



Federica



Facoltà di Scienze  
Matematiche  
Fisiche Naturali

## Laboratorio di Algoritmi e Strutture Dati

Prof. Aniello Murano

### Alberi Binari di Ricerca

Corso di Laurea  
Codice insegnamento  
Email docente  
Anno accademico


Informatica  
13917  
murano@na.infn.it  
2007/2008

Lezione numero: 11


Parole chiave: **Alberi Binari**, **Ricerca Binaria**,  
**Visite di Alberi**

[next](#)



Federica

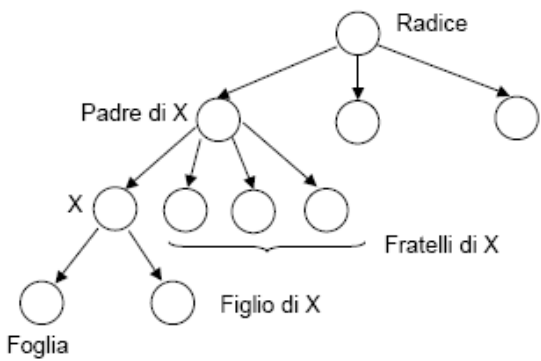


Facoltà di Scienze  
Matematiche  
Fisiche Naturali

16/11/2007

## Alberi

L'albero è un tipo astratto di dato utilizzato per rappresentare relazioni gerarchiche tra oggetti.



[back](#)

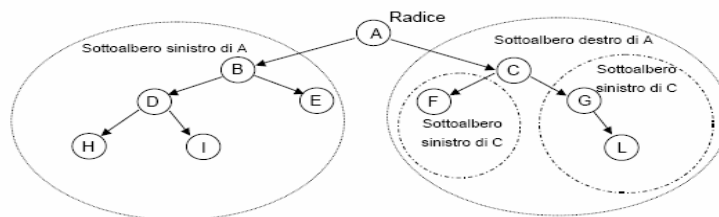
X

[next](#)

## Alberi binari

**Un albero binario può essere facilmente rappresentato in modo ricorsivo. Infatti, un albero**

- è un oggetto vuoto (cioè è un insieme vuoto di nodi); oppure
- è formato da un nodo A (chiamato radice) e da due sottoalberi, a loro volta alberi binari, chiamati rispettivamente sottoalbero sinistro e sottoalbero destro.



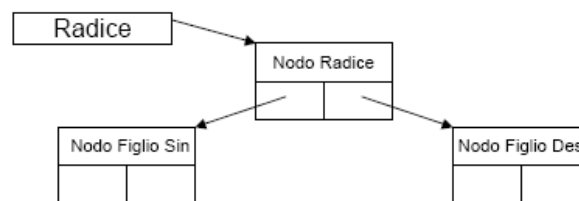
back



next

## Rappresentazione di un albero binario

**Per rappresentare un albero binario si può usare la seguente struttura ricorsiva:**




```
struct nodo { int inforadice;
              struct nodo *sinistro,*destro;
            };
struct nodo *radice;
```

back



next



Federica

16/11/2007

5

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

### Primitive sugli alberi binari

- Per controllare se un nodo è vuoto possiamo usare la seguente funzione:


```
int vuoto (struct nodo *rad)
{
    if(rad) return 0;
    else return 1;
}
```

*rad != 0  
cioè se possiede  
un indirizzo non  
è vuoto*

- Per sapere il valore di un nodo possiamo usare la seguente funzione che ritorna 0 se l'albero è vuoto, altrimenti memorizza nella variabile **val** il valore del nodo

```
int radice(struct nodo *rad, int *val)
{
    int ok=0;
    if !(vuoto(rad)) → se = 1
    {
        *val=rad->inforadice;
        ok=1;
    }
    return ok;
}
```

back
✖
next



Federica

16/11/2007

6

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

### Altre Primitive


**Per avere il punt. al figlio sinistro (destro) di un nodo:**

```
struct nodo *sinistro (struct nodo *rad)
{
    struct nodo *risultato=NULL;
    if !(vuoto(rad)) risultato=rad->sinistro;
    return risultato;
}
```

