

Esempi di strutture dati e algoritmi

Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Strutture dati e algoritmi



- Esistono delle strutture dati e degli algoritmi che sono usate frequentemente in informatica
- Le strutture dati includono insiemi, pile, code, liste, alberi, grafi,....
- Gli algoritmi includono l'ordinamento, la visita dei grafi, la ricerca di dati in vettori, ...
- Nel seguito studieremo alcune di queste strutture e algoritmi

Franco Scarselli

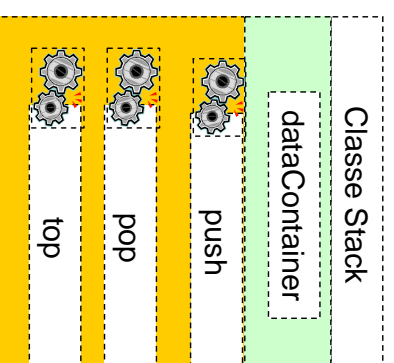
Fondamenti di Informatica I, 2005-2006

Insiemi

- Gli insiemi sono insiemi strutture dati in cui si possono inserire e rimuovere elementi
 - Vi sono molti modi di implementare gli insiemi che differiscono a seconda del modo in cui si possono inserire e rimuovere gli elementi, l'efficienza delle operazioni, l'occupazione di memoria, :
 - **Pile**: gli elementi si possono rimuovere solo in ordine inverso a quello di inserimento
 - **Code**: gli elementi si possono rimuovere solo nello stesso ordine in cui sono stati inseriti
 - **Vettori**: gli elementi possono essere rimossi e inseriti in qualsiasi ordine. Occorre conoscere fin dall'inizio il numero di elementi dell'insieme
 - **Liste**: gli elementi possono essere rimossi e inseriti in qualsiasi ordine. La gestione delle liste è più complessa
- Fondamenti di Informatica I, 2005-2006

La pila (stack)

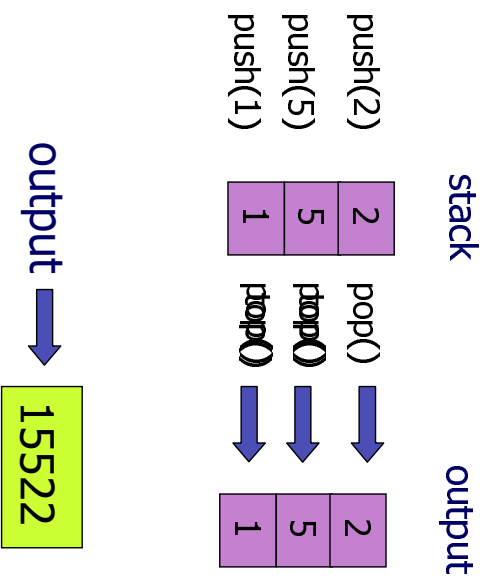
- La pila simula un contenitore in cui gli elementi si trovano uno sopra l'altro
- La pila è un oggetto con tre metodi
 - push: inserisce un elemento nella pila
 - pop: estrae un elemento dalla pila
 - top: restituisce l'elemento in testa alla pila senza rimuoverlo



- La pila implementa una strategia **FILO** (First In, Last Output)

La pila: un esempio

```
Stack myStack=new Stack(100);
myStack.push(1);
myStack.push(5);
myStack.push(2);
int a=myStack.pop();
System.out.print(a);
System.out.print(myStack.top());
System.out.print(myStack.pop());
System.out.print(myStack.top());
System.out.print(myStack.pop());
```



Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Implementazione di una pila

- Per implementare una pila si utilizza un array che funge da contenitore e un indice che indica dove si trova l'elemento in testa alla pila
- Il costruttore permette di inizializzare il contenitore con un numero di elementi voluto

```
class Stack{
    private int dataContainer[];
    private int head;
    Stack(int length) {
        dataContainer=new int[length];
        head=0;
    }
    ...
}
```

Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Implementazione di una pila: pop, push e top

```
class Stack{  
    ...  
    public int top() {  
        return dataContainer[head];  
    }  
    public int pop() {  
        return dataContainer[head];  
        head--;  
    }  
    public void push(int in) {  
        dataContainer[head]=in;  
        head++;  
    }  
}
```

Franco S
005-2006

Quando si usano le pile?

