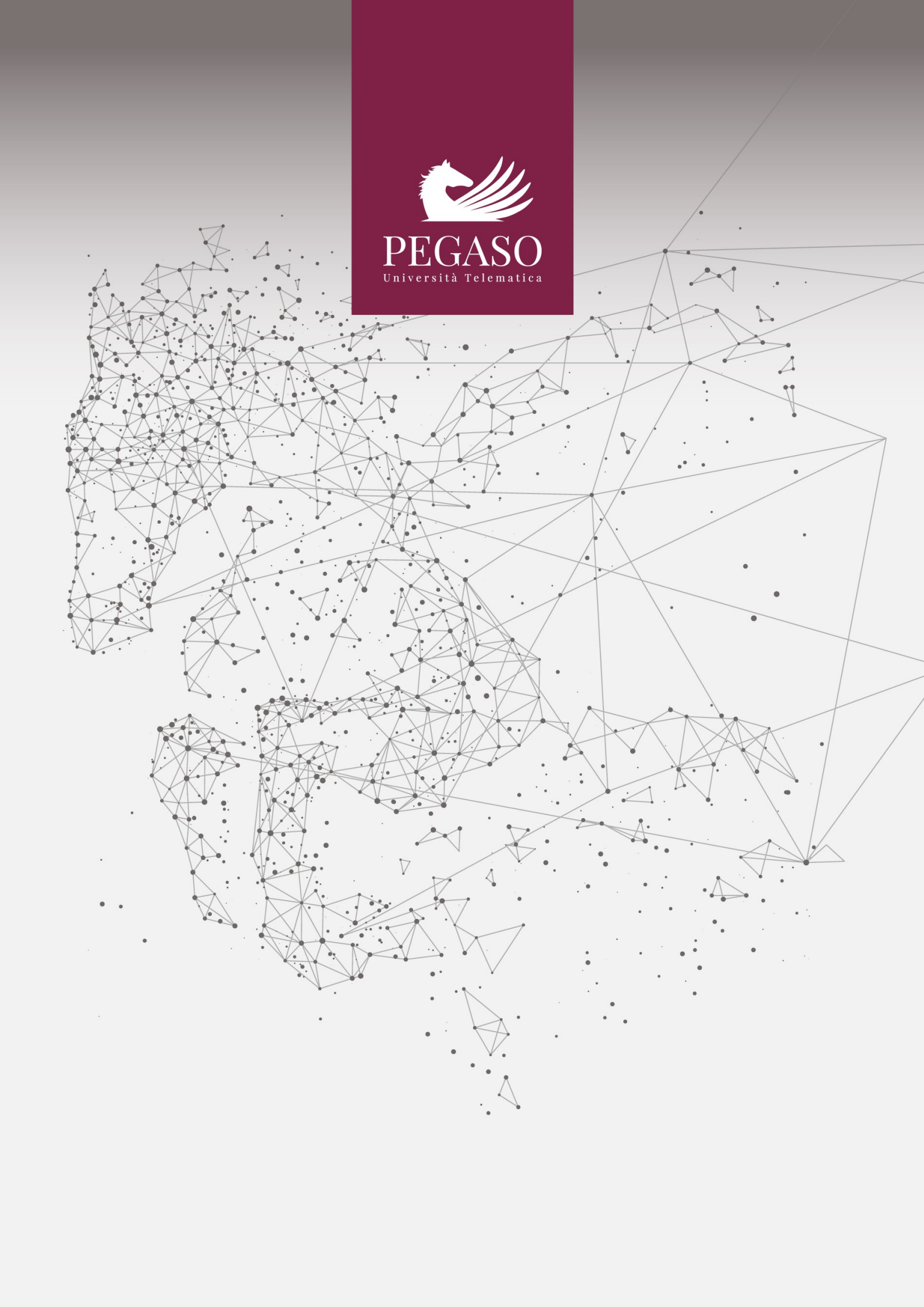
PEGASO
Università Telematica

Indice

1. Introduzione alle strutture dati

Una struttura dati è una rappresentazione organizzata e logica di un insieme di dati in un computer, che permette di accedere, modificare e gestire i dati in modo efficiente.

Le strutture dati sono utilizzate per risolvere problemi specifici, come la gestione di grandi quantità di dati, la ricerca di informazioni specifiche, l'elaborazione di dati e la risoluzione di problemi di ottimizzazione.

