Duomenų struktūros

SET DUOMENŲ TIPAS

Praėjusios paskaitos kartojimas

Masyvai ir sąrašai (angl. lists) prisimena ne tik elementų reikšmes, tačiau ir elementų įdėjimo tvarką, Set duomenų tipas įsimena tik įdėjimo tvarką. Kuriais atvejais geriau naudoti Set duomenų tipą nei masyvus ar sąrašus?

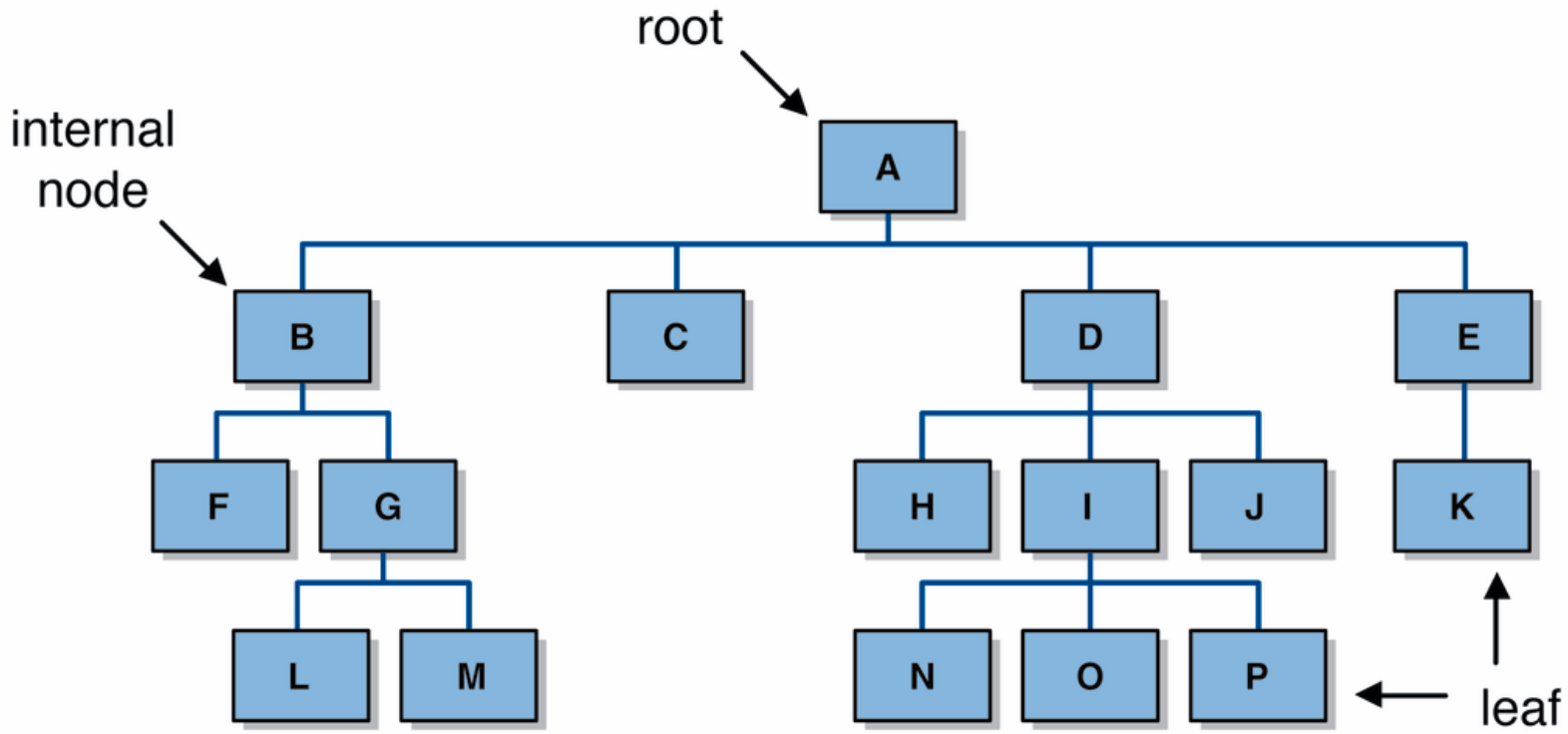
Kuo skiriasi sąrašo iteratorius nuo set iteratoriaus? Kodėl jie skiriasi?