**Per costruire un nodo (o un albero a partire da due sottoalberi):**

```
struct nodo * costruisci(struct nodo *s, int r, struct nodo *d)
{
    struct nodo *aux;
    aux=(struct nodo*)malloc(sizeof(struct nodo));
    if (aux) {
        aux->inforadice=r;
        aux->sinistro=s; aux->destro=d; }
    return aux;
}
```


back
✖
next



Federica

16/11/2007

7



Facoltà di Scienze  
Matematiche  
Fisiche Naturali


## Visita di un albero binario

**Oltre alle operazioni primitive, si definiscono delle operazioni di visita ovvero di analisi dei nodi di un albero in determinato ordine.**

**Di seguito analizziamo le seguenti visite di un albero:**

- Visita in Preordine
- Visita in Ordine
- Visita in Postordine


back
✖
next



Federica

16/11/2007

8

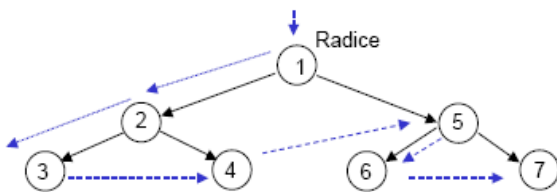


Facoltà di Scienze  
Matematiche  
Fisiche Naturali

## Visita in Preordine

**Nella visita in Preordine, se l'albero non è vuoto:**

- Si analizza la radice dell'albero;
- Si visita in preordine il sottoalbero sinistro;
- Si visita in preordine il sottoalbero destro.



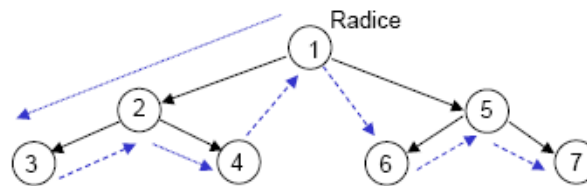
**Nella visita in preordine del precedente albero i nodi verrebbero visitati nel seguente ordine: 1, 2, 3, 4, 5, 6, 7**

back
✖
next

## Visita in Ordine

**Nella visita in ordine, se l'albero non è vuoto:**

- Si visita in ordine il sottoalbero sinistro;
- Si analizza la radice dell'albero;
- Si visita in ordine il sottoalbero destro.



**Nella visita in ordine del precedente albero i nodi verrebbero visitati nel seguente ordine: 3, 2, 4, 1, 6, 5, 7**

back

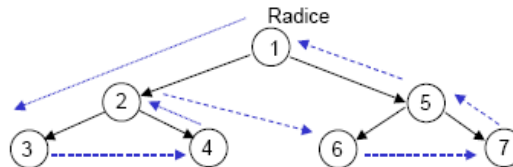


next

## Visita Postordine

**Nella visita in postordine, se l'albero non è vuoto:**

- Si visita in postordine il sottoalbero sinistro;
- Si visita in postordine il sottoalbero destro;
- Si analizza la radice dell'albero.



**Nella visita in ordine del precedente albero i nodi verrebbero visitati nel seguente ordine: 3, 4, 2, 6, 7, 5, 1**

back



next



Federica

16/11/2007

11

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

### Codice per la visita di un albero

```


void visita_in_preordine(struct nodo *radice) {
    if(radice) {    printf("%d ",radice->inforadice);
                    visita_in_preordine(radice->sinistro);
                    visita_in_preordine(radice->destra); } }

void visita_in_ordine(struct nodo *radice){
    if(radice) {    visita_in_ordine(radice->sinistro);
                    printf("%d ",radice->inforadice);
                    visita_in_ordine(radice->destra); } }

void visita_in_postordine(struct nodo *radice) {
    if(radice) {    visita_in_postordine(radice->sinistro);
                    visita_in_postordine(radice->destra);
                    printf("%d ",radice->inforadice); } }