Un insieme può essere considerato come una raccolta di elementi senza alcuna organizzazione specifica, ad esempio un insieme di numeri interi, che sono semplicemente una raccolta di numeri senza alcuna organizzazione specifica.

Una struttura dati, invece, fornisce una rappresentazione organizzata e logica dei dati all'interno di un insieme. Ad esempio, una lista è una struttura dati che rappresenta un insieme di elementi in modo ordinato e accessibile tramite indici. Oppure un albero binario è una struttura dati che rappresenta un insieme di elementi in modo gerarchico.

In sintesi, un insieme è una raccolta di elementi mentre una struttura dati è un modo per organizzare e rappresentare questi elementi in modo che possano essere gestiti e utilizzati in modo più efficiente.

I tipi di strutture dati più comuni sono:

- **Array**: una collezione ordinata di elementi, tutti dello stesso tipo, accessibili tramite un indice.
- **Liste**: una collezione ordinata di elementi, di qualsiasi tipo, accessibili tramite un indice. Le liste possono essere singolarmente collegate o a doppia.
- **Stack**: una collezione di elementi che supporta solo due operazioni principali: push (inserimento) e pop (estrazione). Gli elementi vengono inseriti e rimossi dalla cima dello stack.
- **Coda**: una collezione di elementi che supporta solo due operazioni principali: enqueue (inserimento) e dequeue (estrazione). Gli elementi vengono inseriti dalla coda e rimossi dalla testa.
- **Alberi**: una collezione di elementi gerarchicamente organizzati, in cui ogni elemento ha zero o più figli. Ci sono diversi tipi di alberi, come gli alberi binari, gli alberi di ricerca e gli alberi B.
- **Grafi**: una collezione di nodi (o vertici) e archi che collegano i nodi. Ci sono diversi tipi di grafi, come i grafi non orientati, i grafi orientati, i grafi ponderati e i grafi non ponderati.

- **Hash Table**: una struttura dati che utilizza una funzione hash per mappare gli elementi a una posizione specifica in un array. Le tabelle hash sono utilizzate per la ricerca efficiente di elementi in grandi insiemi di dati.

Una tassonomia classica delle strutture dati prevede tre dimensioni ortogonali tra loro:

- Statica o dinamica
- Compatta o sparsa
- Basata o non basata sull'ordinamento delle chiavi

Una struttura **dinamica** è una struttura che è pensata per aggiungere o togliere elementi durante l'esecuzione di un algoritmo; un array ad es. è considerato una struttura **statica**. Possiamo dunque dire che le strutture dati statiche sono quelle in cui la dimensione e la configurazione degli elementi non cambiano durante l'esecuzione del programma, mentre le strutture dati dinamiche sono quelle in cui la dimensione e la configurazione degli elementi possono cambiare durante l'esecuzione del programma.

Quando si parla di struttura **compatta** o **sparsa** ci si riferisce alla posizione fisica degli elementi in memoria; tipicamente una struttura dinamica è sparsa, non si possono cioè fare ipotesi sulla posizione fisica degli elementi in memoria; gli array sono invece una struttura compatta: indipendentemente da dove è memorizzato, possiamo assumere che sia fatto in modo tale da avere n posizioni fisiche vicine tra loro.

A volte ci si riferisce a queste anche in termini di strutture dati **lineari** e **non lineari**: le strutture dati lineari sono quelle in cui gli elementi sono organizzati in una sequenza lineare e sono accessibili solo attraverso gli estremi della sequenza (come ad es. gli array, le liste e gli stack); le strutture dati non lineari sono quelle in cui gli elementi non sono organizzati in una sequenza lineare e possono essere accessibili attraverso più punti (ad es. gli alberi e i grafi).

Quando si parla di **ordinamento** ci si riferisce al fatto se gli elementi sono disposti in maniera dipendente dal valore delle chiavi: in tal caso la struttura si basa su un ordinamento, altrimenti no. Una lista o un albero di ricerca sono tipicamente strutture ordinate mentre una tabella hash non lo è.

