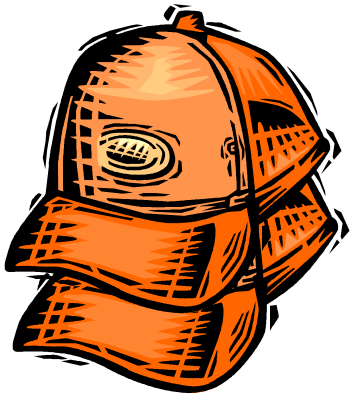
Binary Search Trees

Dawson Zhou



(recap)

Required for `Set`:

- `size`
- `isEmpty`
- `contains`
- `add`
- no duplicates

Idea: vector

Idea: vector

```
add(38)
```

38

Idea: vector

```
add(12)
```

Idea: vector

```
add(9)
```

Idea: vector

. . .

38	12	9	75	2	65	23

Idea: vector

```
add(2)
```

Idea: vector

`add(2)`

38	12	9	75	2	65	23

Idea: vector

`add` runs in `O(n)` time

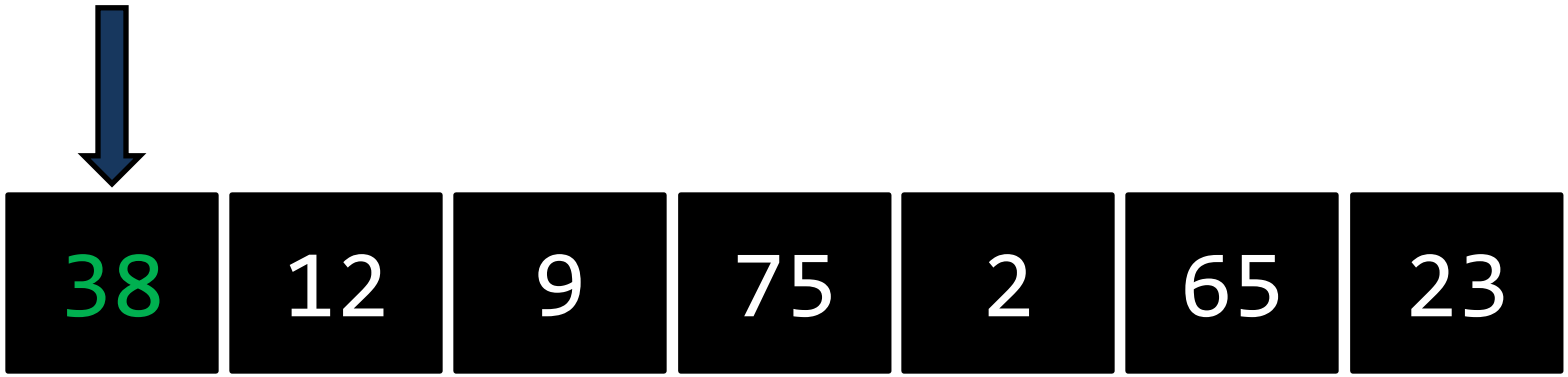
Idea: vector

`contains(38)`

38	12	9	75	2	65	23

Idea: vector

`contains(38)`



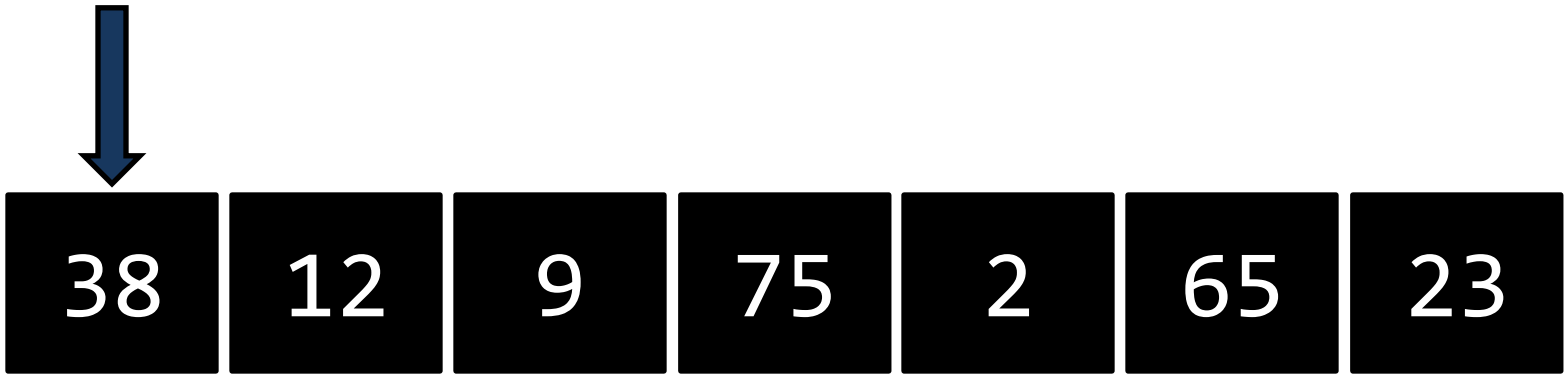
Idea: vector

`contains(2)`

38	12	9	75	2	65	23

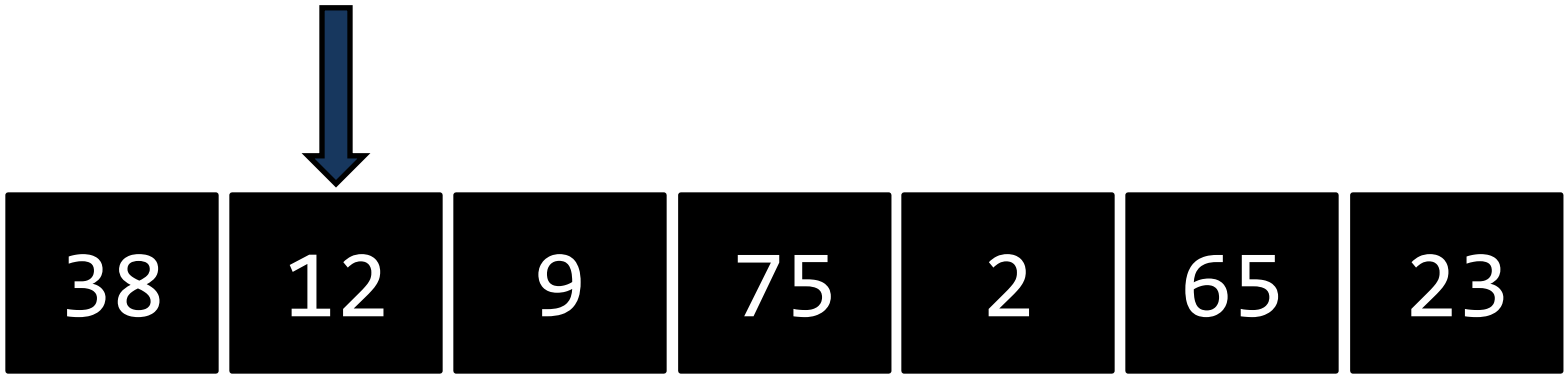
Idea: vector

`contains(2)`



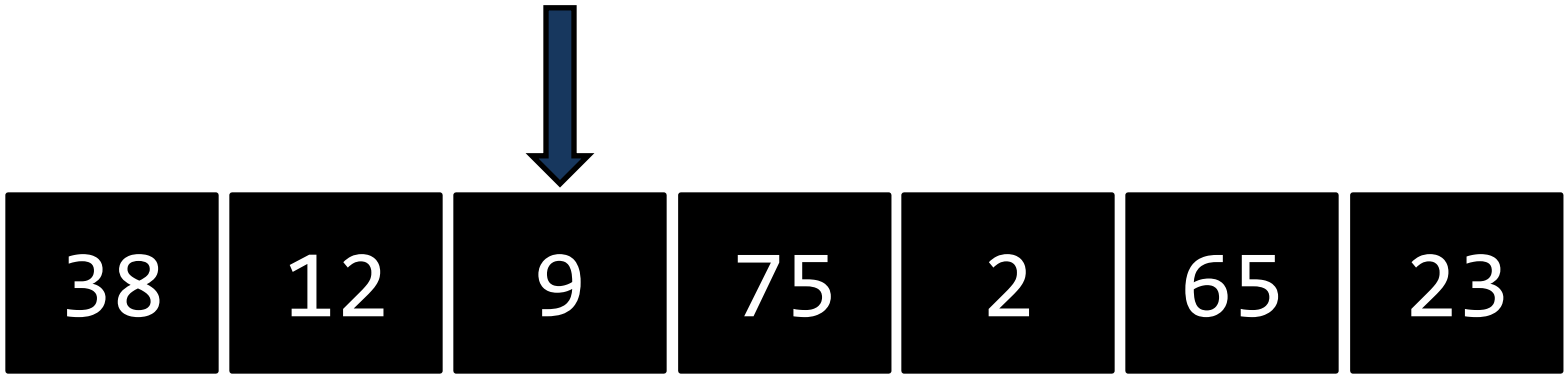
Idea: vector

`contains(2)`



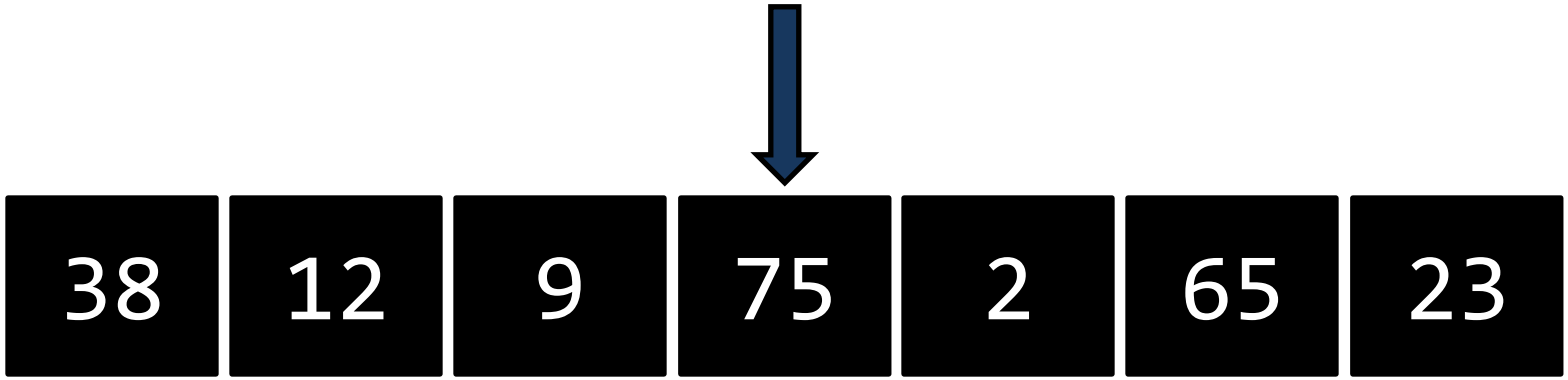
Idea: vector

`contains(2)`



Idea: vector

`contains(2)`



Idea: vector

`contains(2)`

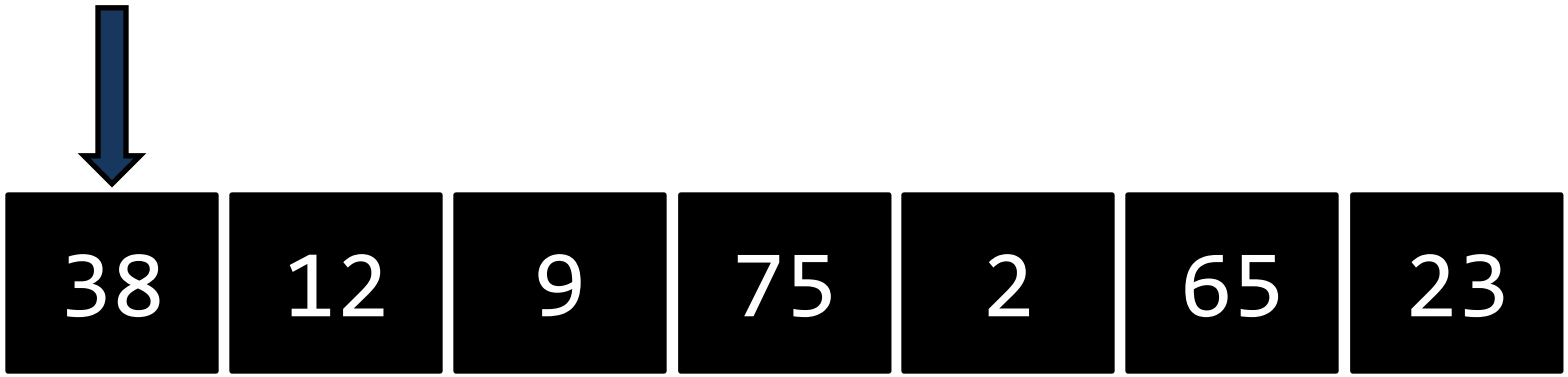
Idea: vector

```
contains(0)
```

38	12	9	75	2	65	23