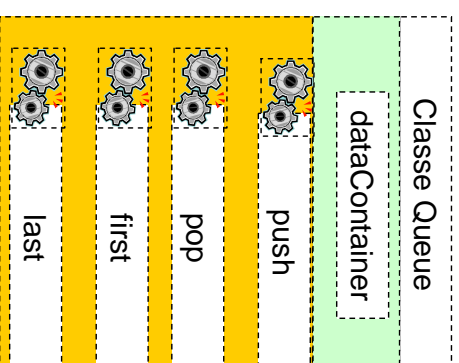
- Si usano in tipicamente nei compilatori e in vari problemi in informatica in cui l'ultimo elemento inserito deve estratto per primo

Esempio

- Un sistema operativo è in grado di gestire processi a priorità diversa (ad es. da priorità 1 a priorità 10). Se mentre si sta eseguendo un processo a ne arriva uno a priorità più alta, quello in esecuzione viene inserito in una pila e si passa all'esecuzione di quello a priorità più alta. Eventualmente successivamente ne può arrivare uno a priorità ancora per cui quelle precedentemente subentrato viene messo in testa alla pila. In seguito, quando il processo a priorità maggiore sarà terminato, gli processi verranno ripresi in ordine di priorità, cioè inverso a come sono stati fermati.

La coda (queue)

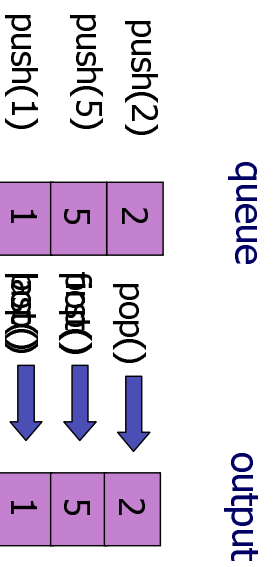
- La coda simula un contenitore in cui gli elementi si trovano in fila, uno dietro l'altro
- La pila è ha metodi simili alla pila
 - push: inserisce un elemento nella coda
 - pop: estrae un elemento dalla coda
 - first: restituisce l'elemento in testa alla coda senza rimuoverlo
 - last: restituisce l'elemento in coda alla coda senza rimuoverlo



- La pila implementa una strategia **FIFO** (First In, First Output)
 Franco Scarselli
 Fondamenti di Informatica I, 2005-2006

La coda: un esempio

```
Stack myQueue=new Stack(100);
myQueue.push(2);
myQueue.push(5);
System.out.print(myQueue.first());
System.out.print(myQueue.last());
System.out.print(myQueue.pop());
myQueue.push(1);
System.out.print(myQueue.pop());
System.out.print(myQueue.pop());
System.out.print(myQueue.pop());
```



output → 25251

Implementazione di una coda

- Per implementare una coda si utilizza un array che funge da contenitore
- Si utilizzano inoltre due indici, che indicano dove si trovano il primo elemento che è stato inserito nella coda e l'ultimo elemento
- L'implementazioni di `first()` e `last()` sono immediate

Franco Scarselli

Fondamenti di In

```
class Queue{
    private int dataContainer[];
    private int first, last;

    Stack(int length) {
        dataContainer=new int[length];
        first=0;
        last=length-1;
    }

    public int first() {
        return dataContainer[first];
    }

    public int last() {
        return dataContainer[last];
    }
}
```

Implementazione di una coda: `pop()` e `push()`

- Per implementare `push()` e `pop()`, si usa la coda in maniera circolare
- Quando `first` arrivano alla testa dell'array si riparte dal fondo ...
- Quando `last` arriva al fondo dell'array si riparte dall'inizio...