2. Albero

Gli alberi sono una delle strutture dati più importanti e versatili in informatica e sono utilizzati in molti aspetti del calcolo. Ecco alcuni esempi di utilizzo comune degli alberi:

- **Rappresentazione gerarchica di dati**: gli alberi sono spesso utilizzati per rappresentare dati gerarchici, come ad esempio la struttura di una directory di file in un sistema operativo o la struttura di categorie e sottocategorie in un catalogo di prodotti.
- **Alberi di ricerca**: gli alberi di ricerca sono utilizzati per organizzare e cercare rapidamente dati, come ad esempio un indice di un libro o un database di informazioni sulle persone.
- **Alberi decisionali**: gli alberi decisionali sono utilizzati per modellare processi decisionali, ad esempio per prevedere l'esito di una partita di calcio o per determinare se un cliente è idoneo per un prestito.
- **Compressione dei dati**: gli alberi sono utilizzati per la compressione dei dati, come ad esempio l'albero di Huffman, che viene utilizzato per codificare i simboli in modo efficiente.
- **Calcolo matematico**: gli alberi sono utilizzati anche in matematica per rappresentare relazioni tra elementi e per effettuare calcoli, ad esempio gli alberi di espressione sono utilizzati per valutare espressioni aritmetiche.

In termini di **performance**, gli alberi possono essere molto efficienti per le operazioni di inserimento, cancellazione e ricerca, ma possono essere meno efficienti per le operazioni di enumerazione o iterazione rispetto ad altre strutture dati come le liste o le matrici.

Un albero **ben equilibrato** garantisce tempi di esecuzione più rapidi per molte operazioni, mentre un albero scorrettamente equilibrato può causare tempi di esecuzione molto lenti.

Gli alberi possono essere inoltre modificati dinamicamente, il che significa che possono essere modificati durante l'esecuzione del programma, a differenza di strutture dati come le matrici che sono fisse una volta definite.

In sintesi, è una struttura dati molto versatile che consente di rappresentare e gestire i dati in modo efficace, in particolare per le relazioni gerarchiche e i problemi di ricerca e ottimizzazione.

È difficile fare una tassonomia completa degli alberi, infatti esistono diverse sottoclassi e varianti con caratteristiche specifiche; indichiamo di seguito una tassonomia generale al fine di evidenziare le diverse classi di alberi esistenti.

Ecco una possibile tassonomia degli alberi:

- **Alberi binari**: sono alberi in cui ogni nodo ha al massimo due figli.
- **Alberi B**: sono alberi binari che mantengono un equilibrio dinamico della struttura per garantire tempi di esecuzione rapidi per le operazioni di ricerca, inserimento e cancellazione.
- **Alberi AVL**: sono alberi binari bilanciati in cui la differenza di altezza tra i due sottoalberi di un nodo non supera uno.
- **Alberi rosso-nero**: sono alberi binari in cui ogni nodo è colorato di rosso o nero per garantire un equilibrio dinamico della struttura.
- **Alberi T**: sono alberi binari che utilizzano un criterio di equilibrio basato sulla quantità di informazioni memorizzate in ogni nodo.
- **Alberi k-ari**: sono alberi in cui ogni nodo ha un numero qualsiasi di figli compreso tra k.
- **Alberi di Huffman**: sono alberi di codifica usati per la compressione dei dati.
- **Alberi di ricerca**: sono alberi binari in cui i valori di ogni nodo sono maggiori dei valori dei nodi figlio a sinistra e minori dei valori dei nodi figlio a destra.
- **Alberi di navigazione**: sono alberi usati per la navigazione gerarchica di informazioni, come ad esempio i file system.

Da un punto di vista "strutturale", un albero consiste di un insieme di **nodi** e un insieme di **archi** orientati che connettono coppie di nodi, con le seguenti proprietà:

- Un nodo dell'albero è designato come nodo **radice** (**root**)
- Ogni nodo n, a parte la radice, ha esattamente un arco entrante
- Esiste un **cammino** unico dalla radice ad ogni nodo
- Ogni nodo che non presenta archi uscenti è detto **foglia** (**leaf**)
- L'albero è **connesso** (è connesso se per ogni coppia di nodi x, y, esiste un cammino da x ad y)

Un albero è dato da:

- un insieme vuoto, oppure
- un nodo radice e zero o più **sottoalberi**, ognuno dei quali è un albero; la radice è connessa alla radice di ogni sottoalbero con un arco orientato.

