AVL Tree. Treap. Trie

Seminar 4.

Plan for the day

	Seminar 1	Seminar 2
All together	1. Repeat rotation methods 2. Implement AVL insertion	1. Treap class & insertion 2. Revise all past
Pro level	1. Implement AVL insertion 2. Treap class & insertion	1. Trie class & insertion

Plan for the day

tips for Pro

1. All 4 rotation cases for AVL insertion:
    - How balance value impacts the choice of rotation?
    - Find the second parameter, affecting the rotation choice

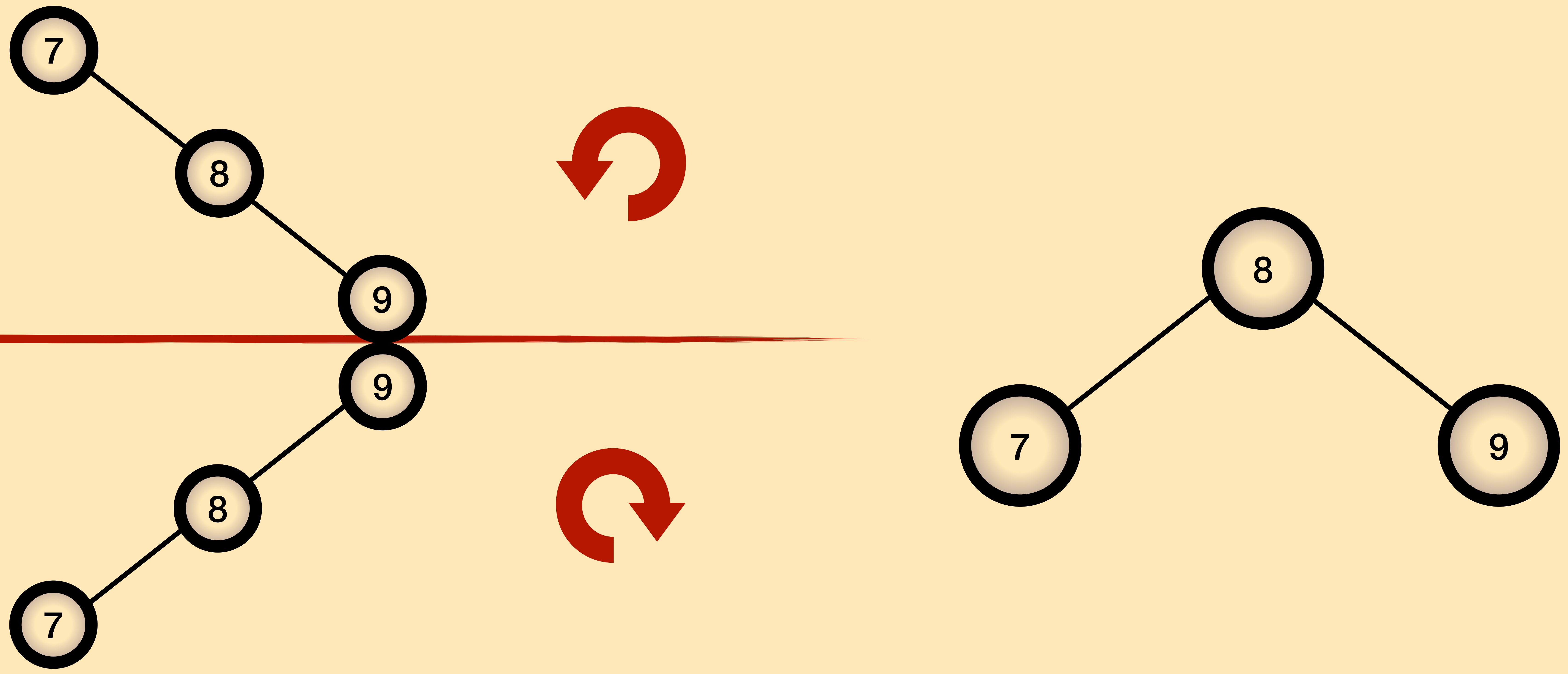
2. Treap:
    - Just use rotation for priority instability

3. Trie
    - Save empty list of letters for each node

AVL: rotation cases

Rotation to left (LL)

