a cura di Andrea de Prisco

Le strutture informative: gli alberi

di Anna Pugliese

Dal nome senza dubbio derivante dall'analogia strutturale rispetto al cugino botanico, la struttura dati «albero» costituisce uno dei maggiori esempi di eleganza, duttilità e, per dirla tutta, genialità, riscontrabile nel campo dell'organizzazione dei dati.
Questo numero della rubrica è dedicato alla trattazione delle più generali forme di alberi dal punto di vista topologico, ed alla presentazione di esempi di impiego di strutturazioni ad albero

L'albero

Partiamo direttamente dalla rappresentazione grafica di ciò che la topologia definisce un «albero libero»; essa è riportata in figura 1.

L'albero libero è la forma più generale di albero.

A partire da questo, è possibile, mediante restrizioni, definire delle forme di alberi particolari.

Le strutture informative comunemente assumono la forma di alberi, di cui sia stato stabilito, fra i possibili nodi, quello che assumerà il ruolo di «RADICE» dell'albero.

Tali particolari alberi liberi, vengono detti semplicemente «alberi».

La figura 2 riporta due esempi di alberi entrambi derivati dall'albero libero in figura 1, ma diversi nella scelta della radice: il nodo A per l'albero di figura 2a, ed il nodo B per quello di figura 2b.

Nota la radice dell'albero, si determina automaticamente la suddivisione dei nodi dell'albero in «LIVELLI», così come illustra la stessa figura 2.

Infine, un'altra importante considerazione da fare, consiste nella possibilità di considerare significativo l'ordine in cui appaiono i nodi figli di uno stesso padre.

È questo il caso che più frequentemente si presenta fra le strutture dati ad albero.

L'albero, non è la più generale fra le strutture composte da nodi ed archi, esso è infatti derivato da una più generale struttura detta «GRAFO», mediante l'imposizione di opportune restrizioni. La struttura che se ne ricava, l'albero appunto, è dotata di proprietà molto particolari, fra le quali probabilmente spicca la ricorsività.

Prima di esaminare le implicazioni che la struttura topologica albero ha sulla strutturazione ad albero delle informazioni, è interessante dedicare un paragrafo di questo articolo alla presentazione di un po' di aspetti più squisitamente formali.

La definizione degli alberi

Percorso semplice

Dato un insieme N di nodi, ed un insieme A di archi, congiungenti coppie di nodi di N, si definisce percorso semplice, tra un nodo n_a ed un nodo n_b, una sequenza di archi:

$[(n_{1,1},n_{1,2}),(n_{2,1},n_{2,2}),(n_{3,1},n_{3,2}),\ldots,(n_{m,1},n_{m,2})]$

Dove:

$(n_{i,1},n_{i,2})$ appartiene ad A per $i=1,2,3,\ldots,m$;
$n_{i,2}=n_{i+1,1}$ per $i=1,2,3,\ldots,m-1$;
$(n_{1,1},n_{i,2})<>(n_{j,1},n_{j,2})$ per $j=1,2,\ldots,i-1,i+1,\ldots m$;
$n_{1,1}=n_a$; $n_{m,2}=n_b$;

In altre parole un percorso semplice è costituito da una sequenza di archi che, partendo dal nodo n_a, attraversano un qualsiasi numero di nodi intermedi senza mai passare due volte dallo stesso nodo, e giungono sul nodo n_b.

Ciclo

Un ciclo è definito come un percorso semplice tra due nodi, n_a ed n_b, tale che $n_a=n_b$.

Struttura connessa

Un insieme di nodi ed un insieme di archi, formano una struttura connessa, se presa una qualsiasi coppia di nodi, n_a, ed n_b, esiste sempre un percorso semplice congiungente n_a con n_b.