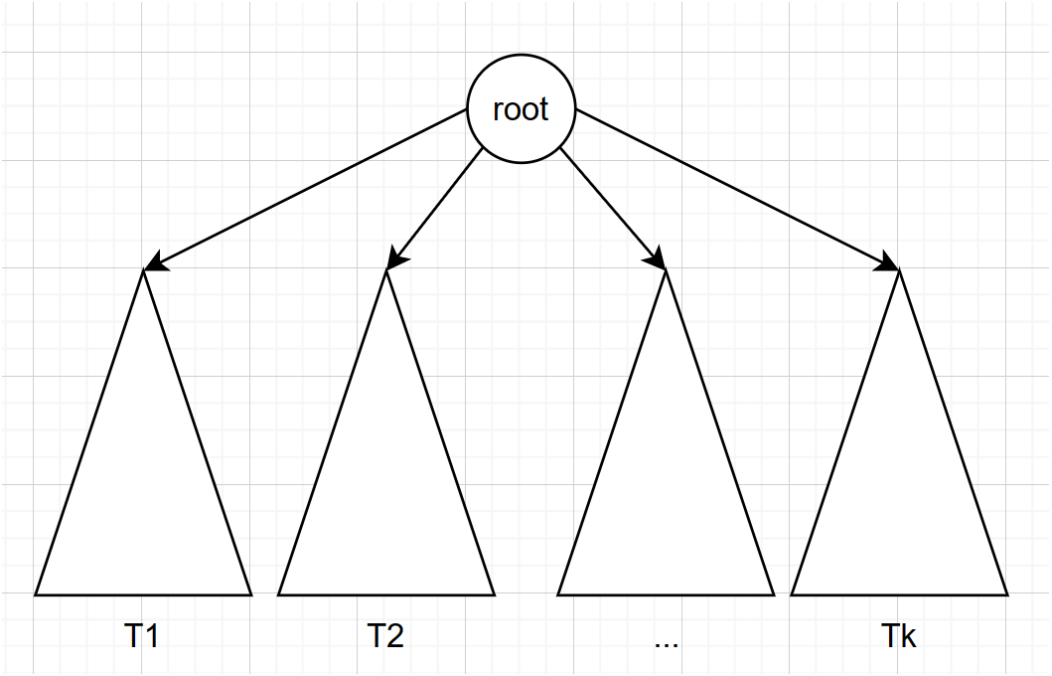

Figura 1: Radice e sottoalberi

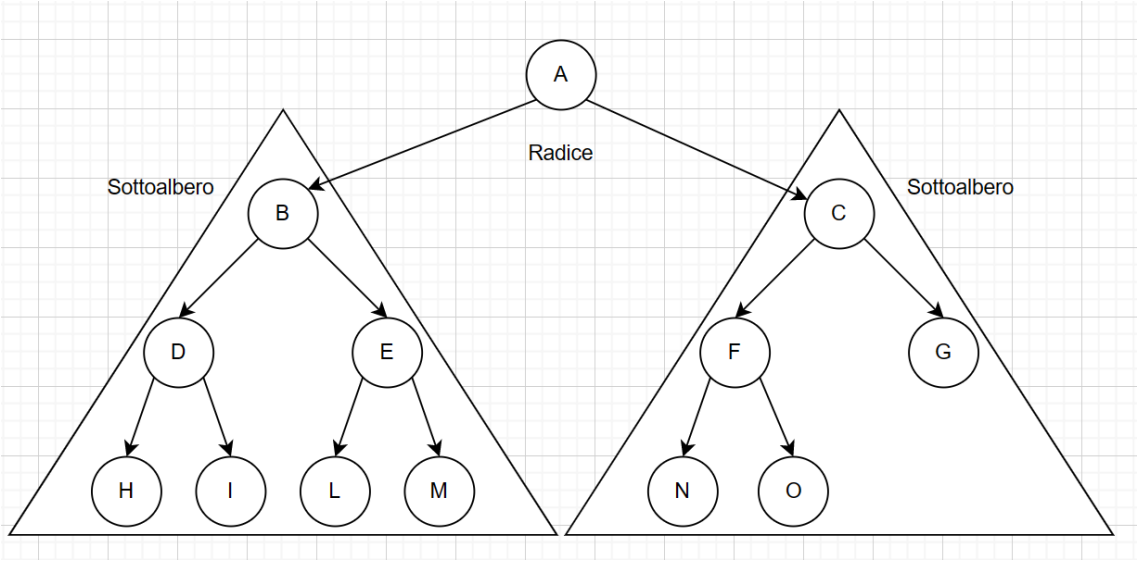

Figura 2: Radice e sottoalberi

- A è la **radice**
- B e C sono le radici dei sottoalberi
- D ed E sono fratelli e sono **figli** di B
- B è il **padre** di D ed E
- I nodi H, I, L, M, N ed O sono le **foglie** dell'albero mentre gli altri nodi sono detti **interni**

Parlando in termini di **teoria dei grafi** si può dire che un grafo (non orientato) senza cicli e connesso è detto **albero**: un albero radicato è una coppia < T, r > dove T è un albero e r è un suo nodo, detto radice.

