Algoritmi e Strutture di Dati

Capitolo 3 - Tipi di dato e strutture di dati

Introduzione

- **Dato**
  - **In un linguaggio di programmazione, un dato è un valore che una variabile può assumere**
- **Tipo di dato astratto**
  - **Un modello matematico, dato da una collezione di valori e un insieme di operazioni ammesse su questi valori**
- **Tipi di dato primitivi**
  - **Forniti direttamente dal linguaggio**
  - **Esempi: int (+,-,*,/, %), boolean (!, &&, ||)**

Tipi di dati

- "Specifica" e "implementazione" di un tipo di dato astratto
  - *Specifica*:
    "manuale d'uso", nasconde i dettagli implementativi all'utilizzatore
  - Implementazione:
    realizzazione vera e propria
- Esempi
  - Numeri reali vs IEEE754
  - Pile vs Pile basate su vettori, Pile basate su puntatori
  - `java.util.Map` vs `java.util.TreeMap`

Strutture dati

- I dati sono spesso riuniti in insiemi detti *strutture di dati*
  - sono particolari tipi di dato, caratterizzati più dall'*organizzazione dei dati* più che dal tipo dei dati stessi
  - il tipo dei dati contenuti può essere addirittura parametrico
- Una struttura di dati è composta quindi da:
  - un modo sistematico di organizzare i dati
  - un insieme di operatori che permettono di manipolare la struttura
- Alcune tipologie di strutture di dati:
  - *lineari* / *non lineari* (presenza di una sequenza)
  - *statiche* / *dinamiche* (variazione di dimensione, contenuto)
  - *omogenee* / *disomogenee* (dati contenuti)

Strutture di dati

Tipo	Java	C++	Python
Sequenze	List, Queue, Deque LinkedList, ArrayList, Stack, ArrayDeque	list, forward_list vector stack queue, deque	list tuple
Insiemi	Set TreeSet, HashSet, LinkedHashSet	set unordered_set	set, frozenset
Dizionari	Map HashTree, HashMap, LinkedHashMap	map unordered_map	dict
Alberi	-	-	-
Grafi	-	-	-

Sequenza

- **Struttura di dati**
  - **Dinamica e lineare**
  - **Contenente elementi generici (Item), potenzialmente anche duplicati**
  - **Ordine all'interno della sequenza è importante**
- **Interfaccia**
  - **E' possibile aggiungere / togliere elementi, specificando la posizione**
    - $s = s_1, s_2, \ldots, s_n$
    - **L'elemento s_i è in posizione pos_i**
    - **Esistono le posizioni (fittizie) pos_0, pos_{n+1}**
  - **E' possibile accedere *direttamente* ad alcuni elementi (testa / coda)**
  - **E' possibile accedere *sequenzialmente* a tutti gli altri elementi**

Sequenza

SEQUENCE

% Restituisce **true** se la sequenza è vuota
boolean empty()

% Restituisce **true** se p è uguale a pos_0 oppure a pos_{n+1}
boolean finished(POS p)

% Restituisce la posizione del primo elemento
POS head()

% Restituisce la posizione dell'ultimo elemento
POS tail()

% Restituisce la posizione dell'elemento che segue p
POS next(POS p)

% Restituisce la posizione dell'elemento che precede p
POS prev(POS p)

per scorrere la lista

Sequenza

% Inserisce l'elemento v di tipo ITEM nella posizione p.
% Ritorna la nuova posizione, che diviene il predecessore di p
POS insert(POS p, ITEM v)

% Rimuove l'elemento contenuto nella posizione p.
% Ritorna il successore di p, che diviene successore del predecessore di p
POS remove(POS p)

% Legge l'elemento di tipo ITEM contenuto nella posizione p
ITEM read(POS p)

% Scrive l'elemento v di tipo ITEM nella posizione p
write(POS p, ITEM v)

Insiemi (Set)

- Struttura dati "generale": *insieme dinamico*
  - Può crescere, contrarsi, cambiare contenuto
  - Operazioni base: inserimento, cancellazione, verifica contenimento
  - Il tipo di insieme (= struttura) dipende dalle operazioni
- Elementi
  - Elemento: oggetto "puntato" da un riferimento/puntatore
  - Composto da:
    - campo chiave di identificazione
    - dati satellite
    - campi che fanno riferimento ad altri elementi dell'insieme

Insiemi dinamici

SET

% Restituisce la cardinalità dell'insieme
integer size()

% Restituisce **true** se x è contenuto nell'insieme
boolean contains(ITEM x)

% Inserisce x nell'insieme, se non già presente
insert(ITEM x)

% Rimuove x dall'insieme, se presente
remove(ITEM x)

% Restituisce un nuovo insieme che è l'unione di A e B
SET union(SET A, SET B)

