Differenze tra Alberi Binari di Ricerca e Alberi Rosso-Neri

di Pagliocca Marco

1 Alberi Binari di Ricerca

Un **albero binario di ricerca** é una struttura dati molto utile per gestire informazioni che devono essere ordinate rispetto ad una qualche proprietá, definita **chiave**. Vengono usati in molte circostanze, spesso per implementare un dizionario o una coda con prioritá.
Innanzitutto sono *Alberi Binari*, cioé alberi radicati, dove ogni nodo presenta i campi "key","left" (puntatore figlio sx),"right" e "p" (puntatore al padre). Dopodiché vi é la proprietá di ricerca:

- se y si trova nel sotto-albero di radice x.left, allora y.key $\leq$ x.key
- se y si trova nel sotto-albero di radice x.right, allora y.key $\geq$ x.key

É garantito cosí l'ottenimento di una struttura dati nella quale l'operazione di *Ricerca* viene molto semplificata. Ma non solo, anche l'*inserimento*, la *cancellazione*, la *ricerca del successore*, etc, vengono eseguiti in O(h), dove h é l'altezza dell'ABR.
Va da sé che lo scopo principale é il raggiungimento dell'altezza piú bassa possibile, cioé **logN**, con N il numero di elementi in esso inseriti. Purtroppo gli ABR non presentano un meccanismo di gestione sufficiente a garantire un'altezza minima, la quale puó divergere fino a N, ad esempio quando vengono inseriti elementi di valore crescente(o decrescente).
A tal proposito vengono utilizzati gli **Alberi Rosso-Neri**, nei quali si ha che h $\leq$ 2log(N+1), e dunque h = O(logN).
Essi sono ABR che presentano 5 proprietá aggiuntive:

1. ogni nodo presenta un attributo "color" che puó essere rosso o nero;

2. la radice é nera;

3. le foglie sono nere;

4. se un nodo é rosso, allora i suoi figli sono neri;

5. tutti i cammini da ogni nodo alle foglie contengono lo stesso numero di nodi neri.

Tuttavia i benefici ottenuti da questa struttura dati non sono gratuiti, ma é necessario implementare le operazioni che modificano l'albero in modo da non violare le proprietá. Dunque, nelle operazioni di inserimento e cancellazione, viene chiamata una funzione che utilizza procedure come *Left/Right-Rotate* e ripristina eventuali proprietá perse.
Tali algoritmi richiedono tempi di esecuzione asintoticamente inferiori alla restante parte, per cui il costo computazione delle principali operazioni negli Alberi Rosso-Neri é **O(logN)**.
Sebbene il costo asintotico non aumenti, l'utilizzo di algoritmi di *Fix-Up*, genera, in certe circostanze, evidenti ritardi nei tempi registrati, facendo sí che nemmeno gli Alberi Rosso-Neri siano la "pallottola d'argento" per ogni situazione.

2 Test condotti

L'analisi precedente ci fornisce un chiaro criterio di realizzazione sui test da eseguire per mettere in luce le differenze tra ABR e ARN.
In particolar modo si concentreranno sulle operazioni di:

- **Inserimento** di un numero costante di elementi all'interno dell'albero, al variare della sua dimensione;

- **Ricerca** di un valore estremo (ovvero su una foglia) al variare della dimensione dell'albero.

Per ottenere un buon campionamento verranno inseriti 'pacchetti' di 10 elementi alla volta, fino al raggiungimento di alberi di dimensione N = 990, limite oltre il quale si ha un eccessivo utilizzo della memoria Stack fornita al programma.

Specifiche del calcolatore: gli esperimenti vengono eseguiti su un *ASUS Laptop X541UV* a 8GB di memoria RAM e con processore Intel Core i7-6500U a 2.50GHz di frequenza. L'architettura a 64-bit ospita come Sistema Operativo Windows 10 Home.