Franco Scarselli

Fondame

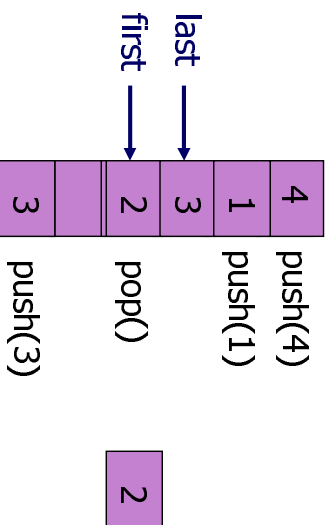
```
class Queue{
    ...
    public void push(int in) {
        last++;
        if (last==dataContainer.length){
            last=0;
        }
        dataContainer[last]=in;
    }

    public int pop() {
        return dataContainer[first];
        first++;
        if (first==dataContainer.length){
            first=0;
        }
    }
}
```

La coda circolare.... come funziona

- Si supponga di avere una coda di 6 elementi che inizialmente contiene 2,3
- Si usa una coda circolare

```
myQueue.push(1);  
myQueue.push(4);  
myQueue.push(3);  
myQueue.pop(2);
```

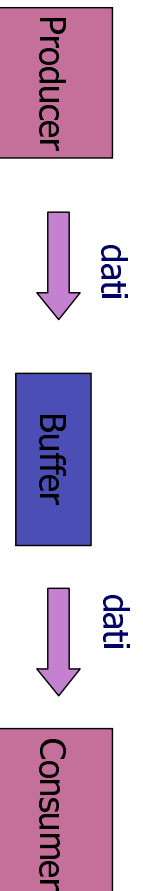


Quando si usano le code?

- Si usano, ad esempio, per realizzare buffer o canali di comunicazione

Esempio

- Due servizi, due processi, due periferiche, due risorse comunicano fra loro. Un servizio (producer) produce dei dati che vengono ricevuti e elaborati dall'altro (consumer). Poiché i due servizi possono viaggiare a velocità diverse occorre avere un buffer (implementato con una coda) che contenga temporaneamente i dati se il consumer non riesce ad elaborarli in tempo.



I vettori (array)

- I vettori sono strutture dati predefinite in Java e in quasi tutti i linguaggi
- I vettori permettono di implementare altre strutture dati: ad esempio, pile e code
- I vettori permettono di implementare anche insiemi in cui si inserisce e si rimuove gli elementi in qualsiasi ordine:
 - se agli elementi da inserire è possibile associare un numero unico (**allocazione per chiave, ad accesso diretto**), allora si mette l'elemento nella posizione corrispondente all'indice
 - altrimenti si può mettere ogni oggetto in un posto qualsiasi (**allocazione sequenziale**)

Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Un esempio: la biblioteca

- Un libro è definito da un titolo, gli autori e un numero di catalogo. Il numero di catalogo è un numero sequenziale assegnato ai libri mano che arrivano in biblioteca
- Per l'implementazione si usa un array di libri. In posizione n si mette il libro avente il numero di catalogo (**chiave**) n.

```
class libro{
    String titolo, autori;
    int num;

    libro(String t, String a, int n){
        titolo=t;
        autori=a;
        num=n;
    }
}
```

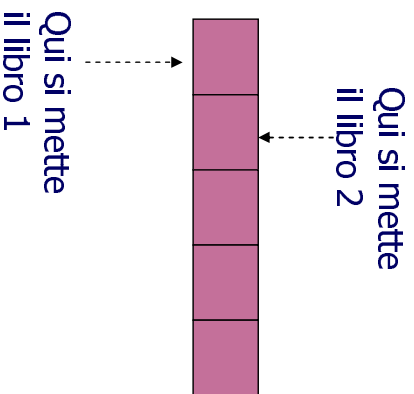
```
class biblioteca{
    private libro insiemelibri;

    public biblioteca(int n){
        insiemelibri= new libro[n];
        ...
    }
}
```


