Alberi

CORDA – Informatica

A. Ferrari

Testi da

Marco Bernardo Edoardo Bontà

Dispense del Corso di

Algoritmi e Strutture Dati

Albero - definizione

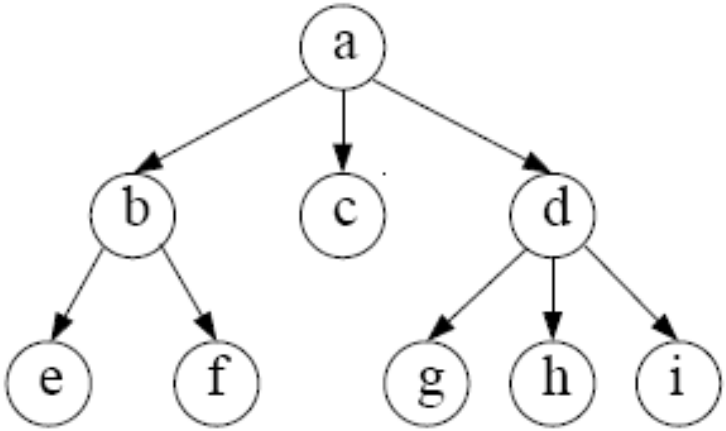
Si dice albero con radice, o semplicemente albero, una tripla $T = (N, r, \mathcal{B})$ dove N è un insieme di nodi, $r \in N$ è detto radice e $\mathcal{B}$ è una relazione binaria su N che soddisfa le seguenti proprietà:

- Per ogni $n \in N$, $(n, r) \notin \mathcal{B}$.
- Per ogni $n \in N$, se $n \neq r$ allora esiste uno ed un solo $n' \in N$ tale che $(n', n) \in \mathcal{B}$.
- Per ogni $n \in N$, se $n \neq r$ allora n è raggiungibile da r, cioè esistono $n'_1, \ldots, n'_k \in N$ con $k \geq 2$ tali che $n'_1 = r$, $(n'_i, n'_{i+1}) \in \mathcal{B}$ per ogni $1 \leq i \leq k-1$, ed $n'_k = n$.

Siano $T = (N, r, \mathcal{B})$ un albero ed $n \in N$. Si dice sottoalbero generato da n l'albero $T' = (N', n, \mathcal{B}')$ dove N' è il sottoinsieme dei nodi di N raggiungibili da n e $\mathcal{B}' = \mathcal{B} \cap (N' \times N')$.

Sia $T = (N, r, \mathcal{B})$ un albero e siano $T_1 = (N_1, n_1, \mathcal{B}_1)$ e $T_2 = (N_2, n_2, \mathcal{B}_2)$ i sottoalberi generati da $n_1, n_2 \in N$. Allora $N_1 \cap N_2 = \emptyset$ oppure $N_1 \subseteq N_2$ oppure $N_2 \subseteq N_1$.

Albero



Figli, fratelli, nodi, foglie

Sia $T = (N, r, \mathcal{B})$ un albero:

- Se $(n, n') \in \mathcal{B}$, allora n è detto padre di n' ed n' è detto figlio di n.
- Se $(n, n_1), (n, n_2) \in \mathcal{B}$, allora n_1 ed n_2 sono detti fratelli.
- I nodi privi di figli sono detti nodi esterni o foglie, mentre tutti gli altri nodi sono detti nodi interni.
- Gli elementi di $\mathcal{B}$ sono detti rami.

Grado, livello, altezza, larghezza

Sia $T = (N, r, \mathcal{B})$ un albero:

- Si dice grado di T il massimo numero di figli di un nodo di T:
$$d(T) = \max_{n \in N} |\{n' \in N \mid (n, n') \in \mathcal{B}\}|$$
- Si dice che:
  - r è al livello 1.
  - Se $n \in N$ è al livello i ed $(n, n') \in \mathcal{B}$, allora n' è al livello $i + 1$.
- Si dice altezza o profondità di T il massimo numero di nodi che si attraversano nel percorso di T che va dalla radice alla foglia più distante:
$$h(T) = \max\{i \in \mathbb{N} \mid \exists n \in N.\ \ n \text{ è al livello } i\}$$
- Si dice larghezza o ampiezza di T il massimo numero di nodi di T che si trovano allo stesso livello:
$$b(T) = \max_{1 \leq i \leq h(T)} |\{n \in N \mid n \text{ è al livello } i\}|$$

