Laboratorio I
a.a. 2024/2025

Corso B

Esercitazione su TypeScript

Calendario

- **Oggi:** esercitazione guidata
- **Giovedì 27 marzo:** simulazione prova in itinere
- **Martedì 1 aprile:** non c’è lezione
- **Giovedì 3 aprile:** simulazione prova in itinere
- **Martedì 8 aprile:** prova in itinere

Esercizio 1: Crea sconti

Scrivere in TypeScript una funzione chiamata **`createDiscount`** che accetti un parametro opzionale **`discount`**, con valore predefinito del 10%. La funzione deve restituire un'altra funzione che, dato un prodotto **`p`** di tipo **`Product`** (con proprietà **`name`**, **`price`** e **`category`**) applica lo sconto specificato al prezzo di **`p`**.

Esercizio 2: Ordina prezzi

Scrivere in TypeScript una funzione generica chiamata **`sortByPrice`** che accetti i seguenti parametri:

- un array di oggetti, ognuno dei quali possiede una proprietà numerica **`price`**;
- un secondo parametro di tipo enumerazione, che indica l'ordine di ordinamento: crescente o decrescente.

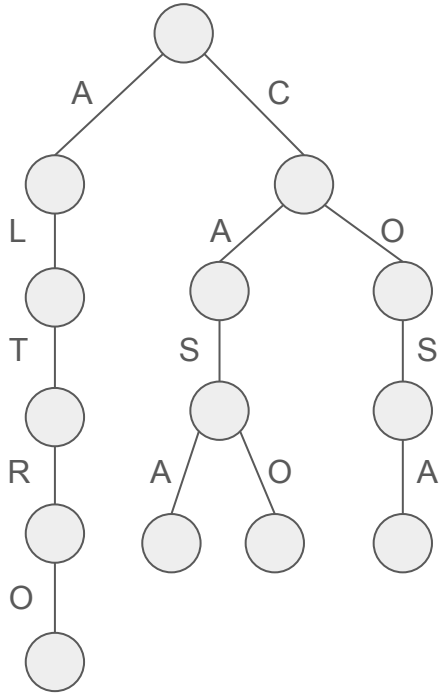
La funzione deve ordinare l'array in base al valore della proprietà **`price`** in base al parametro di ordinamento specificato.

Esercizio 3: Trie

Scrivere in TypeScript una classe **`Trie`** che rappresenti l'omonima struttura dati e che supporti le seguenti operazioni:

- **`insert(key)`**: inserisce la stringa **`key`** nel trie.
- **`lookup(key)`**: restituisce true se e solo se la stringa **`key`** è presente nel trie.
- **`size`**: proprietà di sola lettura che indica il numero di chiavi attualmente memorizzate nel trie.
- **`prefixSearch(prefix)`**: restituisce un generatore che produce tutte le stringhe memorizzate nel trie che iniziano con il prefisso **`prefix`**.

Trie per le stringhe: {altro, casa, caso, cosa}

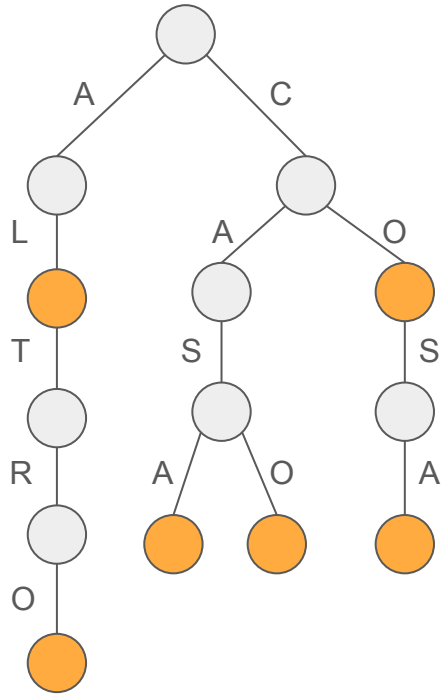


Esercizio 3: Trie

Scrivere in TypeScript una classe **`Trie`** che rappresenti l'omonima struttura dati e che supporti le seguenti operazioni:

- **`insert(key)`**: inserisce la stringa **`key`** nel trie.
- **`lookup(key)`**: restituisce true se e solo se la stringa **`key`** è presente nel trie.
- **`size`**: proprietà di sola lettura che indica il numero di chiavi attualmente memorizzate nel trie.
- **`prefixSearch(prefix)`**: restituisce un generatore che produce tutte le stringhe memorizzate nel trie che iniziano con il prefisso **`prefix`**.

Trie per le stringhe: {altro, casa, caso, cosa, al, co}



Alcune stringhe possono terminare in un nodo interno (come le stringhe al e co nell'esempio). Per distinguere nodi che rappresentano stringhe valide nell'insieme, ogni nodo contiene un flag

Esercizio 4: Trie con valori

Modificare la classe **`Trie`** per trasformarla in una struttura dati che funzioni come mappa chiave-valore, dove le chiavi sono stringhe e i valori sono di tipo generico **`T`**. In particolare, aggiornare i seguenti metodi:

- **`insert(key, value)`**: inserisce l'associazione **`key`** → **`value`** nel trie, eventualmente aggiornando il valore esistente per **`key`**.
- **`lookup(key)`**: restituisce il valore associato alla stringa **`key`**, oppure **`undefined`** se la stringa non è presente.
- **`prefixSearch(prefix)`**: restituisce un generatore che produce tutte le coppie **`[key,value]`** memorizzate nel trie, dove **`key`** ha come prefisso la stringa **`prefix`**.

Esercizio 5: Reduce su valori del trie

Estendere la classe **`Trie<T>`** aggiungendo un metodo **`reduce(f, initialValue)`** che restituisce il risultato dell'applicazione della funzione di riduzione **`f(accumulator, currentValue)`** a ciascun valore di tipo **`T`** memorizzato nel trie, usando **`initialValue`** come valore iniziale dell'accumulatore.

Q & A