Albero libero

Un albero libero AL è definito come un insieme N di nodi ed un insieme A di archi, che uniscono coppie di nodi di N, tali che siano verificate due qualsiasi tra le seguenti tre condizioni:

1 - AL non contiene alcun ciclo.
2 - AL è connesso.
3 - AL contiene esattamente n−1 archi (dove n è il numero di nodi in N).

Una caratteristica molto interessante di questa definizione, è costituita dal fatto che due qualsiasi delle tre condizioni, implicano necessariamente la terza.

Se un nodo dell'albero libero viene

designato come radice, dando così vita ad un albero, è possibile darne una definizione nuova che non fa esplicito riferimento agli archi, che vengono a costituire un semplice mezzo di ordinamento dei nodi.

Albero

Un albero è un insieme N di uno o più nodi, tale che:
1 - Un particolare nodo di N è designato come radice.
2 - I rimanenti nodi, possono essere ripartiti negli insieme disgiunti N_1, N_2,...,N_m (m>=0), ciascuno dei quali viene detto «sottoalbero della radice», ed è a sua volta un albero.

Visto che siamo in vena di definizioni, diamo fin d'ora la definizione di un particolare tipo di albero che, come vedremo nel seguito, costituisce una struttura molto importante.

Albero binario

Un albero binario è un insieme N di nodi tali che:
1 - Se N non è vuoto, un nodo di N è designato come radice.
2 - I rimanenti nodi possono essere ripartiti nei due insieme disgiunti N_1 ed N_2, ciascuno dei quali è a sua volta un albero binario.

Notiamo che, dal punto di vista concettuale, l'albero binario non dovrebbe essere considerato come un caso particolare di albero, e questo per due motivi: innanzitutto un albero binario può anche essere vuoto, mentre un albero no (ma questa è solo una disquisizione accademica, tant'è che le strutture informative ad albero prevedono sempre il caso di albero vuoto), in secondo luogo, quando un nodo dell'albero binario ha un solo figlio, questo non necessariamente appartiene al sottoalbero N1 del padre (comunemente detto sottoalbero sinistro), ma potrebbe benissimo essere il suo figlio destro (appartenente cioè al suo sottoalbero destro). In altre parole, con gli alberi binari, non basta dire «il figlio unico di quel nodo», per riferirsi ad un nodo il cui fratello è vuoto, poiché ogni nodo conserva sempre la sua identità di figlio destro o sinistro indipendentemente dall'esistenza o meno del fratello.

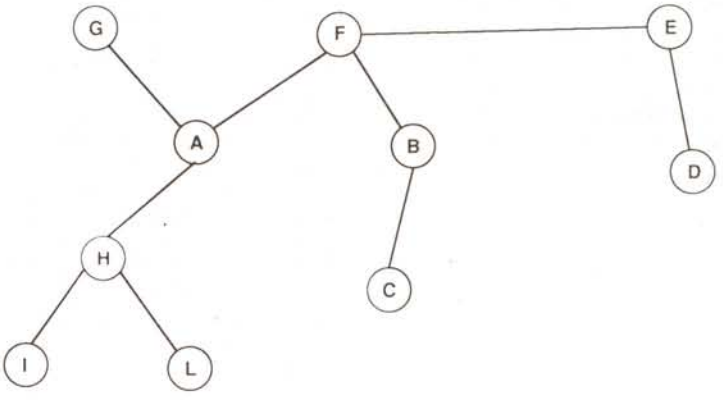
Strutturazione ad albero delle informazioni

Dopo aver esaminato le caratteristiche strutturali degli alberi, vediamo ora come essi possano essere utilizzati per rappresentare informazioni sotto forma di dati strutturati.