```

back
✖
next



Federica

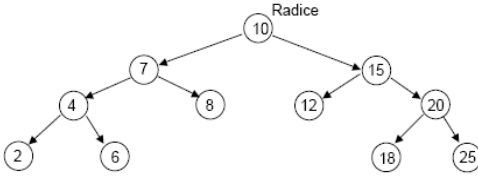
16/11/2007

12

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

### Alberi binari di ricerca (ABR)

- Un albero binario di ricerca (ABR) è un albero binario in cui per ogni nodo dell'albero  $N$  tutti i nodi del sottoalbero sinistro di  $N$  hanno un valore minore o uguale di quello di  $N$  e tutti i nodi del sottoalbero destro hanno un valore maggiore di quello del nodo  $N$ .




```

graph TD
    10((10)) --> 7((7))
    10 --> 15((15))
    7 --> 4((4))
    7 --> 8((8))
    4 --> 2((2))
    4 --> 6((6))
    15 --> 12((12))
    15 --> 20((20))
    20 --> 18((18))
    20 --> 25((25))


```

- Il vantaggio principale di tale organizzazione è nella **ricerca**.
- Ogni volta che bisogna ricercare un elemento, il confronto del valore di un nodo dell'albero permette di eliminare dalla fase di ricerca o il sottoalbero corrente di destra o quello di sinistra.

back
✖
next

 Federica

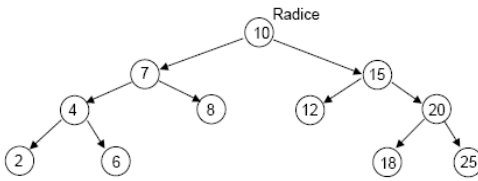
16/11/2007

 13

Facoltà di Scienze  
Matematiche  
Fisiche Naturali


### Osservazione

**Dato un ABR, la visita in "ordine" restituisce una lista ordinata crescente dei valori contenuti nell'albero**




**Nella visita in ordine del precedente albero, i nodi sono infatti visitati nell'ordine: 2,4,6,7,8,10,12,15,18,20,25**

back ✖ next

 Federica

16/11/2007


 14

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

### Esercizio

**Scrivere una funzione in linguaggio C che preso in input un albero binario con  $n$  elementi valuti in tempo  $O(n)$  se l'albero è un ABR.**

back ✖ next



Federica

16/11/2007


15

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

## Ricerca in un albero binario di ricerca


- Per trovare un numero R si procede nel seguente modo:
  1. Se l'albero è vuoto l'elemento non è presente;
  2. Se la radice dell'albero == R l'elemento è stato trovato;
  3. Se la radice dell'albero > R la ricerca viene condotta nel sottoalbero sinistro;
  4. Altrimenti la ricerca viene condotta nel sottoalbero destro;

R=6



- La ricerca può essere realizzata mediante una funzione ricorsiva che nei casi 3 e 4 invoca se stessa.

back
✖
next



Federica

16/11/2007

16

Facoltà di Scienze  
Matematiche  
Fisiche Naturali

## Ricerca in un albero binario di ricerca


### Versione iterativa della ricerca

```

int ricerca (struct nodo *radice, int r)
{
    int trovato=0;
    while(radice && trovato==0)
    {
        if(radice->inforadice==r)
            trovato=1; /* Trovato */
        else if(radice->inforadice > r)
            /* Cerca nel sottoalbero sinistro */
            radice=radice->sinistro;
        else /* Cerca nel sottoalbero destro */
            radice=radice->destra;
    }
    return trovato;
}
  
```


back
✖
next




**Federica**  
UNIVERSITÀ

16/11/2007

17



**Facoltà di Scienze  
Matematiche  
Fisiche Naturali**

## Ricerca in un albero binario di ricerca

### Versione ricorsiva della ricerca


```

int ricerca (struct nodo *radice, int r)
{
    int trovato=0;
    if !(vuoto (radice)) /*else non trovato poiché ABR vuoto */
    {
        if(radice->inforadice==r) return 1; /* Trovato */
        else if(radice->inforadice > r) /* Cerca nel sottoalbero sx */
            trovato=ricerca(radice->sinistro,r);
        else /* Cerca nel sottoalbero destro */
            trovato=ricerca(radice->destro,r);
    }
    return trovato;
}
        
```

back


✖

next


**Federica**  
UNIVERSITÀ

16/11/2007

18




**Facoltà di Scienze  
Matematiche  
Fisiche Naturali**

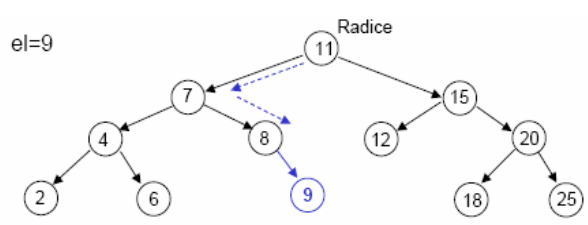
## Inserimento di un nuovo nodo in un ABR

