Azzolini Riccardo 2019-04-30

Operazioni sugli alberi binari di ricerca

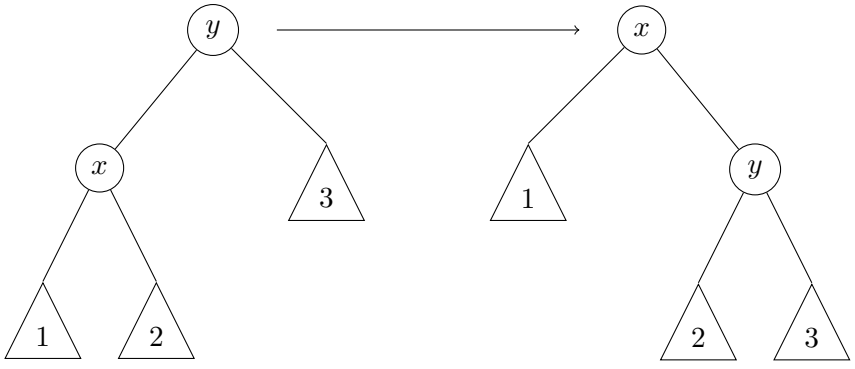
1 Rotazioni

Procedura RDX(v)
  begin
    $u \coloneqq \mathrm{sx}(v)$;
    $\mathrm{sx}(v) \coloneqq \mathrm{dx}(u)$;
    $\mathrm{dx}(u) \coloneqq v$;
    return u;
  end

Procedura RSX(v)
  begin
    $u \coloneqq \mathrm{dx}(v)$;
    $\mathrm{dx}(v) \coloneqq \mathrm{sx}(u)$;
    $\mathrm{sx}(u) \coloneqq v$;
    return u;
  end

Le operazioni di rotazione scambiano il livello di due nodi, mantenendo le proprietà degli alberi binari di ricerca.

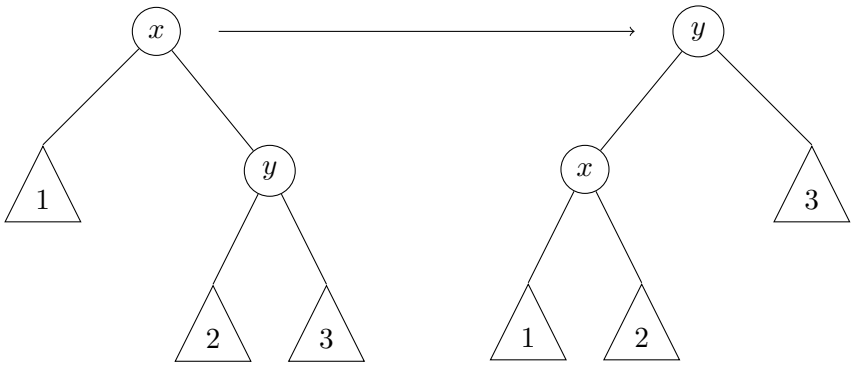
- Rotazione a destra:



  In seguito alla rotazione:
  - x e tutti i nodi del sottoalbero 1 salgono di un livello;
  - y e i nodi del sottoalbero 3 scendono di un livello;

- i nodi del sottoalbero 2 non cambiano livello.

- Rotazione a sinistra:



  In seguito alla rotazione:

  - x e tutti i nodi del sottoalbero 1 scendono di un livello;
  - y e i nodi del sottoalbero 3 salgono di un livello;
  - i nodi del sottoalbero 2 non cambiano livello.

Le rotazioni sono utili per l'implementazione di altre operazioni.

2 Inserimento alla radice

Osservazione: Se ogni operazione lascia alla radice il nodo su cui agisce, i nodi usati più di frequente rimangono ai primi livelli, cioè nelle posizioni alle quali si accede più rapidamente.

Può quindi essere utile inserire un nodo alla radice:

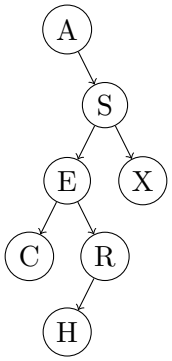
```
Procedura InsertR(v, x)
  begin
    if v = NULL then return CreaNodo(x);
    if x ≤ Key(v) then
      begin
        sx(v) := InsertR(sx(v), x);
        v := RDX(v);
      end
    else
      begin
        dx(v) := InsertR(dx(v), x);
        v := RSX(v);
      end
```