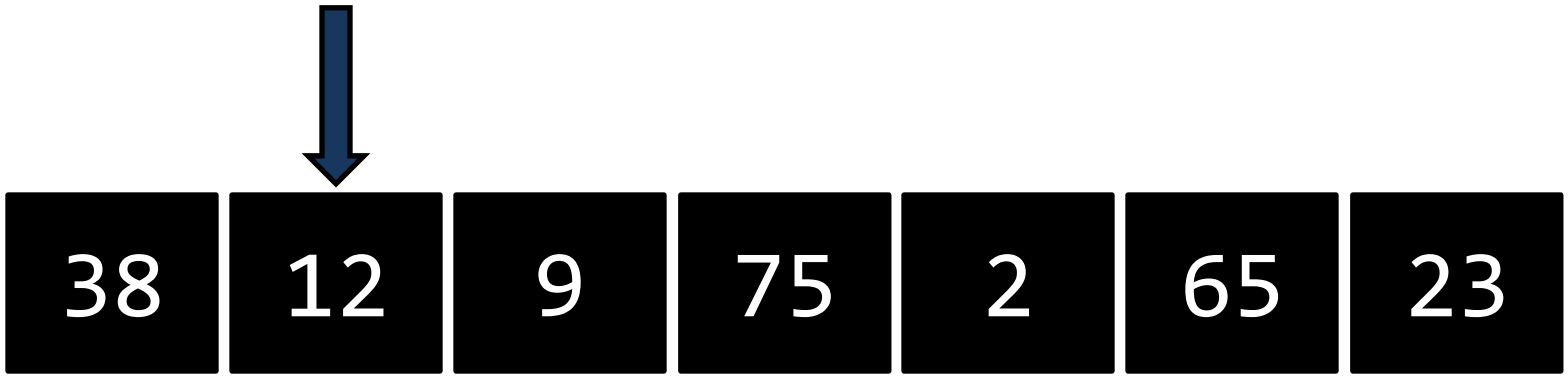
Idea: vector

`contains(0)`



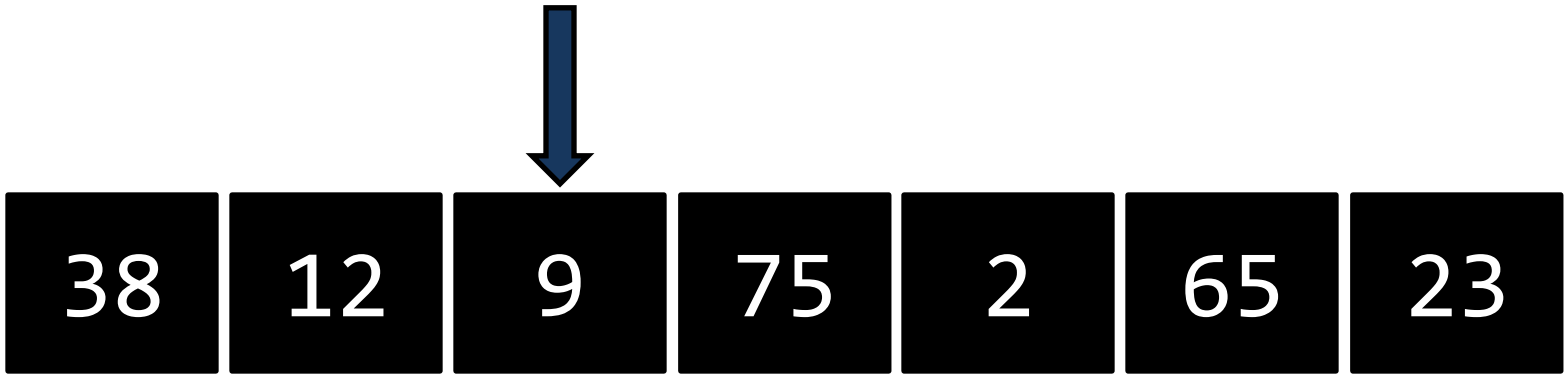
Idea: vector

`contains(0)`



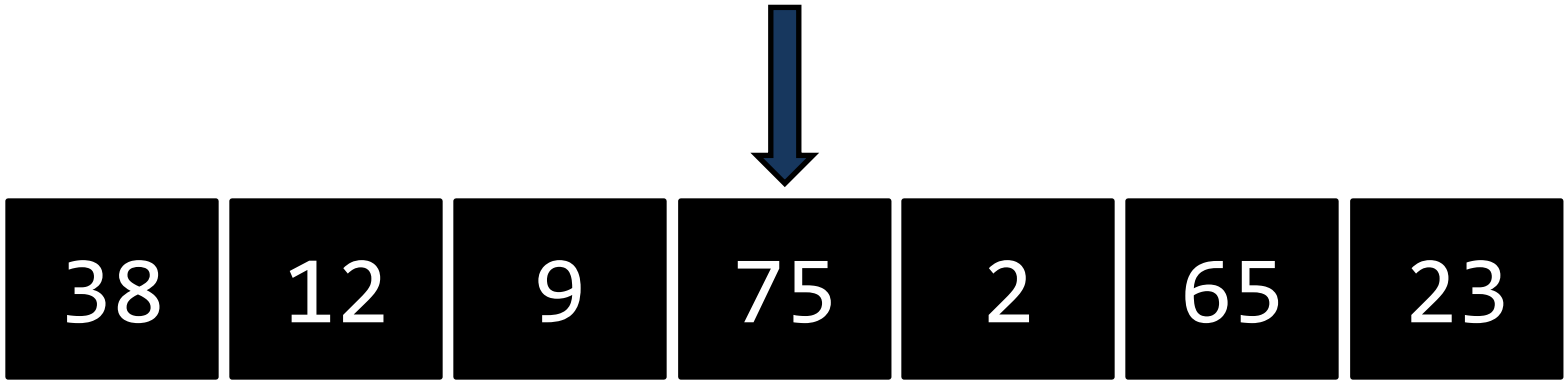
Idea: vector

`contains(0)`



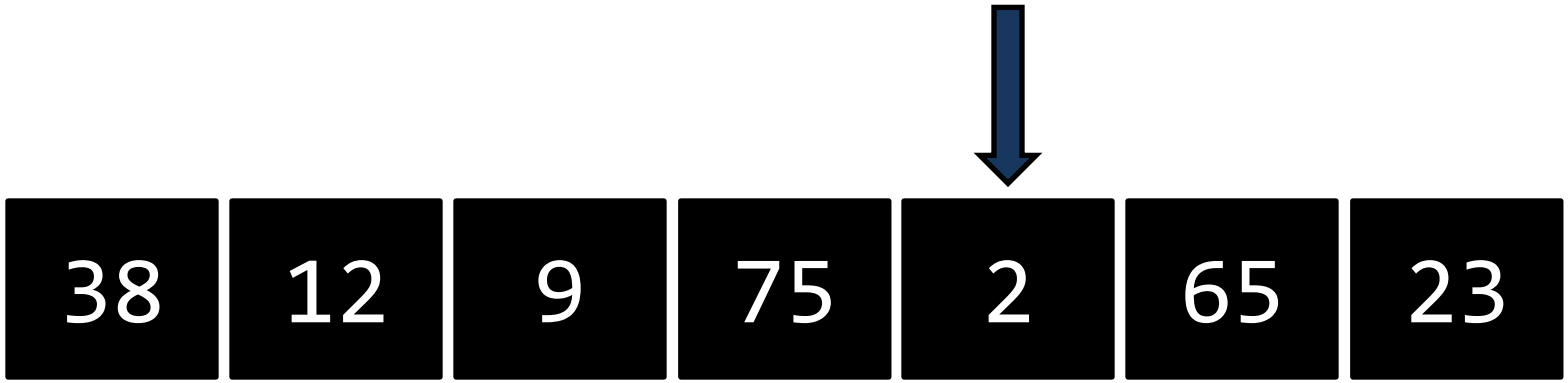
Idea: vector

`contains(0)`



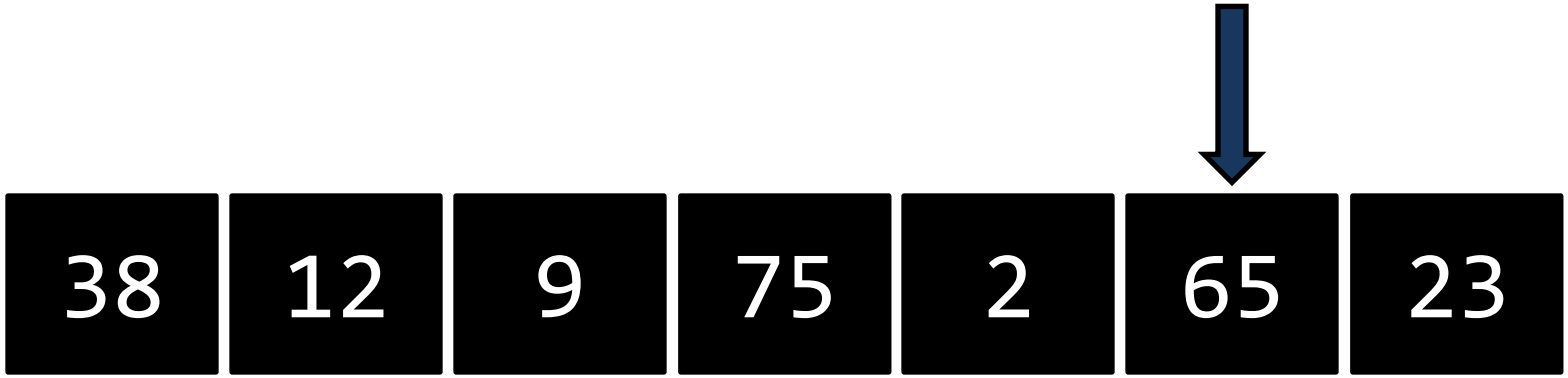
Idea: vector

`contains(0)`



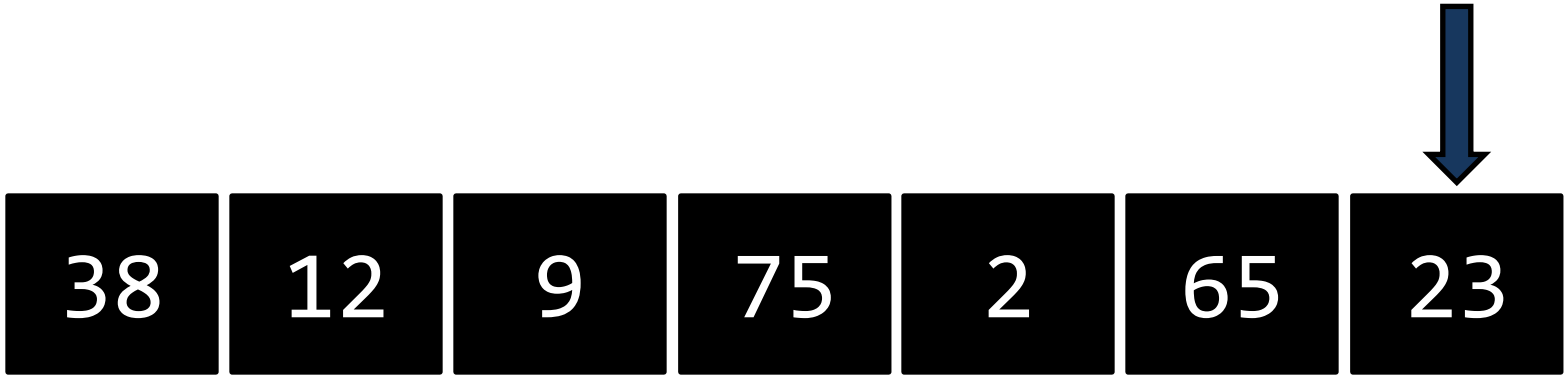
Idea: vector

`contains(0)`



Idea: vector

`contains(0)`



Idea: vector

`contains(0)`

38	12	9	75	2	65	23

Idea: vector

`contains` runs in `O(n)` time

The runtime behavior of your data structure is terrible!

Required for `Set`:

- `size`
- `isEmpty`
- `contains`
- `add`
- no duplicates

Required for `Set`:

- `size`
- `isEmpty`
- `contains`
- `add`
- no duplicates

NOT required:

no need to track sequence of elements

Idea: sorted vector

2	9	12	23	38	65	75

Binary Search (demo)

Idea: sorted vector

`add` runs in `O(n)` time

2	9	12	23	38	65	75

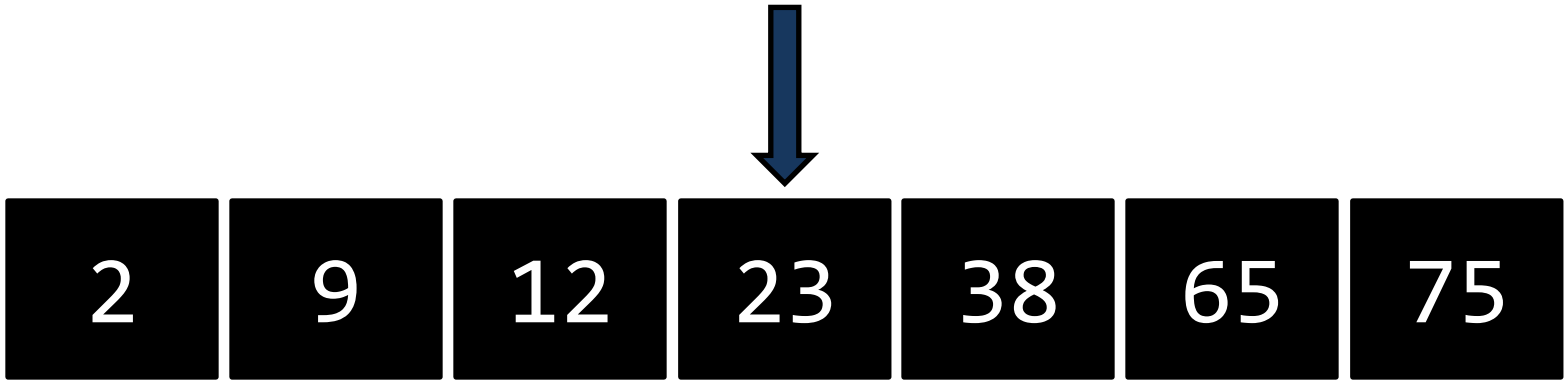
Idea: sorted vector

```
contains(12)
```