Cominciamo col notare che gli archi degli alberi, sebbene in linea di principio non siano dotati di verso, finiscano poi con l'esserlo nello stesso momento in cui ne viene stabilita la radice. Stabilita infatti la radice, e quindi la suddivisione in livelli dei nodi dell'albero, si ha che ogni arco dell'albero congiunge sempre un nodo del livello i con un nodo del livello i+1, e può dunque essere considerato come orientato da quello a questo. Ogni arco quindi, porta con sé una relazione che potremmo chiamare «PADRE-FIGLIO» e che induce un ordinamento parziale tra i nodi dell'albero. Osserviamo ancora una volta l'albero di figura 2a. Potremmo dire, ad esempio, che essendo A il padre di F, A è maggiore di F; a sua volta F è maggiore di E, E è maggiore di D, e transitivamente concludiamo che A è maggiore di D (anche se avremmo potuto supporre esattamente il contrario). Niente è possibile affermare, invece, circa la relazione esistente tra L e B. Tuttavia, è possibile ottenere un secondo ordinamento parziale, stabilendo che, tra i figli dello stesso padre, il primo (quello più sinistra), è maggiore di tutti gli altri, il secondo è minore soltanto al primo, e così via.

A questo punto abbiamo tutto ciò che ci serve per definire un ordinamento totale sui nodi di un albero. Di tali ordinamenti, se ne possono definire

Figura 1



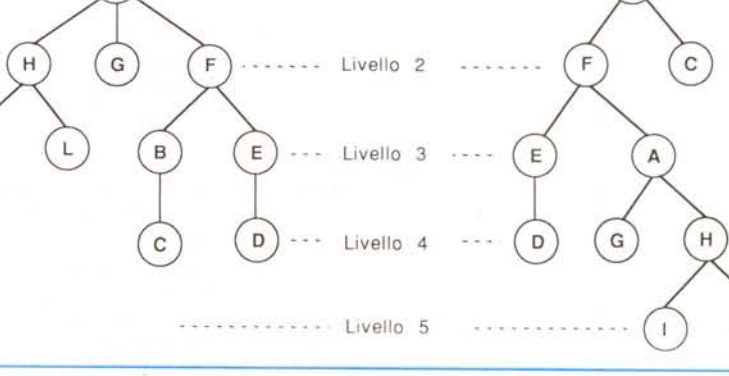


Figura 2 - Due alberi ricavati dall'albero libero di figura 1, scegliendo come radice il nodo A (figura 2a) ed il nodo B (figura 2b).

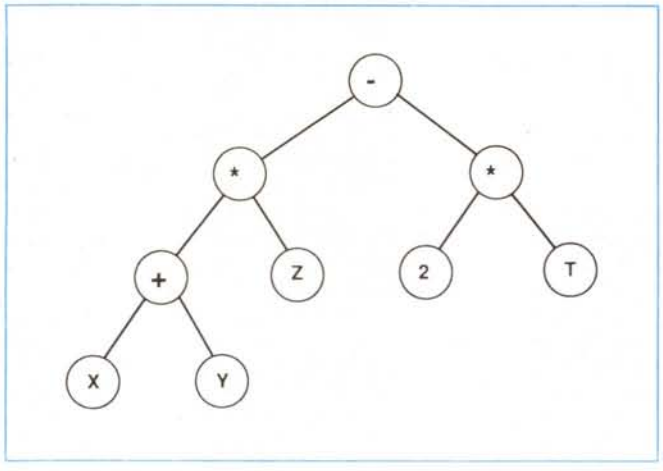


Figura 3 - Un'espressione algebrica rappresentata mediante un albero binario.

tanti, ed è difficile sceglierne uno come quello standard. Ora, quello che è accaduto è che alcuni fra i possibili ordinamenti, hanno acquistato particolare significatività, ed in tal modo si sono guadagnati un nome e l'onore di essere riportati come esempi. Ci riferiamo all'ordinamento ANTICIPATO ed a quello DIFFERITO, accompagnati, nel caso di alberi binari, da un terzo: l'ordinamento SIMMETRICO.

