Benchmark sort algorithms & Trees

Seminar 2.

sorts

sorts

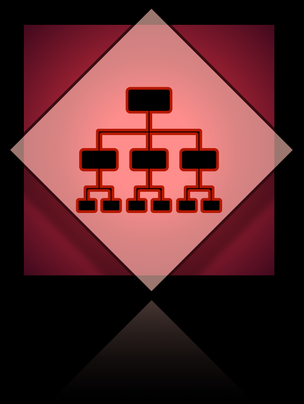
Trees

trees

1. Brand-new sorts

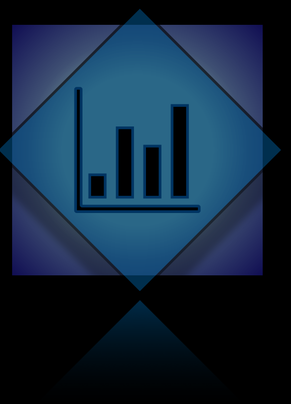
Merge sort

Insertion sort



Merge sort

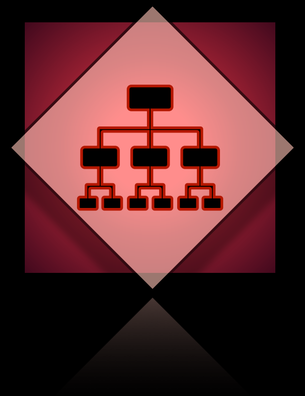
Heap sort



Count sort

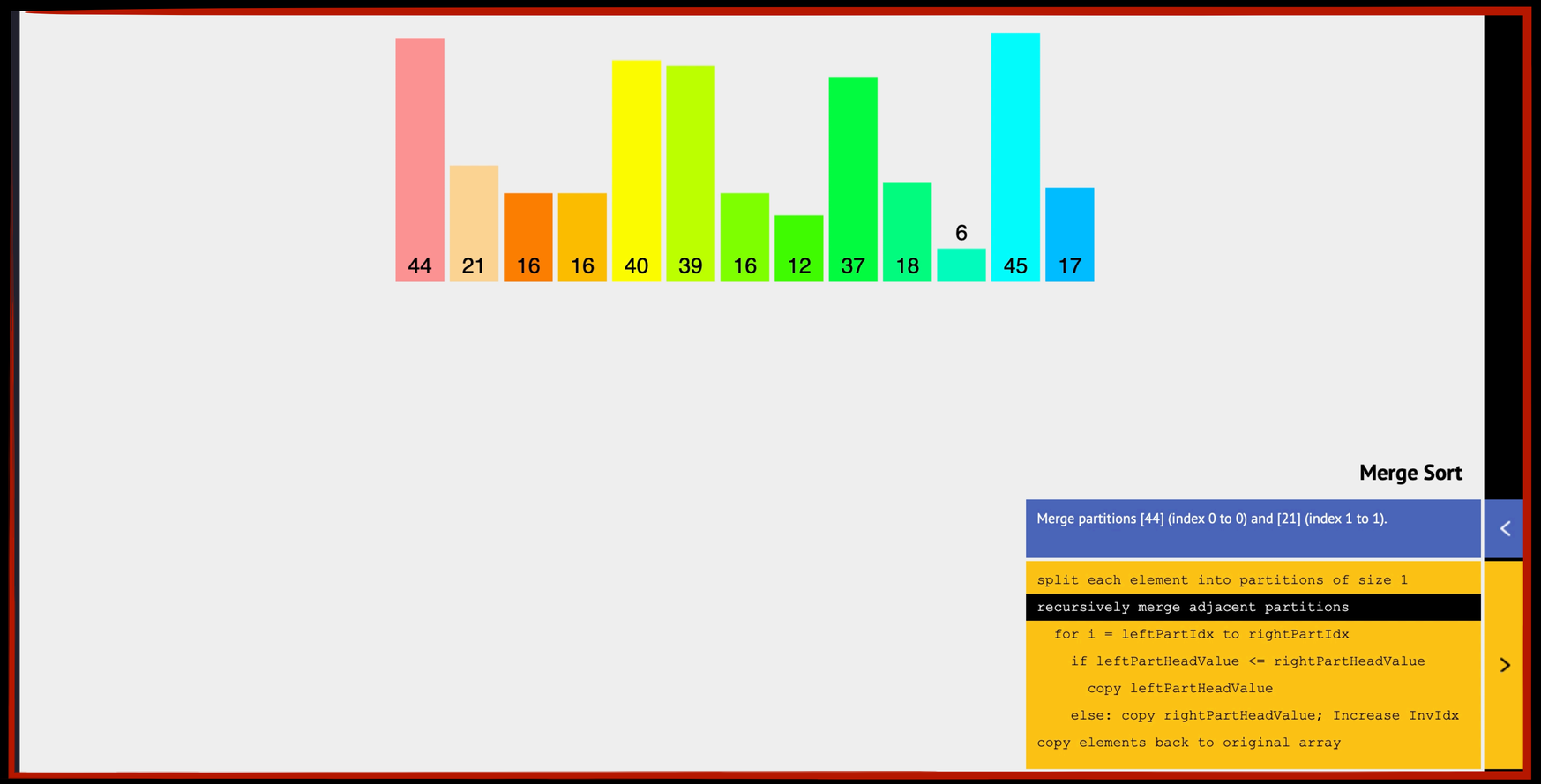
1. Brand-new sorts

Merge sort



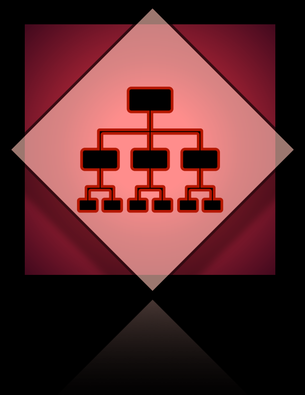
Merge sort

- divide-and-conquer technique
- separate each object into particular bin
- join & sort bins iteratively



1. Brand-new sorts

Merge sort



Merge sort

- divide-and-conquer technique
- separate each object into particular bin
- join & sort bins iteratively

Code the following:

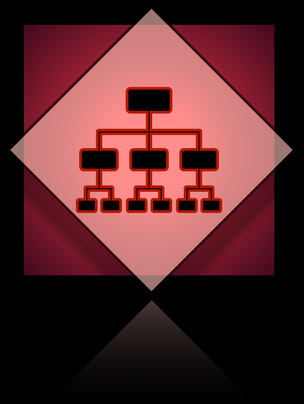
1. **Divide recursively the array**
2. **Start merging as you've reached the bottom**

```
void mergeSortAlgorithm(int A[], int l, int r)
{
    if(l >= r)
        return
    int mid = l + (r - l)/2
    mergeSortAlgorithm(A, l, mid)
    mergeSortAlgorithm(A, mid + 1, r)
    merge(A, l, mid, r)
}
```