end

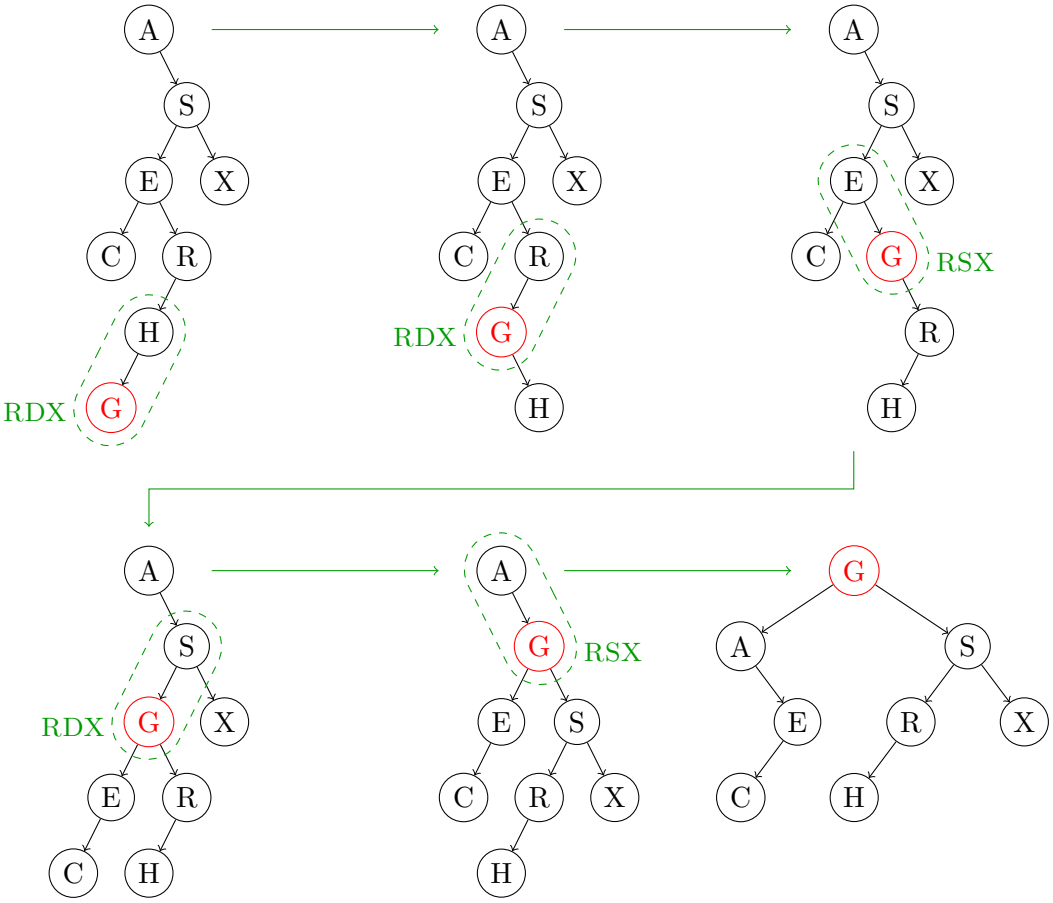
Questa procedura è implementata in modo ricorsivo. Il caso base si ha quando l'albero è vuoto: viene creato un nuovo nodo, che diventa la radice. Altrimenti:

1. si inserisce ricorsivamente il nuovo nodo nel sottoalbero corretto (qui si è scelto di consentire valori duplicati, ponendoli nel sottoalbero sinistro);
2. si usano delle rotazioni per far risalire tale nodo fino alla radice, ripercorrendo al contrario il cammino effettuato al punto 1 per l'inserimento.

2.1 Esempio



Si inserisce nell'albero il valore G:



3 Costo di costruzione

Ogni inserimento ha costo pari all'altezza dell'albero.

Di conseguenza, la costruzione di un albero mediante n inserimenti ha costo massimo quando tale albero è degenere (il che si verifica, per esempio, se la sequenza di dati inseriti è ordinata), perché allora l'i-esimo inserimento richiede i confronti:

$$\sum_{i=1}^{n-1} i = \frac{n(n+1)}{2} \sim \frac{n^2}{2} = \Theta(n^2)$$

Per l'analisi del caso medio, si ipotizza che i dati siano una permutazione casuale di lunghezza n. Il primo elemento di tale permutazione diventa quindi la radice (perché è il primo a essere inserito). Siccome la permutazione è casuale, ogni valore ha la stessa probabilità di essere la radice:

$$\forall x,\, 1 \leq x \leq n, \quad P(\text{radice} = x) = \frac{1}{n}$$

Poi, dopo il primo inserimento, i restanti $n-1$ dati si confrontano con la radice prima di essere inseriti nel sottoalbero sinistro o destro. Al termine, il sottoalbero sinistro conterrà $k-1$ valori, e quello destro ne conterrà $n-k$, con $1 \leq k \leq n$. Si può così ricavare l'equazione di ricorrenza per il numero di confronti nel caso medio:

$$T_n = n - 1 + \frac{1}{n}\sum_{k=1}^{n}(T_{k-1} + T_{n-k}), \quad T_0 = T_1 = 0$$

dove

- $n-1$ sono i confronti degli elementi successivi con la radice;
- $\frac{1}{n}\sum_{k=1}^{n}(T_{k-1} + T_{n-k})$ è il numero di confronti necessario, in media, per costruire i due sottoalberi.

Quest'equazione è analoga a quella del Quicksort, e ha perciò la stessa soluzione: il numero medio di confronti necessari per costruire un BST con n dati è $\Theta(n \log n)$. Si ricava quindi che il costo medio di ciascun inserimento è $\Theta(\log n)$, che corrisponde anche all'altezza media dell'albero risultante.