2	9	12	23	38	65	75

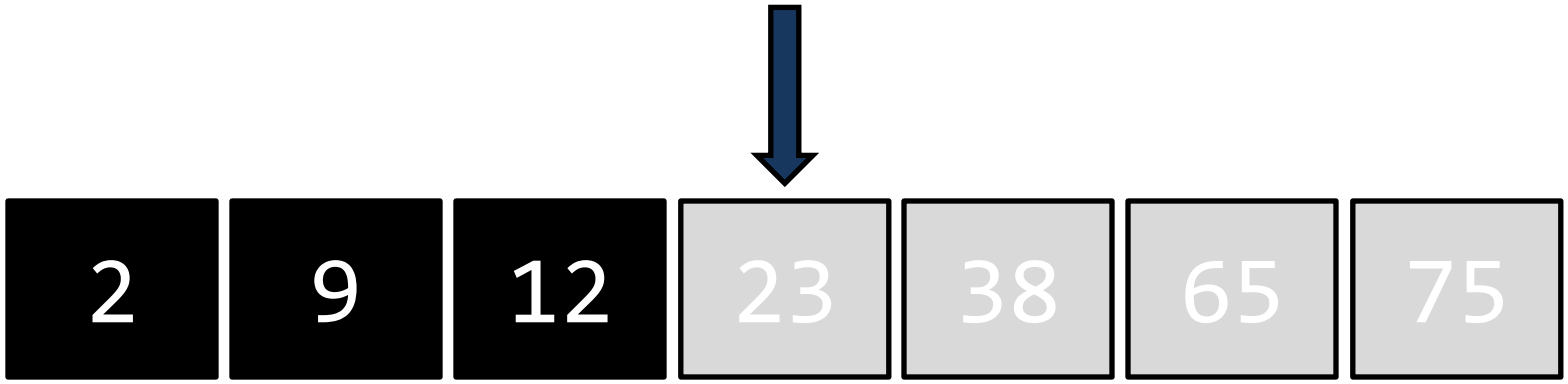
Idea: sorted vector

`contains(12)`



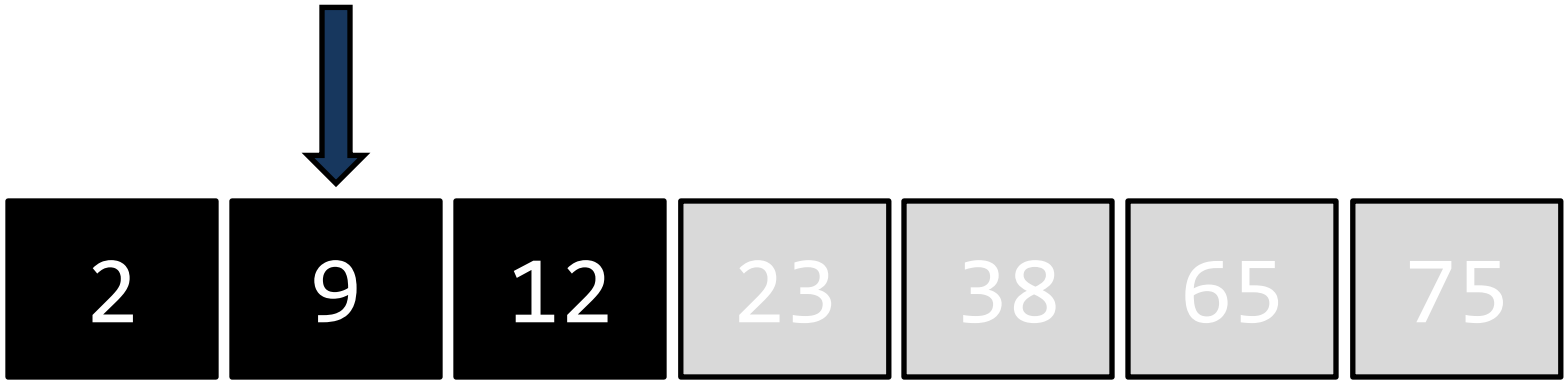
Idea: sorted vector

`contains(12)`



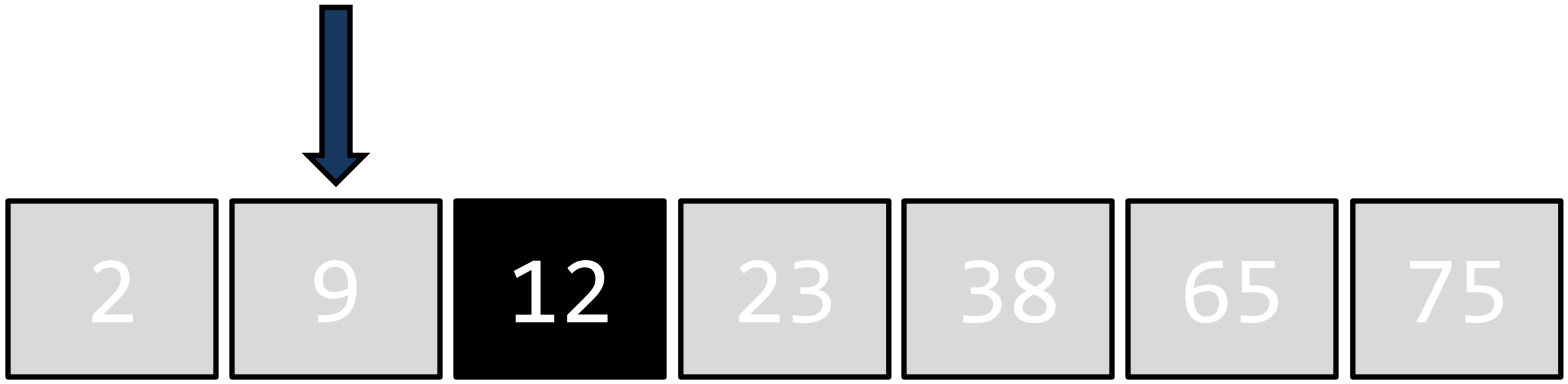
Idea: sorted vector

`contains(12)`



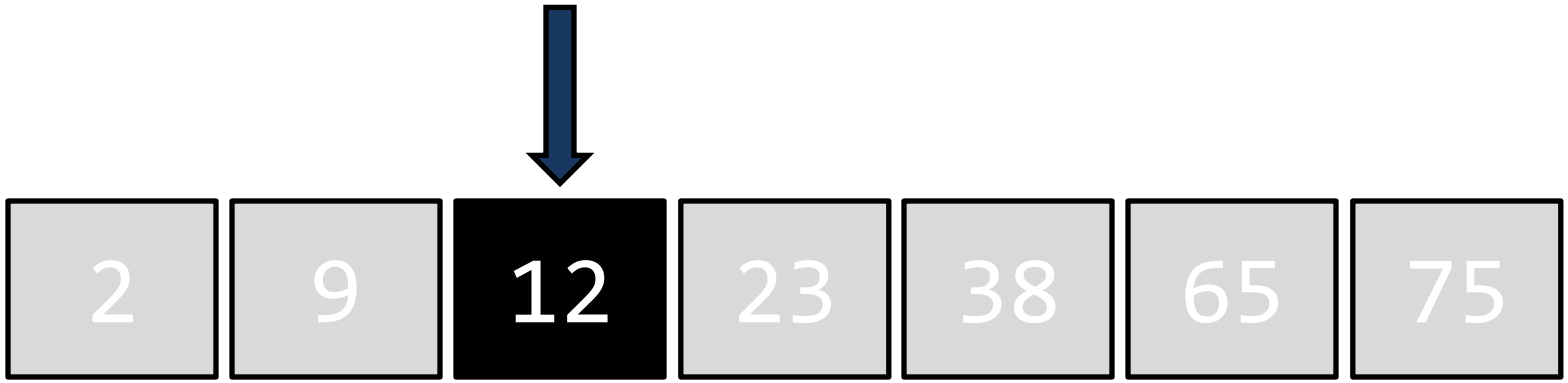
Idea: sorted vector

`contains(12)`



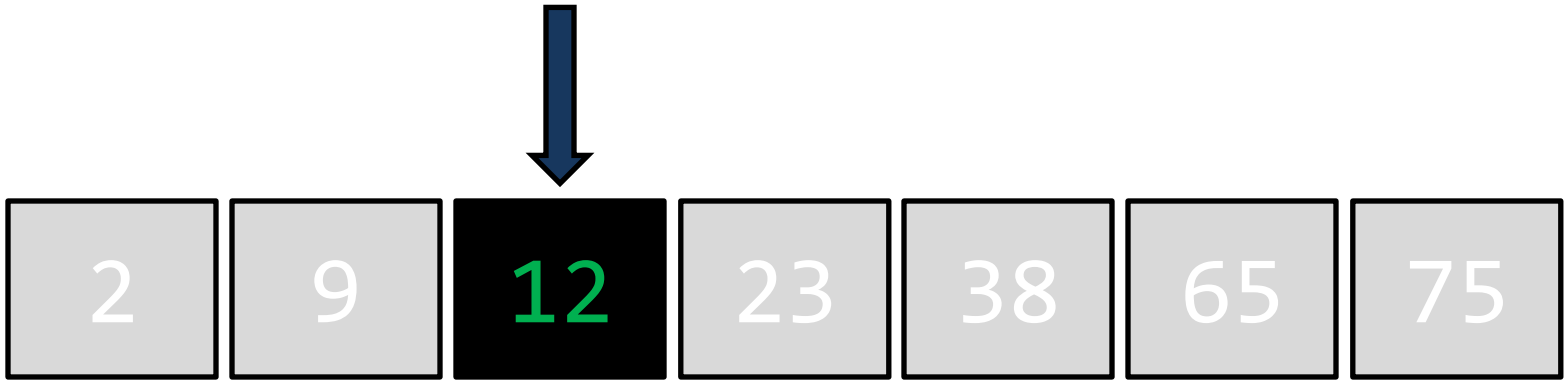
Idea: sorted vector

`contains(12)`



Idea: sorted vector

`contains(12)`



Idea: sorted vector

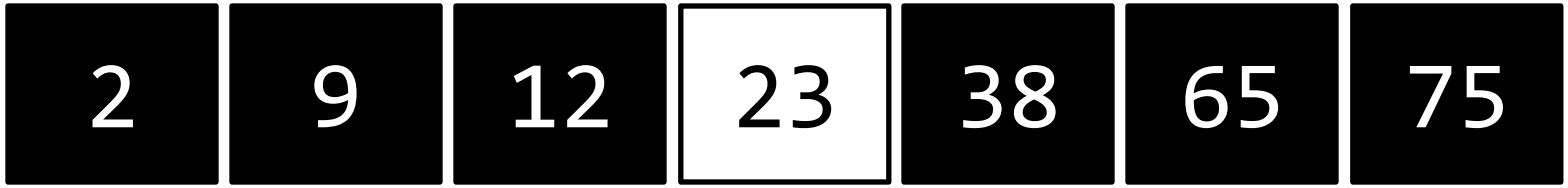
`contains` runs in `O(log n)` time

2	9	12	23	38	65	75