1. Brand-new sorts

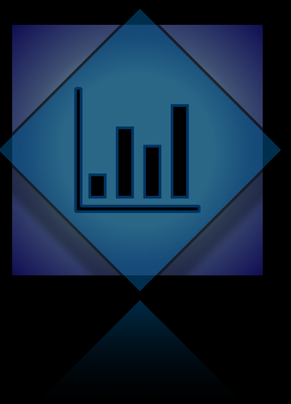
Merge sort

Insertion sort



Merge sort

Heap sort



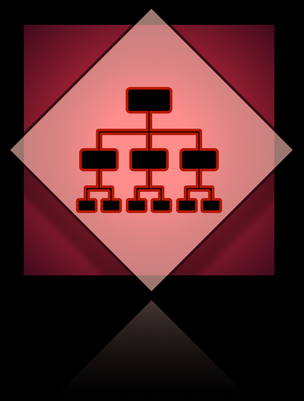
Count sort

1. Brand-new sorts

Preface

Insertion sort

WORKS SLOWLY



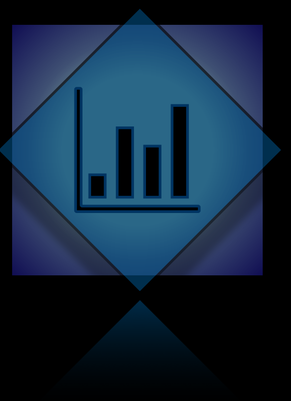
Merge sort

HIGH MEMORY

Heap sort

LOSES BY SPEED

BAD Locality Of Reference



Count sort

SMALL POSITIVE INTs ONLY

1. Brand-new sorts

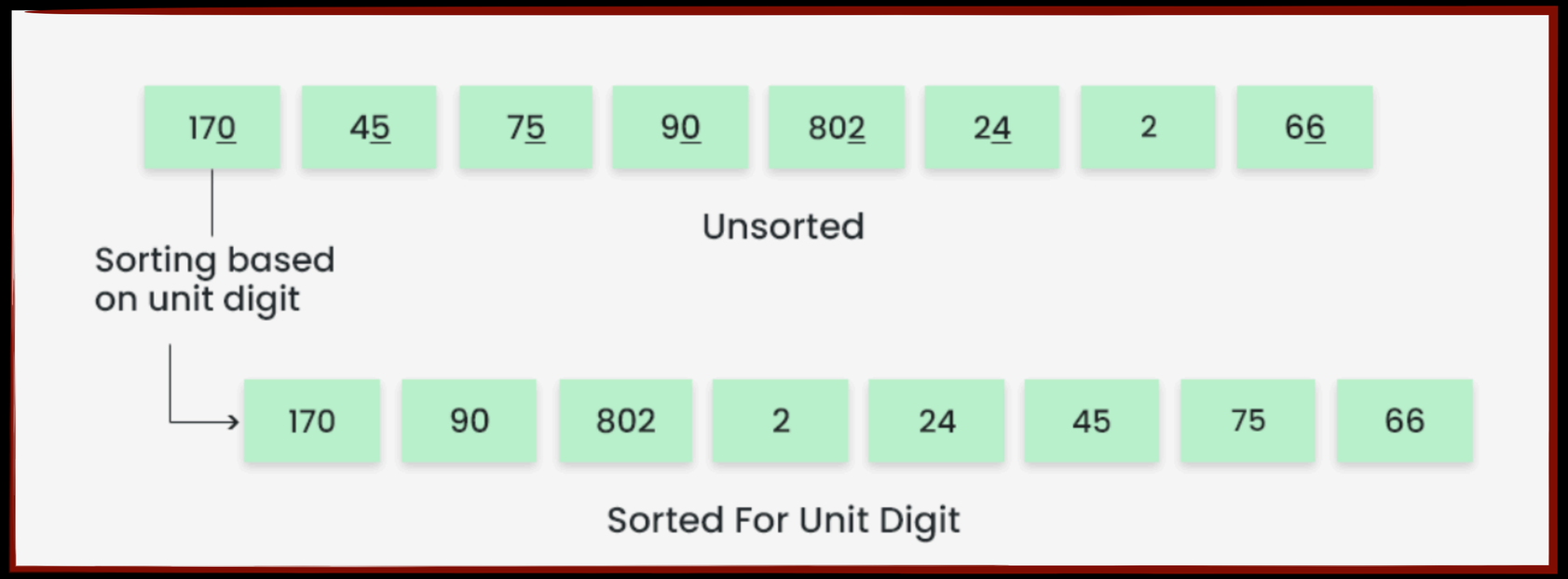
Radix: what's the point?

Radix sort

1. Brand-new sorts

Radix: what's the point?