Adottando una definizione ricorsiva, un albero radicato (non vuoto) è:

- o un singolo nodo
- o una radice connessa a un insieme di alberi

Di seguito riportiamo alcune forme di "albero" ed il loro utilizzo:

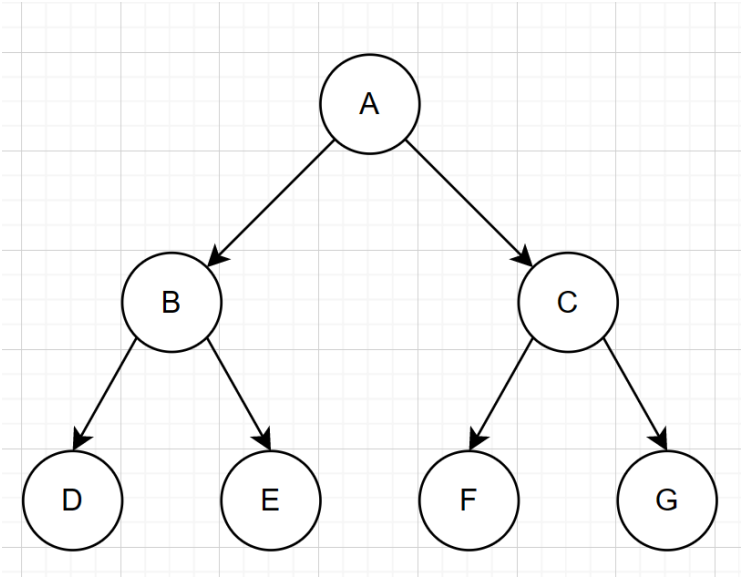

Figura 3: Albero binario

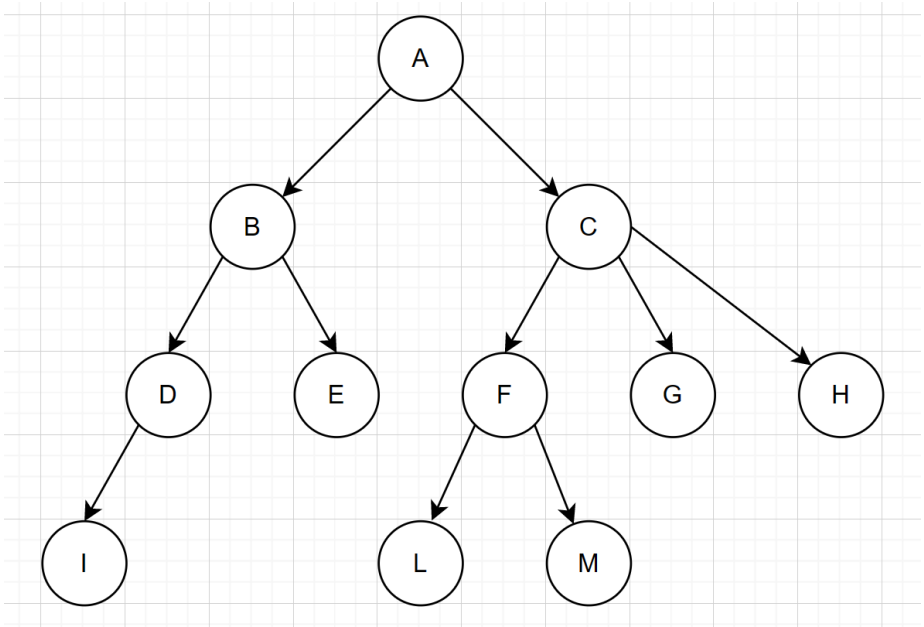

Figura 4: Albero generico

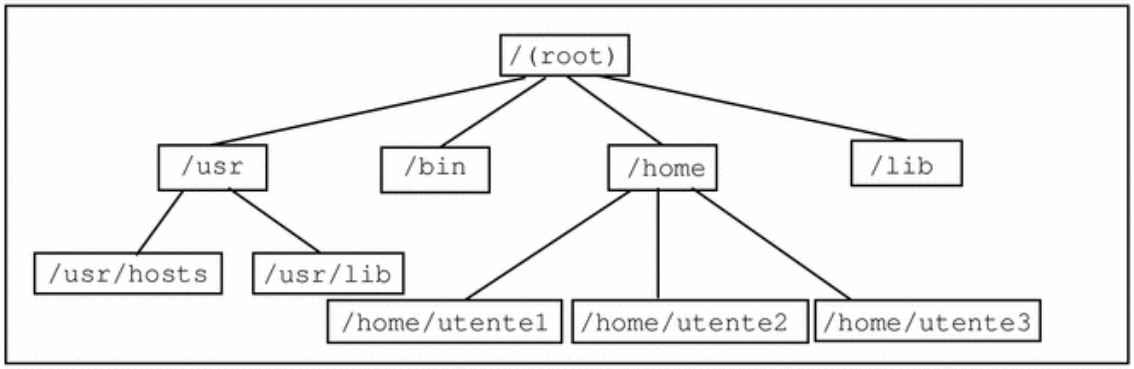