Albero binario

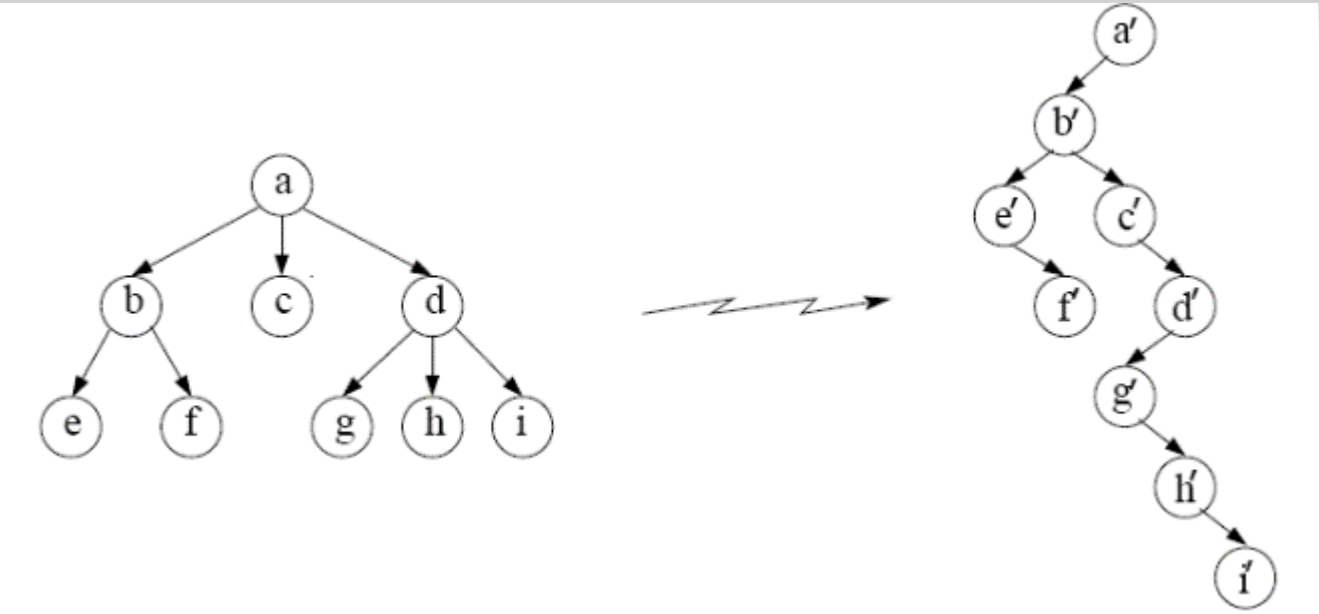
Un albero $T = (N, r, \mathcal{B})$ è detto binario se:

- $\mathcal{B} = \mathcal{B}_{sx} \cup \mathcal{B}_{dx}$.
- $\mathcal{B}_{sx} \cap \mathcal{B}_{dx} = \emptyset$.
- Per ogni $n, n_1, n_2 \in N$, se $(n, n_1) \in \mathcal{B}_{sx}$ (risp. $\mathcal{B}_{dx}$) ed $(n, n_2) \in \mathcal{B}_{sx}$ (risp. $\mathcal{B}_{dx}$), allora $n_1 = n_2$.