Tipologia dei dati: Per avere una buona visione sulle differenze si considerano, per ogni operazione:

Elementi strettamente crescenti forniti da un vettore ordinato, i cui valori vengono aumentati di 10 unitá ad ogni nuovo inserimento;

Elementi di valore costante forniti da un vettore di valori costanti. Il valore numerico di per sé risulta ininfluente al fine dell'analisi;

Elementi di valore casuale forniti da un vettore generato tramite un algoritmo pseudo-casuale.

Le prime 2 tipologie sono usate al fine di mettere in luce gli svantaggi degli ABR rispetto agli ARN, in quanto determinano l'inserimento 'unilaterale' degli elementi, ovvero si raggiunge il caso critico di h = N.
La terza tipologia mostra in modo piú evidente i ritardi temporali dovuti all'utilizzo di operazioni aggiuntive negli ARN. Dunque mette in luce il vantaggio degli ABR rispetto agli ARN.

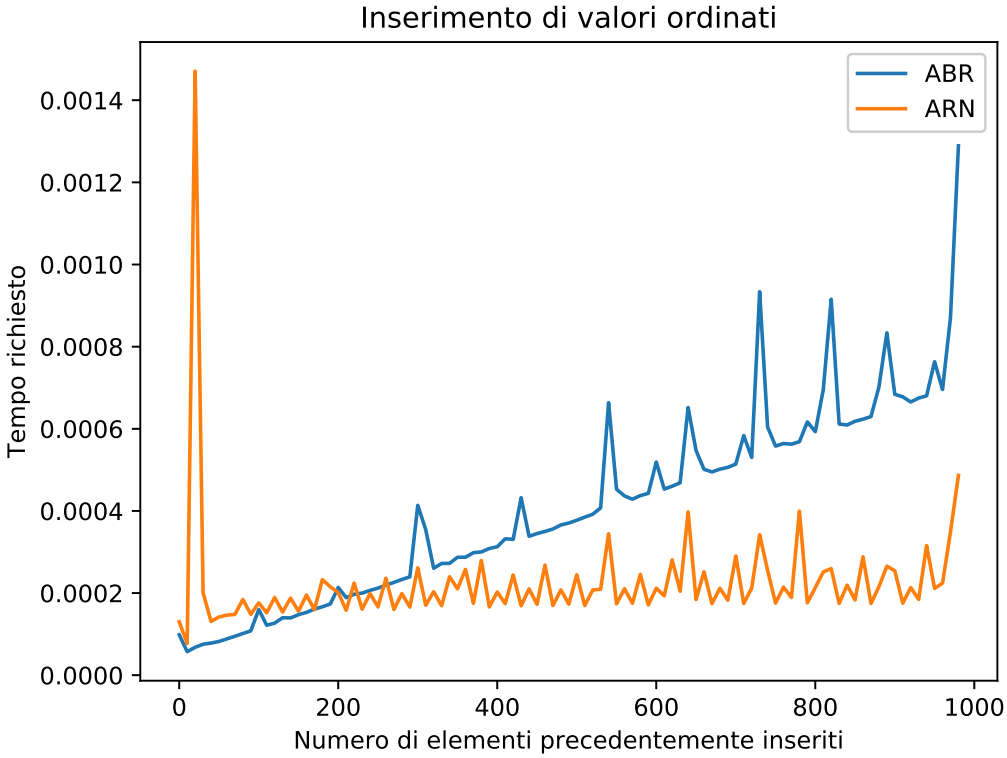
Misurazioni temporali: Il timer verrá fatto partire un attimo prima che venga effettuata la chiamata a funzione (dell'inserimento e della ricerca) e verrá terminato non appena ha concluso l'esecuzione. In questo modo i tempi saranno indipendenti dalla costruzione dei vettori che forniscono i dati.