Figura 5: File System

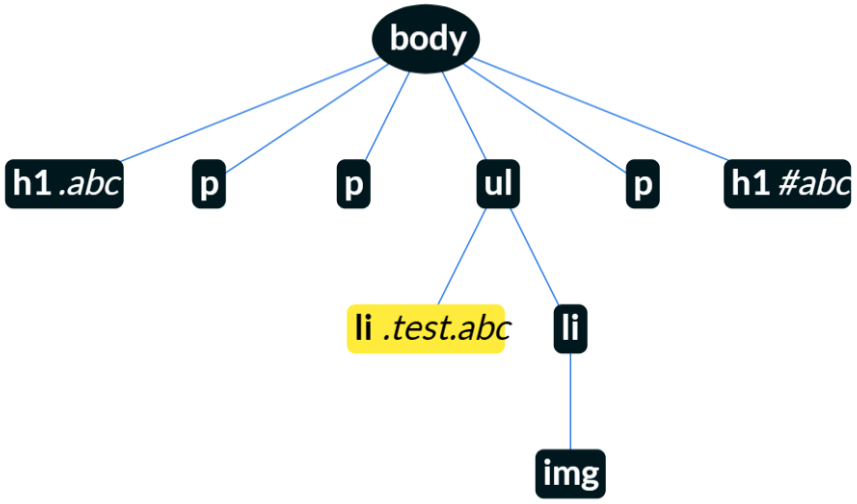

Figura 6: DOM tree

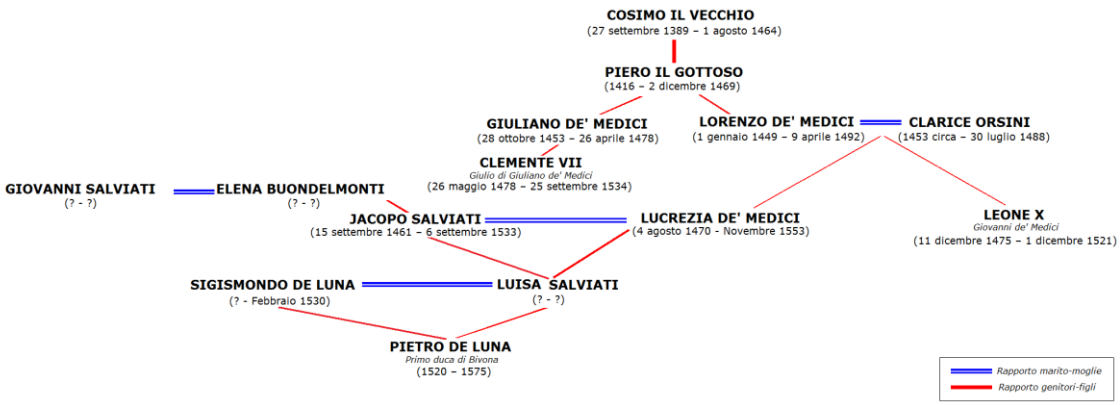

Figura 7: Albero genealogico

3. Proprietà di un albero

Intendiamo per proprietà di un albero le sue caratteristiche che possono essere misurate o quantificate. Riportiamo di seguito alcune caratteristiche di cui si deve tener conto quando si parla di "alberi":