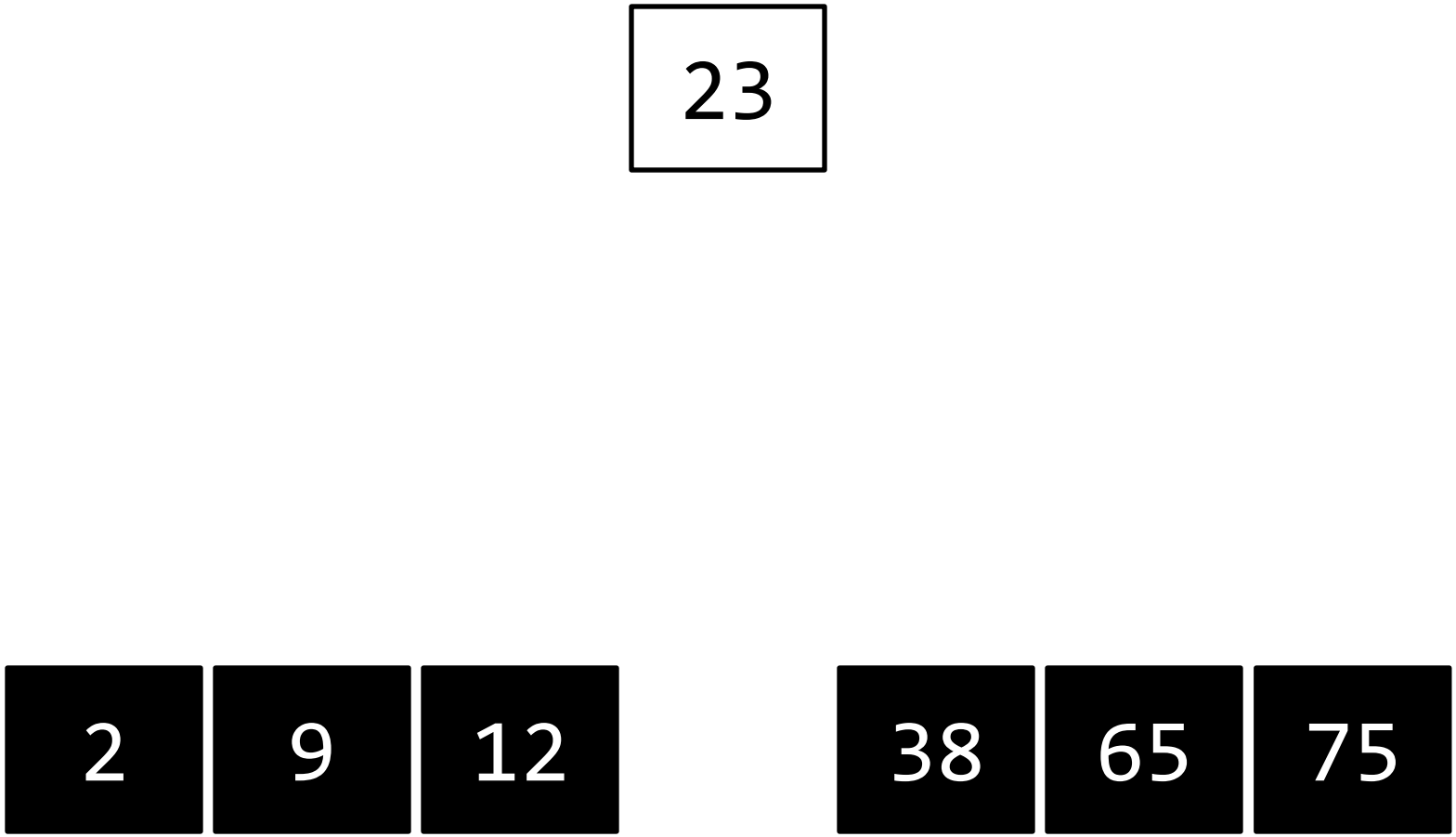
Idea: binary search tree

2	9	12	23	38	65	75

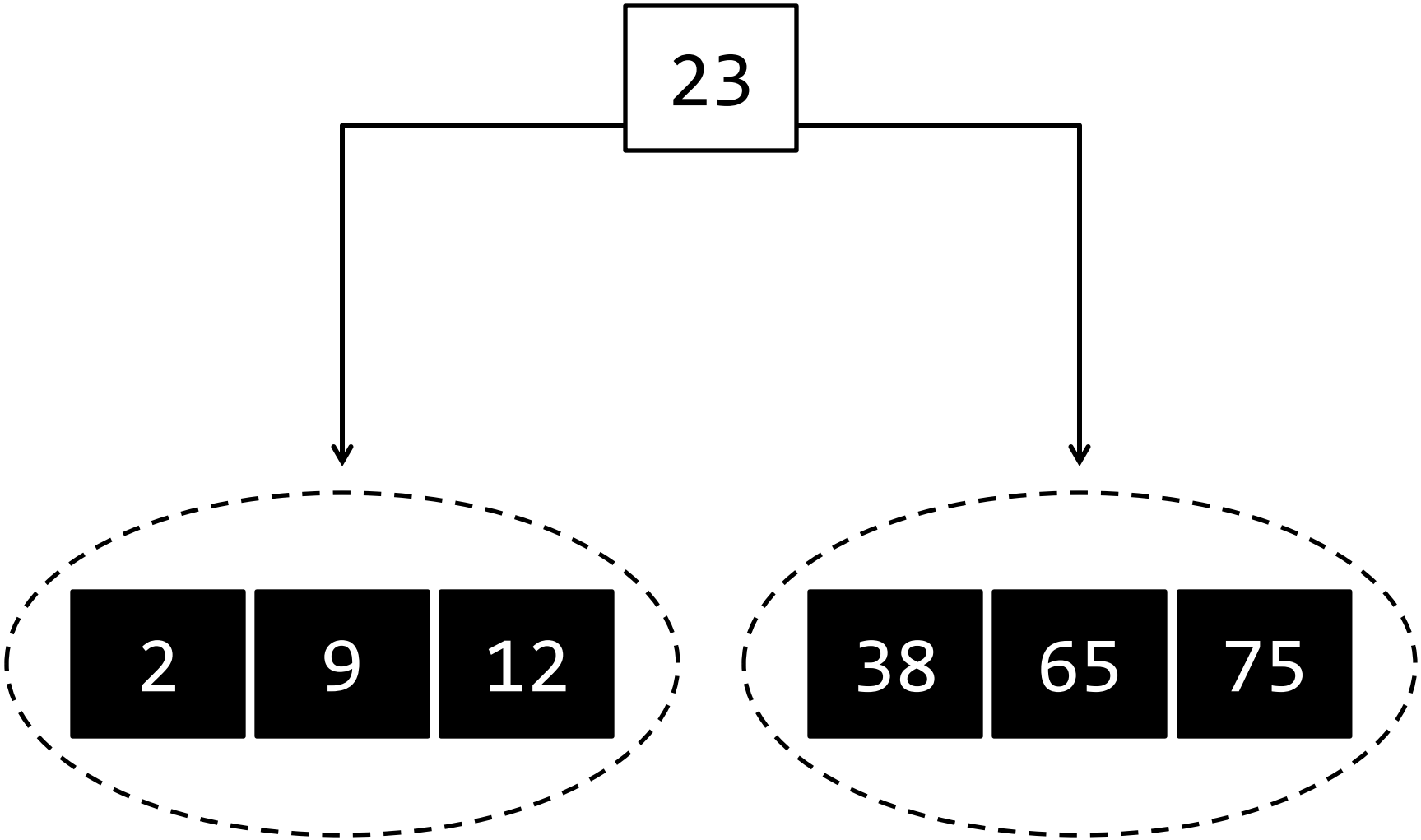
Idea: binary search tree



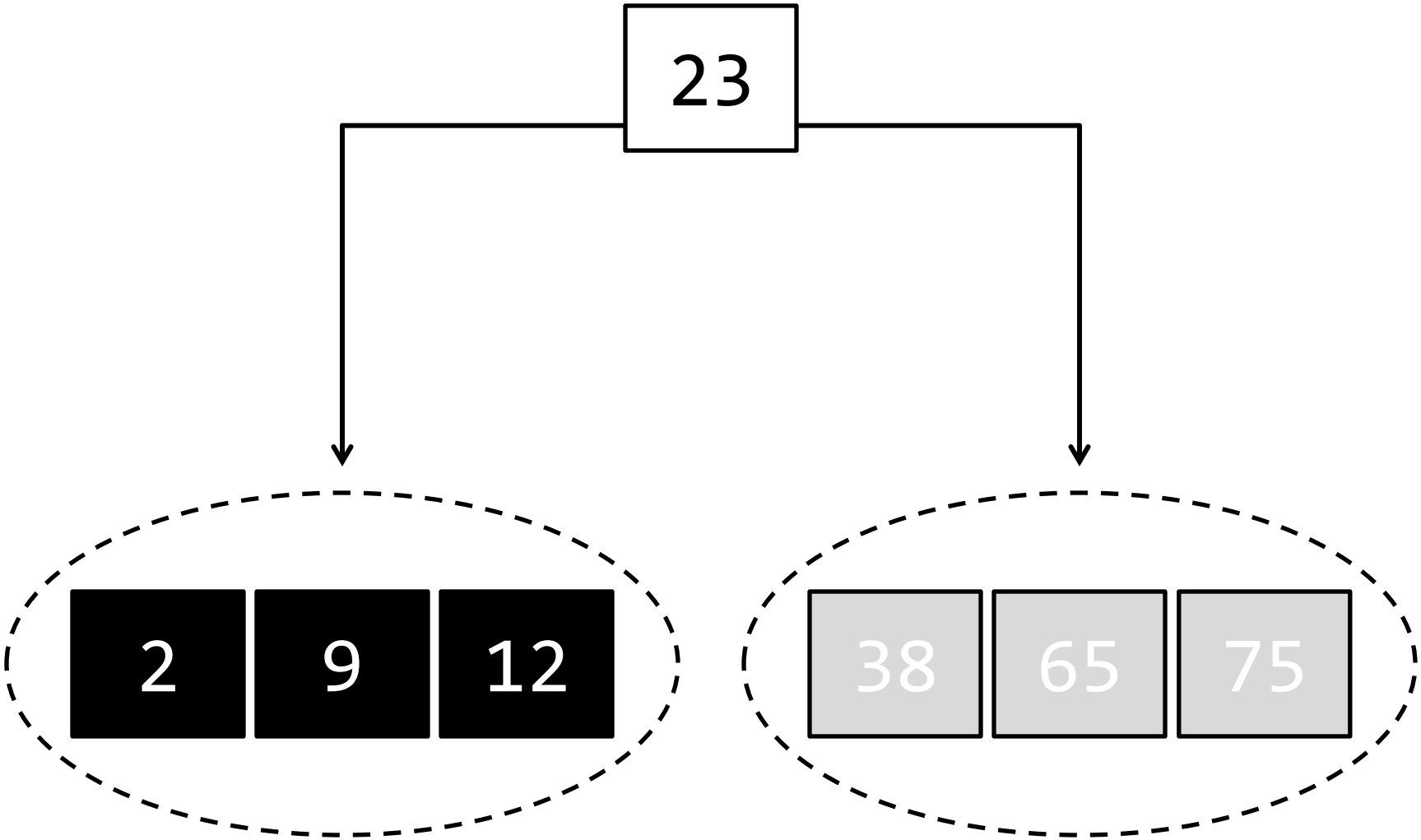
Idea: binary search tree



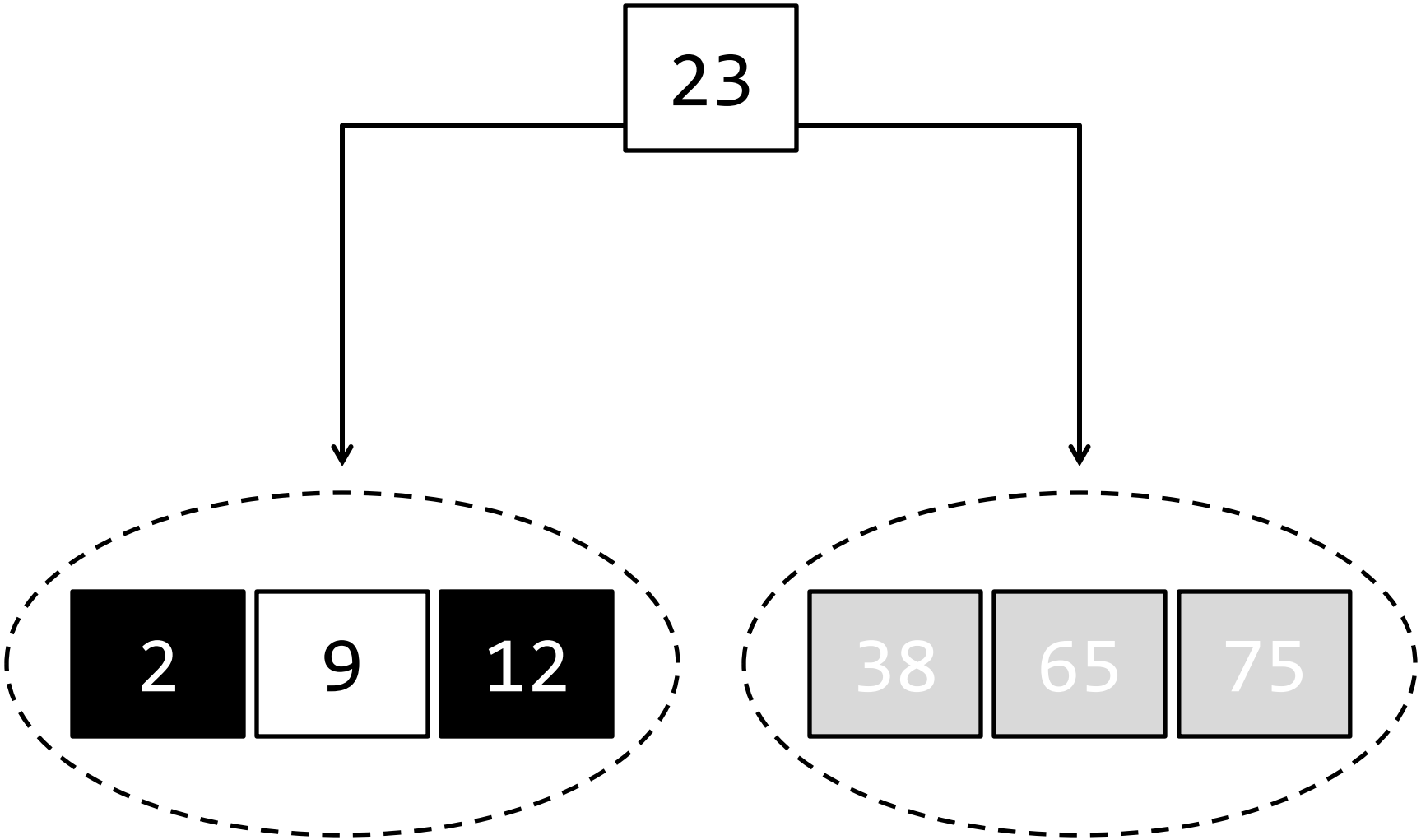
Idea: binary search tree



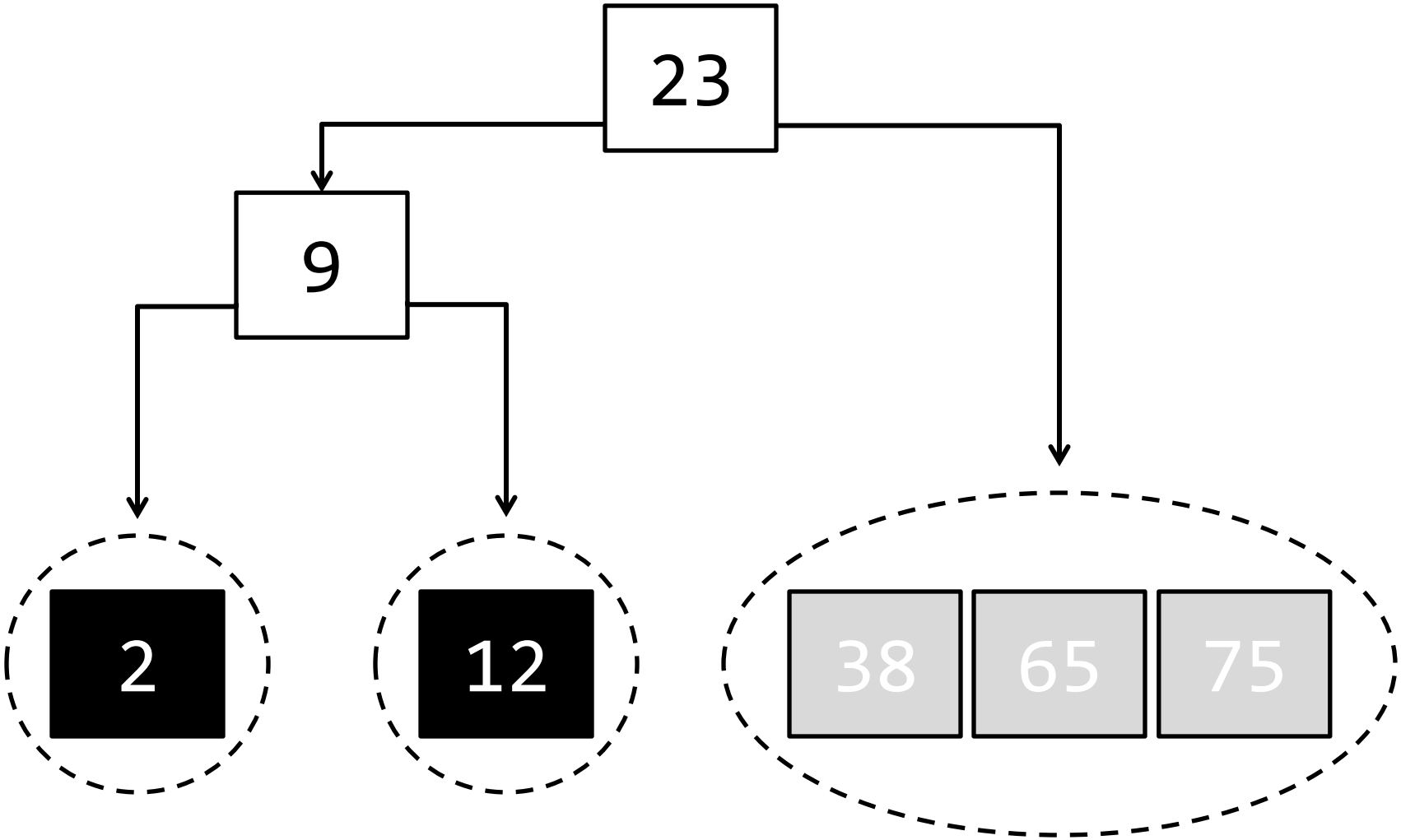
Idea: binary search tree



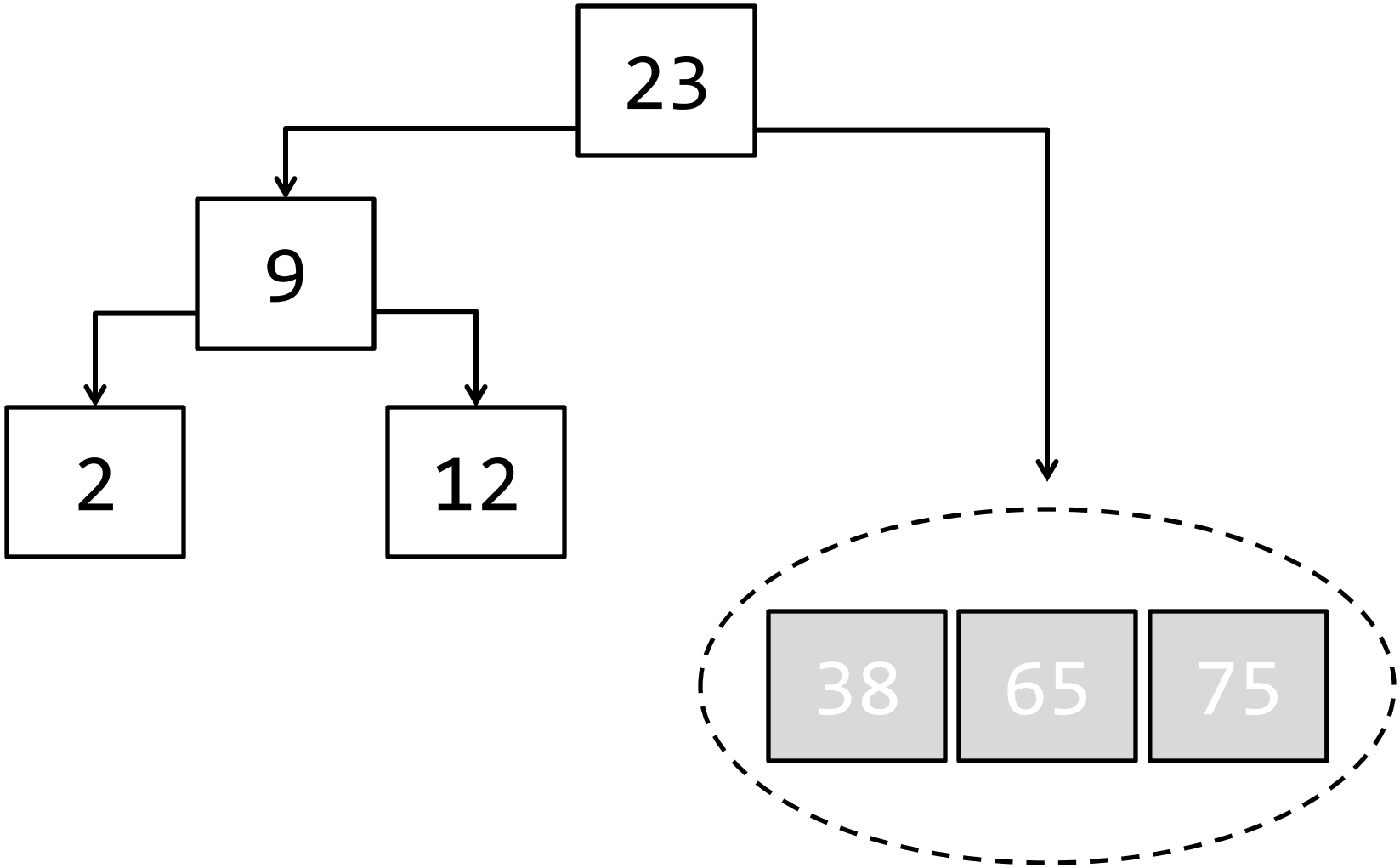
Idea: binary search tree



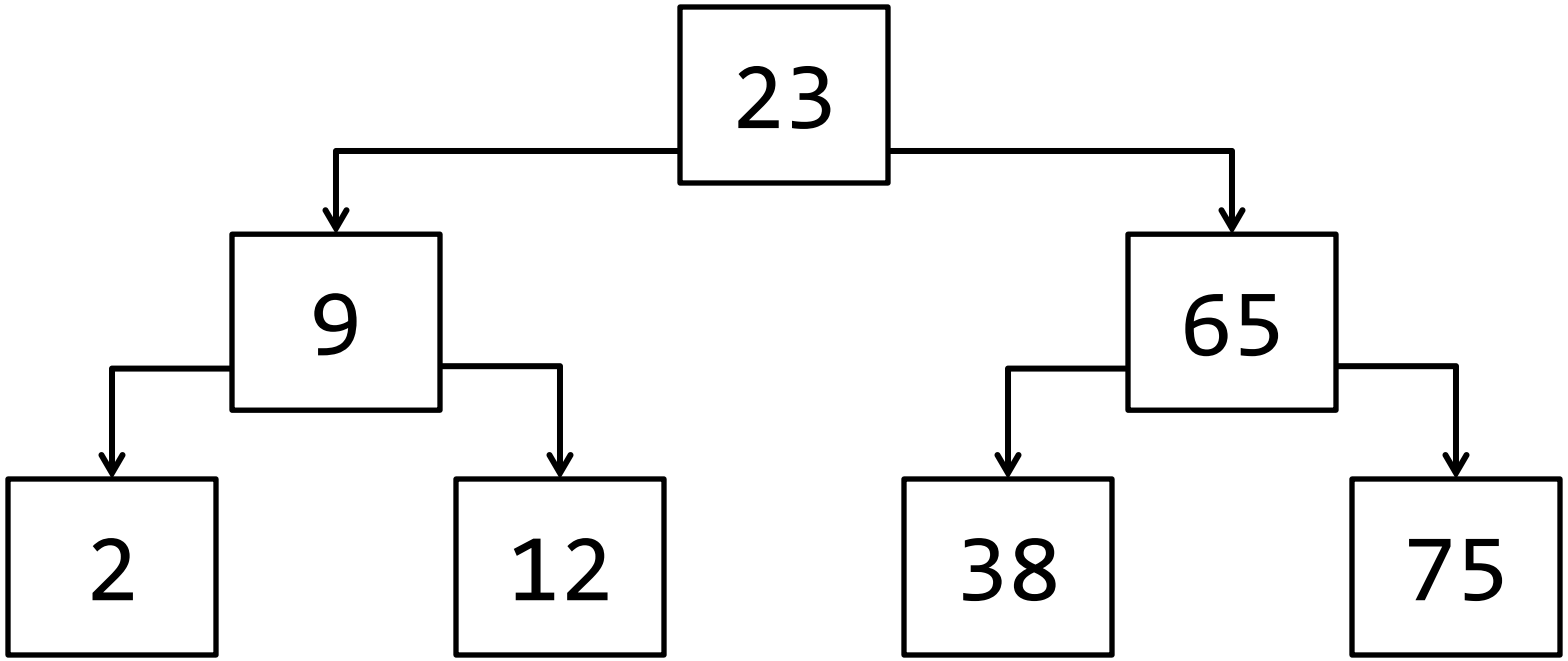
Idea: binary search tree



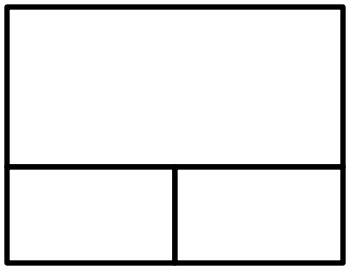
Idea: binary search tree



