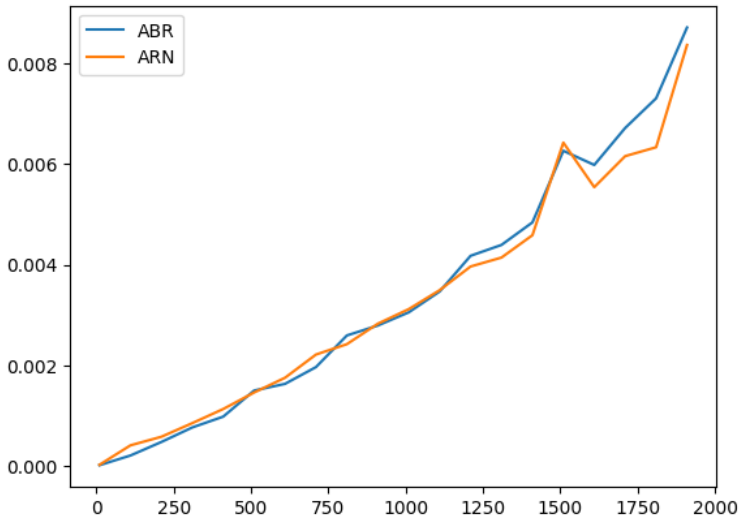

Figure 3: Confronto con sequenza ordinata

Dimensione	ABR	ARN
10	1.3603e-05	2.1889e-05
110	0.0002	0.0004
210	0.0004	0.0005
410	0.0009	0.0011
510	0.0014	0.0014
610	0.0016	0.0017
810	0.0025	0.0024
910	0.0027	0.0028
1010	0.0030	0.0036
1210	0.0041	0.0039
1310	0.0043	0.0041
1410	0.0048	0.0045
1610	0.0059	0.0055
1710	0.0067	0.0061
1810	0.0073	0.0063

5.4 Ricerca sequenza random

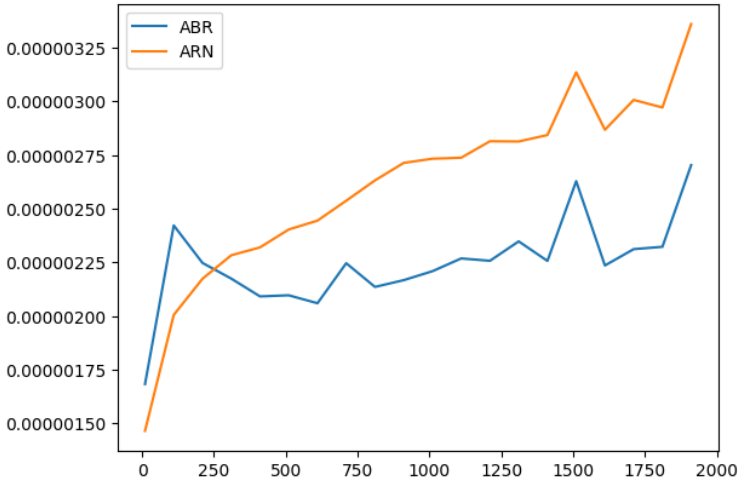

Figure 4: Confronto con sequenza random

Dimensione	ABR	ARN
10	1.684e-06	1.466e-06
110	2.422e-06	2.006e-06
210	2.248e-06	2.174e-06
410	2.092e-06	2.320e-06
510	2.097e-06	2.404e-06
610	2.060e-06	2.445e-06
810	2.136e-06	2.632e-06
910	2.168e-06	2.714e-06
1010	2.209e-06	2.733e-06
1210	2.258e-06	2.814e-06
1310	2.348e-06	2.813e-06
1410	2.257e-06	2.843e-06
1610	2.236e-06	2.868e-06
1710	2.312e-06	3.007e-06
1810	2.323e-06	2.972e-06

6 Conclusioni

L'esito dei test ha in parte confermato le predizioni teoriche. Come è ben visibile dai grafici, il tempo di esecuzione di inserimento e ricerca su alberi binari rosso neri è nettamente inferiore perchè impedisce la creazione di un albero totalmente sbilanciato. Nel caso invece di sequenze randomiche, la ricerca e l'inserimento hanno tempi di esecuzione simii per entrambe le strutture dati.