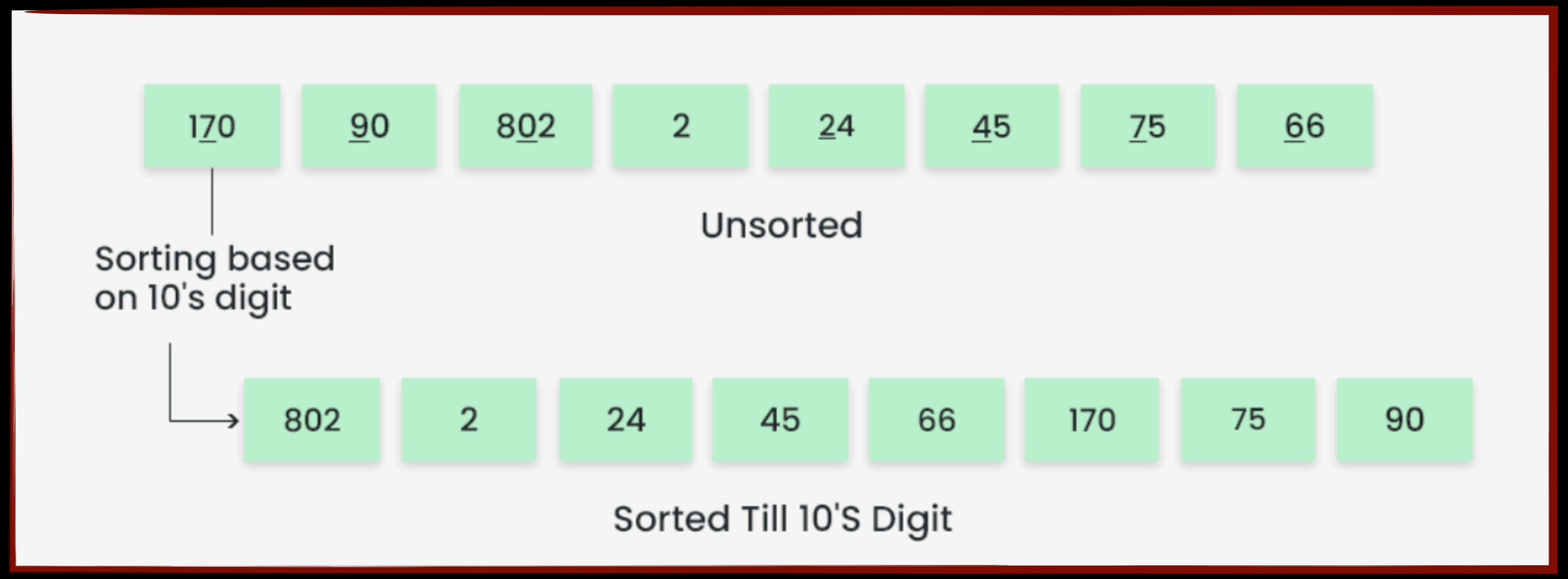
Radix sort



1. Brand-new sorts

Radix: what's the point?

Radix sort



1. Brand-new sorts

Radix: what's the point?

Radix sort

- Define max order & length
- Iterate through each order
- Sort by counting sort

1. Brand-new sorts

Radix: what's the point?

Radix sort

- Define max order & length
- Iterate through each order
- Sort by counting sort

3256 2898 7795 7589 8781 6421 1569 8243 6034 2130 5115

Radix Sort

```
create 10 buckets (queues) for each digit (0 to 9)
for each digit placing
  for each element in list
    move element into respective bucket
  for each bucket, starting from smallest digit
    while bucket is non-empty
      restore element to list
```

1. Brand-new sorts

Radix: inference

Radix sort

Time worst	
Time best	
Memory	

Code the following:

1. **Least Significant Digit**

extra:

2. **Most Significant Digit**
3. **+ recursion**
4. **+ insertion**

1. Brand-new sorts

Radix: inference

	Radix sort
Time worst	$O((n+b)\cdot k)$
Time best	$\Omega((n+b)\cdot k)$
Memory	$O(n+b)$