Un esempio: la biblioteca

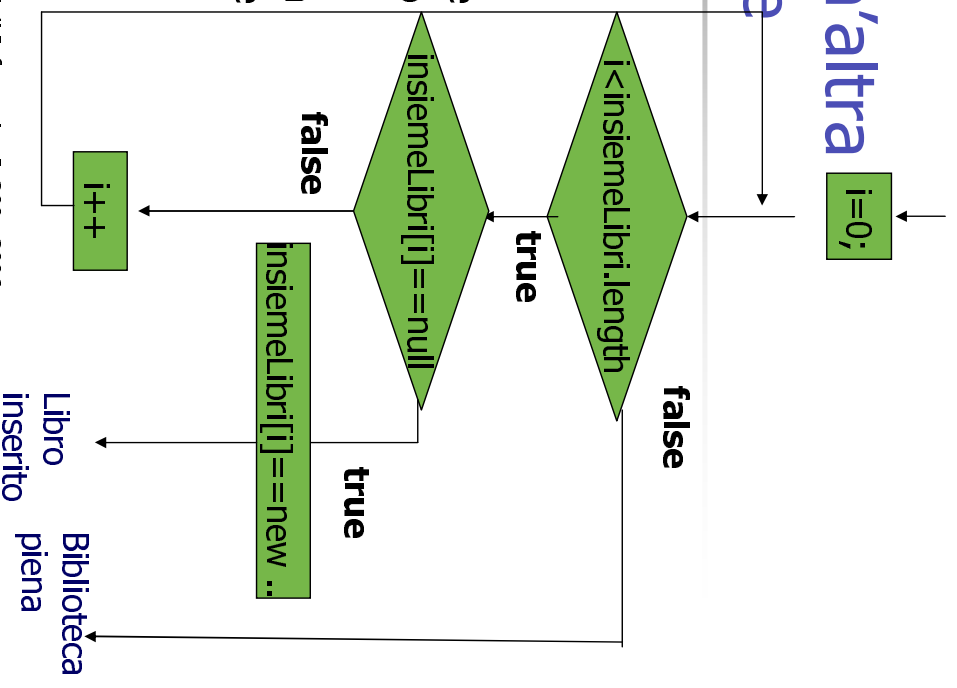
- Poiché ogni libro può occupare solo una posizione dell'array, la rimozione e l'inserimento sono banali

```
class biblioteca{  
    private libro insiemeLibri;  
  
    public void insert(String titolo, String autori, int num){  
        insiemeLibri[num] = new libro(titolo, autori,num);  
    }  
  
    public void remove(int num){  
        insiemeLibri[num]=null;  
    }  
}
```



La biblioteca: un'altra implementazione

- In una seconda implementazione della biblioteca ogni libro può assumere un qualsiasi posto
- Per l'inserimento si cerca la prima posizione libera (in cui c'è null) e ci si inserisce il libro



La biblioteca: un'altra implementazione II

In Java

```
class biblioteca{  
  
    public void insert(String titolo, String autori, int num){  
        int i;  
        for(i=0;i<insiemelibri.length;i++){  
            if(insiemelibri[num]==null) {  
                insiemeLibri[num]=new libro(titolo,autori,num);  
                break;  
            }  
        }  
        if(i==insiemelibri.length){  
            System.out.println("Biblioteca piena...");  
        }  
    }  
}
```

Franco Scarsi

La biblioteca: un'altra implementazione III

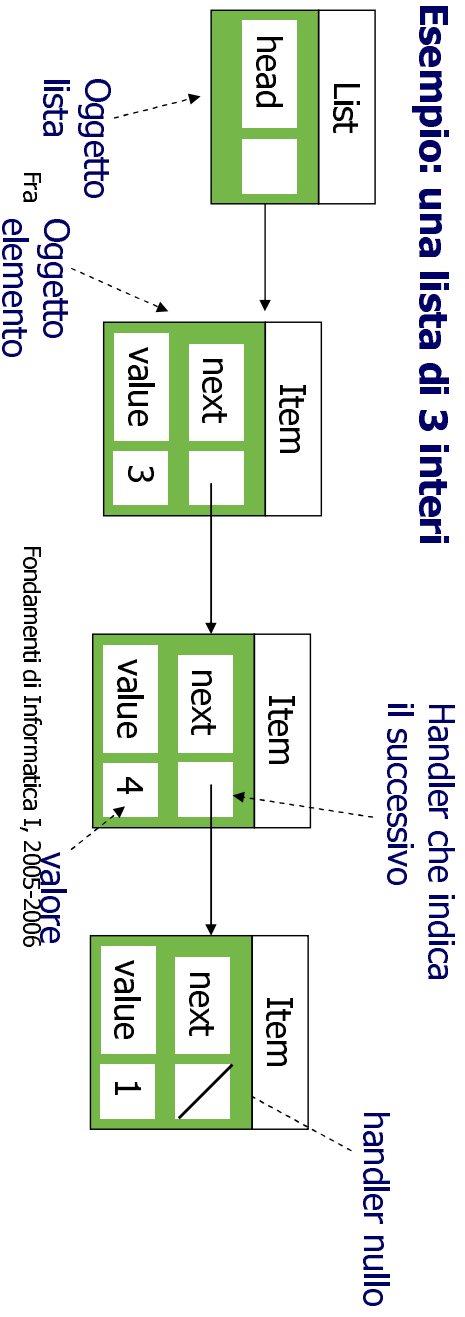
La rimozione funziona in modo simile all'inserimento: occorre cercare l'elemento da eliminare

```
public void remove( int num){  
    int i;  
    for(i=0;i<insiemelibri.length;i++){  
        if(insiemelibri[num].num==v) {  
            insiemeLibri[num]=null;  
            break;  
        }  
    }  
    if(i==insiemelibri.length){  
        System.out.println("Libro non trovato...");  
    }  
}
```