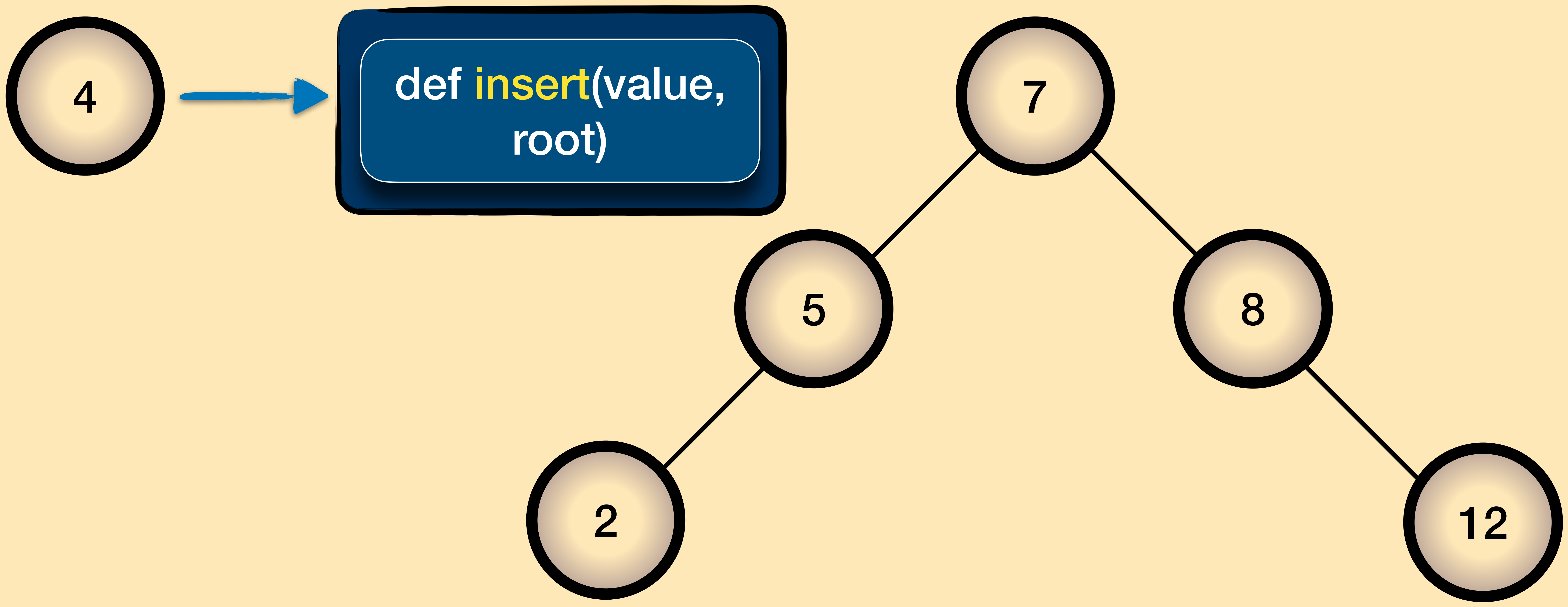
Rotation to right (RR)