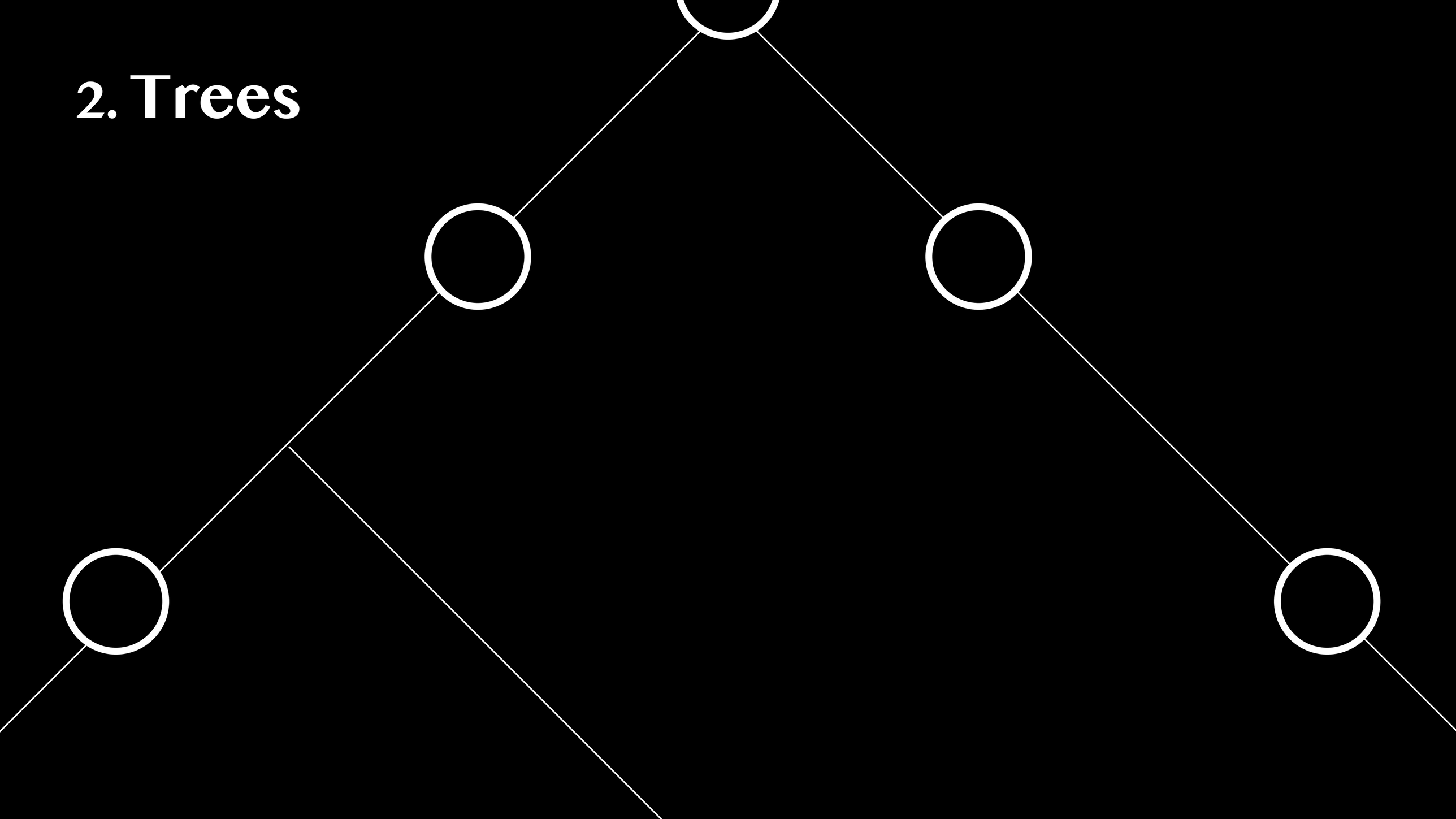
Code the following:

1. **Least Significant Digit**

extra:

2. **Most Significant Digit**
3. **+ recursion**
4. **+ insertion**

2. Trees



2. Trees

What is a tree?

2. Trees

What is a tree?

- Hierarchical data structure
- Non-linear access to data

2. Trees

What is a tree?

- Hierarchical data structure
- Non-linear access to data

- Nodes
  - - entity, containing key & pointers to parents & children
  - Leaf / external - no-children's node
  - Degree - № of branches to node

2. Trees

What is a tree?

- Hierarchical data structure
- Non-linear access to data

- Nodes
- Tree
  - Height - № of edges from root to the deepest leaf
  - Forest - collection of disjoint trees
  - Subtree - a node & all its descendants

2. Trees

and binary tree?

- max. of two child nodes for each parent
- leaves lean towards left of the tree

2. Trees

and binary tree?

- max. of two child nodes for each parent
- leaves lean towards left of the tree

- Full binary tree
  - each node has either 0 or 2 children
- Perfect binary tree
  - each interior node has either 2 children & all leaves are on the same level

2. Trees

and binary tree?

- max. of two child nodes for each parent
- leaves lean towards left of the tree

- Interdependency of № of nodes & height

$n \leq \quad \ldots \quad \leftrightarrow h \geq \quad \ldots$

2. Trees

and binary tree?