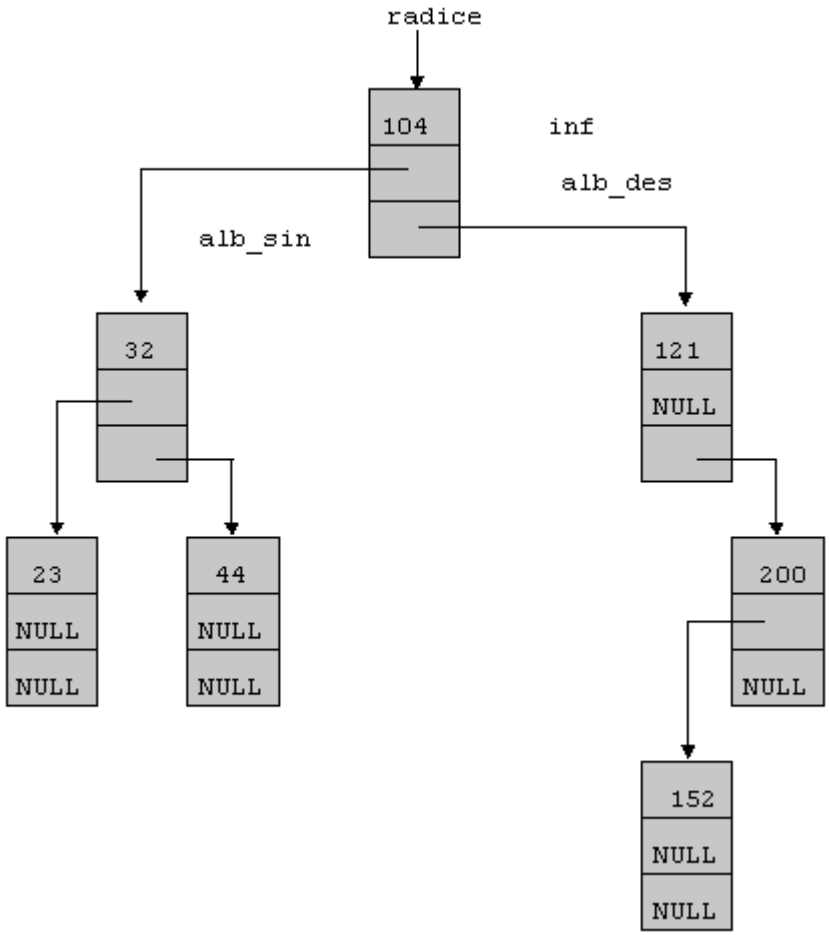
Se $(n, n') \in \mathcal{B}_{sx}$ (risp. $\mathcal{B}_{dx}$), allora n' è detto figlio sinistro (risp. destro) di n.

Trasformazione da albero ad albero binario

- Creare i nuovi nodi $n', n'_1, \ldots, n'_k$.
- Mettere n'_1 come figlio sinistro di n'.
- Per ogni $i = 1, \ldots, k-1$, mettere n'_{i+1} come figlio destro di n'_i.



Rappresentazione



Esercizio

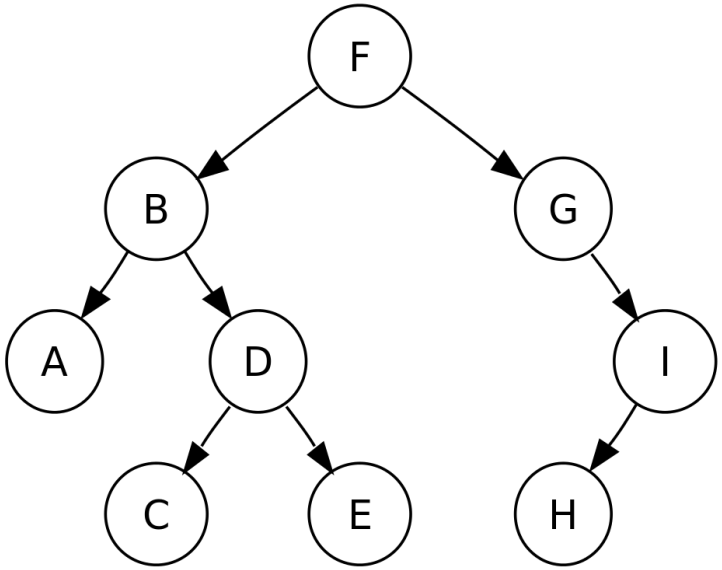
- Definire una struttura dati che permetta di rappresentare un albero binario
- Per semplicità l'informazione associata ad ogni nodo si considera che sia un numero intero