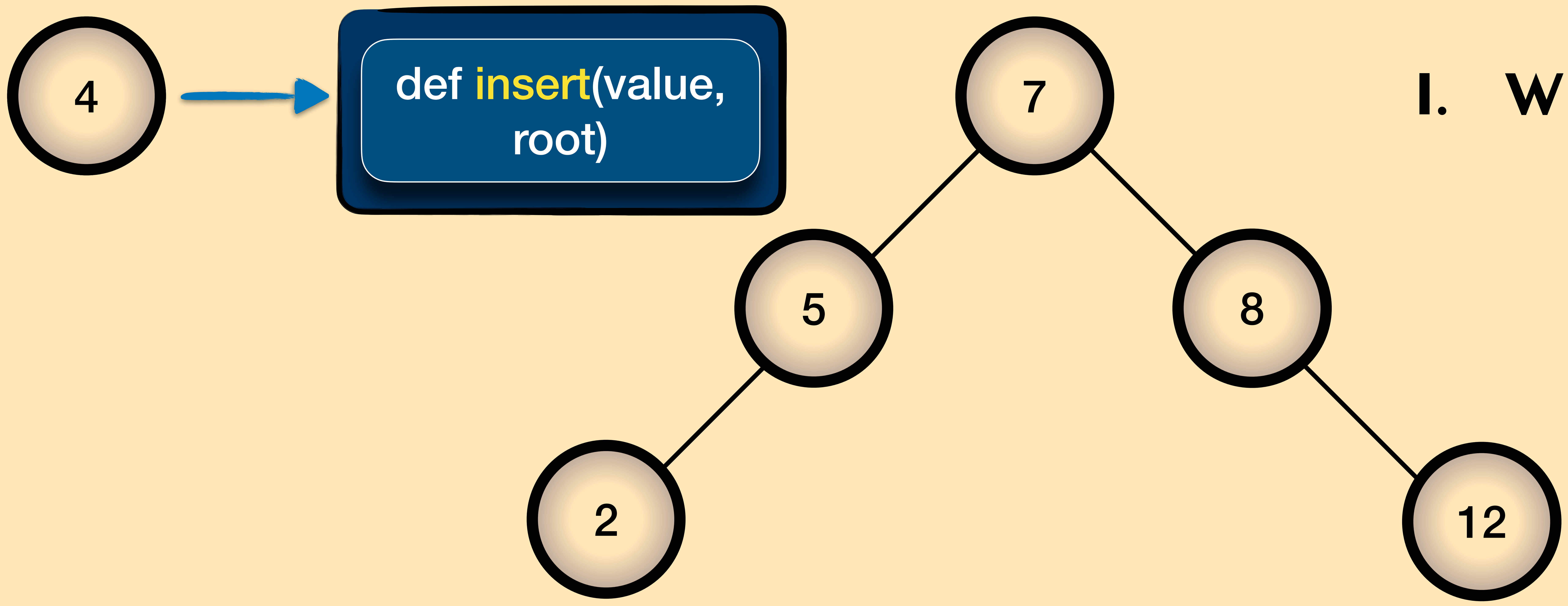
Left right rotation (LR)

Right left rotation (RL)

AVL: insertion



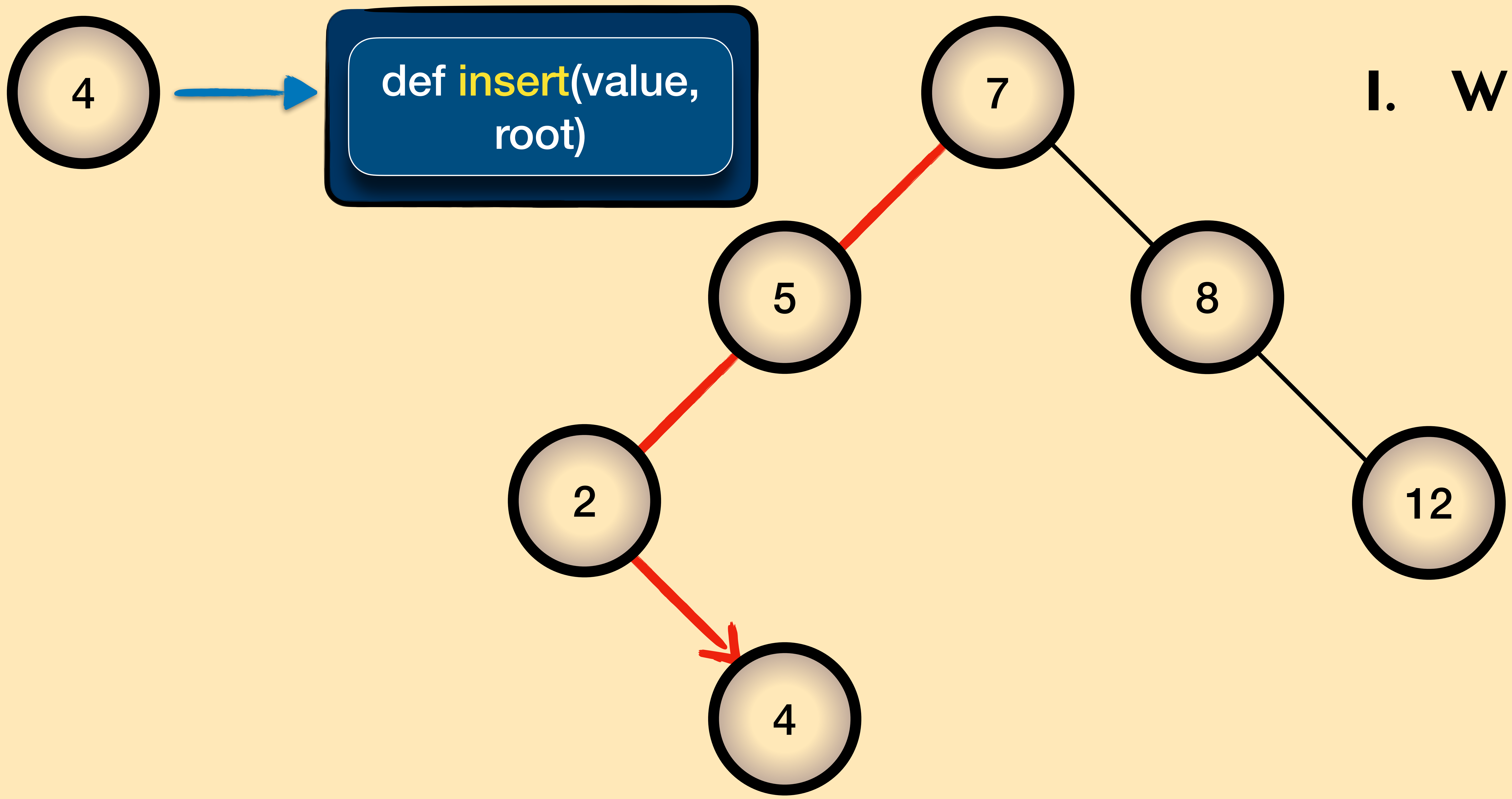
AVL: insertion



```
def insert(value,
root)
```