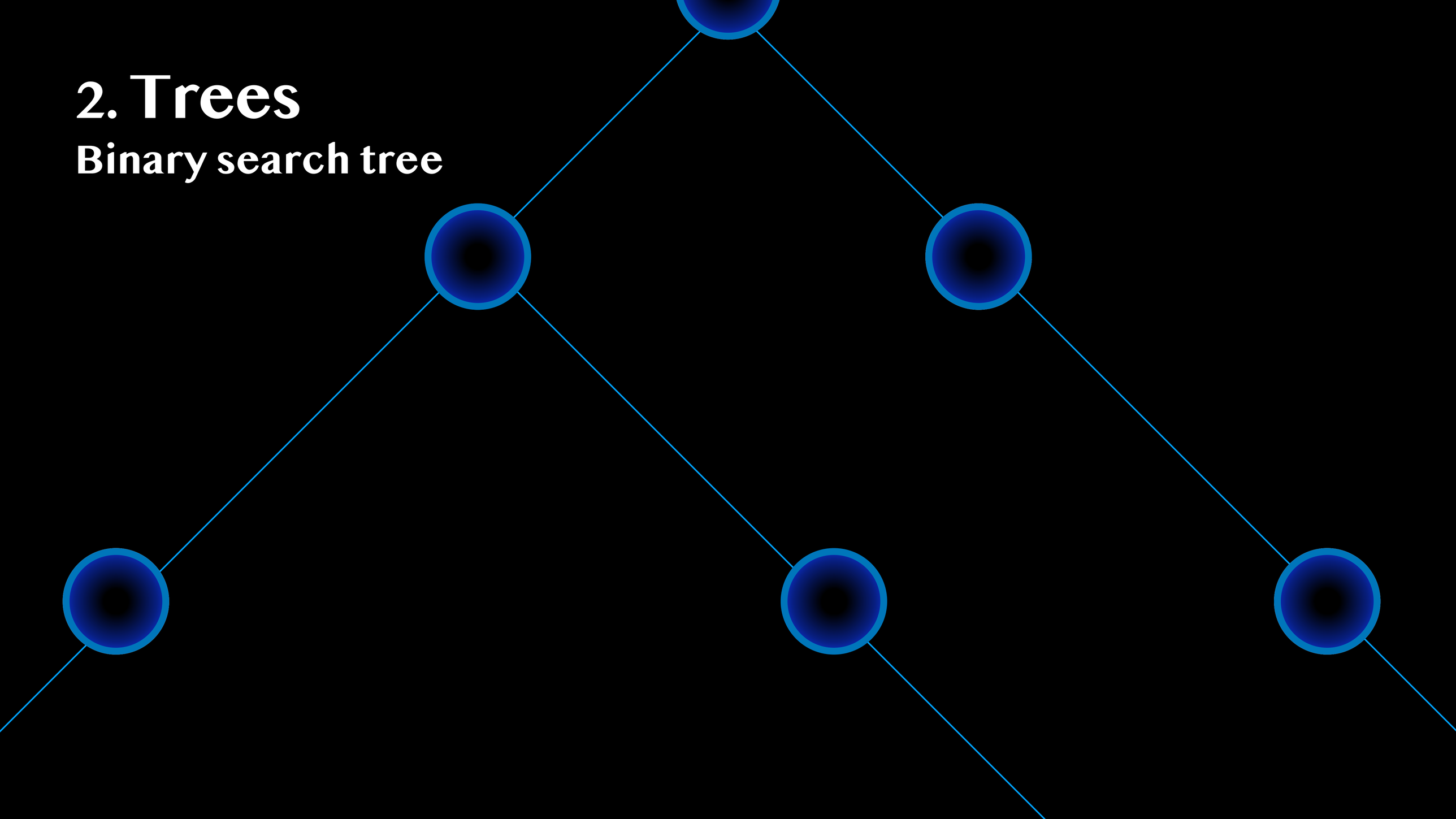
- max. of two child nodes for each parent
- leaves lean towards left of the tree