Le liste

- La lista è una sequenza di elementi concatenati
- Ogni elemento contiene
 - Il valore del contenuto
 - L'handler dell'elemento successivo

Esempio: una lista di 3 interi

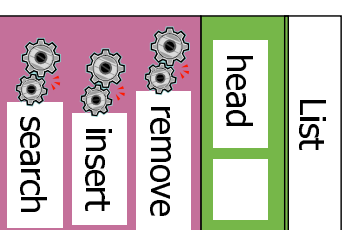
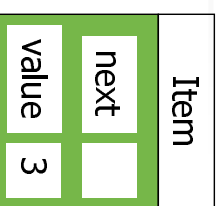


Implementazione di una lista

- Ogni elemento della lista viene implementato con una classe...
- Ovviamente anche la lista è un oggetto che contiene l'handler del primo elemento:
 - I metodi tipici sono inserimento, ricerca e rimozione di un elemento

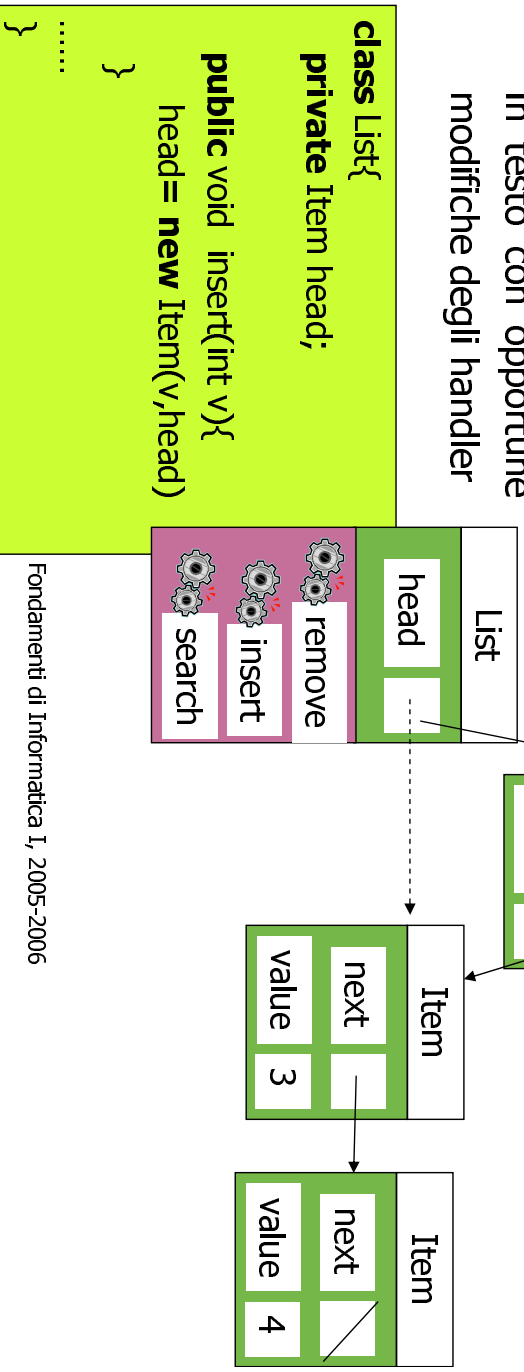
```
class Item{
    int value;
    Item next;
    Item(int v, Item n){
        value=v;
        next=n;}
}

class List{
    Item head;
    void insert(int v){
        .....
    }
}
```