1. **Where to insert?**

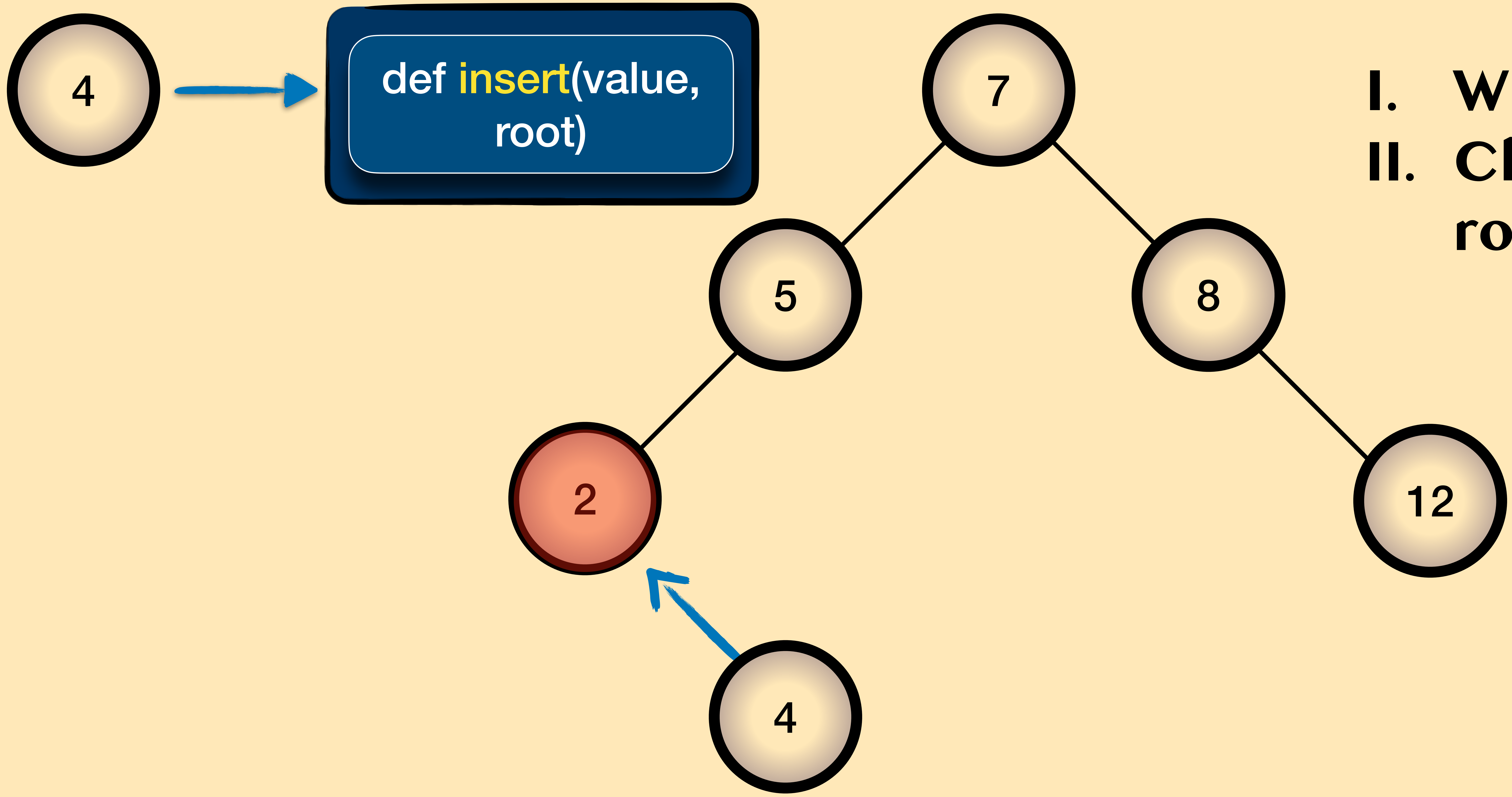
AVL: insertion



```
def insert(value,
root)
```