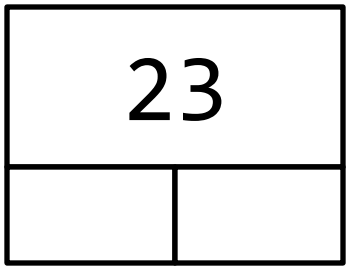
Idea: binary search tree



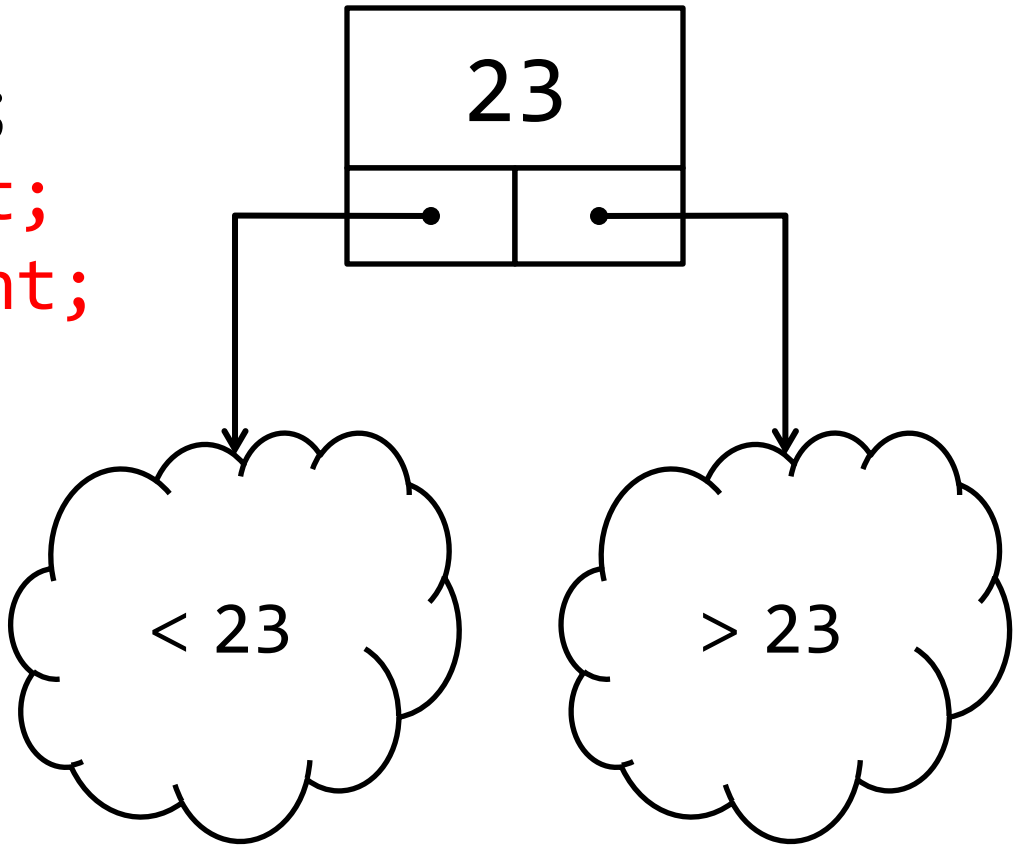
```
struct Node {
    int value;
    Node* left;
    Node* right;
};
```



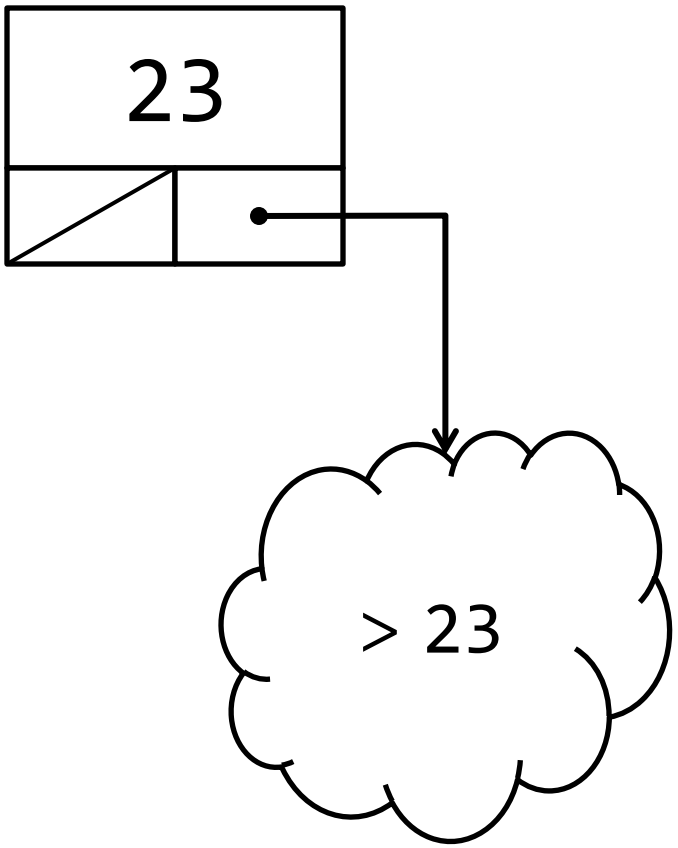
```
struct Node {
    int value;
    Node* left;
    Node* right;
};
```



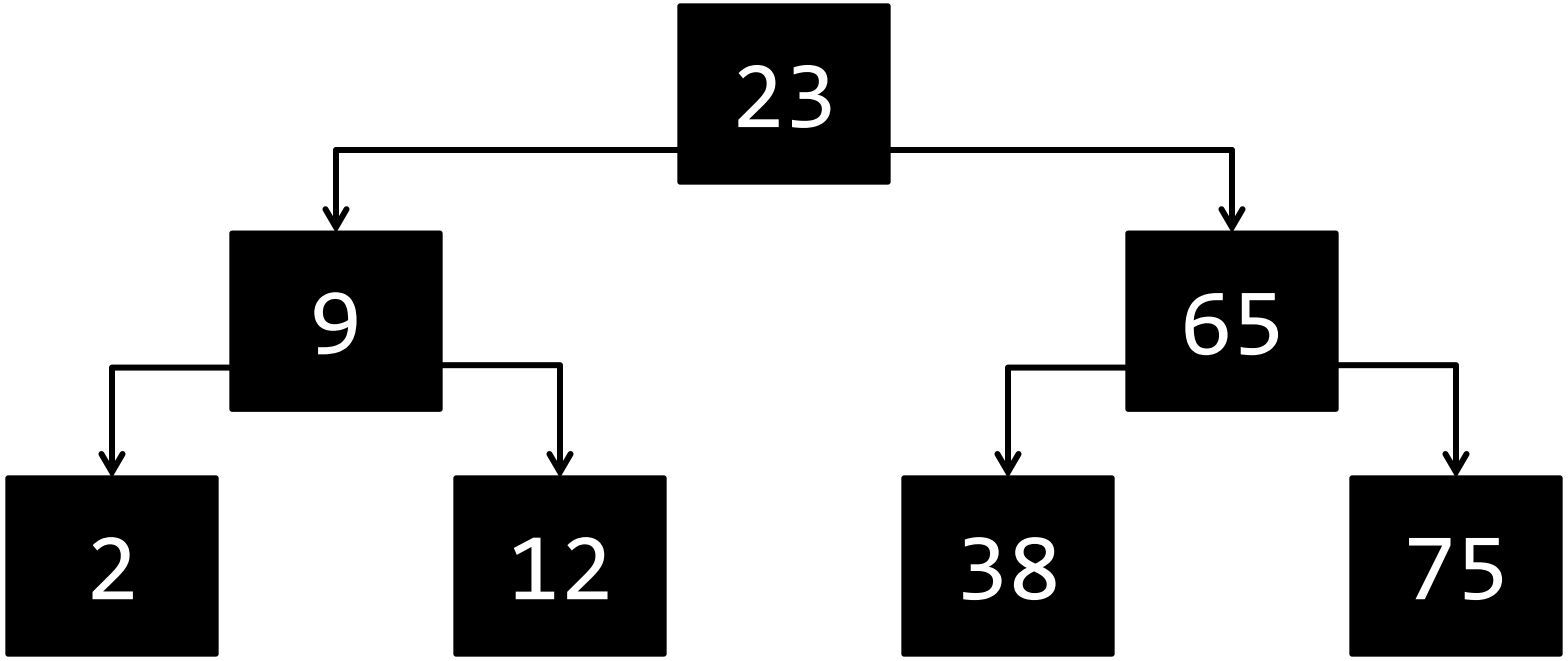
```
struct Node {
    int value;
    Node* left;
    Node* right;
};
```



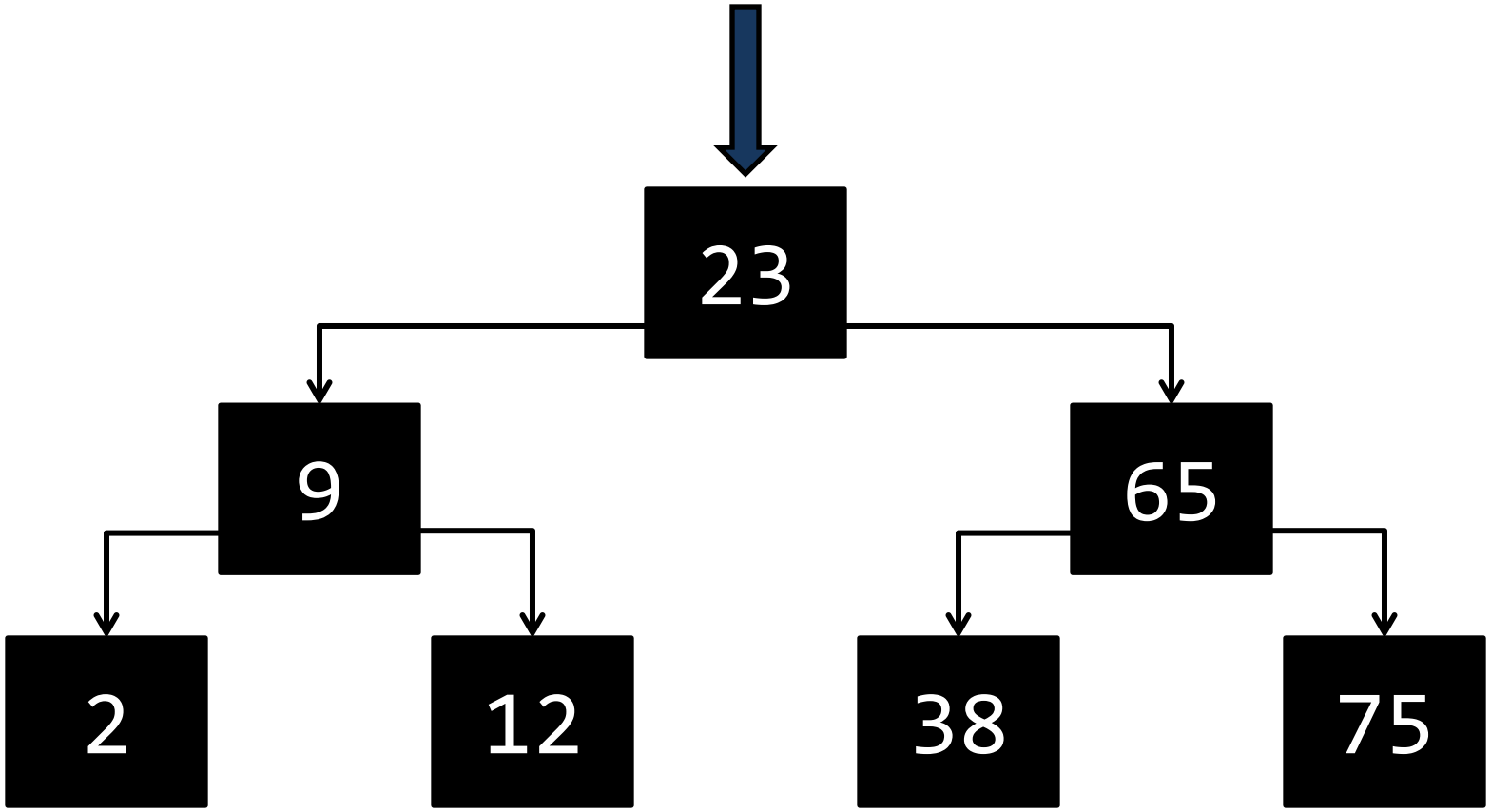
```
struct Node {
    int value;
    Node* left;
    Node* right;
};
```



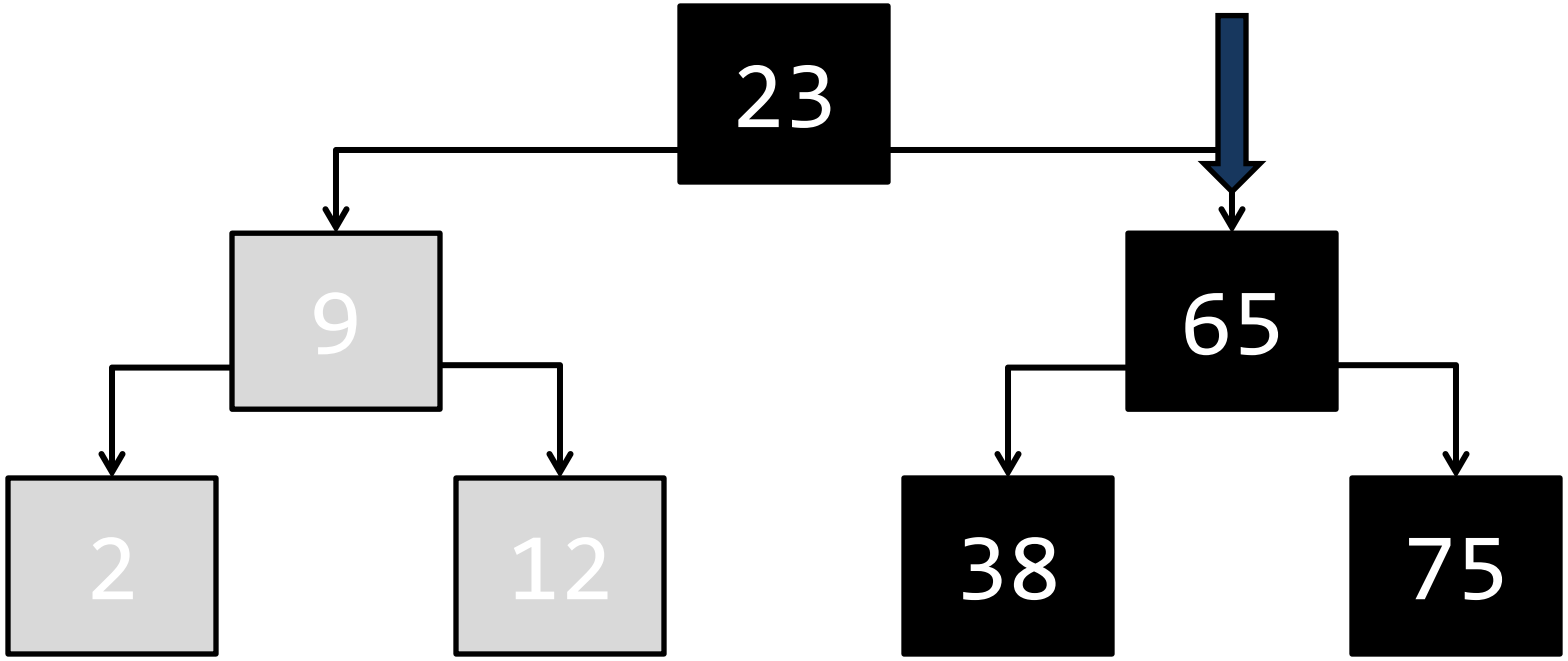
contains(38)