% Restituisce un nuovo insieme che è l'intersezione di A e B
SET intersection(SET A, SET B)

% Restituisce un nuovo insieme che è la differenza di A e B
SET difference(SET A, SET B)

Dizionari (Dictionary o Map)

- Il dizionario rappresenta il concetto matematico di relazione univoca
  - Relazione $R : D \rightarrow C$
  - Insieme D è il dominio (elementi detti chiavi)
  - Insieme C è il codominio (elementi detti valori)
  - Associazione chiave-valore
- Operazioni ammesse:
  - ottenere il valore associato ad una particolare chiave (se presente), o nil
  - inserire una nuova associazione chiave- valore, cancellando eventuali associazioni precedenti;
  - rimuovere un'associazione chiave-valore esistente

Dizionari

DICTIONARY

% Restituisce il valore associato alla chiave k se presente, **nil** altrimenti
ITEM lookup(ITEM k)

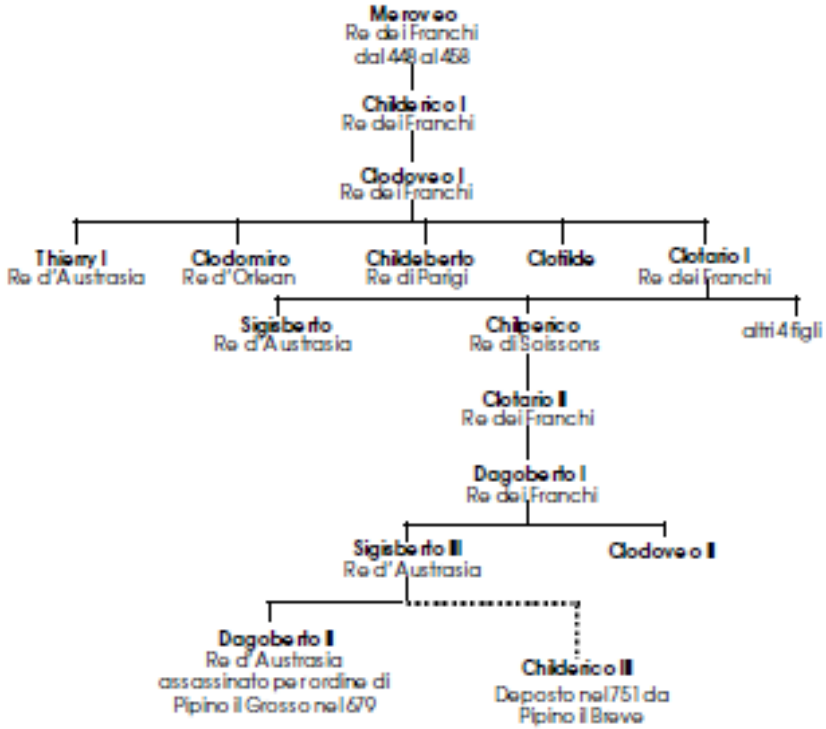
% Associa il valore v alla chiave k
insert(ITEM k, ITEM v)

% Rimuove l'associazione della chiave k
remove(ITEM k)

Alberi e grafi

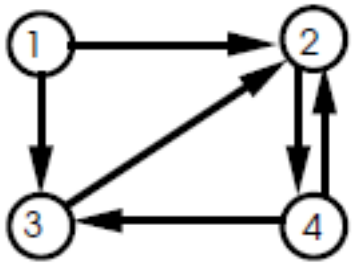
✦Un albero ordinato

- ✦è formato da un insieme finito di elementi detti nodi
- ✦un nodo particolare è designato come radice
- ✦i rimanenti nodi, se esistono, sono partizionati in insiemi ordinati e disgiunti, anch'essi alberi ordinati



✦Grafi

- ✦Insiemi di nodi e archi che connettono i nodi



✦Operazioni

- ✦Visita: ispezione completa di tutti i nodi di un albero o di un grafo
- ✦Specifica completa più avanti

Discussione

- Le specifiche viste finora possono essere arricchite:
  - Operatori `min()` e `max()` nel tipo di dato Set, se esiste ordinamento totale
  - Concetti di insieme e dizionario sono collegati
    - Insieme delle chiavi / insieme dei valori
    - Scorrere tutte le chiavi

Realizzazione

- Alcune realizzazioni sono “naturali”
  - Sequenza ↔ lista
  - Albero astratto ↔ albero basato su puntatori
- Esistono tuttavia realizzazioni alternative
  - Insieme come vettore booleano
  - Albero come vettore dei padri
- La scelta della struttura di dati ha riflessi sull’efficienza e sulle operazioni ammesse
  - Dizionario come hash table: lookup in tempo *O(1)*, ma niente ordinamento
  - Dizionario come albero: lookup in tempo *O(log n)*, con ordinamento