Algoritmi di visita

- La visita consiste nell'accesso una e una sola volta a tutti i nodi dell'albero.
- Per gli alberi binari sono possibili più algoritmi di visita che generano sequenze diverse (per ordine) di nodi
  - visita in ordine anticipato
  - visita in ordine simmetrico
  - visita in ordine posticipato (differito)

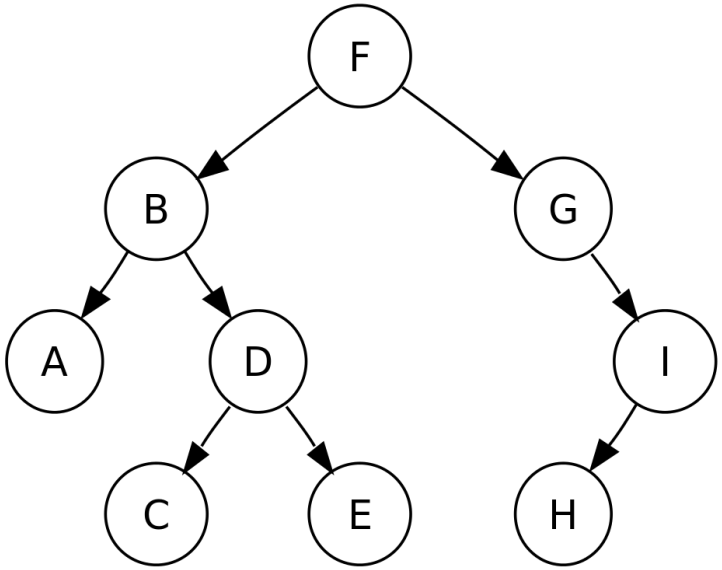
Visita in ordine anticipato

- Visita la radice
- Visita il sottoalbero sinistro in ordine anticipato
- Visita il sottoalbero destro in ordine anticipato
- Lista dei nodi:



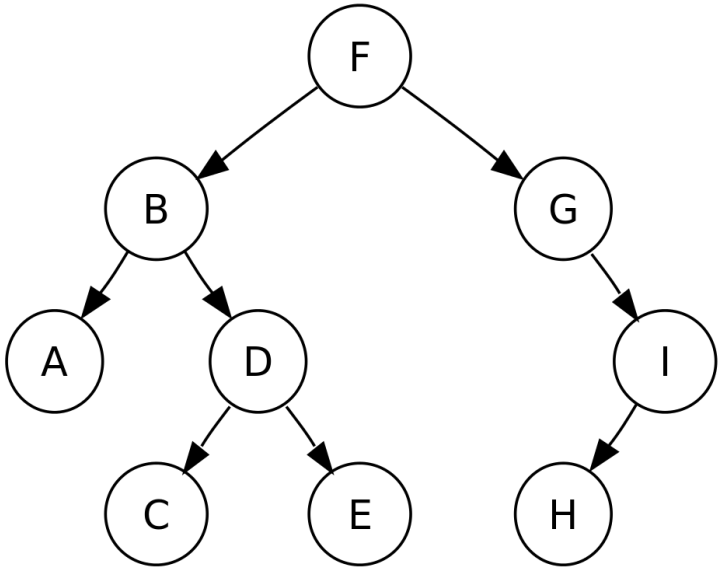
Visita in ordine simmetrico

- Visita il sottoalbero sinistro in ordine simmetrico
- Visita la radice
- Visita il sottoalbero destro in ordine simmetrico
- Lista dei nodi:



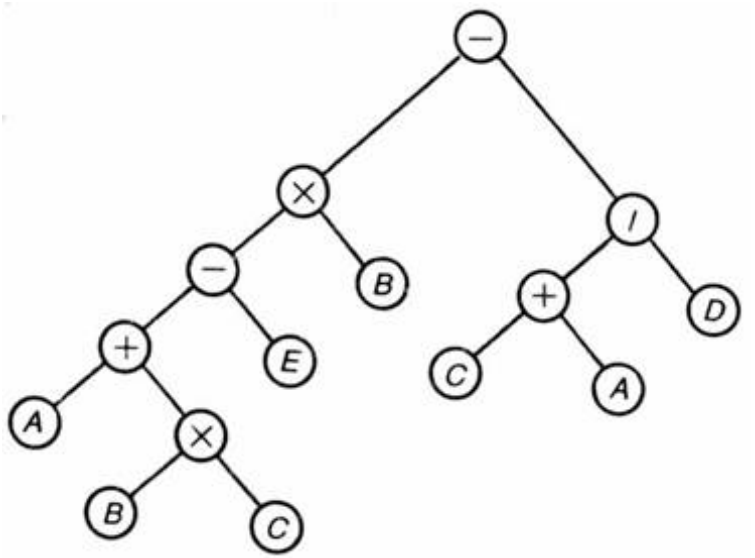
Visita in ordine posticipato

- Visita il sottoalbero sinistro in ordine posticipato
- Visita il sottoalbero destro in ordine posticipato
- Visita la radice
- Lista dei nodi:



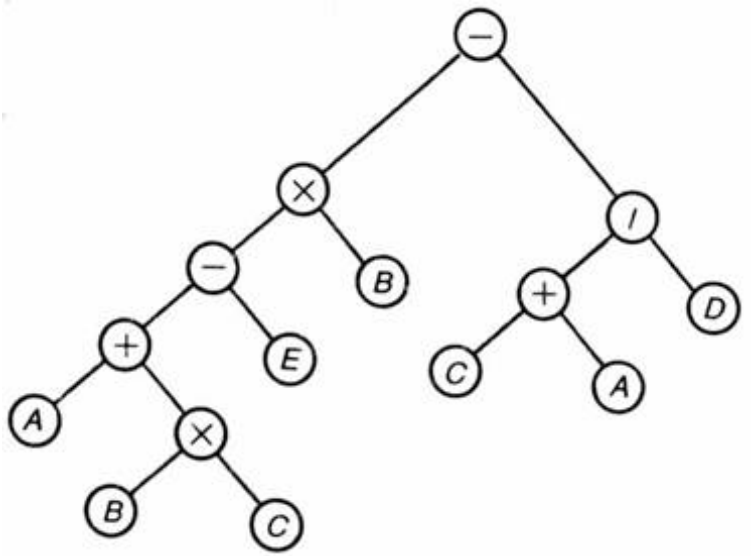
Alberi ed espressioni

- Ogni nodo che contiene un operatore è radice di un sottoalbero
- Ogni foglia contiene un valore costante o una variabile



Esercizio

- Definire la sequenza di nodi che si ottiene visitando l'albero in ordine
- Anticipato
- Simmetrico
- Differito



alberi – espressioni – algoritmi di visita - matematica – informatica ???

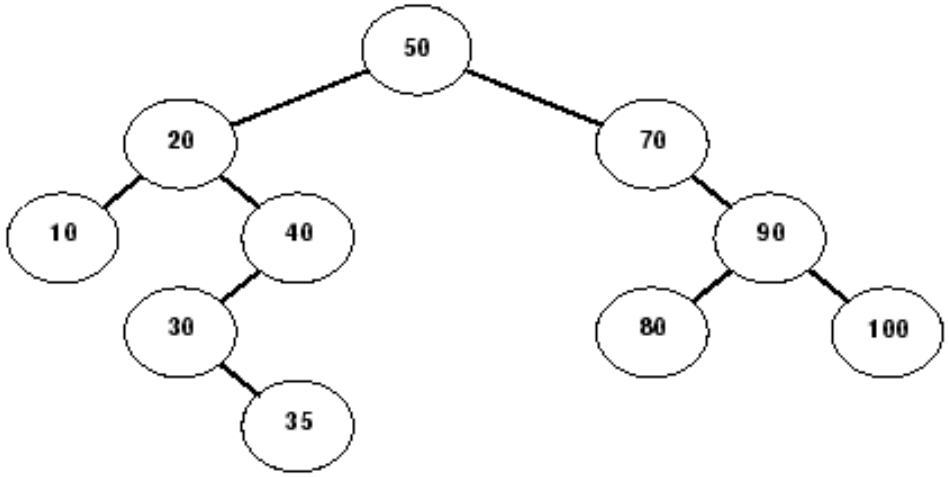
- Jan Łukasiewicz
  - http://it.wikipedia.org/wiki/Jan_%C5%81ukasiewicz
- Notazione polacca
  - http://it.wikipedia.org/wiki/Notazione_polacca

