Algoritmi (modulo di laboratorio)

Corso di Laurea in Matematica

Roberto Cordone

DI - Università degli Studi di Milano

Lezioni:	Martedì 8.30 - 10.30 in aula 8	Mercoledì 10.30 - 13.30 in aula 2
	Giovedì 15.30 - 18.30 in aula 2	Venerdì 10.30 - 12.30 in aula 3
Ricevimento:	su appuntamento (Dipartimento di Informatica)	
E-mail:	roberto.cordone@unimi.it	
Pagina web:	http://homes.di.unimi.it/~cordone/courses/2023-algo/2023-algo.html	
Sito Ariel:	https://mgoldwurma.ariel.ctu.unimi.it	

Lezione 12: Alberi binari — Milano, A.A. 2022/23

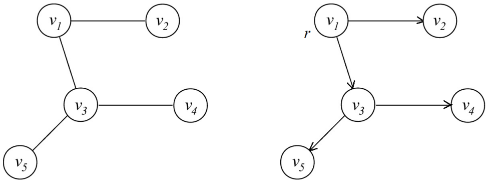
Alberi

Un albero $T = (V, E)$ è un grafo

- connesso: ogni coppia di vertici è legata da un cammino
- aciclico: nessun cammino si richiude su sé stesso

Quindi, ogni coppia di vertici è legata esattamente da un cammino

Albero radicato è un albero con un vertice r marcato come radice
È orientato dalla radice attraverso i nodi interni sino alle foglie



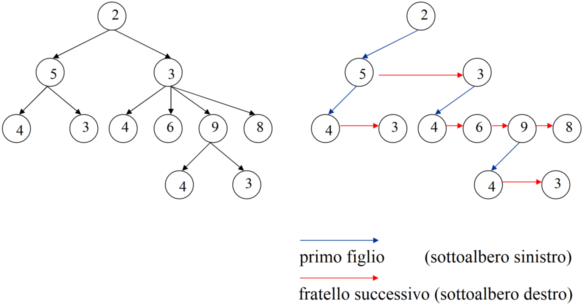
Albero ordinato ha una relazione di ordine totale sui figli di ciascun nodo

Albero binario ha una al massimo due figli per ciascun nodo

Alberi ordinati

Un generico albero ordinato si può rappresentare con un albero binario:

- il sottoalbero sinistro di un nodo nell'albero binario rappresenta il primo figlio nell'albero ordinato
- il sottoalbero destro di un nodo nell'albero binario rappresenta il fratello successivo nell'albero ordinato



Applicazioni tipiche sono gli alberi genealogici e i sistemi di classificazione

Albero binario: struttura dati astratta

Un albero binario T su un insieme U ha una definizione ricorsiva: è

- un insieme vuoto (caso base) oppure
- una terna ordinata (a_r, T_s, T_d) con
  1. $a_r \in U$ (radice)
  2. T_s albero binario su U (sottoalbero sinistro)
  3. T_d albero binario su U (sottoalbero destro)

La definizione non è tautologica perché il caso base arresta la ricorsione, ma il numero dei nodi non è soggetto ad alcun limite

Le operazioni di proiezione e sostituzione usano un insieme di posizioni P per accedere ai nodi dell'albero

- dato un albero T, solo la posizione della radice r è nota
- dato un albero T e una posizione p si può ricavare direttamente solo
  - l'informazione associata al nodo in posizione p
  - le posizioni delle radici dei suoi due sottoalberi
  - (eventualmente) la posizione del nodo padre

La situazione è del tutto analoga a quella delle liste con la stessa necessità di definire posizioni fittizie (*vedi Lezione* 7)