Prima ancora di passare alla descrizione di questi ordinamenti sui nodi degli alberi, è il caso di sottolineare come accada assai spesso che una applicazione faccia uso di una strutturazione ad albero dei dati, sui quali essa definisce implicitamente un ordinamento, che riveste un grosso significato, ma solo per l'applicazione in oggetto. Per dirla in breve: gli alberi si prestano ad essere utilizzati in svariati modi, anche concettualmente molto diversi tra loro. Il fatto di aver definito in sommario, l'albero, come una struttura dati duttile e geniale, è dovuto proprio alla caratteristica degli alberi di permettere la memorizzazione dei dati in maniera articolata; ogni dato infatti, occuperà una posizione corrispondente a quella di un nodo, ed essendo i nodi posti in relazione fra loro mediante archi, queste stesse relazioni resteranno valide per i dati dei nodi stessi, e potranno essere sfruttate per esprimere esse stesse informazioni aggiuntive.

Gli esempi che faremo, chiariranno ulteriormente queste considerazioni.

Torniamo ora agli ordinamenti totali definibili sui nodi di un albero, ed osserviamo che grazie a tali ordinamenti, è possibile «visitare» un albero, vale a dire scorrere i suoi nodi, con un algoritmo che permetta di esaminarli tutti. Le due cose sono così legate tra loro, che dato un algoritmo di visita esso definisce un ordinamento sui nodi dell'albero (il primo nodo visitato è il più grande di tutti, e così via) e viceversa.

Descriviamo allora, gli ordinamenti totali sugli alberi, fornendo gli algoritmi di visita corrispondenti.

Visita in ordine anticipato

1 Esamina la radice
Se n>=0 è il numero di sottoalberi della radice:
2 Visita il primo sottoalbero in ordine anticipato.
3 Visita il secondo sottoalbero in ordine anticipato.

.....

n+1 Visita l'ultimo sottoalbero in ordine anticipato.

Visita in ordine differito

Se n >=0 è il numero di sottoalberi della radice:
1 Visita il primo sottoalbero in ordine differito.
2 Visita il secondo sottoalbero in ordine differito.

.....

n Visita l'ultimo sottoalbero in ordine differito.
n+1 Esamina la radice.

Visita in ordine simmetrico (solo alberi binari)

1 Visita il sottoalbero sinistro in ordine binario simmetrico.
2 Esamina la radice.
3 Visita il sottoalbero destro in ordine binario simmetrico.

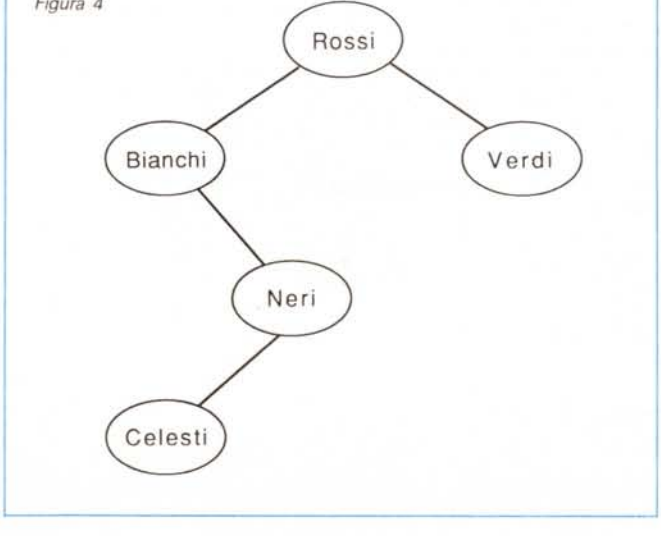
Applichiamo, a scopo esemplificativo, i primi due algoritmi all'albero di figura 2b. L'ordinamento che ne risulta è il seguente:

Ordinamento anticipato :BFEDAGHILC
ordinamento differito :DEGILHAFCB

Il terzo algoritmo (il binario simmetrico) non può essere applicato all'albero di figura 2b poiché esso non è un albero binario. In verità tale albero è molto simile (apparentemente) ad un albero binario, avendo ogni nodo non più di due figli, ma se si osserva il nodo D, si capisce che non è possibile stabilire se esso è il figlio destro oppure il sinistro del nodo E, quindi non sapremmo se esaminare E prima oppure dopo aver esaminato D.