- Se l'albero è vuoto, viene creato un nuovo nodo;
- Se l'elemento è minore o uguale alla radice dell'albero, l'inserimento va fatto nel sottoalbero sinistro;
- Se l'elemento è maggiore o uguale alla radice dell'albero, l'inserimento va fatto nel sottoalbero destro;

radice




e|=9




back

✖

next


**Federica**  
UNIVERSITÀ

16/11/2007



Facoltà di Scienze  
**Matematiche**  
 Fisiche Naturali

### Inserimento di un nuovo nodo in un ABR


```

struct nodo *inserisci (struct nodo *radice, int e)
{ struct nodo *aux;
  if (vuoto(radice)) /* Creazione di un nuovo nodo */
  {
    aux=(struct nodo*)malloc(sizeof(struct nodo));
    if(aux)
    {
      aux->info=e;
      aux->sx=aux->dx=NULL;
      radice=aux;
    }
    else printf("Memoria non allocata");
  }
  else if(e<radice->info) radice->sx = inserisci(radice->sx, e);
  else if(e>radice->info) radice->dx = inserisci(radice->dx, e);
  /* altrimenti il valore è già nell'ABR e non si fa niente */
  return radice;
}
```


**struttura**  

info	
sx	dx

back
✖
next


**Federica**  
UNIVERSITÀ

16/11/2007



Facoltà di Scienze  
**Matematiche**  
 Fisiche Naturali

### Inserimento di un nuovo nodo tramite l'uso di puntatori a puntatori

```

void inserisci (struct nodo **radice, int e)
{ struct nodo *aux;
  if(*radice==NULL)
  { /* Creazione di un nuovo nodo */
    aux=(struct nodo*)malloc(sizeof(struct nodo));
    if(aux)
    {
      aux->info=e;
      aux->sx=aux->dx=NULL;
      *radice=aux;
    }
    else printf("Memoria non allocata");
  }
  else if((*radice)->info>e) inserisci(&(*radice)->sx,e);
  else if((*radice)->info<e) inserisci(&(*radice)->dx,e);
}
```

**struttura**  

info	
sx	dx

back
✖
next

**Osservazioni sulla slide precedente**

- L'invocazione della funzione **inserisci** dipende da come è stato definito l'ABR nella funzione chiamante (che può anche essere main). Di seguito mostriamo due possibili casi:
  - L'ABR è definito con singolo puntatore:
 

```
struct nodo * radice = NULL;
```

 allora la funzione inserisci sarà invocata con
 

```
inserisci(&(radice),valore);
```
  - L'ABR è definito con doppio puntatore:
 

```
struct nodo **radice;
```

```
radice=(struct nodo**)malloc(sizeof(struct nodo));
```

```
* radice=NULL;
```

 allora saraprima la funzione inserisci sarà invocata con
 

```
inserisci(radice,valore);
```

back ✖ next

**Ricerca minimo in un ABR**

- Se il sottoalbero sinistro è vuoto, il minimo è la radice.
- Altrimenti il minimo è da cercare nel sottoalbero sinistro

Radice

Radice

```
int ricerca_minimo (struct nodo *radice)
{ /* per semplicità assumiamo tutti i valori dell'ABR positivi*/
  int min=0;
  if !(vuoto(radice)) {
    if(radice->sx==NULL) minimo=radice->info;
    else min= ricerca_minimo(radice->sx);
  }
  return min;
}
```

back ✖ next

This document was created with Win2PDF available at <http://www.win2pdf.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.  
This page will not be added after purchasing Win2PDF.