3 Esperimenti

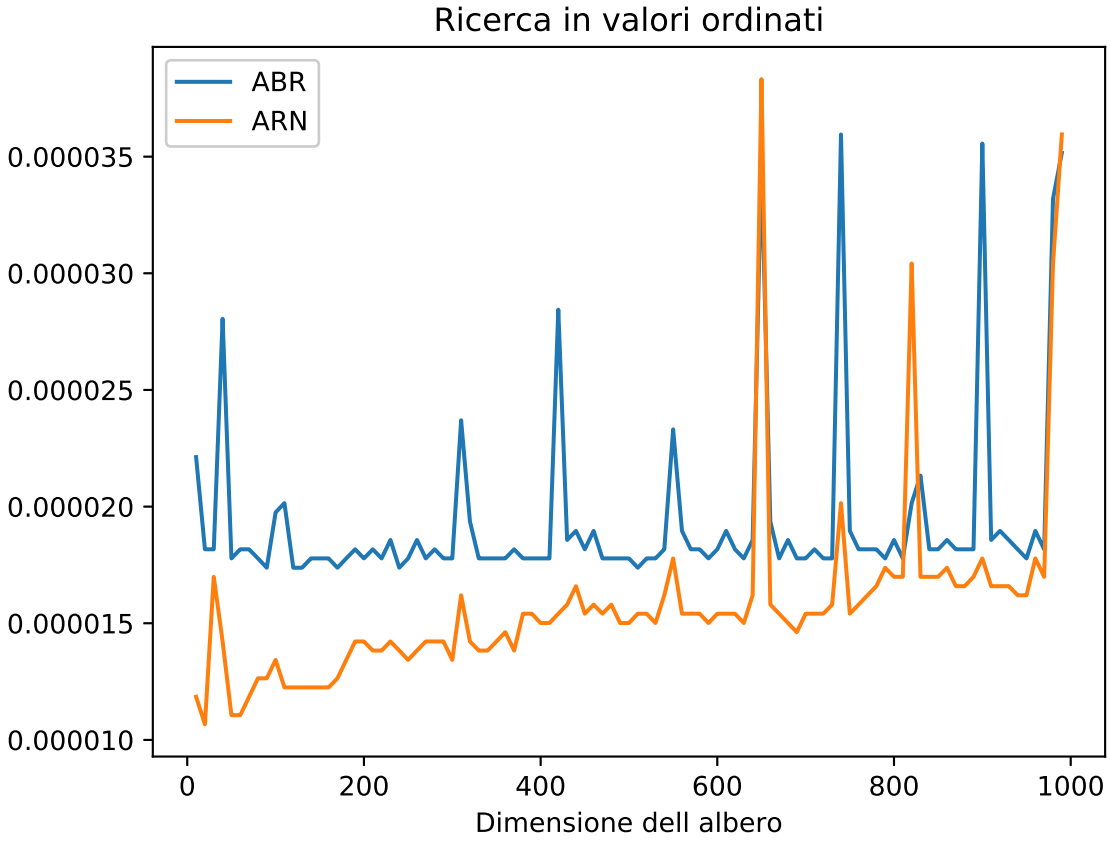
Tutti i dati degli esperimenti svolti sono memorizzati nel file di testo allegato *OutputABReARN*. I grafici che seguono sono ottenuti utilizzando per l'ascissa il numero di elementi nell'albero e per ordinata il tempo, in secondi, necessario all'esecuzione dell'algoritmo.

3.1 Elementi strettamente crescenti

L'inserimento di valori strettamente crescenti fa sí che l'ABR si trasformi in una lista concatenata di altezza N, dunque molto inefficiente. Di fatto il tempo necessario per inserire lo stesso numero di elementi aumenta in modo pressapoco lineare. Nell'ARN, invece, viene mantenuto un profilo pressapoco costante.
Si noti come i picchi repentini nei due alberi corrispondono all'inserimento degli stessi valori: infatti sono dovuti al raggiungimento delle foglie.