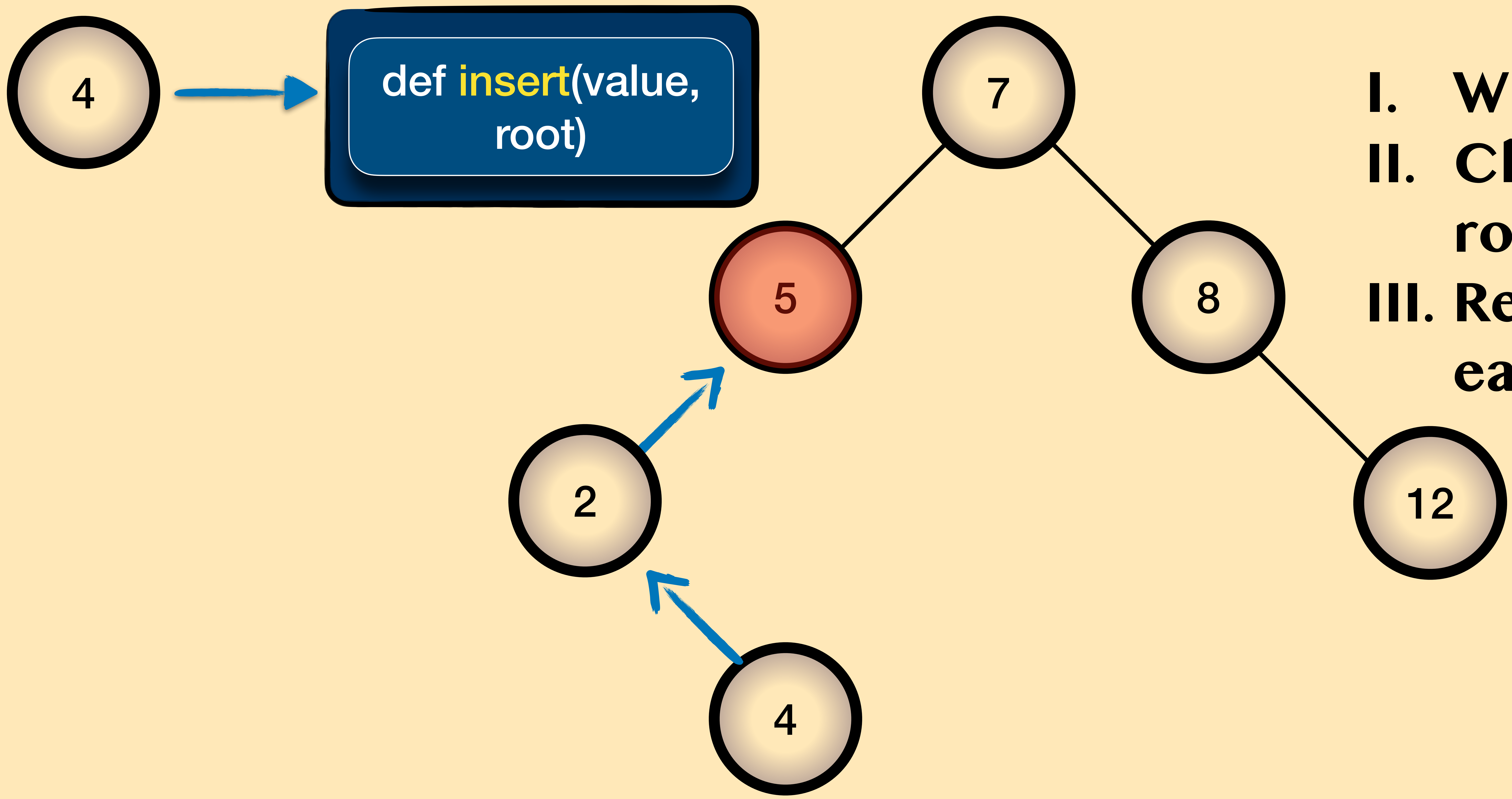
1. Where to insert?