Implementazione di una lista: inserimento

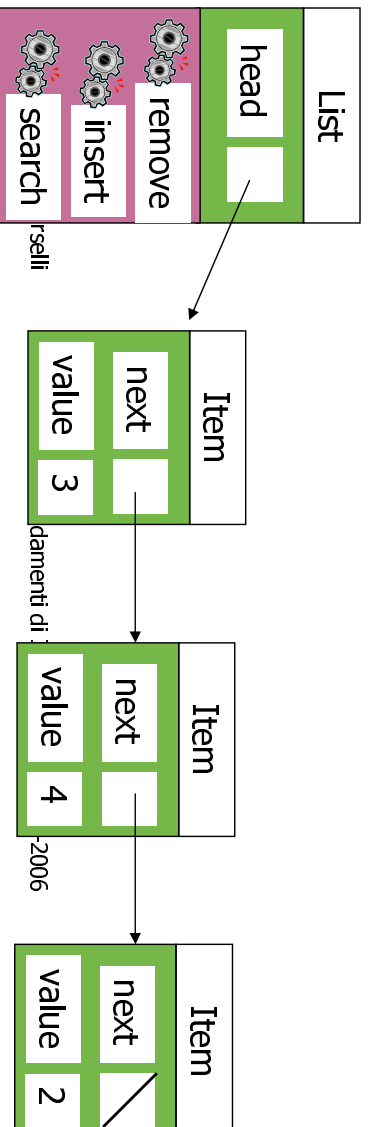
- L'inserimento
tipicamente avviene
aggiungendo l'elemento
in testo con opportune
modifiche degli handler



Implementazione di una lista: ricerca

- Per cercare un valore occorre scorrere tutta la lista partendo da head
- Se si trova l'elemento che contiene il valore si restituisce l'handler altrimenti si restituisce il valore **null**

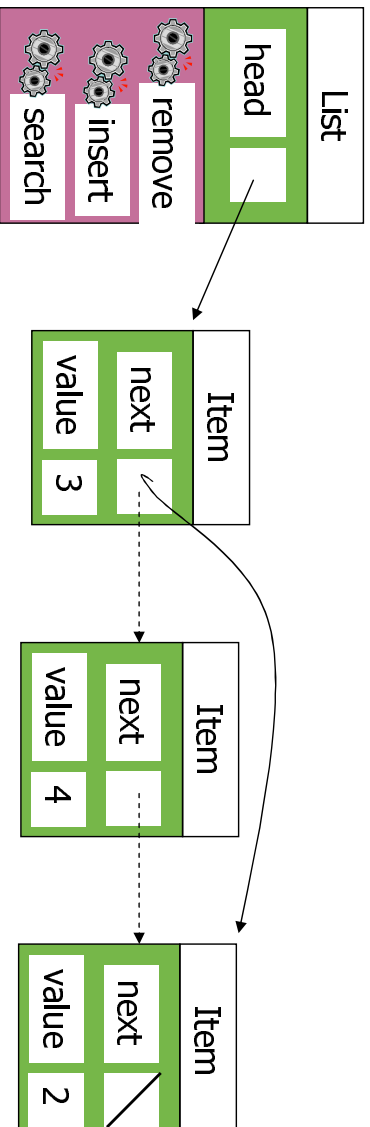
```
class List{  
    .....  
    public Item search(int v){  
        .....  
    }  
}
```



Implementazione di una lista: rimozione

- Per rimuovere un elemento occorre prima scorrere tutta la lista per individuarlo, poi si eliminaaggiustando gli handler
- Occorre mantenere due handler mentre si scorre la lista

Rimozione di 4



Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Implementazione di una lista: rimozione

```
public void remove(int v){
    Item it=head, previous=null;
    boolean found=false;

    while(it!=null){
        if (it.value==v) {
            found=true;
            break;}
        previous=it;
        it=it.next;}

    if (found){
        if (it==head){head=head.next;}
        else {previous.next=it.next;}
    }
}
```

Mantiene l'handler precedente..