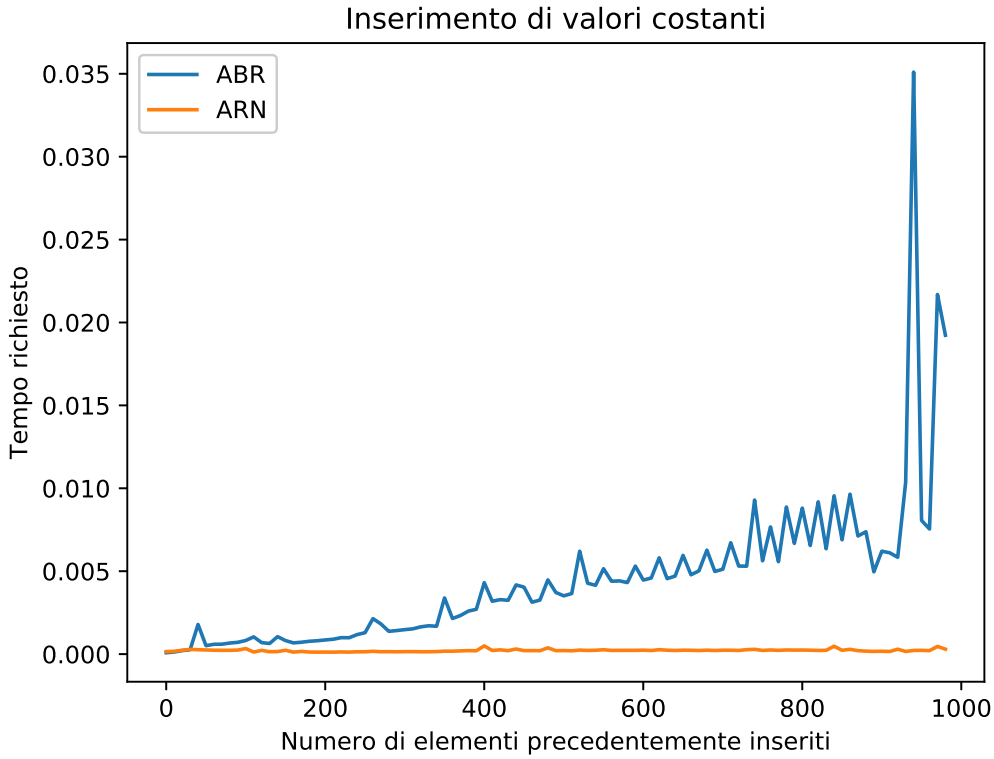


Anche nella ricerca i picchi temporali corrispondono ad elementi corrispondenti, tuttavia per l'ARN il tempo richiesto é generalmente inferiore.
Vale la pena notare che nell'ABR non si ha un aumento lineare, come nell'inserimento, sebbene ad ogni ricerca successiva il numero di elementi da visitare aumenta.
Nell'ARN, invece, si ha un crescita logaritmica, come da previsione.

