Alberi

Alberi: definizioni

Alberi Binari

Esercizi su alberi binari: metodi ricorsivi

Struttura dati per alberi generici

Alberi

- Gli alberi sono strutture dati naturalmente **ricorsive**
- Un albero è un particolare **grafo diretto** in cui un **nodo** particolare è detto **radice**
- Ogni altro nodo dell'albero, tranne la radice, ha **un solo nodo padre**; un arco parte dal padre verso il figlio, che è radice di un **sottoalbero**
- Ogni nodo di un albero può avere zero o più nodi figli, quindi zero o più **sottoalberi**

Alberi

- I figli di un nodo sono una **lista ordinata**
- Un nodo che non ha figli è detto **foglia**
- Un nodo che ha almeno un figlio è detto **nodo interno**
- Dalle definizioni sopra discende che in un albero **non ci sono cicli**
- Un **cammino** è una sequenza di nodi e archi alternati che parte dalla radice e finisce in una foglia

Alberi

- L'albero **più piccolo** è formato da un solo nodo che è sia radice che foglia
- Esiste anche il concetto di **albero vuoto**, che non ha né nodi né archi
- Un insieme di alberi disgiunti si chiama **foresta**
- I nodi di un albero possono contenere una qualsiasi informazione, ad esempio un numero intero `int` o `Integer`, una `String`, un `Object` in generale

Alberi Binari

- Il tipo di alberi più semplice è quello degli alberi binari
- In un albero binario un nodo può avere zero, uno o due figli
- Nel caso di due figli, essi si chiamano sottoalbero **sinistro** e sottoalbero **destro**
- Per gli alberi binari definiamo una classe **`BinTree<E>`** dove **`E`** è una generica classe che rappresenta l'informazione associata ai nodi

Alberi Binari

- Si veda il codice allegato per la definizione della classe
- Si noti che le variabili istanza **`left`** e **`right`** sono di tipo **`BinTree<E>`** cioè la classe stessa è ricorsiva
- I metodi per le visite sono tutti ricorsivi
- In generale, per risolvere un problema su un albero binario si ragiona ricorsivamente (per induzione) sui sottoalberi

Alberi Binari

- Si consideri ad esempio il problema di calcolare l'altezza di un albero binario
- L'altezza di un albero è la massima lunghezza di un cammino nell'albero
- Il cammino vuoto che comincia nella radice che è anche foglia è lungo zero
- Il problema si risolve in maniera semplice ragionando ricorsivamente
- Si consideri il metodo ricorsivo **`getHeight()`**

Alberi Binari

- Basta considerare il caso base, cioè quando l'albero è il più piccolo possibile
- In questo caso l'altezza è 0
- Nel caso in cui ci sia un solo sottoalbero, basta calcolare ricorsivamente l'altezza del sottoalbero e aggiungere 1
- Nel caso in cui ci siano entrambi i sottoalberi, basta calcolare ricorsivamente l'altezza dei sottoalberi, prendere il massimo e aggiungere 1

Alberi Binari

- Per nostra comodità definiamo una classe

```
public class BinTreeMeasurable<E extends IntMeasurable>
```

- La classe è equivalente a **`BinTree<E>`**, ma:
- Si richiede che la classe **`E`** implementi l'**`interface IntMeasurable`**
- In questo modo è possibile ottenere un numero intero associato ad ogni nodo
- È sufficiente chiamale il metodo **`getMeasure`**() sull'elemento di tipo **`E`** associato al nodo

Alberi Binari

- Si consideri ad esempio il metodo **`getSumOfInternalNodes()`**
- Deve calcolare la somma degli interi associati ai nodi interni di un albero binario
- Nel caso dell'albero più piccolo basta chiamare il metodo **`getMeasure()`** sull'oggetto associato alla radice/foglia
- Nel gli altri casi si calcola la somma sui sottoalberi e si somma all'intero associato alla radice

Alberi Generici

- Nel caso si voglia rappresentare un albero generico basta prevedere che ogni nodo abbia una lista di figli
- Abbiamo definito la classe **`Tree<E>`** che al suo interno contiene una variabile di tipo **`List<Tree<E>>`**
- In questo modo tutti i figli di un nodo sono ordinati (da sinistra verso destra) e sono tutti sottoalberi
- In caso di radice/foglia la lista sarà vuota

Esercitazione

- Si veda il codice allegato per tutti gli esercizi fatti in classe o proposti per casa