Introduzione agli Algoritmi e alle Strutture Dati

Alberi

Dr. Emanuela Merelli

Argomenti della lezione

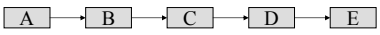
- Tipi di Dato Astratto
  - Albero
- *Concetto di*
  - *Struttura dati dinamiche*
  - *Lista non lineare*

Lista Lineare

è un insieme ordinato di *n* elementi, tutti dello stesso tipo T, ogni elemento ha un **predecessore** e un **successore**

Proprietà:

1. X_1 è il primo elemento della lista
2. X_i è preceduto da xi-1 e seguito da xi+1
3. X_n è l'ultimo elemento della lista
4. N= lunghezza della lista
5. N=0 ⇒ Lista Vuota

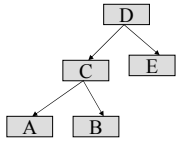


L(A, B, C, D)

Relazione di precedenza: per accedere ad un elemento si deve accedere a tutti quelli che lo precedono

Lista Non Lineare

è una lista i cui elementi possono essere a loro volta una lista



L(D (C (A, B), E))

Lista non Lineare come Tipo di Dato Astratto

Dominio dei valori

- Insieme dinamico basato su elementi di tipo omogeneo

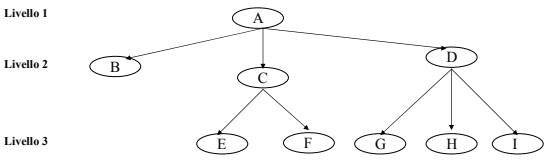
Operazioni su Lista non Lineare

- Creazione della lista non lineare
- Inserire di un elemento
- Visita dell'albero

Alberi

Un albero è un insieme finito di elementi detti nodi di cui uno è chiamato **radice**.

I rimanenti elementi, se esistono, formano insiemi disgiunti chiamati **sottoalberi** della radice, ciascuno dei quali è ancora un **albero**



Alberi

Grado di un nodo è il numero di sottoalberi di quel nodo. Un nodo di grado 0 è detto **nodo terminale** o **foglia.**

Un nodo X è discendente di un nodo Y se esiste una successione di nodi $x_1 \rightarrow x_2 \rightarrow, \ldots \rightarrow x_n$ tale che $Y=x_1 \rightarrow x_2 \rightarrow, \ldots \rightarrow x_n = X$ e x_{i+1} è discendente diretto di x_i. Tale successione prende il nome di **cammino.**

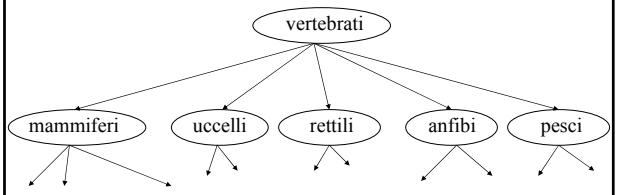
Lunghezza del cammino è il numero di nodi del cammino meno uno.

Livello di un nodo X è uguale ad i+1 se i è il livello del padre di X

Altezza di un albero è il livello massimo delle foglie

Esempi di utilizzo di alberi

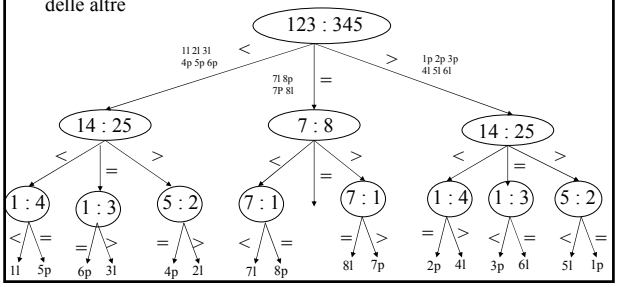
Problemi di classificazione



Problemi di decisione

Problema delle 8 monete

Data 8 monete, ne potrebbe esistere una falsa di diverso peso, utilizzando una bilancia a due piatti, individuare con sole 3 pesate la moneta falsa, e qualora esista se pesa di più o di meno delle altre

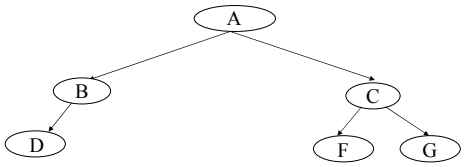


Alberi Binari

Definiamo un albero binario ricorsivamente. Un albero binario è una struttura definita su un insieme finito di elementi detti nodi, che può essere o

- Vuoto, non contiene nodi

È composto da tre insieme disgiunti di nodi: una radice, un albero binario di sinistra e un albero binario di destra

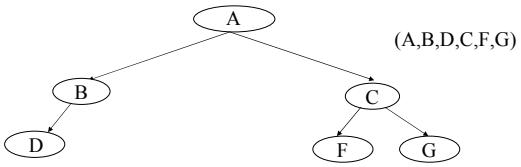


Criteri di attraversamento di alberi

- Visita in ordine anticipato
- Visita in ordine differito
- Visita in ordine simmetrico

Visita in ordine anticipato

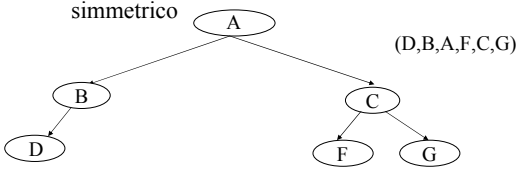
1. Esamina la radice (se N>0 visita sottoalberi)
2. Visita il sottoalbero di sinistra in ordine anticipato
3. Visita il sottoalbero di destra in ordine anticipato



(A,B,D,C,F,G)

Visita in ordine simmetrico

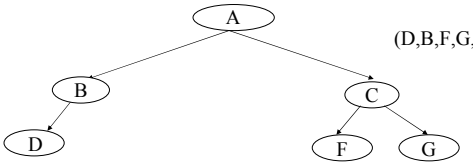
1. Visita il sottoalbero di sinistra in ordine simmetrico
2. Esamina la radice (se N>0 visita sottoalberi)
3. Visita il sottoalbero di destra in ordine simmetrico



(D,B,A,F,C,G)

Visita in ordine differito

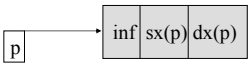
1. Attraversa sottoalbero sinistro in ordine differito
2. Attraversa sottoalbero destro in ordine differito
3. Esamina radice



(D,B,F,G,C,A)

Allocazione di alberi in memoria

Un albero binario di ricerca viene allocato in memoria come una lista non lineare dove ogni elemento ha il formato seguente:




```
Type nodo = RECORD
        inf: CHAR; dx: ^ nodo; sx:^ nodo
        END;
```

Allocazione Dinamica\

```
Type Albero=ARRAY[1..n}
      nodo= RECORD
inf : CHAR; dx:INTEGER; sx:INTEGER
```

Allocazione Statica\

Algoritmo per creare ricorsivamente un albero binario

```
VAR
Radice:^nodo; valore:CHAR;
Radice:=nil;
WRITELN("inserisci lista di caratteri, per terminare 0");
READLN(valore);
WHILE valore <> "0" DO BEGIN  crea(radice, valore); readln(valore) END;
WRITELN("visita anticipata"); anticipata(radice);
WRITELN ("visita differita"); differita(radice);
WRITELN("visita simmetrica"); simmetrica(radice);
END.
```

Esercizio

Progettare le procedure ricorsive: crea, anticipata, differita e simmetrica