Per capire meglio l'ordine di visita definito su un albero, può essere utile applicare una notazione parentesizzata alla sequenza di nodi che scaturisce dall'ordinamento sull'albero stesso. Applicando tale notazione, raggrupperemo i nodi di un albero nel seguente modo:
(Sottoalbero 1)(Sottoalbero 2)...(Sottoalbero n) Radice

Figura 4



nel caso di ordine differito, mentre avremo:
Radice (Sottoalbero 1)(Sottoalbero 2)...(Sottoalbero n)
nel caso di ordine anticipato.

La notazione si applica ricorsivamente ai sottoalberi della radice.

Applicando questa notazione dell'albero di figura 2b, avremo:
Ordine anticipato: B(F(E(D))(A(G)(H(I)(L))))(C)
Ordine differito: (((D)E)((G)((I)(L)H)A)F)(C)B

Questa notazione, oltre ad essere maggiormente espressiva (cosa discutibile!), costituisce un'effettiva definizione dell'albero, essendo l'espressione parentesizzata univocamente corrispondente ad un albero.

A questo punto, abbiamo messo abbastanza carne sul fuoco per poter esaminare concreti esempi di applicazione dei concetti fin qui presentati. Un'interessante applicazione degli ordini di visita degli alberi (in particolare di quelli binari), si presenta nel caso di elaborazione di espressioni algebriche.

Consideriamo l'espressione algebrica:
(X+Y)Z−2T,
essa potrebbe essere memorizzata in una struttura ad albero binario così come illustra la figura 3.

Proviamo allora a visitare l'albero di figura 3 in ordine anticipato; la sequenza ottenuta è la seguente:
-*+XYZ*2T,
che con notazione parametrizzata sarebbe:
-(*(+(X)(Y))(Z))(*(2)(T)),
che costituisce la notazione polacca diretta (o prefissa) dell'espressione (che prende il nome dal matematico polacco Lukasiewicz).

La notazione polacca inversa (o postfissa) della stessa espressione è invece ottenibile da una lettura dell'albero in ordine differito:
(((X)(Y)+)(Z)*)((2)(T)*)-.

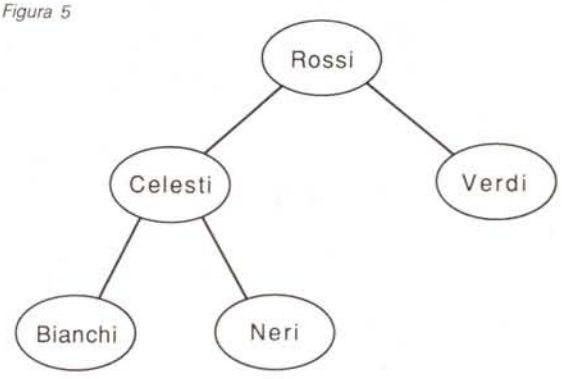
Visitando, infine, l'albero in ordine binario simmetrico, otterremo:
(((X)+(Y))*(Z))-((2)*(T))
che è la ben nota notazione infissa dell'espressione algebrica (altro che Insieme di Mandelbrot!, n.d.a.d.p.).

Problemi di ordinamento

A conclusione di questa trattazione, vogliamo parlare di un problema molto diffuso nel campo dell'archiviazione dei dati: l'ordinamento dei dati e le soluzioni rese possibili mediante l'impiego di strutture ad albero.

Parlando di ordinamento dei dati, intendiamo qui riferirci ad un ordinamento significativo per i dati in sé, quale ad esempio potrebbe essere l'ordinamento lessicografico di stringhe. Mantenere ordinato un archivio di nominativi è un esempio di tal genere.