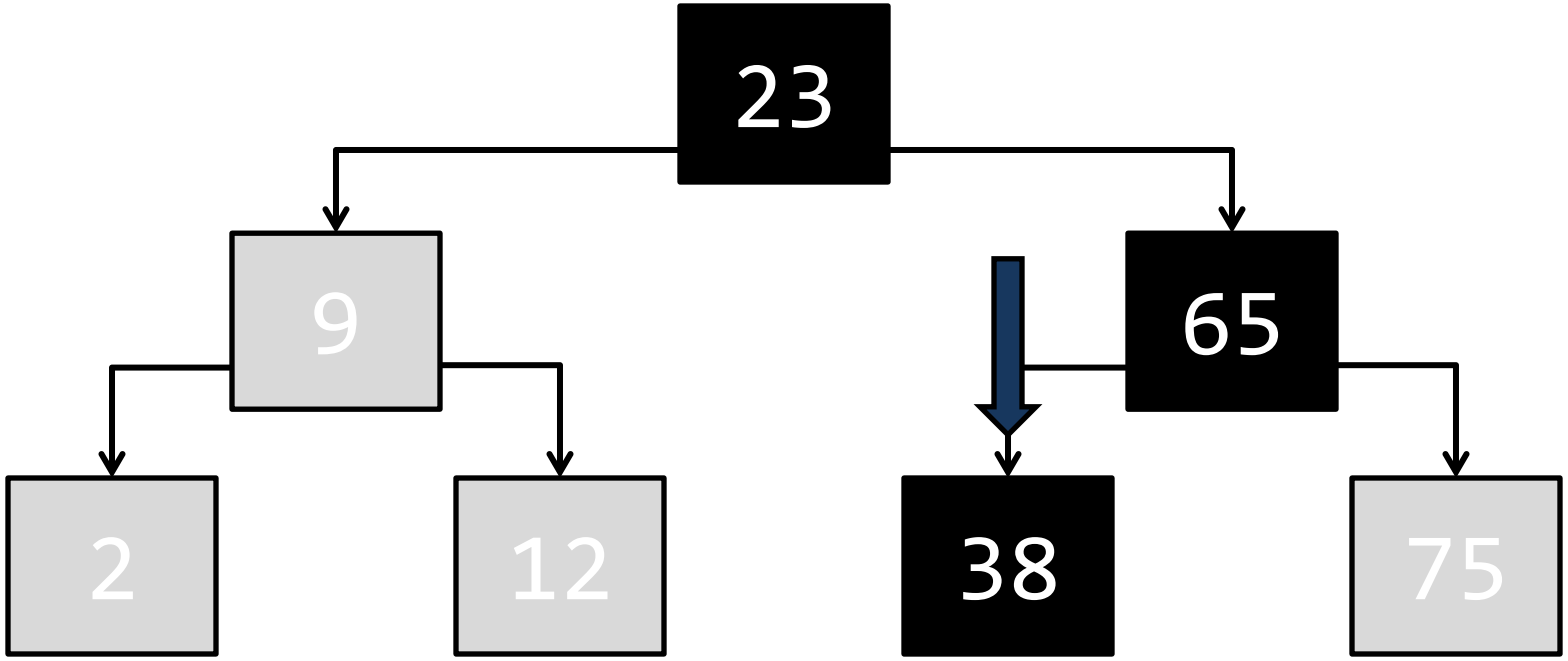
contains(38)



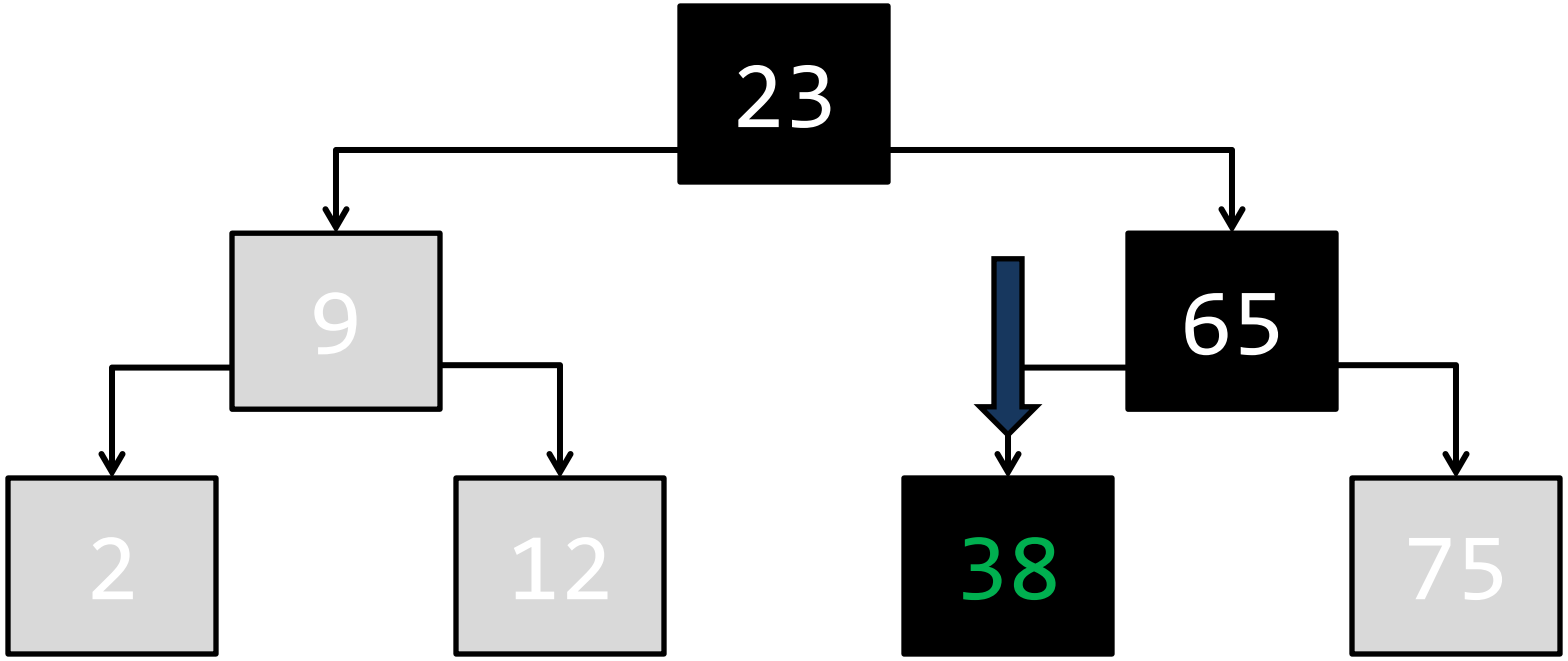
contains(38)



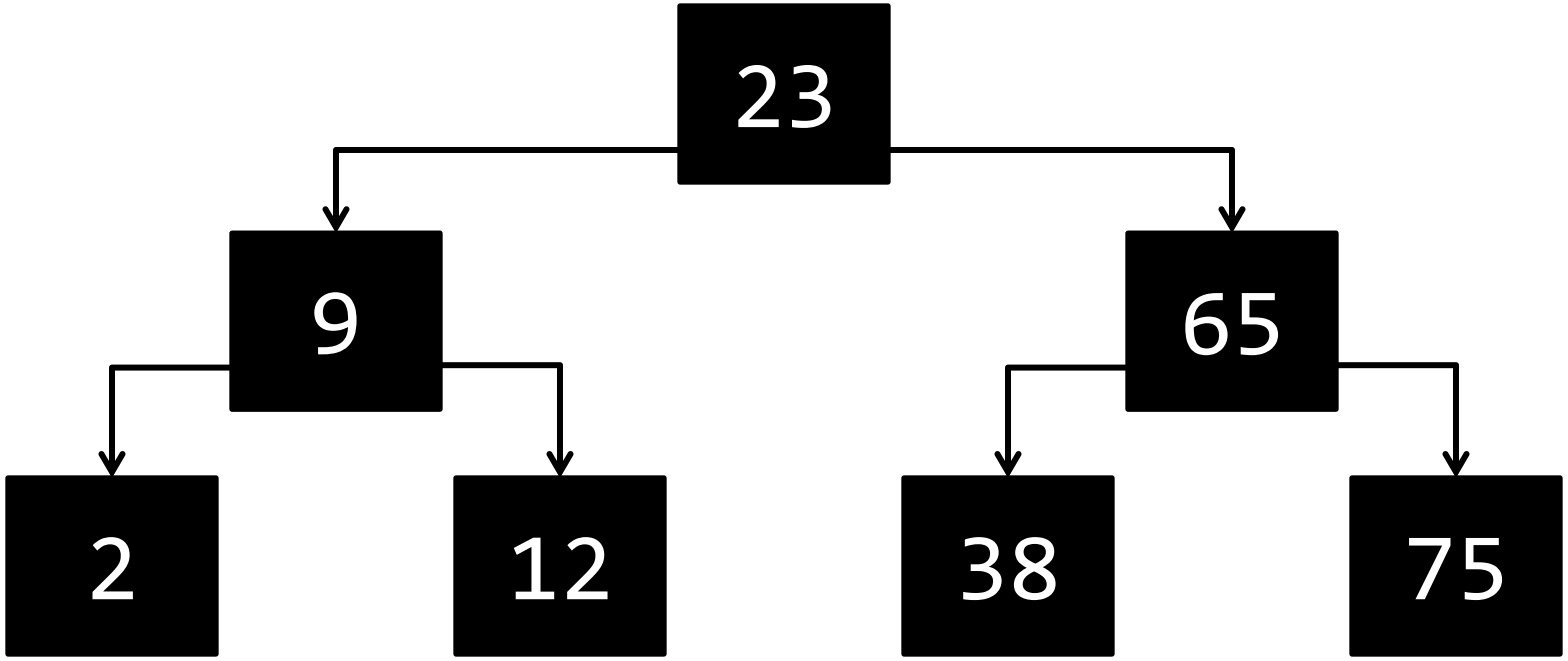
contains(38)

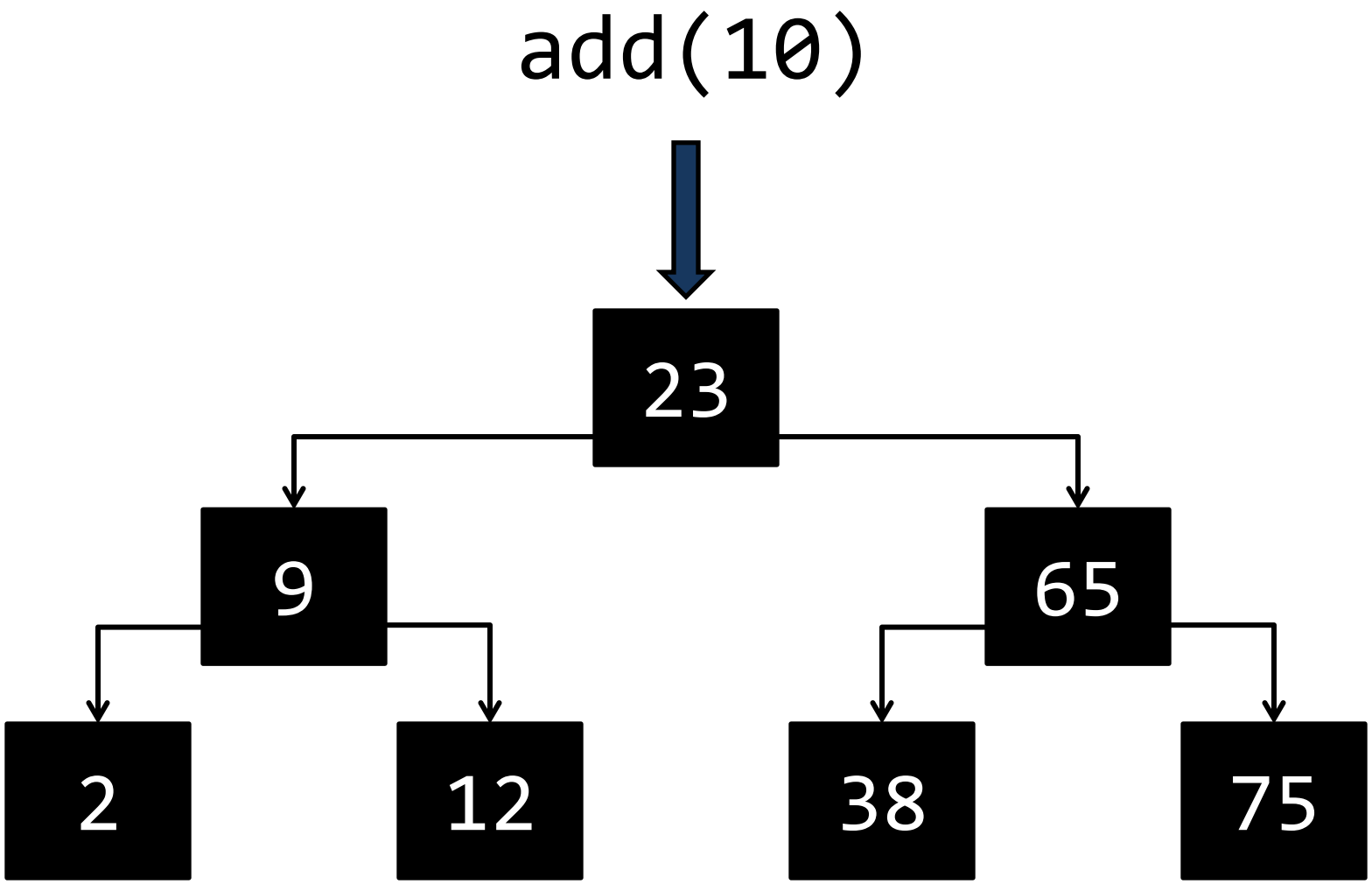


contains(38)

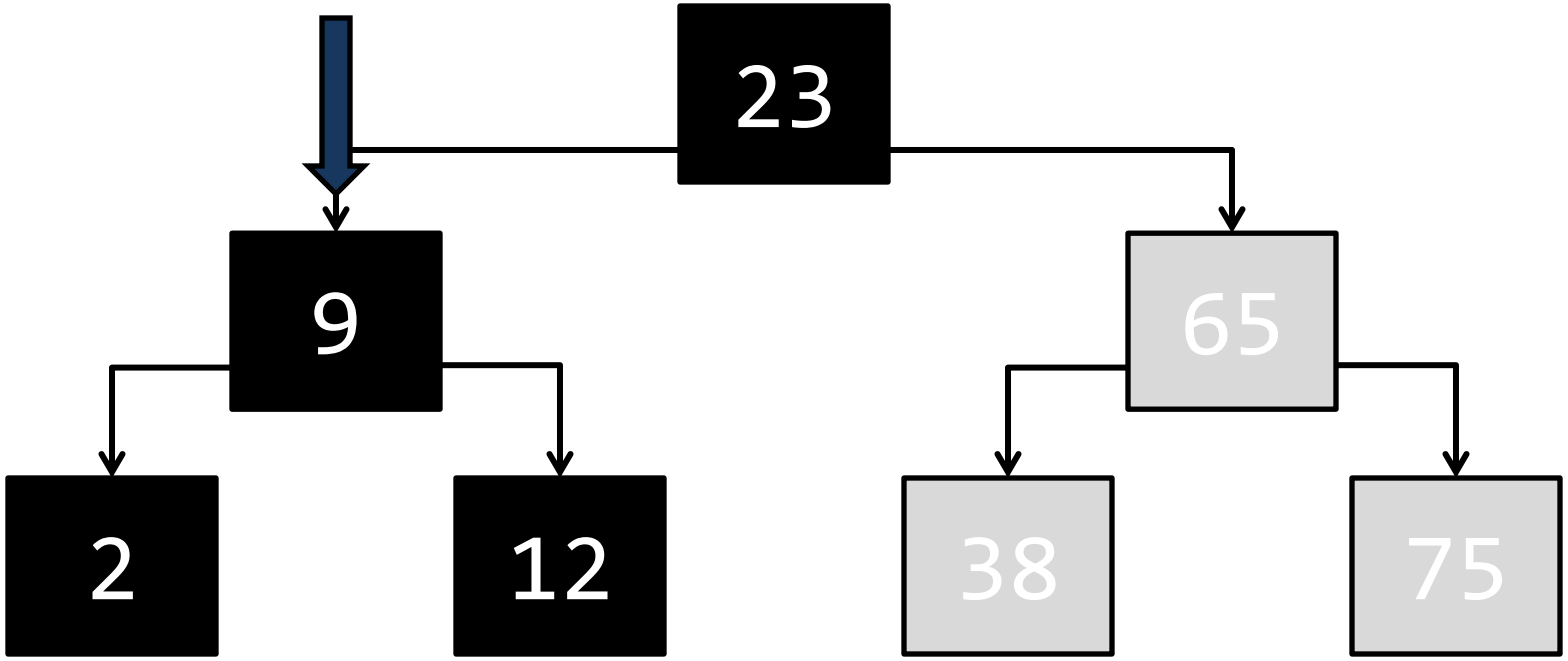


add(10)

