Azzolini Riccardo 2019-04-30

Alberi 2-3

1 Albero 2-3

Un **albero 2-3** per un insieme ordinato V è un albero tale che

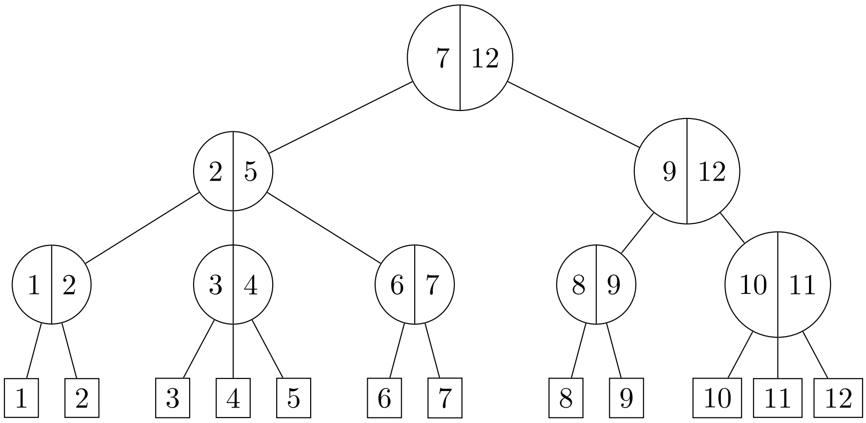
- ogni nodo interno ha 2 o 3 figli: F_1, F_2, F_3;
- ogni nodo interno ha 2 etichette, corrispondenti ai massimi valori presenti, rispettivamente, nel primo e nel secondo sottoalbero:

$$L(v) = \max F_1(v) \qquad M(v) = \max F_2(v) \qquad L(v) < M(v)$$

- i dati sono contenuti solo nelle foglie (ciascuna contiene un singolo valore $x \in V$), mentre i nodi interni formano una “mappa di navigazione ai dati”;
- tutte le foglie hanno la stessa profondità;
- è ordinato: dato un nodo v,
  - il primo sottoalbero contiene valori $x_1 \leq L(v)$ (per la precisione, contiene $L(v)$ e altri valori minori di $L(v)$);
  - il secondo sottoalbero contiene valori $L(v) < x_2 \leq M(v)$ ($M(v)$ e altri valori maggiori di $L(v)$ e minori di $M(v)$);
  - il terzo sottoalbero (se presente) contiene valori $x_3 > M(v)$.

Osservazione: Un albero 2-3 non può contenere un solo dato, perché allora la radice avrebbe un singolo figlio, mentre la definizione ne richiede almeno due. Per poter gestire un singolo dato è necessario estendere la definizione, specificando che la radice può essere una foglia.

1.1 Esempio



2 Numero di dati e altezza

Lemma: Sia n il numero di dati in un albero 2-3 di altezza h. Si ha allora che:

$$2^h \leq n \leq 3^h$$

Dimostrazione: Si effettua per induzione su h.

- Se $h = 1$, la proprietà è verificata immediatamente per la definizione di albero 2-3.
- Se $h > 1$, allora:
  - la radice ha almeno 2 sottoalberi, T_1 e T_2, entrambi di altezza $h - 1$, che implica

    $$n \geq N_{\text{dati}}(T_1) + N_{\text{dati}}(T_2) \geq 2^{h-1} + 2^{h-1} = 2^h$$

    per ipotesi induttiva;
  - la radice ha al massimo 3 sottoalberi, T_1, T_2 e T_3, di altezza $h - 1$, e quindi per ipotesi induttiva

    $$n \leq N_{\text{dati}}(T_1) + N_{\text{dati}}(T_2) + N_{\text{dati}}(T_3) \leq 3^{h-1} + 3^{h-1} + 3^{h-1} = 3^h \qquad \square$$

Osservazione: Da questo lemma si ricava che un albero 2-3 con n dati ha altezza $\Theta(\log n)$, perché

$$2^h \leq n \leq 3^h \iff \begin{cases} 2^h \leq n \\ n \leq 3^h \end{cases} \iff \begin{cases} h \leq \log_2 n \\ \log_3 n \leq h \end{cases} \iff \log_3 n \leq h \leq \log_2 n$$

quindi gli alberi 2-3 sono **bilanciati** (e, di conseguenza, non possono essere degeneri).

3 Ricerca del minimo

Procedura $\textsc{Min}(v)$
  begin
    if $v = \texttt{NULL}$ **then return** `NULL`;
    while $F_1(v) \neq \texttt{NULL}$ **do**
      $v \coloneqq F_1(v)$;
    return v;
  end

- Se l'albero è vuoto, viene restituito `NULL`.
- Altrimenti, si scende a sinistra il più possibile, passando ogni volta dal padre al primo figlio (F_1), fino a raggiungere la foglia contenente il valore minimo, che viene quindi restituita.

4 Ricerca del massimo

Procedura $\textsc{Max}(v)$
  begin
    if $v = \texttt{NULL}$ **then return** `NULL`;
    while $F_2(v) \neq \texttt{NULL}$ **do**
      if $F_3(v) = \texttt{NULL}$ **then** $v \coloneqq F_2(v)$
        else $v \coloneqq F_3(v)$;
    return v;
  end

La procedura è simile a $\textsc{Min}$, ma c'è una leggera asimmetria, perché a ogni passo è necessario determinare qual è il sottoalbero più a destra, dato che F_3 può non esistere.

5 Ricerca

Procedura FIND(v, x)
begin
if FOGLIA$(F_1(v))$ **then return** v;
if $x \leq L(v)$ **then return** FIND$(F_1(v), x)$;
if $x \leq M(v)$ **or** $F_3(v) =$ `NULL` **then return** FIND$(F_2(v), x)$
else return FIND$(F_3(v), x)$;
end

Procedura ISMEMBER(v, x)
begin
$p \coloneqq$ FIND(v, x);
if ISFIGLIO(p, x) **then return** `Vero`
else return `Falso`;
end

- FIND restituisce il nodo interno che potrebbe essere padre del valore cercato, x. Se tale valore non è presente nell'albero, il nodo restituito è la posizione in cui andrebbe eventualmente inserito (aggiungendolo una nuova foglia, contenente x, come figlio). Questa procedura è quindi utile per tutte le operazioni, come l'inserimento e la cancellazione, che richiedono una prima fase di ricerca del nodo su cui agire.

  A ogni passo, FIND determina in quale sottoalbero scendere in base alle etichette del nodo corrente. In particolare, se $x > M(v)$ ma $F_3(v) =$ `NULL`, è necessario scendere comunque nel secondo sottoalbero, per consentire anche in questo caso l'inserimento.

- ISMEMBER fa uso di FIND per implementare l'operazione Member dell'algebra eterogenea *parti di A (ordinato).*

Il costo di FIND, e quindi anche di ISMEMBER, è $\Theta(\log n)$ in ogni caso, perché si effettua un numero di passi pari all'altezza dell'albero.