Praeitos paskaitos kartojimas

1. Ar sąrašai (pvz. LinkedList) užimtų daugiau vietos saugant tą pačią informaciją nei masyvai? Kodėl?

2. Kam mums reikalingas iteratorius?

Medžio terminologija



Medžio terminologija

A leaf has no children.

An internal node has at least one child.

Siblings have the same parent.

grandparent, grandchild, ancestor, descendant

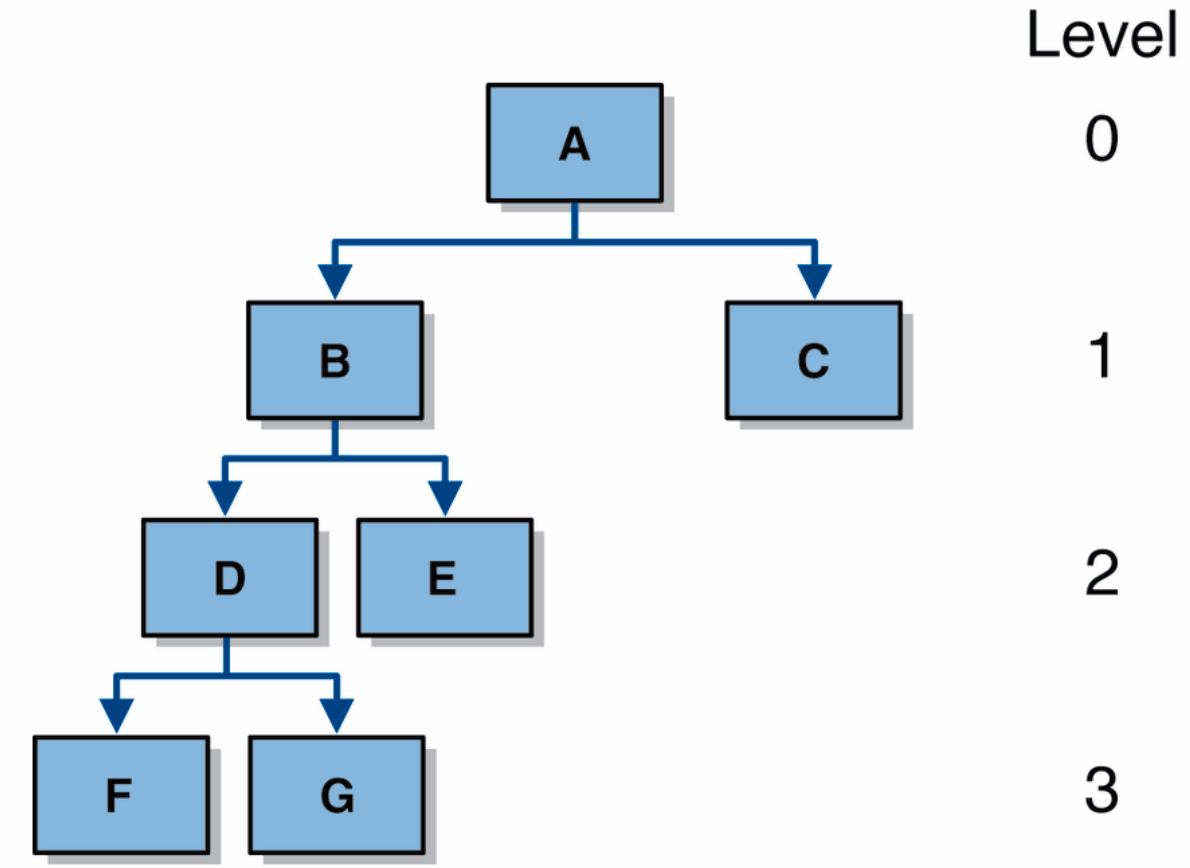
A path is a sequence of nodes n1, n2, … , nk such that ni is the parent of ni+1 for 1 ? i < k.