Alberi binari: operazioni

Sia $\mathcal{T}$ l'insieme di tutti i possibili alberi binari su U

Gli alberi binari ammettono tipicamente le seguenti operazioni

- proiezione: dato un albero e una posizione, fornisce il nodo corrispondente

$$\text{leggenodo} : \mathcal{T} \times P \rightarrow U$$

- sostituzione: dato un albero, una posizione e un nodo inserisce il nodo nell'albero sostituendo quello puntato dalla posizione

$$\text{scrivenodo} : \mathcal{T} \times P \times U \rightarrow \mathcal{T}$$

- verifica di vuotezza: dato un albero, indica se è vuoto

$$\text{alberovuoto} : \mathcal{T} \rightarrow \mathbb{B} \qquad (\text{ovvero } \{0, 1\})$$

- accesso alla radice: dato un albero, fornisce la posizione della radice

$$\text{radice} : \mathcal{T} \rightarrow P$$

Se l'albero è vuoto, restituisce la posizione fittizia $\perp$

Alberi binari: operazioni

Sia $\mathcal{T}$ l'insieme di tutti i possibili alberi binari su U

Gli alberi binari ammettono tipicamente le seguenti operazioni

- figlio sinistro: dato un albero e una posizione, fornisce la posizione della radice del figlio sinistro del nodo nella posizione data

$$\text{figliosinistro} : \mathcal{T} \times P \rightarrow P$$

Se non esiste un sottoalbero sinistro, restituisce $\perp$

- figlio destro: dato un albero e una posizione, fornisce la posizione della radice del figlio destro del nodo nella posizione data

$$\text{figliodestro} : \mathcal{T} \times P \rightarrow P$$

Se non esiste un sottoalbero destro, restituisce $\perp$

- padre: dato un albero e una posizione, fornisce la posizione del nodo padre

$$\text{padre} : \mathcal{T} \times P \rightarrow P$$

Per il nodo radice, che non ha padre, restituisce $\perp$

Alberi binari: operazioni

Sia $\mathcal{T}$ l'insieme di tutti i possibili alberi binari su U

Inserimento e cancellazione di elementi per un albero binario differiscono dalle analoghe operazioni per le liste:

- costruzione: dato un nodo e due alberi binari, restituisce un albero che ammette il nodo come radice, il primo albero come figlio sinistro e il secondo come figlio destro

$$\text{costruiscealbero} : U \times \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$$

- cancsottoalbero: dato un albero e una posizione, cancella dall'albero il sottoalbero che ha come radice il nodo nella posizione data

$$\text{cancsottoalbero} : \mathcal{T} \times P \rightarrow \mathcal{T}$$

Questa differenza è dovuta alla struttura gerarchica!

Alberi binari: operazioni

In matematica basta definire un oggetto per crearlo

Nelle implementazioni concrete, questo in genere non vale
Quindi è opportuno definire

- creazione: crea un albero binario vuoto

$$\text{creaalbero} : () \to \mathcal{T}$$

- distruzione: distrugge un albero

$$\text{distruggealbero} : \mathcal{T} \to ()$$

Alberi binari: implementazione con puntatori

L'idea base è di rappresentare le posizioni con indirizzi di memoria

- l'albero corrisponde allora alla posizione della radice
- ogni elemento dell'albero corrisponde a una struttura con
  - il dato $a \in U$
  - la posizione della radice del sottoalbero sinistro ($\perp$ se non esiste)
  - la posizione della radice del sottoalbero destro ($\perp$ se non esiste)
  - la posizione del nodo padre ($\perp$ se non esiste)

```
#define EMPTY_TREE NULL                    (albero vuoto)
#define NO_NODE NULL                       (posizione esterna all'albero)

typedef nodo *alberobinario;               (l'albero è l'indirizzo della radice)
typedef nodo *posizione;                   (la posizione del nodo è il suo indirizzo)

typedef struct _nodo nodo;
struct _nodo {
   U a;                                    (U è il tipo del nodo generico)
   posizione Ts;                           (posizione della radice del sottoalbero sinistro)
   posizione Td;                           (posizione della radice del sottoalbero destro)
   posizione padre;                        (posizione del nodo padre)
};
```

Alberi binari: implementazione con puntatori