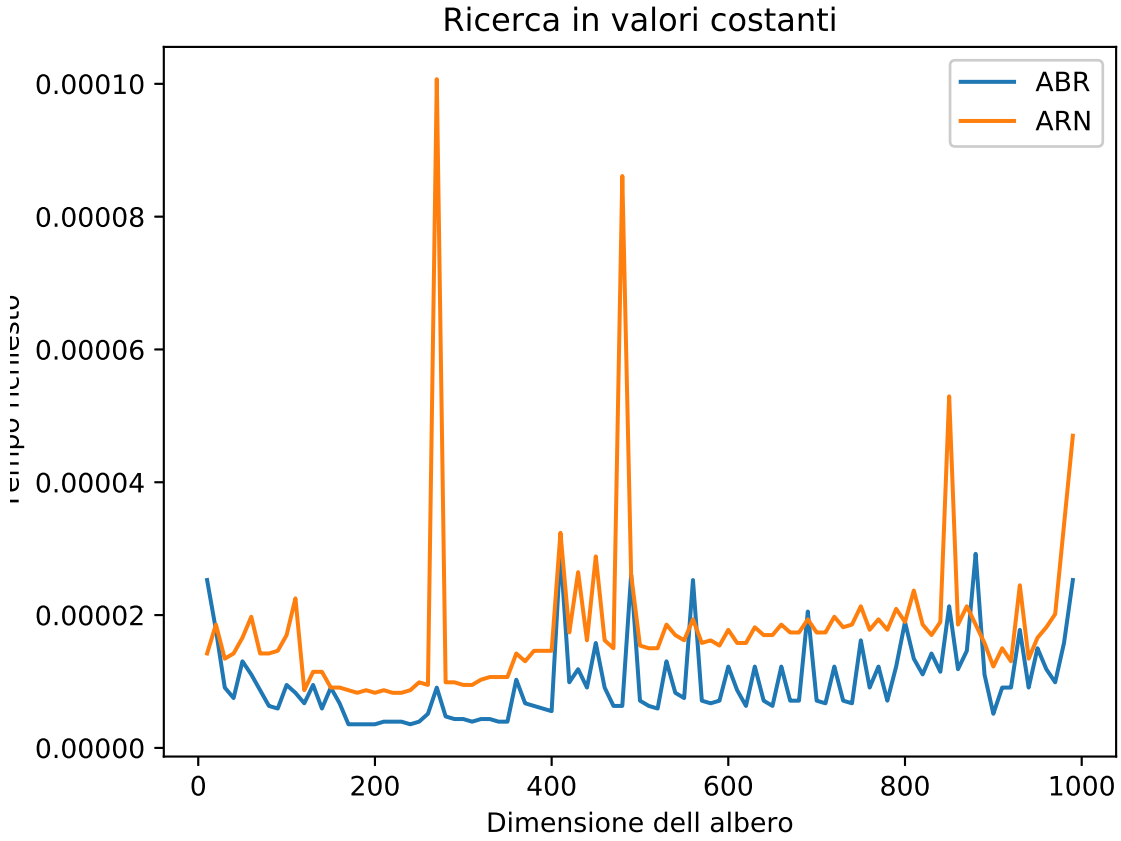


3.2 Elementi costanti

Sicuramente il test che mette in luce il vantaggio degli Alberi Rosso-Neri rispetto agli Alberi Binari di Ricerca é l'inserimento di valori costanti. Dovuta all'implementazione del codice, l'inserimento di valori tutti uguali avviene solo da un lato dell'ABR e fa sí che h = N. Si ha dunque un costo che cresce linearmente con la dimensione, ovviamente peggiore dell'inserimento nell'ARN.