The length of a path is the number of edges in the path.

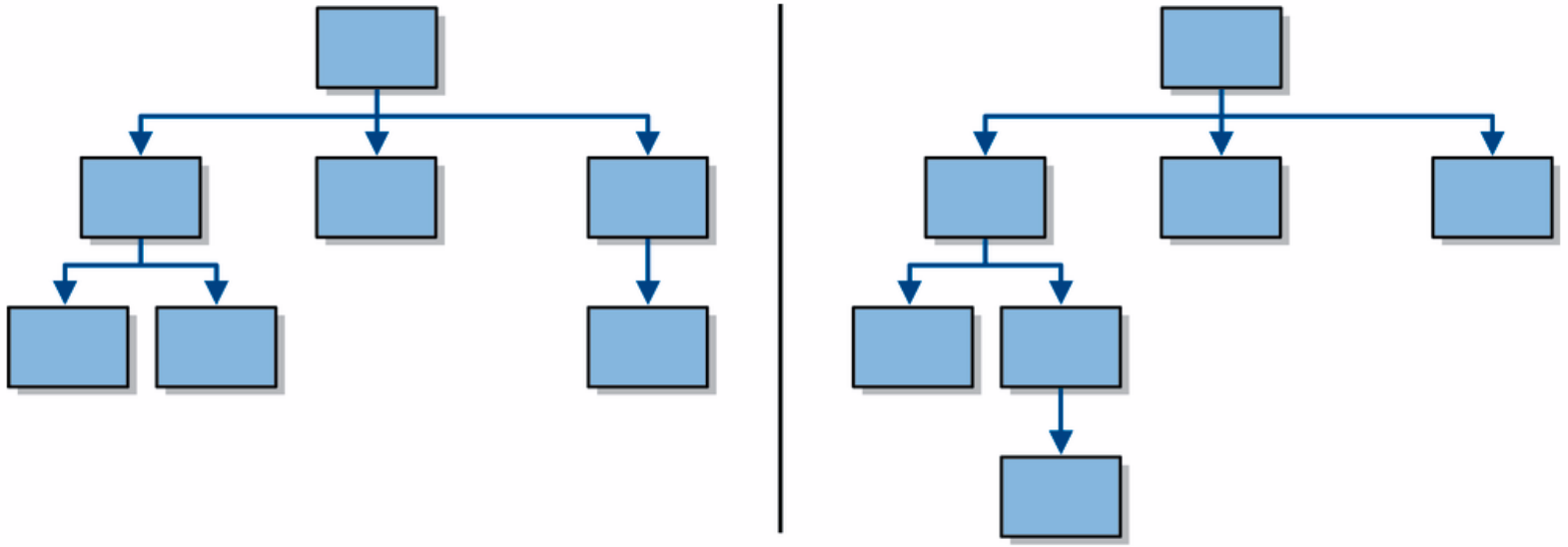
The depth of a node is the length of the path from the root to the node. (Starts at zero!)

The height of a tree: length of the longest path from root to a leaf.