Algoritmi di ricerca

- Con opportune modifiche si può adattare un qualunque algoritmo di visita per ottenere un algoritmo di ricerca
- Nel caso pessimo la ricerca attraverserà tutti nodi dell'albero quindi avrà complessità O(n)

Alberi binari di ricerca

- Un albero binario di ricerca è un albero binario tale che:
  - per ogni nodo che contiene una chiave di valore k
  - ogni nodo del suo sottoalbero sinistro contiene una chiave di valore ≤ k
  - ogni nodo del suo sottoalbero destro contiene una chiave di valore ≥ k



Ricerca in alberi binari di ricerca

- Non è necessario visitare tutti i nodi
- Basta fare un unico percorso tra quelli che partono dalla radice, scendendo ad ogni nodo incontrato che non contiene il valore dato a sinistra o a destra a seconda che il valore dato sia minore o maggiore, rispettivamente, della chiave contenuta nel nodo
- La complessità della ricerca dipende quindi dalla profondità dell'albero

Esercizio 1

- Implementare (in linguaggio C) la struttura dati che permetta di implementare un albero binario
- Implementare gli algoritmi di
  - inserimento
  - visita in ordine anticipato, simmetrico, differito
  - ricerca (dato il valore dell'informazione restituire il Nodo)
  - eliminazione di un nodo

Albero binario – linguaggio C

```
struct Nodo{

 int key;     // informazione associata al Nodo

 struct Nodo* left, right; // sottoalbero sn e ds

};

// L'albero è un puntatore alla radice
// o NULL se vuoto)

typedef Nodo* AlberoBin;
```

Albero binario in C - utilizzo

```
// Creazione di una lista di due nodi
struct Nodo* n1 = malloc(sizeof(Nodo));
struct Nodo* n2 = malloc(sizeof(Nodo));
struct Nodo* n3 = malloc(sizeof(Nodo));
AlberoBin alb;
n1->key = 33; n1->left = n2; n1->right = n3;
n2->key = 12; n2->left = NULL; n2->right = NULL;
n3->key = 45; n3->left = NULL; n3->right = NULL;
alb = n1;
// Inserimento di un nuovo nodo come radice
Struct Nodo* n4 = malloc(sizeof(Nodo));
n4->key = 14; n4->left = alb; n4->right = NULL;
alb = n4;
```

Esercizio 2

- Implementare (in linguaggio C) la struttura dati che permetta di implementare un albero binario di ricerca
- Implementare gli algoritmi di
  - inserimento
  - visita in ordine anticipato, simmetrico, differito
  - ricerca (dato il valore dell'informazione restituire il Nodo)
  - eliminazione di un nodo (vedi suggerimenti)