- Interdependency of № of nodes & height

$$n \leq 2^{h+1} - 1 \leftrightarrow h \geq \log_2(n+1) - 1$$

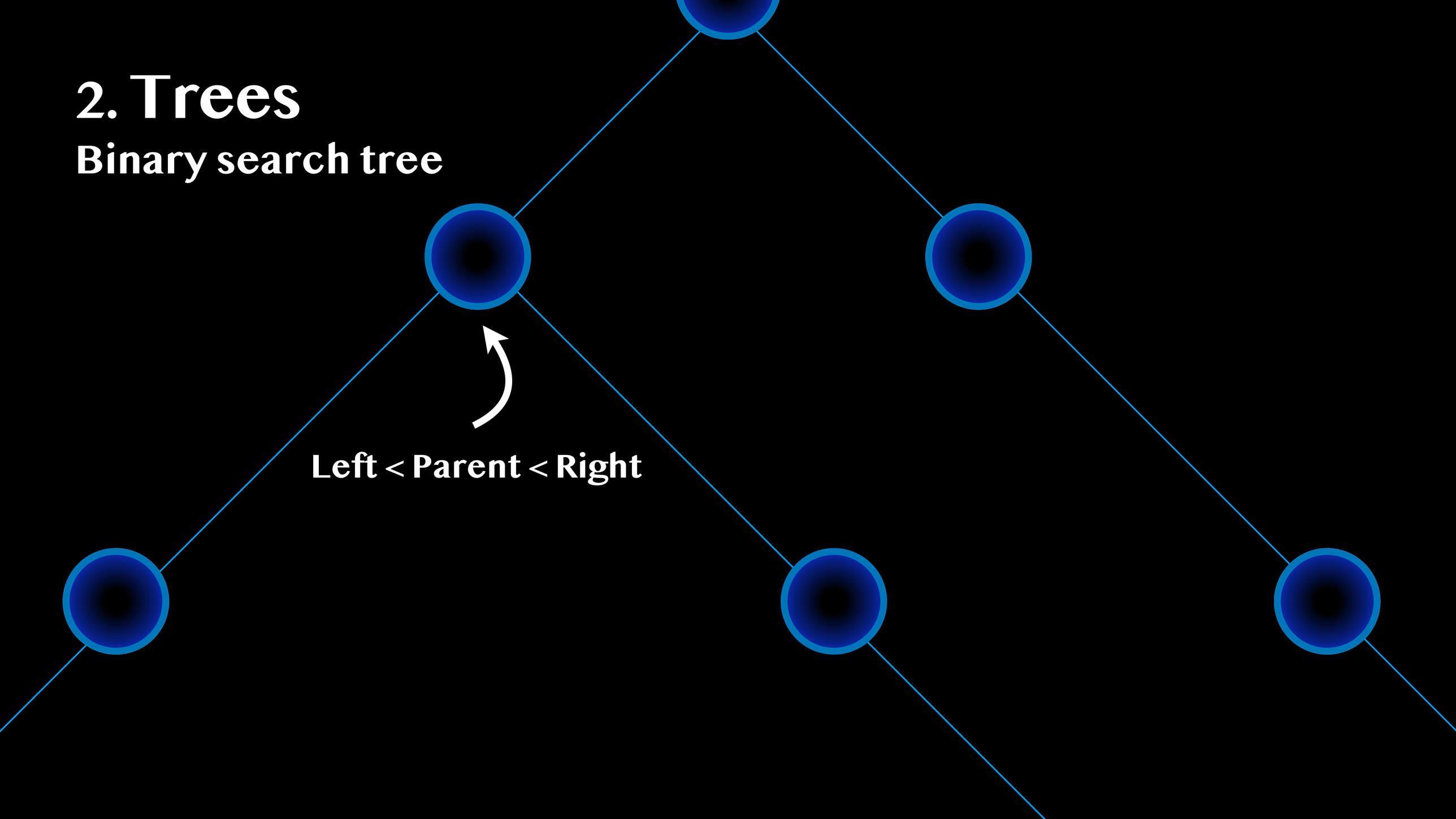
2. Trees

Binary search tree



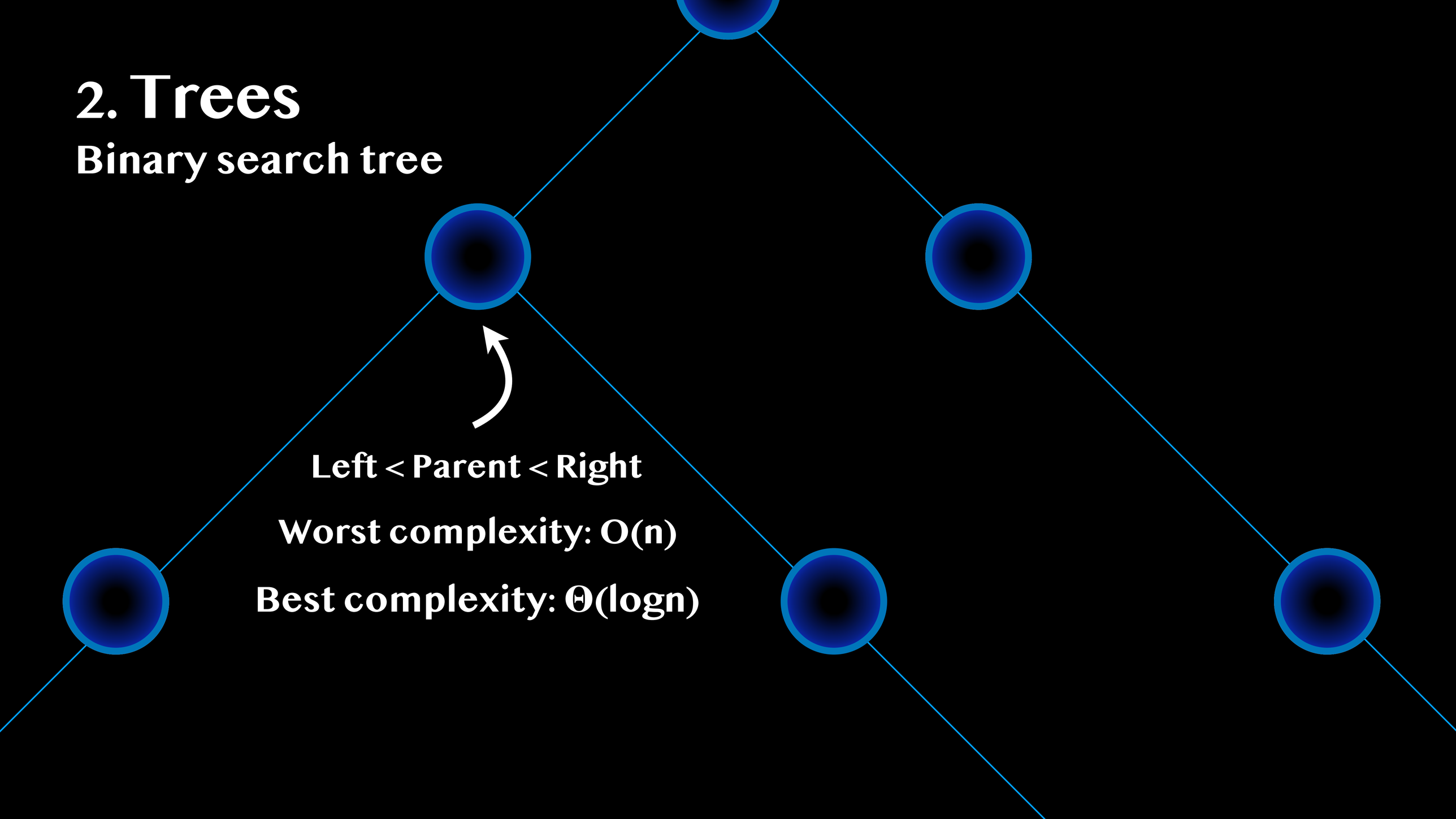
2. Trees

Binary search tree



2. Trees

Binary search tree



Left < Parent < Right

Worst complexity: O(n)

Best complexity: Θ(logn)

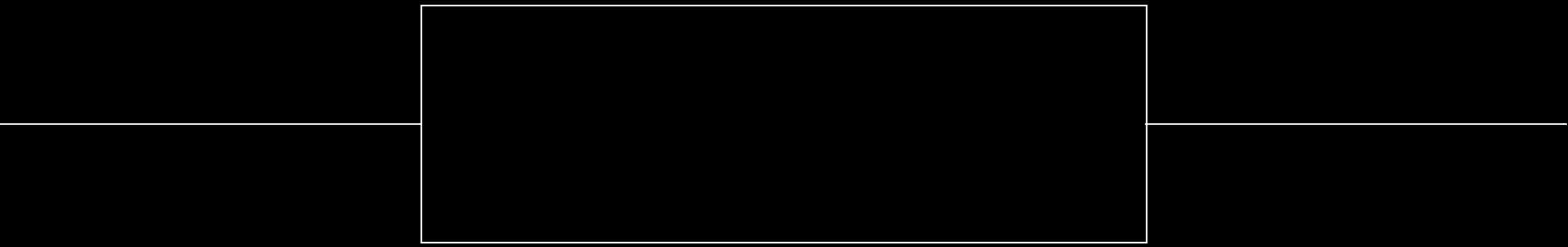
2. Trees
Binary search tree

fun with coding

- Create Node & Tree classes

- Print method: 1)pre- 2)in- 3) post-order
- Insert method: new element as a leaf
- Search method:
    - Recursive
    - Iterative
- Delete method: 3 cases (leaf, one child, both)

3. Balanced trees



3. Balanced trees

Heights of right & left subtrees of each node differ by at most one

3. Balanced trees

AVL is a self balancing tree!

3. Balanced trees

AVL is a self balancing tree!

To be continued...