Si sostituiscono gli handler: si deve fare in modo diverso se v è nel primo...

Si fa qualcosa solo se si è trovato v

Franco Sc



Osservazioni

- Vettori e liste permettono di implementare insieme in cui non si conosce l'ordine con cui saranno inseriti/eliminati gli elementi: ad esempio, una biblioteca, una videoteca, una rubrica ...
- Vettori rispetto alle liste: **velocità di accesso**
 - Nelle liste per accedere ad un elemento occorre scorrere la lista fino a quando non si trova l'elemento desiderato
 - con vettori basati su chiave (allocazione per chiave), si può accedere direttamente all'elemento che si desidera
 - Con i vettori non basati su chiave (allocazione sequenziale) occorre scorrere tutto il vettore per accedere un elemento

Franco Scarselli

Fondamenti di Informatica I, 2005-2006



Osservazioni II

- Vettori rispetto alle liste: **occupazione di memoria**
 - Con i vettori occorre conoscere fin dall'inizio il numero (massimo) di elementi da memorizzare: si spreca spazio e/o si rischia che lo spazio non basti
 - Con le liste non serve sapere quanti elementi occorre allocare, ma nelle liste occorre memorizzare anche gli handler che possono occupare un po' di memoria in più.

Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Alberi

- Gli alberi sono una struttura dati fatta di nodi: ogni nodo contiene
 - un valore
 - gli **handler** ad alcuni sottoalberi (**figli**)

Definizioni

- Gli alberi si dicono **n-ari** se hanno n figli
 - Una classe importante di alberi sono quelli binari (due figli)
- Esiste un solo nodo (detto **radice**) da cui si può raggiungere tutti gli altri
- I nodi senza figli sono detti **foglie**
- I livelli di un albero sono il numero di nodi compresi fra la radice e una delle foglie
- Un albero si dice bilanciato se ogni foglia è allo stesso livello.