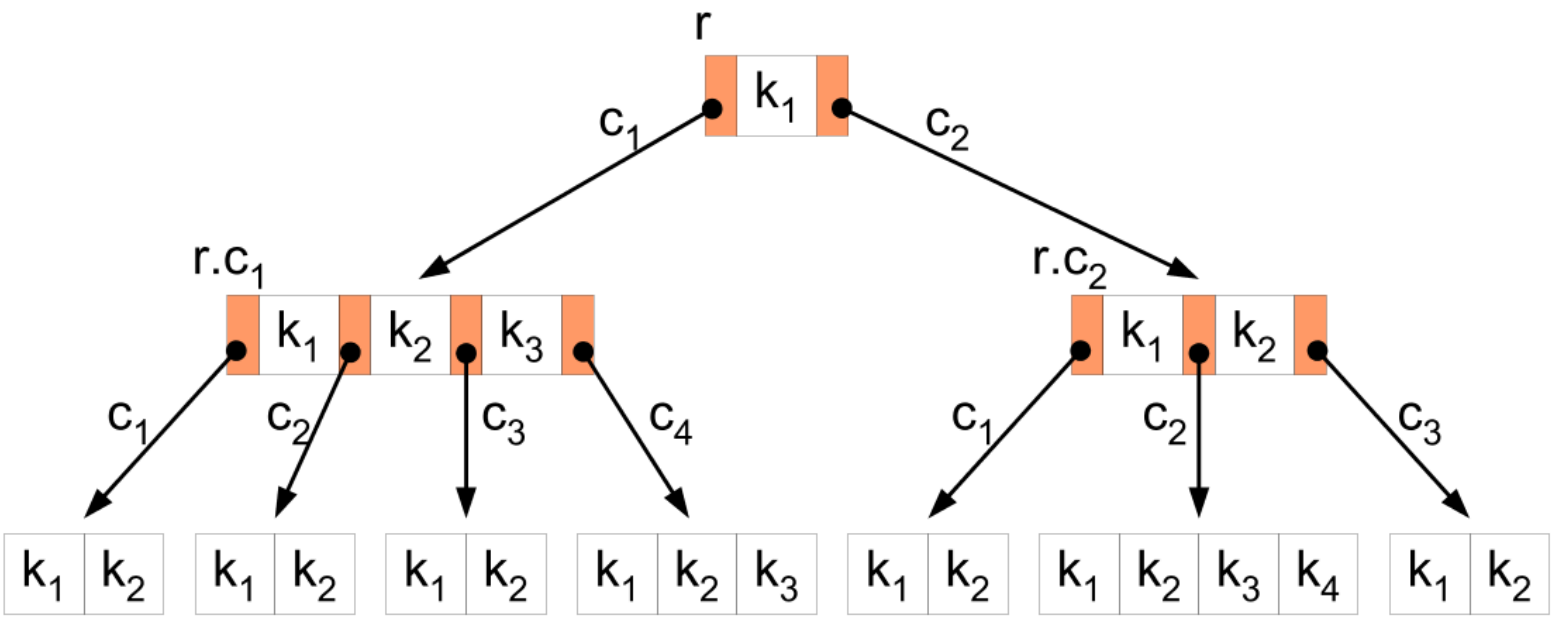
Eliminazione di un nodo

- L'algoritmo di rimozione di un valore da un albero binario di ricerca deve garantire che l'albero binario ottenuto a seguito della rimozione sia ancora di ricerca.
- Se il nodo contenente il valore da rimuovere è una foglia, basta eliminarlo.
- Se il nodo contenente il valore da rimuovere ha un solo figlio, basta eliminarlo collegando suo padre direttamente a suo figlio.
- Se il nodo contenente il valore da rimuovere ha ambedue i figli, si procede sostituendone il valore con quello del nodo più a destra del suo sottoalbero sinistro, in quanto tale nodo contiene la massima chiave minore di quella del nodo da rimuovere (in alternativa, si può prendere il nodo più a sinistra del sottoalbero destro)

B-Albero (B-Tree)

- Struttura dati che permette la rapida localizzazione dei file (Records o keys)
- Deriva dagli alberi di ricerca, in quanto ogni chiave appartenente al sottoalbero sinistro di un nodo è di valore inferiore rispetto a ogni chiave appartenente ai sottoalberi alla sua destra
- E' garantito il bilanciamento: per ogni nodo, le altezze dei sottoalberi destro e sinistro differiscono al più di una unità
- Utilizzati spesso nell'ambito dei database, in quanto permettono di accedere ai nodi in maniera efficiente sia nel caso essi siano disponibili in memoria centrale o in memoria di massa

B-Tree


r
k1
c1
c2
r.c1
k1
k2
k3
r.c2
k1
k2
c1
c2
c3
c4
c1
c2
c3
k1
k2
k1
k2
k1
k2
k1
k2
k3
k1
k2
k1
k2
k3
k4
k1
k2