Azzolini Riccardo 2019-03-19

Rappresentazione e visita di alberi

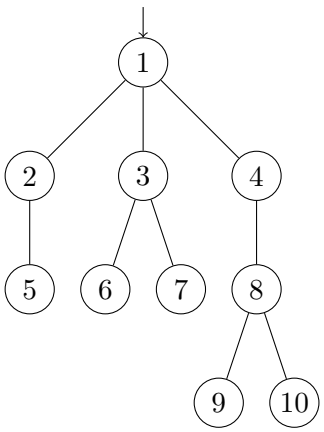
1 Rappresentazione di alberi come grafi

Poiché un albero è un particolare grafo, è possibile utilizzare le stesse rappresentazioni (liste di adiacenza o matrice di adiacenza).

Ci sono però rappresentazioni specializzate che sfruttano le caratteristiche di certi tipi di alberi (con radice, con radice ordinati, ecc.).

2 Rappresentazione di alberi con radice

Per rappresentare gli *alberi con radice* si può utilizzare un vettore P che associa a ogni nodo il suo padre.



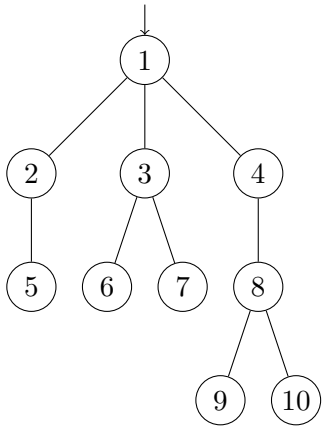
	P
1	0
2	1
3	1
4	1
5	2
6	3
7	3
8	4
9	8
10	8

- *Vantaggio*: risalire da un figlio al padre richiede tempo $O(1)$.
- *Svantaggio*: scendere dal padre a un figlio, o determinare se un nodo è una foglia, richiede una scansione del vettore, cioè tempo $O(n)$, dove n è il numero di nodi dell'albero.

3 Rappresentazione di alberi con radice ordinati

Nel caso degli *alberi con radice ordinati* si può ricorrere a due o tre vettori paralleli:

- F associa a ogni nodo il suo primo figlio (quello più a sinistra);
- S associa a ogni nodo il suo fratello successivo (a destra);
- P (opzionale) associa a ogni nodo il suo padre, ed è utile se è necessario risalire l'albero.



	P	F	S
1	0	2	0
2	1	5	3
3	1	6	4
4	1	8	0
5	2	0	0
6	3	0	7
7	3	0	0
8	4	9	0
9	8	0	10
10	8	0	0

Con questa rappresentazione, il passaggio dal padre a uno qualsiasi dei suoi figli ha complessità in tempo $O(k)$, dove k è il grado dell'albero:

- $O(1)$ per passare dal padre v al primo figlio w;
- $O(k)$ per scorrere i fratelli di w (gli altri figli di v) fino a quello desiderato.

Per percorrere un cammino dalla radice a una foglia, in un albero di grado k e altezza h, serve quindi tempo $O(hk)$.

4 Rappresentazione di alberi di grado k

Dato un *albero con radice ordinato di grado* k, ogni nodo può essere rappresentato mediante un record contenente

- l'informazione associata al nodo;
- k puntatori ai figli.

Il vantaggio di questa rappresentazione è la possibilità di passare in tempo $O(1)$ dal padre a un figlio, ma ci sono alcuni importanti svantaggi:

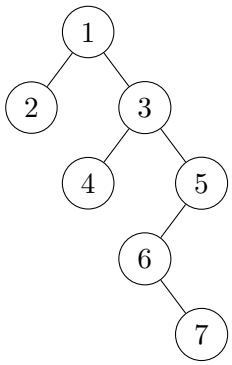
- ogni nodo occupa sempre spazio $\Theta(n)$, perché il record contiene k puntatori indipendentemente dal numero effettivo di figli del nodo;
- è necessario stabilire quali puntatori utilizzare quando il nodo ha $k' < k$ figli:
  - se si scelgono i primi k', nel caso della rimozione di un figlio bisogna spostare indietro tutti i successivi;
  - se invece non si spostano mai i figli successivi a quelli rimossi, rimangono dei "buchi".