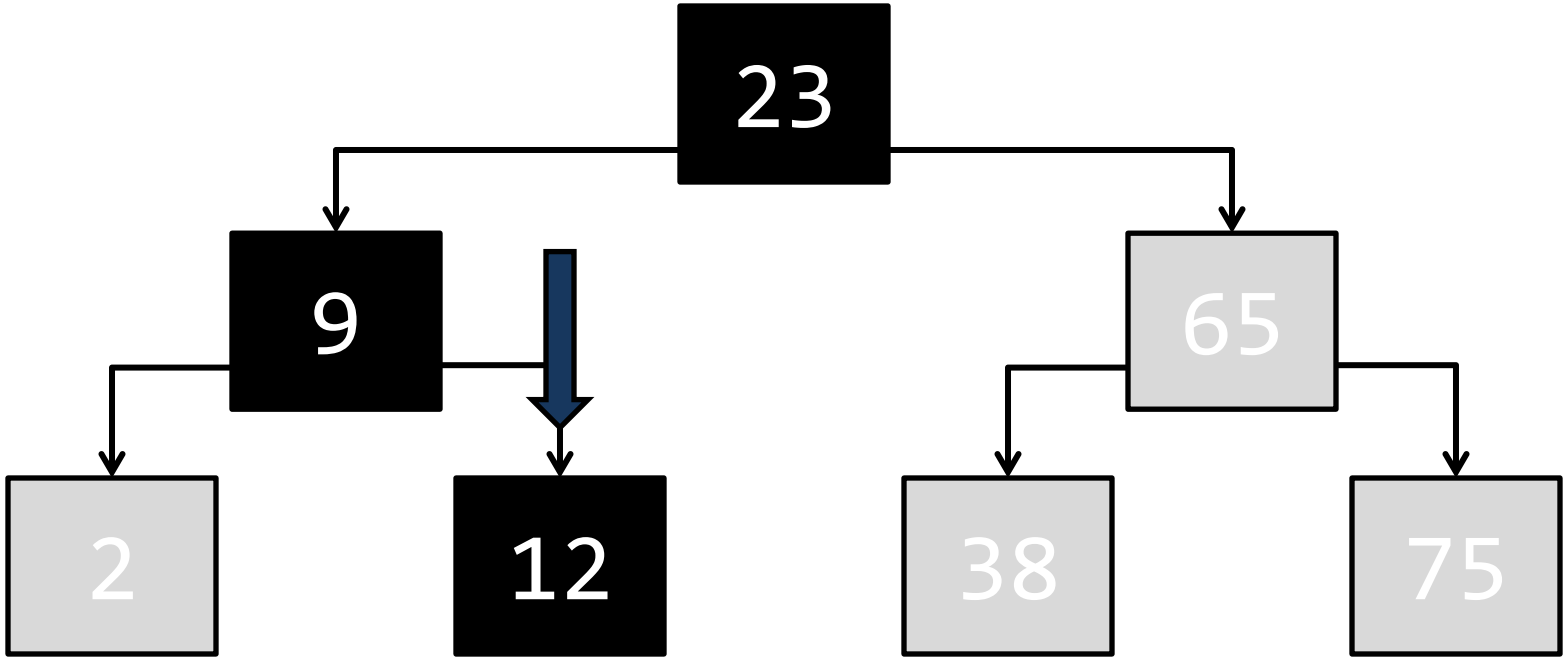



add(10)
23
9
65
2
12
38
75

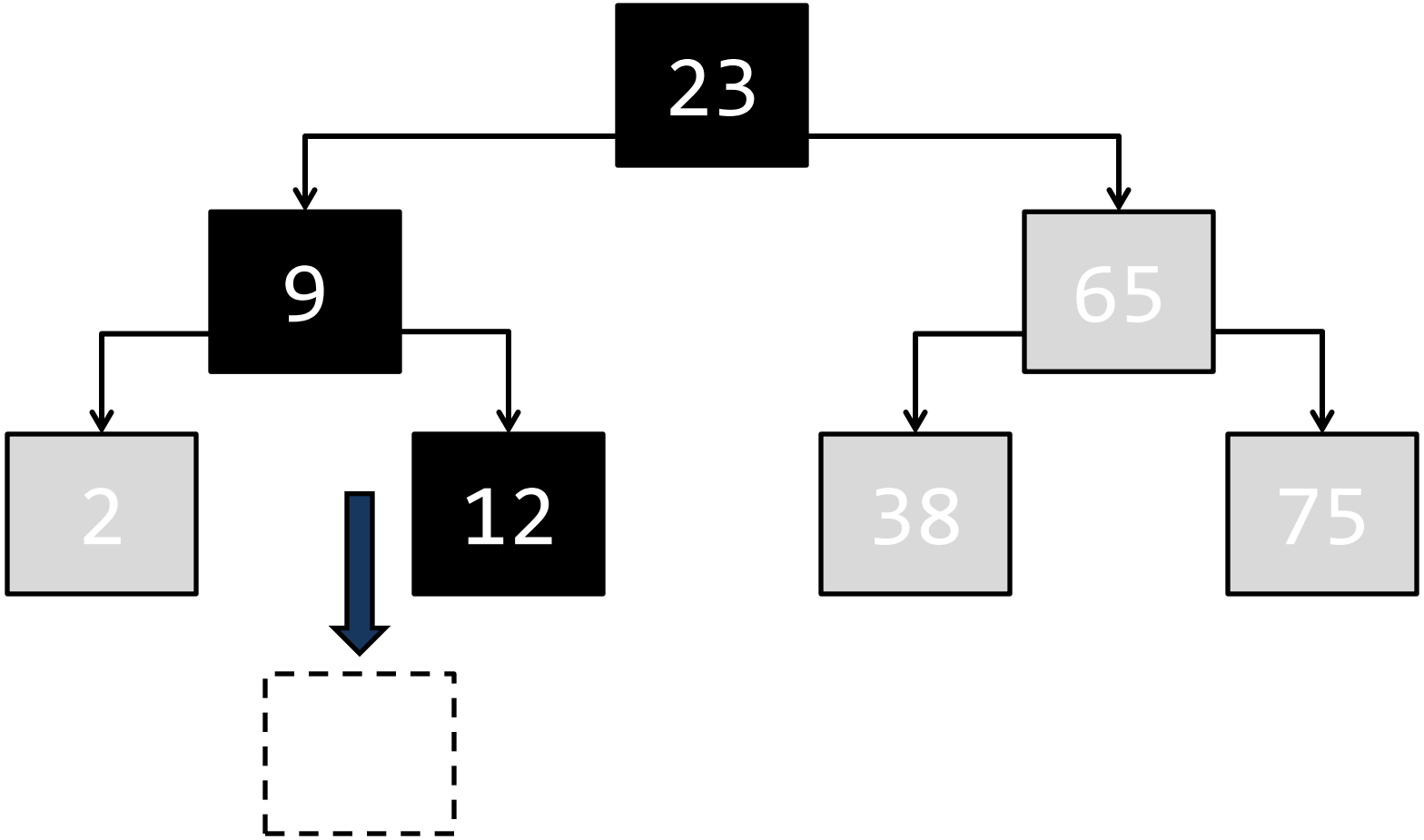
add(10)



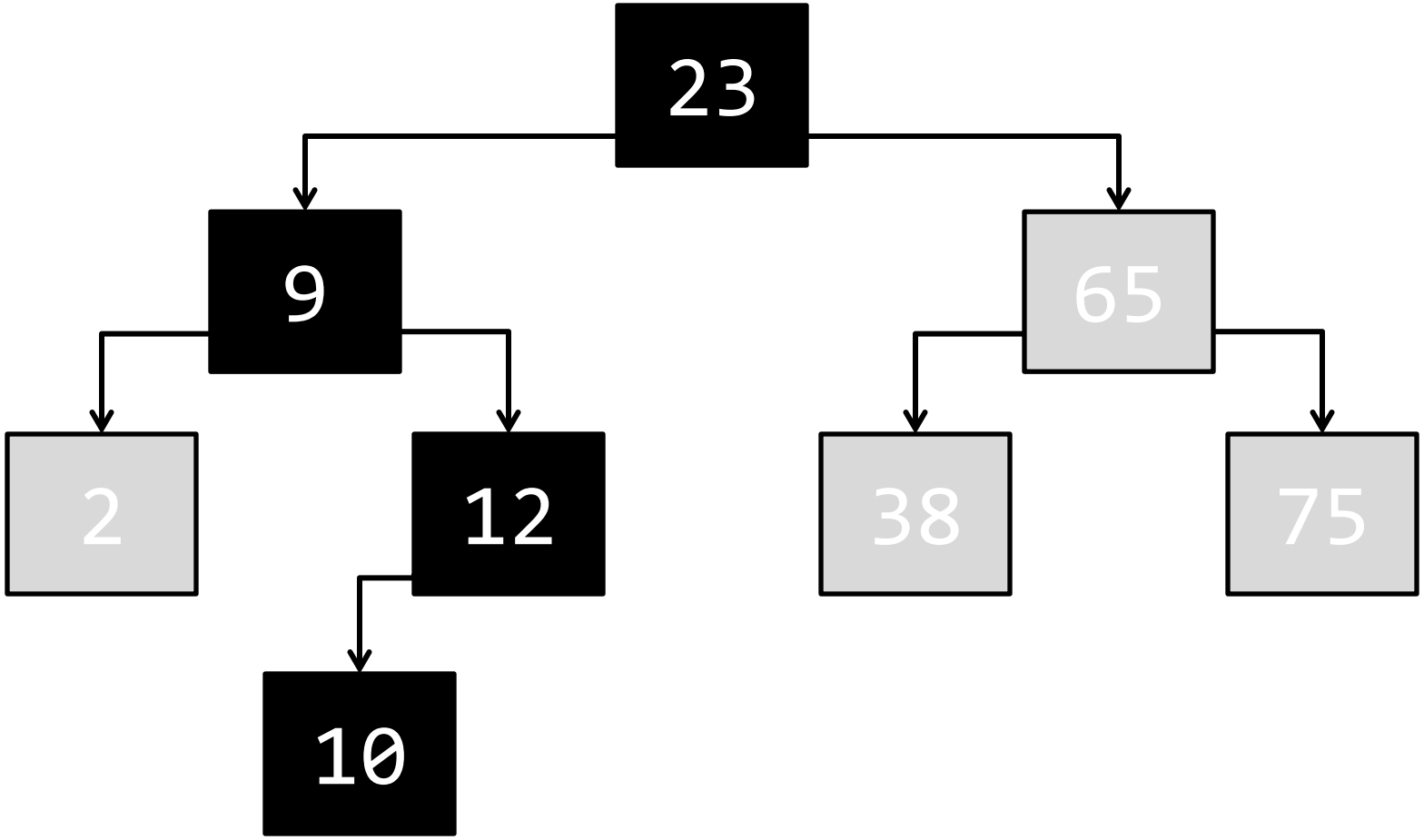
add(10)



add(10)



add(10)



Tastes like efficiency!

Announcement Break

- Apply to section lead! http://cs198.stanford.edu

- Casual dinner for women studying CS is **Thursday, May 23** at **5:30 pm** at the **Gates Patio**. Check Monday's lecture slides to RSVP!

- Keith's office hours **tomorrow** changed to 10 am until noon.

Announcement Break

- Assignment 5 due Friday.
- Second midterm: **Tuesday, May 28** from **7 pm to 10 pm**. Locations on the practice exam!
- Review session: **Sunday, May 26** from **3 pm to 5 pm**. Location TBA.

Binary Search Tree (code)