Franco Scarselli

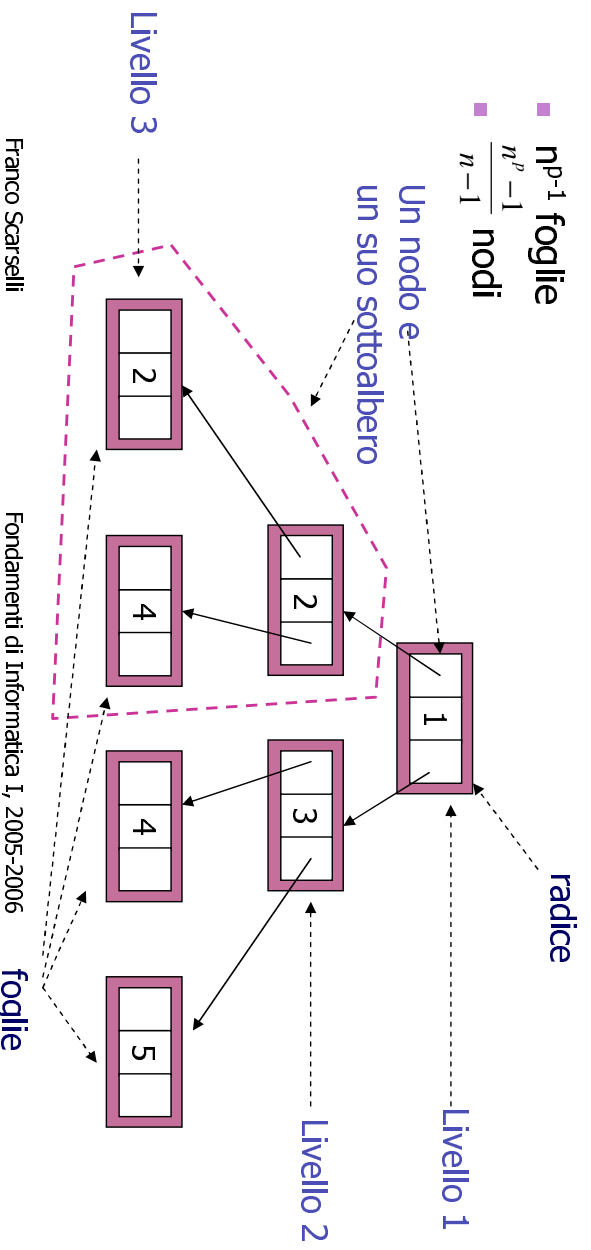
Fondamenti di Informatica I, 2005-2006

Alberi II

- Un albero n -ario bilanciato di profondità p contiene

- n^{p-1} foglie
- $\frac{n^p - 1}{n - 1}$ nodi

Un albero binario bilanciato



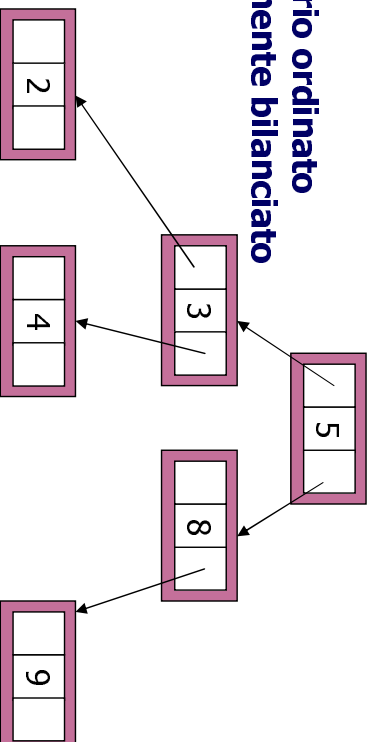
Franco Scarselli

Fondamenti di Informatica I, 2005-2006

Alberi binari ordinati

- Un albero binario è ordinato se per ogni nodo n e il suo valore v_n
 - Il sottoalbero sinistro contiene solo valori minori di v_n
 - Il sottoalbero destro contiene solo valori maggiori di v_n

Un albero binario ordinato non completamente bilanciato



Franco Scarselli

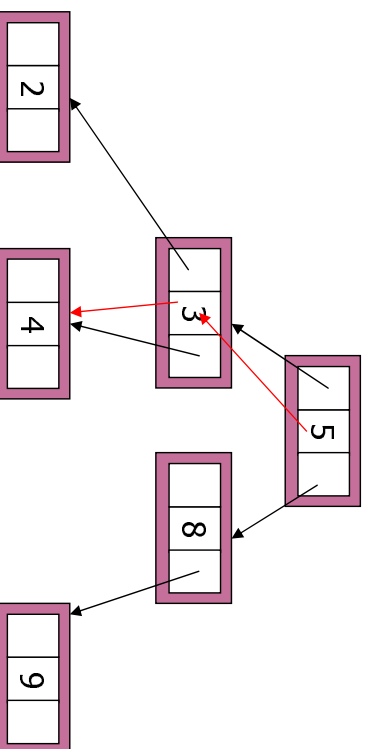
Fondamenti di Informatica I, 2005-2006

Alberi ordinati

- Gli alberi ordinati sono molto importanti e usati in informatica per realizzare insiemi di grandi dimensioni
- Per accedere ad un elemento non occorre visitare tutto l'albero, ma è sufficiente scendere in un ramo dell'albero partendo dalla radice

Per cercare 4,

- si parte dalla radice
- si confronta 4 con 5: si va a sinistra
- si confronta 4 con 3: si va a destra
- si confronta 4 con 4: trovato!



Franco Scarselli

Fondamenti di Informatica I, 2005-2006



Implementazione degli alberi binari

- L'implementazione degli alberi binari usa un oggetto nodo che contiene il valore e i due sottoalberi destro e sinistro
- Gli alberi mettono di solito a disposizione dei metodi per inserire, rimuovere ed, eventualmente, cercare un valore

```
class Node{  
    int value;  
    Node left, right;  
  
    Item(int v, Node l, Node r){  
        value=v;  
        left=l;  
        right=r;}  
}
```

Franco Scarselli

```
class Tree{  
    Node root;  
    void insert(int v){  
        ....  
    }  
    void remove(int v){  
        ....  
    }  
}
```

Fondamenti di Informatica