Entrambi questi problemi si risolvono utilizzando una lista per memorizzare i puntatori ai figli (una soluzione che peraltro si può applicare ad alberi di grado illimitato), ma così facendo il tempo necessario a passare dal padre a un figlio diventa $O(k)$, come nella rappresentazione a vettori paralleli degli alberi con radice ordinati.

5 Rappresentazione di alberi binari

Per gli *alberi binari* è necessario distinguere tra figlio sinistro e figlio destro. Ci sono due possibili rappresentazioni:

- una *statica* (a dimensione fissa), basata su una coppia di vettori paralleli che associano a ogni nodo il figlio sinistro e quello destro;



	SX	*DX*
1	2	3
2	0	0
3	4	5
4	0	0
5	6	0
6	0	7
7	0	0

- una *dinamica*, nella quale ogni nodo è un record contenente
  - l'informazione associata al nodo;
  - un puntatore al figlio sinistro;
  - un puntatore al figlio destro.

Entrambe queste rappresentazioni permettono di passare dal padre a uno dei due figli in tempo $O(1)$.

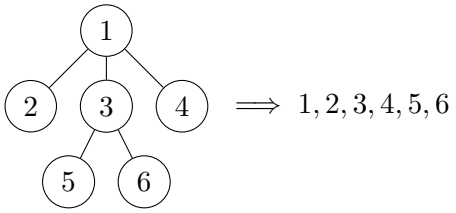
6 Attraversamento di alberi come grafi

Gli stessi metodi di attraversamento (o visita) definiti per i grafi, cioè in ampiezza e in profondità, si possono applicare anche agli alberi. Le proprietà di questi ultimi consentono però di semplificare gli algoritmi di visita: in particolare, l'esistenza di un unico cammino dalla radice a un nodo garantisce che non verrà mai incontrato un nodo già visitato, quindi non è necessario il vettore `nuovo`.

Ad esempio, l'algoritmo di visita in ampiezza, che per gli alberi con radice prende il nome di **attraversamento per livelli** (**level-order**), diventa:

```
public void LevelOrder(v) {
    Queue<Nodo> q = new Queue<Nodo>();
    if (v != null) q.enqueue(v);

    while (!q.isEmpty()) {
        x = q.front(); q.dequeue();
        x.visita();
        for (y in L(x)) q.enqueue(y);
    }
}
```



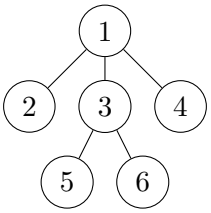
7 Attraversamento di alberi con radice ordinati

Per gli alberi con radice ordinati esistono alcuni metodi di visita in più rispetto a quelli utilizzabili sui grafi.

Per descrivere questi metodi, è utile dare una definizione ricorsiva di un albero con radice ordinato:

- $T = \langle r \rangle$ è l'albero formato dalla sola radice r;
- $T = \langle r, T_1, T_2, \ldots, T_k \rangle$ è l'albero formato dalla radice r e dai sottoalberi $T_1, T_2, \ldots, T_k$.

L'albero



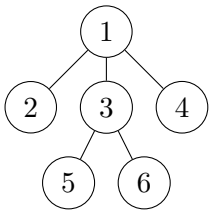
si può quindi scrivere come

$$\langle 1, \langle 2\rangle, \langle 3, \langle 5\rangle, \langle 6\rangle\rangle, \langle 4\rangle\rangle$$

7.1 Ordine anticipato

Per visitare un albero in **ordine anticipato** (**pre-order**)

- se $T = \langle r\rangle$, si visita il nodo r;
- se invece $T = \langle r, T_1, T_2, \ldots, T_k\rangle$:
  1. si visita r;
  2. si visita in ordine anticipato ciascuno dei sottoalberi $T_1, T_2, \ldots, T_k$.

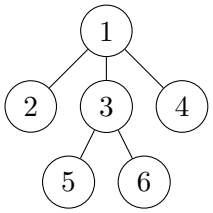


$\Longrightarrow 1, 2, 3, 5, 6, 4$

7.2 Ordine posticipato

Per visitare un albero in **ordine posticipato** o **differito** (**post-order**)

- se $T = \langle r\rangle$, si visita il nodo r;
- se invece $T = \langle r, T_1, T_2, \ldots, T_k\rangle$:
  1. si visitano in ordine posticipato ciascuno dei sottoalberi $T_1, T_2, \ldots, T_k$;
  2. si visita r.



$\Longrightarrow 2, 5, 6, 3, 4, 1$