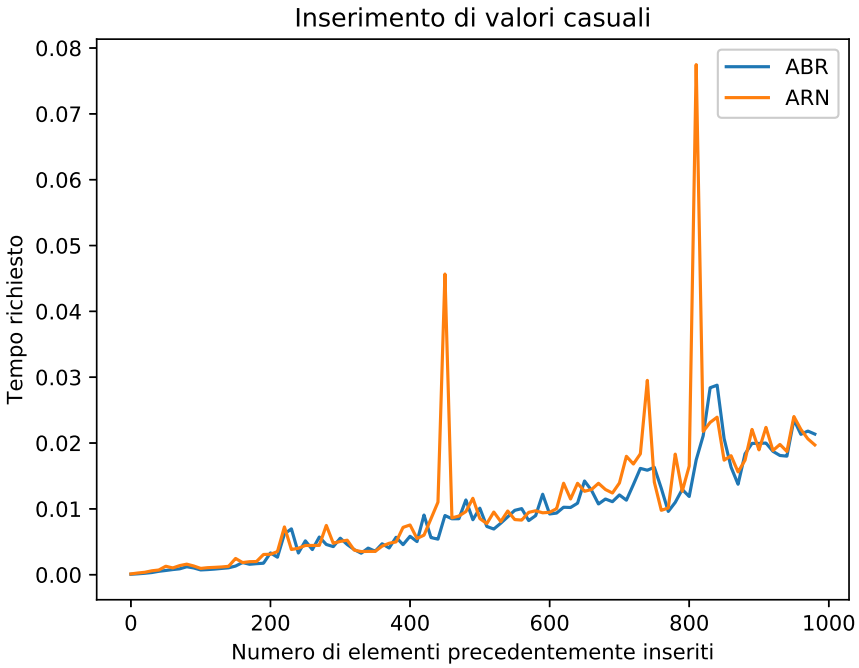


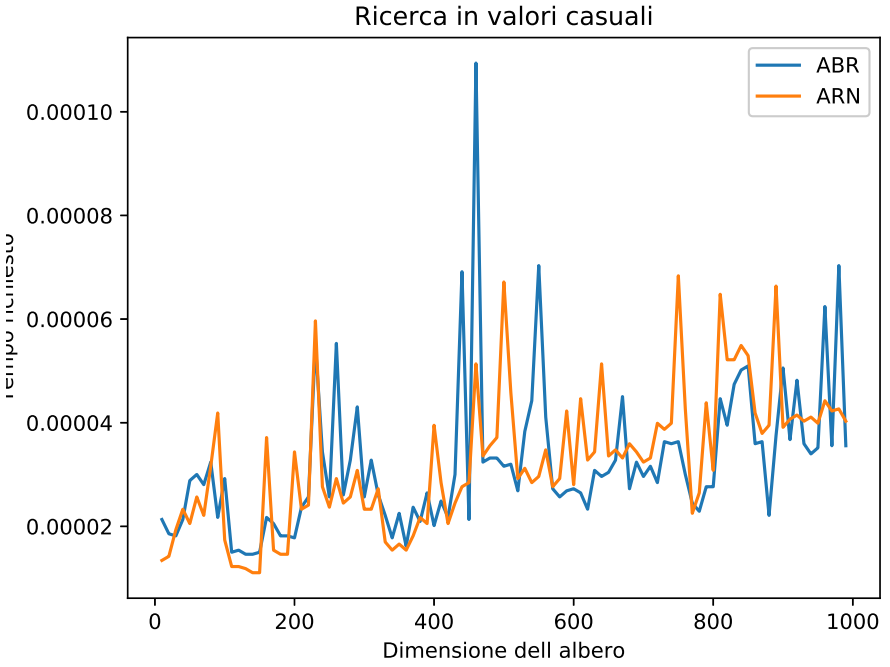
Poiché i valori sono tutti uguali, la ricerca in entrambi gli alberi ha tempi molto rapidi (si veda l'ordine di grandezza del tempo). É sufficiente visitare il primo nodo, la radice, per far terminare l'esecuzione, pertanto ci si aspetterebbe tempi molto uniformi e costanti. Invece ci sono molte variazioni. Questo risultato é utile a mostrare come, all'interno di un calcolatore, bisogna considerare ampi margine di divergenza rispetto ai risultati teorici: una solita operazione eseguita in istanti di tempo diversi dá risultati differenti.



3.3 Elementi casuali

Infine si consideri il caso di inserimento di valori casuali. Il grafico mostra che il tempo di esecuzione nell'inserimento é molto simile fra i due alberi, ma l'ABR appare piú stabile. Infatti i picchi vertiginosi nell'ARN sono dovuti all'utilizzo delle rotazioni, nell'algoritmo con cui ristabilisce le proprietá dell'albero, non utilizzato nell'ABR.
Anche la ricerca rende evidente che in tali circostanze i tempi di esecuzione sono pressapoco uguali. Infatti, rispetto ai precedenti casi, l'altezza dell'ABR non é detto che abbia raggiunto massima estensione, e quindi é probabile, da quanto emerge graficamente, che assuma un valore vicino all'altezza ottimale. Di conseguenza in questo caso i due alberi si assomigliano.


Inserimento di valori casuali
ABR
ARN
Tempo richiesto
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
200
400
600
800
1000
Numero di elementi precedentemente inseriti


Ricerca in valori casuali
ABR
ARN
0.00010
0.00008
0.00006
0.00004
0.00002
0
200
400
600
800
1000
Dimensione dell albero

4 Conclusioni

Da quanto visto si conclude che gli alberi rosso neri sono molto utili quando i dati inseriti hanno una qualche forma di ordinamento o peculiaritá, che porterebbe ad avere tempi di esecuzione critici per gli alberi binari di ricerca. Invece, questi ultimi si prestano bene per essere utilizzati quando non si ha alcuna informazione aggiuntiva sulla tipologia di dati che verranno immessi. Infatti, ci si puó aspettare che i dati siano molto simili ai dati casuali degli esperimenti svolti, per cui si ottengono tempi di esecuzione piú stabili rispetto agli alberi rosso neri.