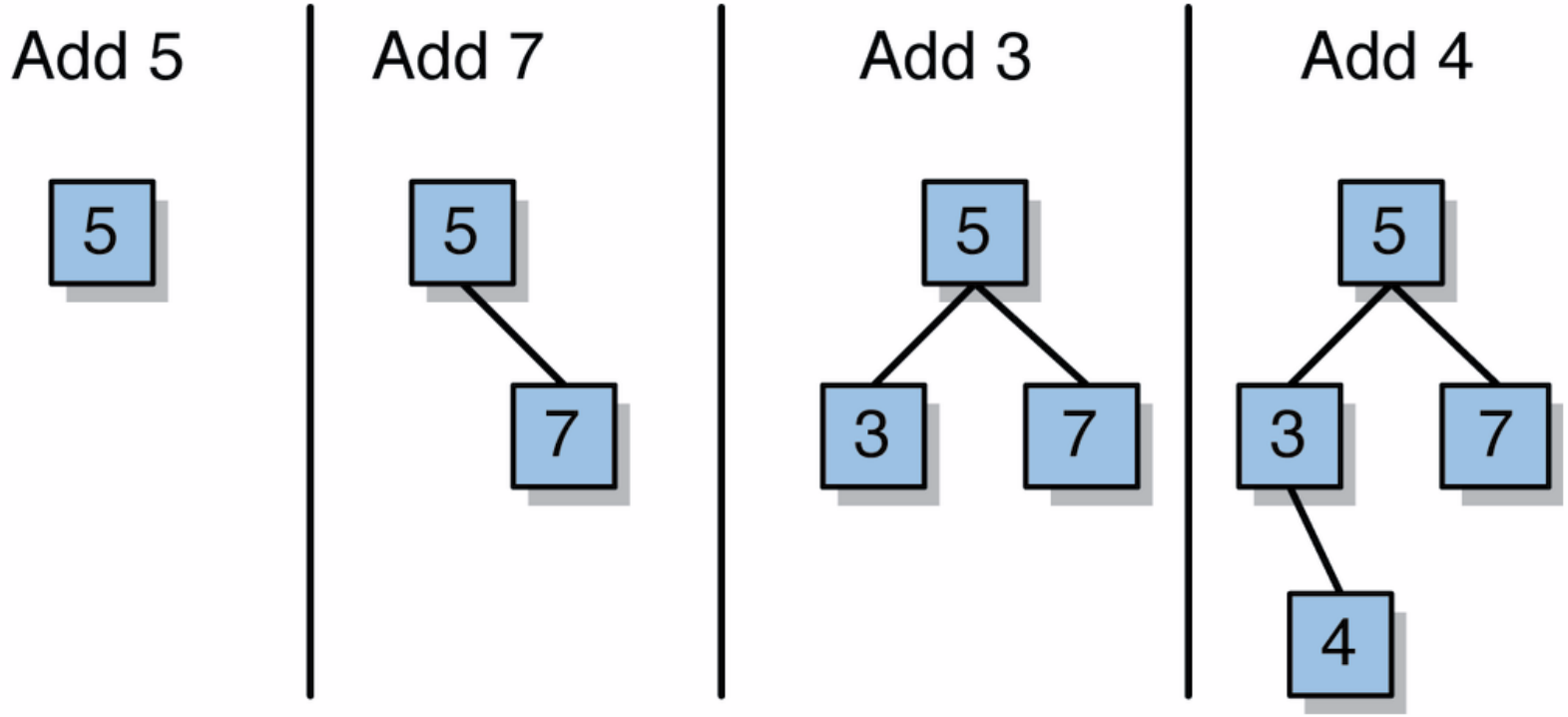
Path length and level



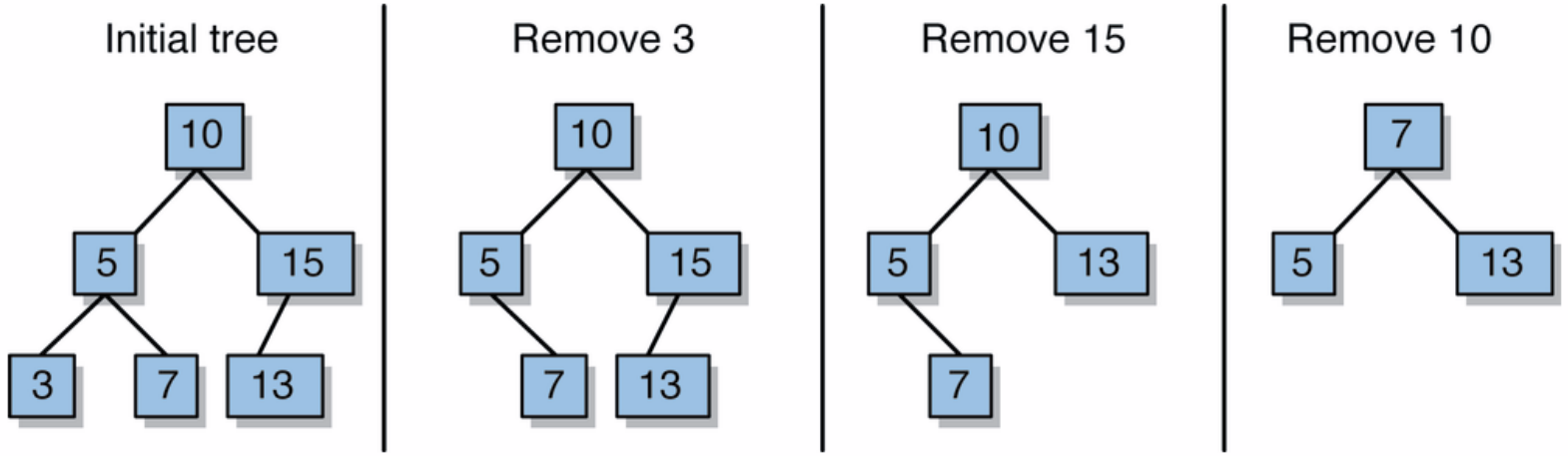
Balanced and unbalanced trees



Adding elements to a binary search tree



Removing elements from a binary search tree



BST remove

Removing an item disrupts the tree structure.