Unsorted Vector

- `add: O(n)`
- `contains: O(n)`

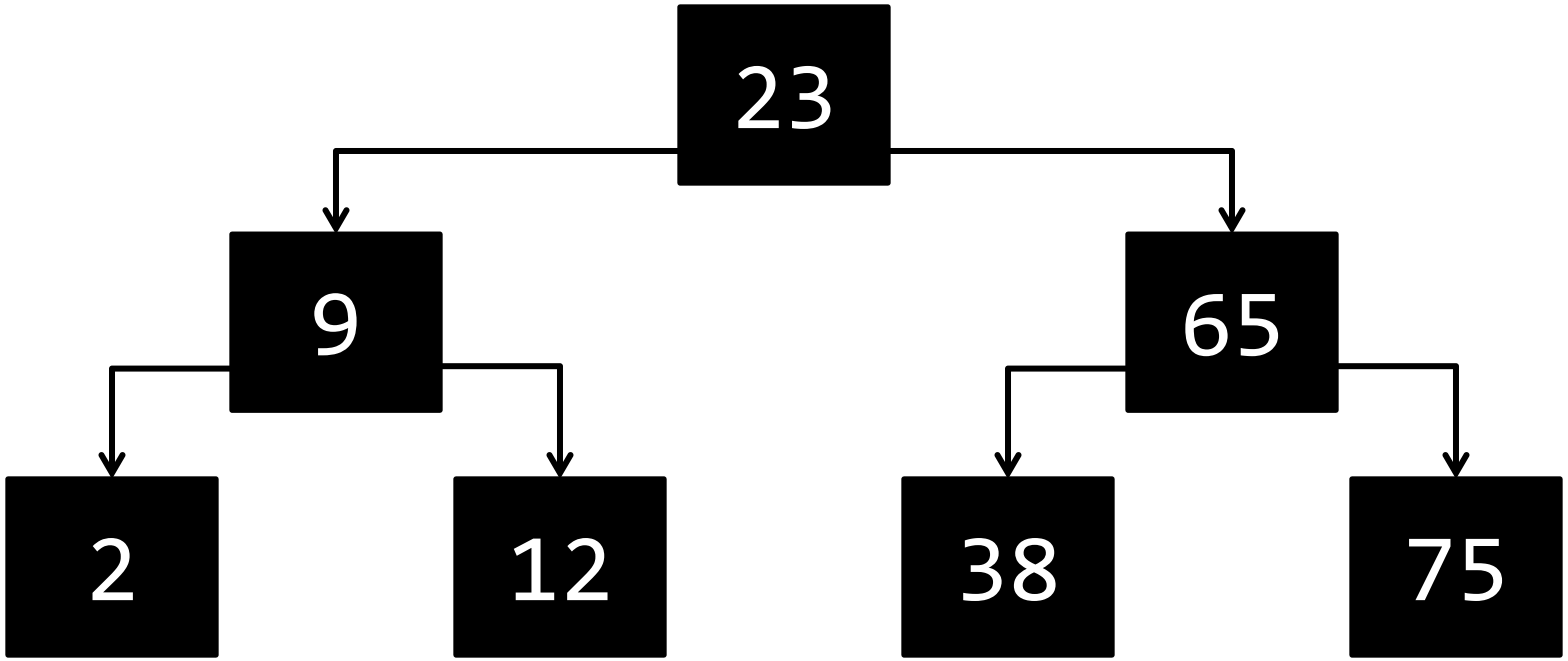
Sorted Vector

- `add: O(n)`
- `contains: O(log n)`

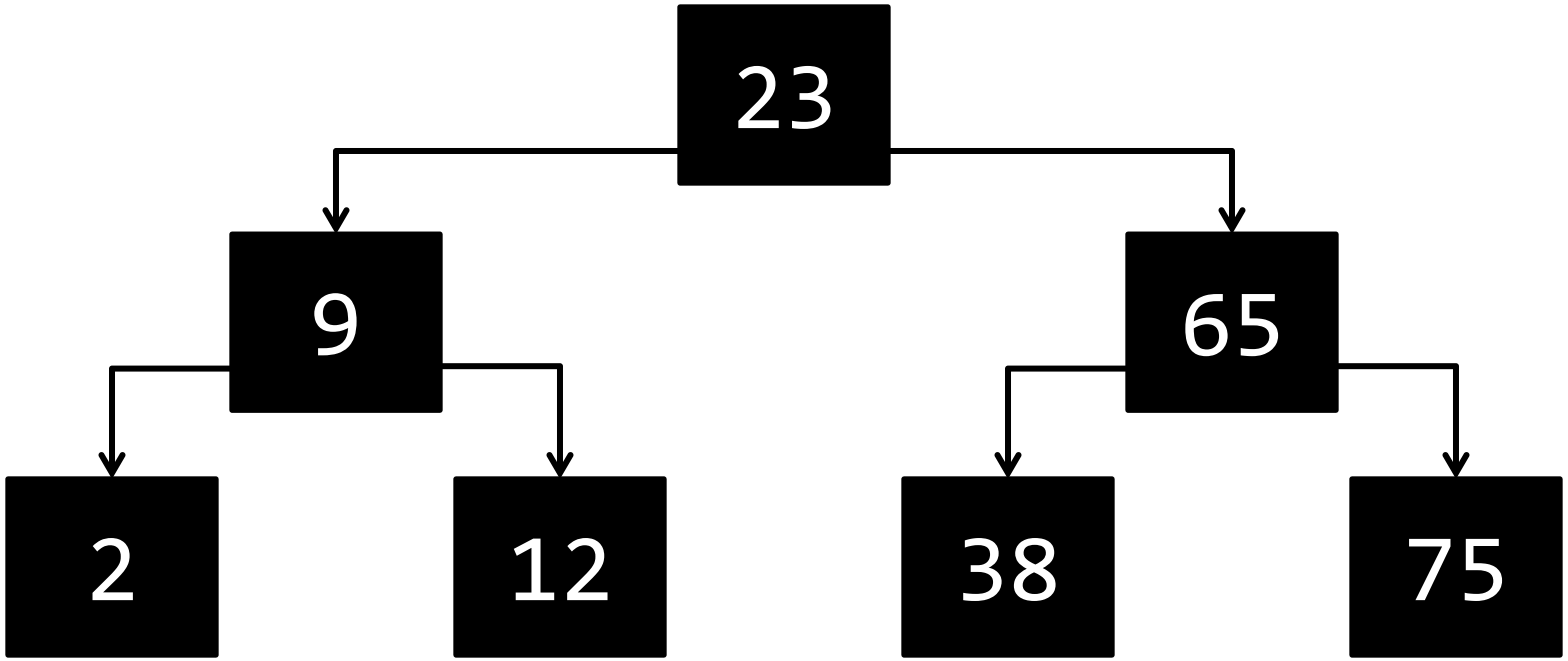
Binary Search Tree

- `add: ??`
- `contains: ??`

BST balance



BST balance



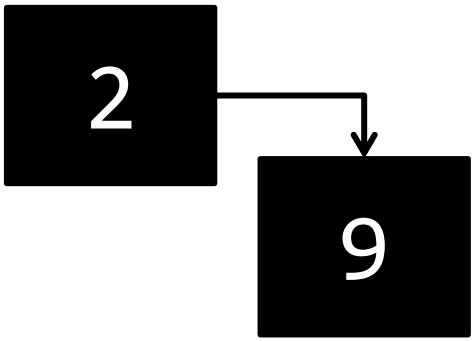
Runtime:
`O(log n)`

BST balance

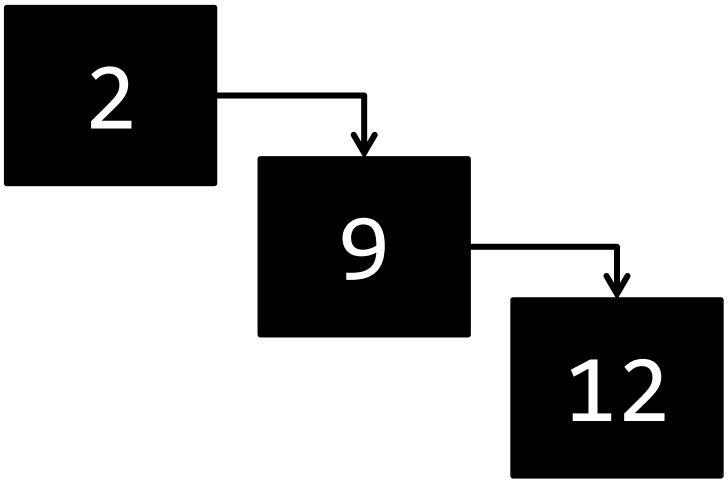
BST balance

2

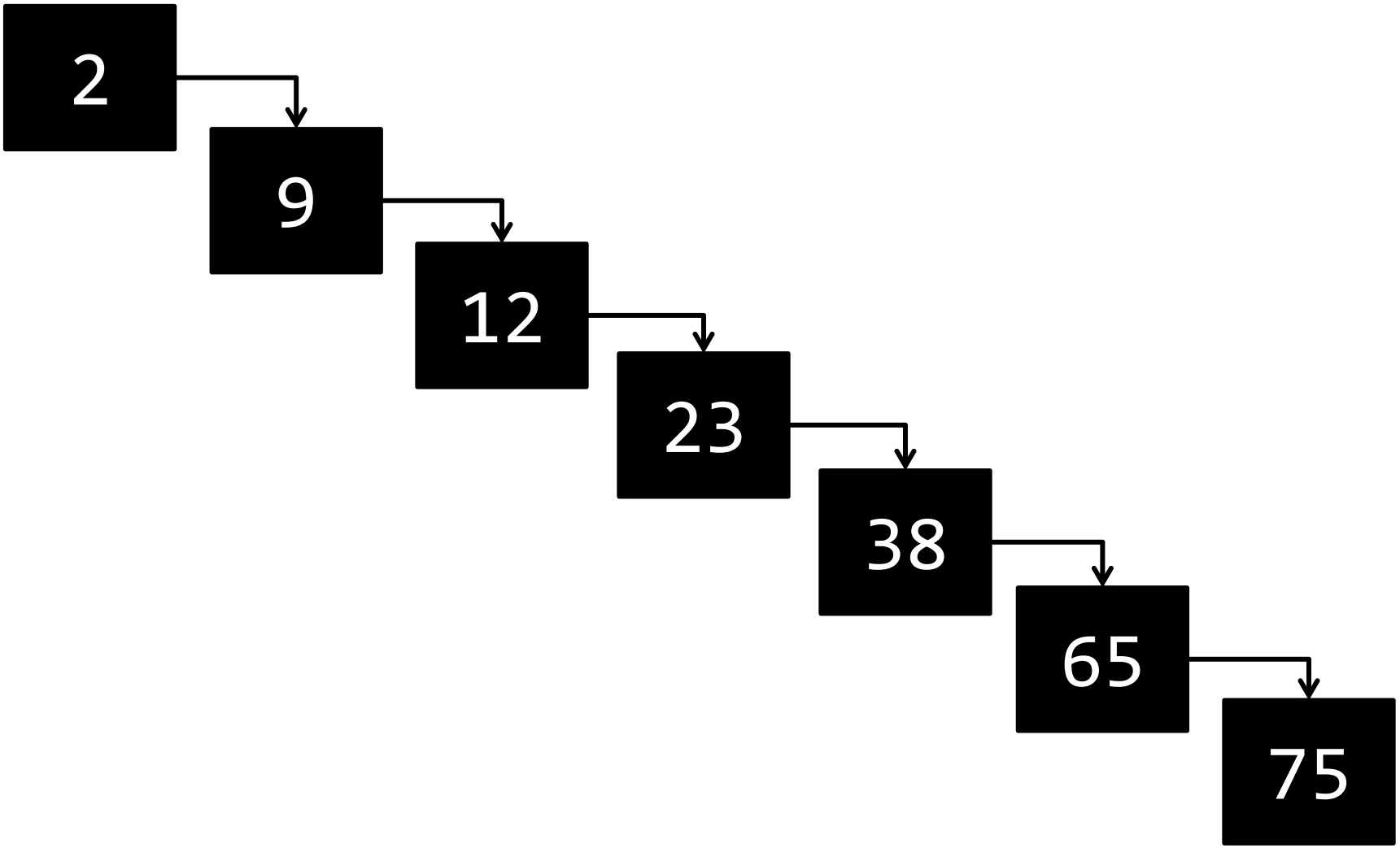
BST balance



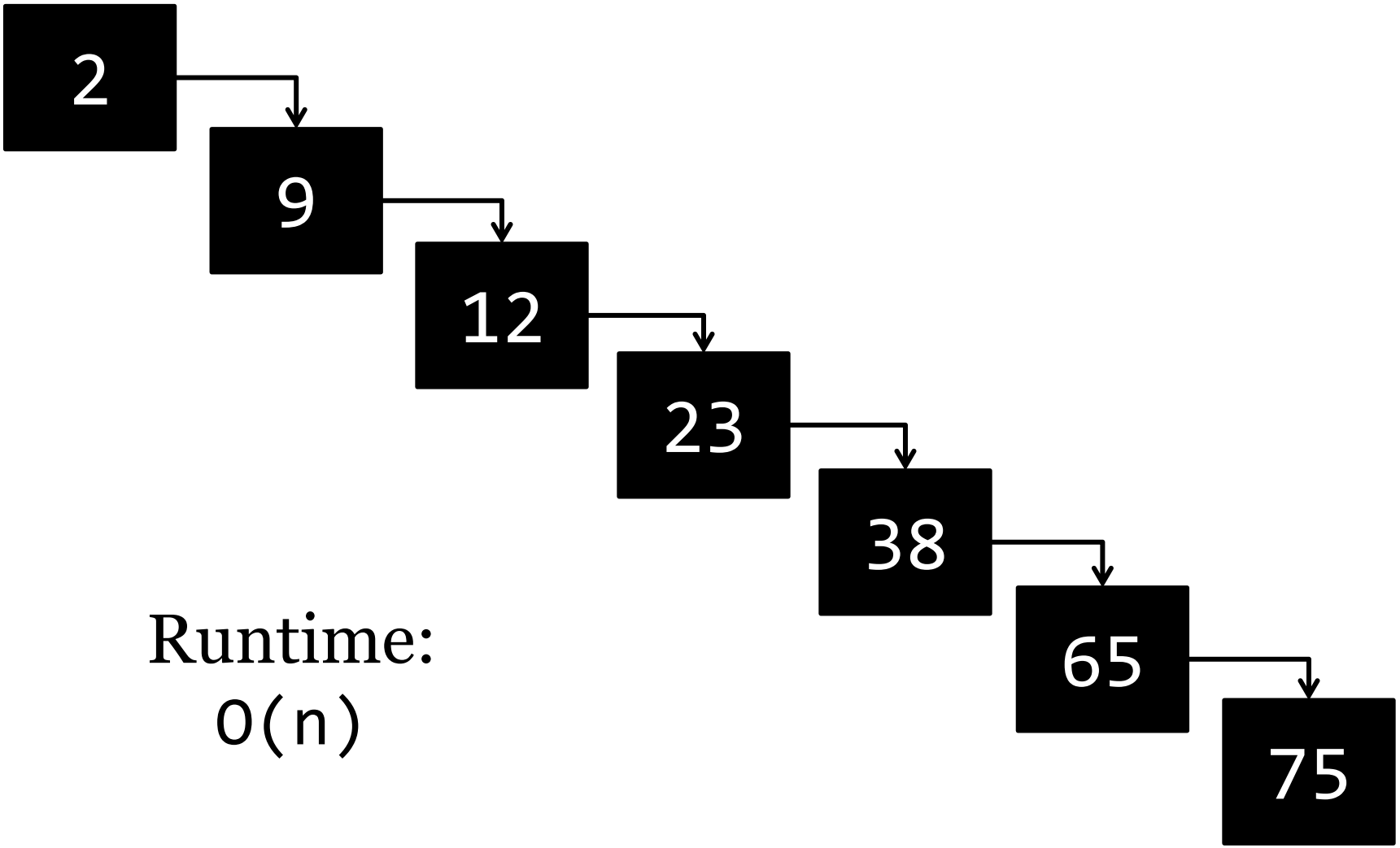
BST balance



BST balance



BST balance



Runtime:
O(n)

BST Depth (code)

Implementations

Dynamic Arrays

Linked Lists

Hash Tables

Binary Search Trees

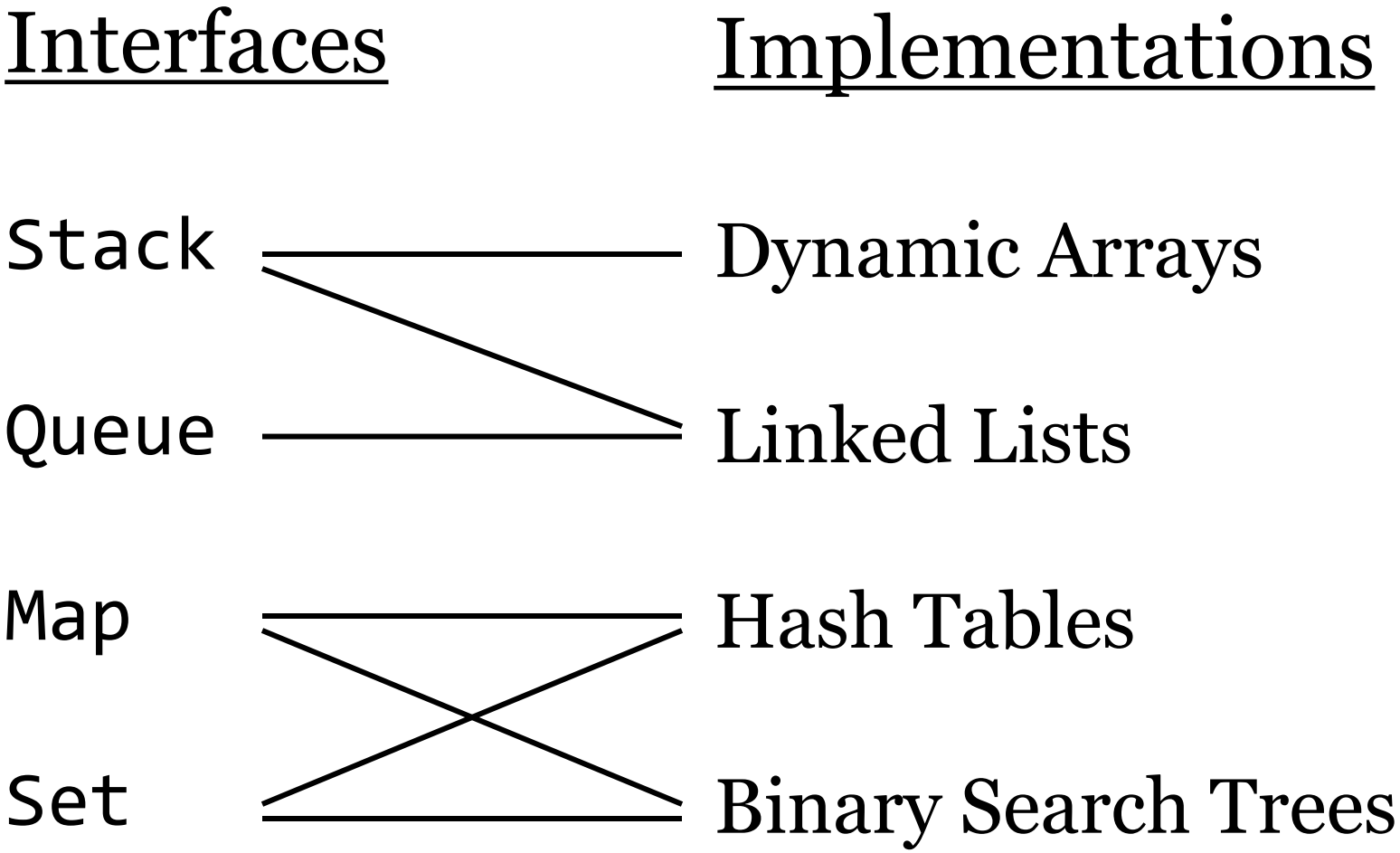
Interfaces	Implementations
`Stack`	Dynamic Arrays
`Queue`	Linked Lists
`Map`	Hash Tables
`Set`	Binary Search Trees

Interfaces	Implementations
`Stack`	Dynamic Arrays
`Queue`	Linked Lists
`Map`	Hash Tables
`Set`	Binary Search Trees

Interfaces	Implementations
`Stack`	Dynamic Arrays
`Queue`	Linked Lists
`Map`	Hash Tables
`Set`	Binary Search Trees

Interfaces	Implementations
`Stack`	Dynamic Arrays
`Queue`	Linked Lists
`Map`	Hash Tables
`Set`	Binary Search Trees

Interfaces	Implementations
`Stack`	Dynamic Arrays
`Queue`	Linked Lists
`Map`	Hash Tables
`Set`	Binary Search Trees


Interfaces
Implementations
Stack
Queue
Map
Set
Dynamic Arrays
Linked Lists
Hash Tables
Binary Search Trees