- **Radice**: è il nodo principale che non ha genitori
- **Foglia**: è un nodo che non ha figli.
- **Profondità di un nodo (Depth)**: la lunghezza del cammino semplice dalla radice al nodo, cioè il numero di archi che separano il nodo dalla radice dell'albero
- **Livello (Level)**: il livello di un nodo è il numero di livelli tra la radice dell'albero e il nodo stesso; il livello della radice è zero e il livello di ogni altro nodo è uno in più rispetto alla profondità l'insieme di nodi alla stessa profondità
- **Altezza albero (Height)**: la profondità massima delle sue foglie, cioè il numero massimo di archi tra la radice dell'albero e una delle sue foglie
- **Grado**: è il numero di sotto alberi del nodo, che è uguale al numero di figli del nodo

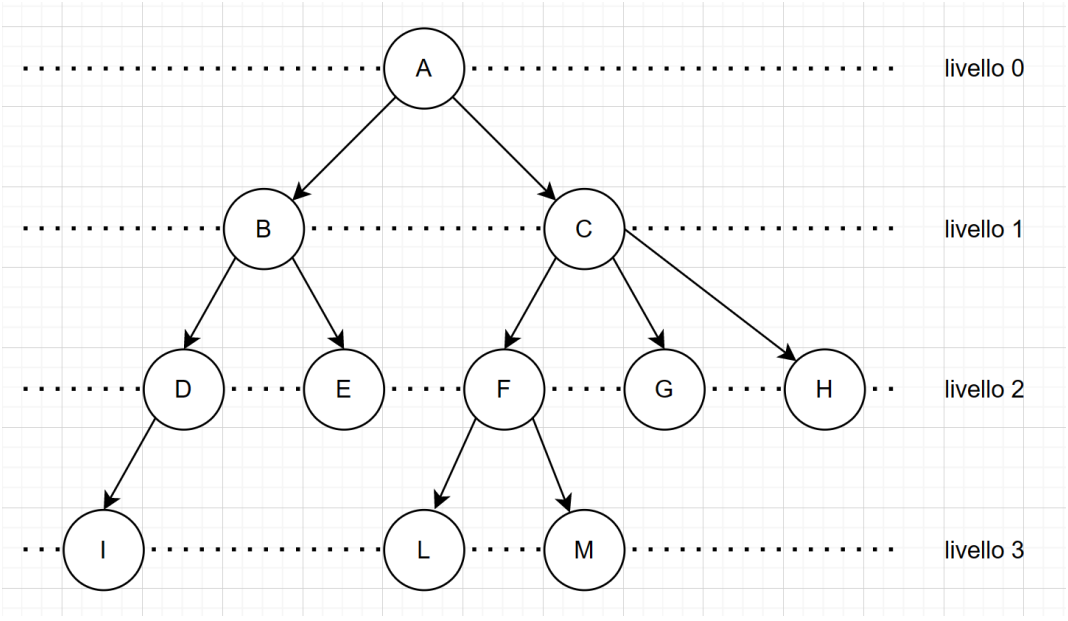

Figura 8: Livelli in un albero

L'albero della fig. 6 ha dunque:

- 3 livelli
  - 2 nodi nel livello 1
  - 5 nodi nel livello 2
  - 3 nodi nel livello 3
- Altezza = 3 (profondità massima dalla radice alle foglie)
- Il grado del nodo B è 2 (2 figli)
- Il grado del nodo C è 3 (3 figli)
- Il grado del nodo D è 1 (1 figlo)
- Il grado del nodo I è 0 (0 figli → è una foglia)

Bibliografia

- Alan Bertossi, Alberto Motresor: Algoritmi e strutture di dati, Città Studi Edizioni, terza edizione
- C. Demetrescu, I. Finocchi, G. F. Italiano: Algoritmi e strutture dati, McGraw-Hill, seconda edizione
- Crescenzi, Gambosi, Grossi: Strutture di Dati e Algoritmi, Pearson/Addison-Wesley
- Sedgewick: Algoritmi in C, Pearson, 2015
- Cormen Leiserson Rivest Stein-Introduzione Agli Algoritmi E Strutture Dati-Prima Edizione