Figura 5



Per comprendere come gli alberi possano essere utilmente impiegati in problematiche di questo genere, evitiamo di girare attorno alla questione se andiamo avanti con esempi concreti.

Supponiamo che la nostra brava lista di nominativi, da tenere ordinata, sia composta da:
Rossi, Bianchi, Verdi, Neri e Celesti.

La figura 4 riportata un albero binario contenente questi nominativi, in modo che essi siano reperibili, in maniera ordinata alfabeticamente, leggendo l'albero stesso con un algoritmo di visita binaria simmetrica. Difatti, se leggiamo l'albero in maniera binaria simmetrica otterremo la seguente sequenza:
Bianchi, Celesti, Neri, Rossi, Verdi.

Cerchiamo di capire la strategia di memorizzazione utilizzata per ottenere l'albero di figura 4.

Supponiamo che i nominativi ci si presentino davanti, nell'ordine specificato dalla sequenza iniziale. Il primo è Rossi. Dal momento che l'albero è vuoto, inseriremo Rossi nella radice stessa dell'albero. Ora tocca al Bianchi. Ci domandiamo se Bianchi è minore o maggiore di Rossi, ed essendo vera la prima ipotesi, sappiamo di dover inserire Bianchi in una posizione tale che venga visitato prima del Rossi, dall'algoritmo di visita che ci interessa, cioè quello binario simmetrico. Tale algoritmo di visita, esamina prima della radice, l'intero sottoalbero sinistro, per cui tutti i nominativi minori del Rossi dovranno finire in tale sottoalbero e tutti quelli maggiori in quello destro (che verrà esaminato dopo la radice). In particolare il Bianchi andrà ad occupare la posizione di figlio sinistro di Rossi (vale a dire la radice del sottoalbero sinistro di Rossi).

È ora la volta del Verdi. Analoghe considerazioni conducono a collocare tale nominativo nel nodo figlio destro di Rossi. Per quanto riguarda il Neri, esso viene prima del Rossi, ma la posizione di figlio sinistro del Rossi stesso è stata già occupata dal Bianchi; peraltro Neri è maggiore di Bianchi e dovrà essere visitato dopo di quest'ultimo; non è difficile convincersi del fatto che la posizione giusta per il nominativo Neri è quella di figlio destro di Bianchi (che l'ordine binario simmetrico pone dopo Bianchi, ma prima di Rossi). Infine il Celesti va posto alla sinistra di Rossi, alla destra di Bianchi e alla sinistra di Neri, cioè come figlio sinistro del nodo contenente Neri.

Un'ultima considerazione è doverosa. Osserviamo la figura 5. L'albero ivi riportato, visitato in ordine binario simmetrico, ottiene anch'esso la corretta sequenza di nominativi.

Essa peraltro, presenta, rispetto al caso della figura 4, l'indubbio vantaggio di avere tre soli livelli e non quattro. Un simile albero è quello che la nostra strategia avrebbe prodotto se l'ordine di presentazione dei nominativi fosse stato diverso, e precisamente:
Rossi, Verdi, Celesti, Neri, Bianchi (o altri simili).

Un albero strutturato in modo tale da avere, a parità di nodi, il minimo numero possibile di livelli, viene detto BILANCIATO. Tali alberi sono molto importanti nei problemi di ordinamento, in quanto velocizzano le operazioni di ricerca binaria sull'albero, cosa che non possiamo trattare per mancanza di spazio. Ci basterà comunque dire, in questa sede, che esistono delle operazioni, dette di BILANCIAMENTO, che possono venir eseguite su alberi come quello di figura 4 per produrre un albero come quello di figura 5. Il lettore particolarmente volenteroso potrebbe provare da solo a tirar giù un algoritmo che svolga questo lavoro di bilanciamento, noi, chissà, ne parleremo un'altra volta. MC