Basic idea:

find the node that is to be removed.

Then “fix” the tree so that it is still a binary search tree.

Remove node and replace it with its successor or predecessor (whichever more convenient)

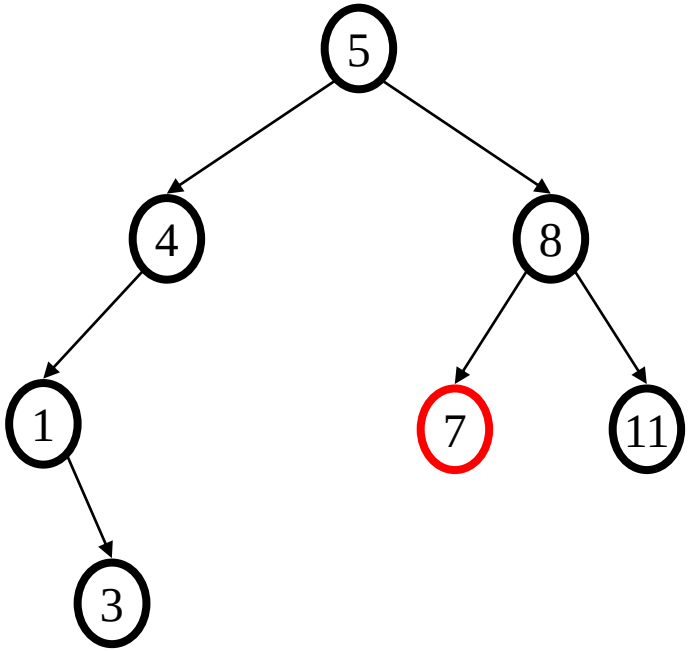
Three cases:

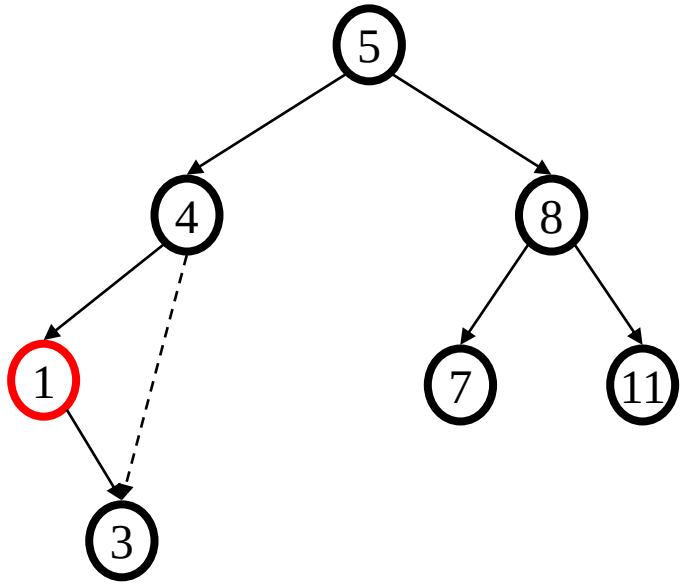
node has no children

node has one child

node has two children

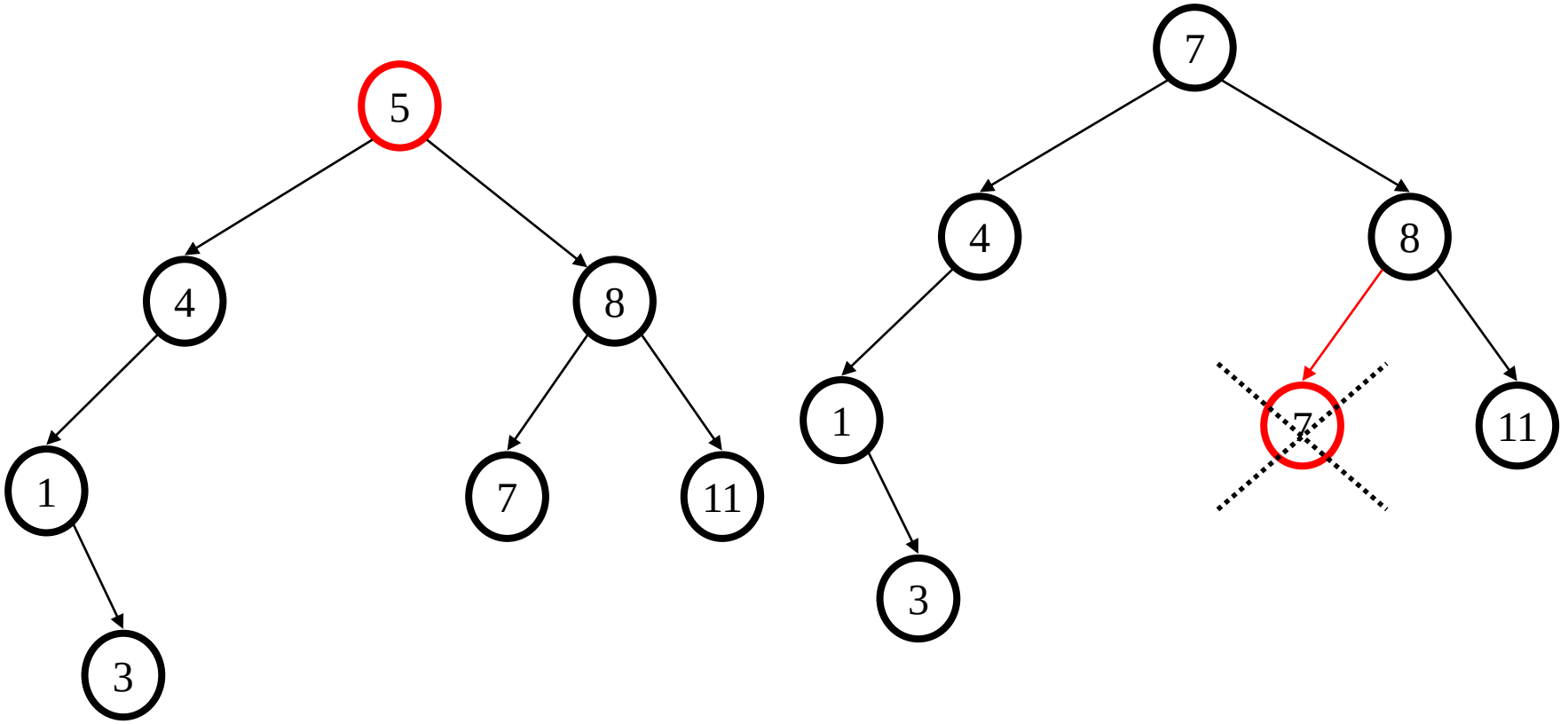
No children, one child





Two children

Replace the node with its successor. Then remove the successor from the tree.



TreeSet

In a TreeSet

- Elements are kept in sorted order
- Elements are stored in nodes.

The nodes are arranged in a tree shape, not in a linear sequence

You can form tree sets for any class that implements the Comparable interface:

Example: String or Integer.

Įrašų talpinimas ir šalinimas

Norėdami patalpinti ar pašalinti įrašus, galime naudoti add ir remove metodus:

```
names.add("Romeo");
names.remove("Juliet");
```

Sąrašai nesaugo dviejų vienodų reikšmių

Dviejų vienodų reikšmių talpinimas yra ignoruojamas

- Sets don't have duplicates.
- Adding a duplicate is ignored.
- Attempting to remove an element that isn't in the set is ignored.

Duomenų paieška sąraše

The contains method tests whether an element is contained in the set:

if (names.contains("Juliet")) . . .

The contains method uses the equals method of the element type

Set iteratorius ir jo naudojimas

To process all elements in the set, get an iterator.

A set iterator visits the elements in the order in which the set implementation keeps them.

```
Iterator<String> iter = names.iterator();    while (iter.hasNext())
{
String name = iter.next();
}
```

- You cannot add an element to a set at an iterator position - A set is unordered.
- You can remove an element at an iterator position. The iterator interface as no previous method.

Pagrindiniai metodai

Table 4 Working with Sets	
`Set<String> names;`	Use the interface type for variable declarations.
`names = new HashSet<>();`	Use a `TreeSet` if you need to visit the elements in sorted order.
`names.add("Romeo");`	Now `names.size()` is 1.
`names.add("Fred");`	Now `names.size()` is 2.
`names.add("Romeo");`	`names.size()` is still 2. You can't add duplicates.
`if (names.contains("Fred"))`	The `contains` method checks whether a value is contained in the set. In this case, the method returns `true`.
`System.out.println(names);`	Prints the set in the format `[Fred, Romeo]`. The elements need not be shown in the order in which they were inserted.

`for (String name : names)`
`{`
`. . .`
`}`
	Use this loop to visit all elements of a set.

`names.remove("Romeo");`	Now `names.size()` is 1.
`names.remove("Juliet");`	It is not an error to remove an element that is not present. The method call has no effect.

Set klasės pasirinkimas

Use a TreeSet if you want to visit the set's elements in sorted order.

Otherwise choose a HashSet.

It is a bit more efficient — if the hash function is well chosen.

Jei talpinami objektai į TreeSet, tuomet turime būti relizavę interfeisą Comparable, jei naudojame HashSet, tuomet objektai turi turėti realizuotą Hash funkciją - hashCode();

Set klasės pasirinkimas

Klasė HashSet ir TreeSet turi tuos pačius metodus ir tą patį interfeisą, todėl galime rašyti:

```
Set<String> names = new HashSet<>();
```

arba

```
Set<String> names = new TreeSet<>();
```

Skirsis tik vidinė realizacija.