AVL: insertion



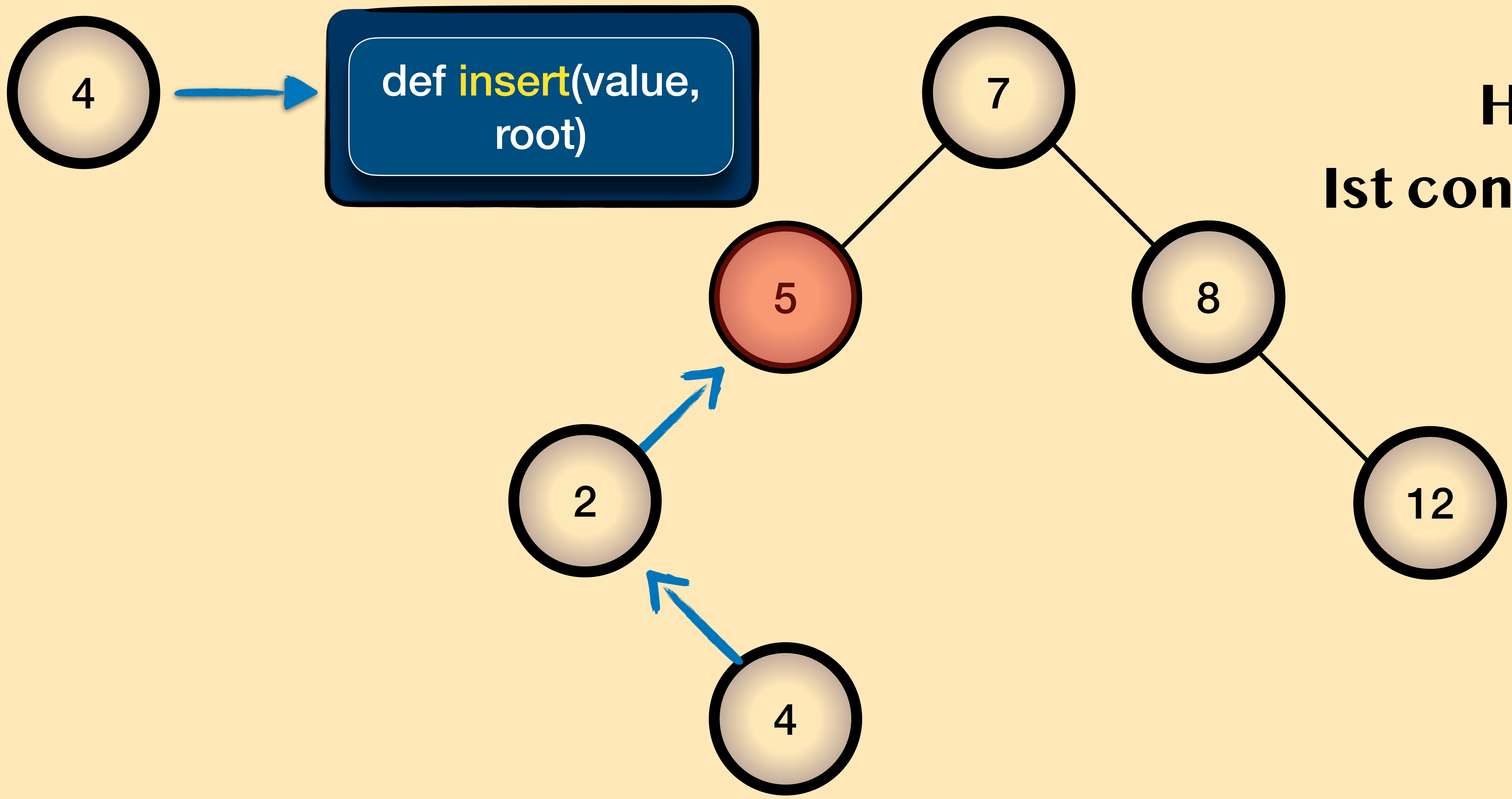
I. Where to insert?
II. Check balance of root & rotate in case

AVL: insertion



I. Where to insert?
II. Check balance of root & rotate in case
III. Recursively check each root

AVL: insertion



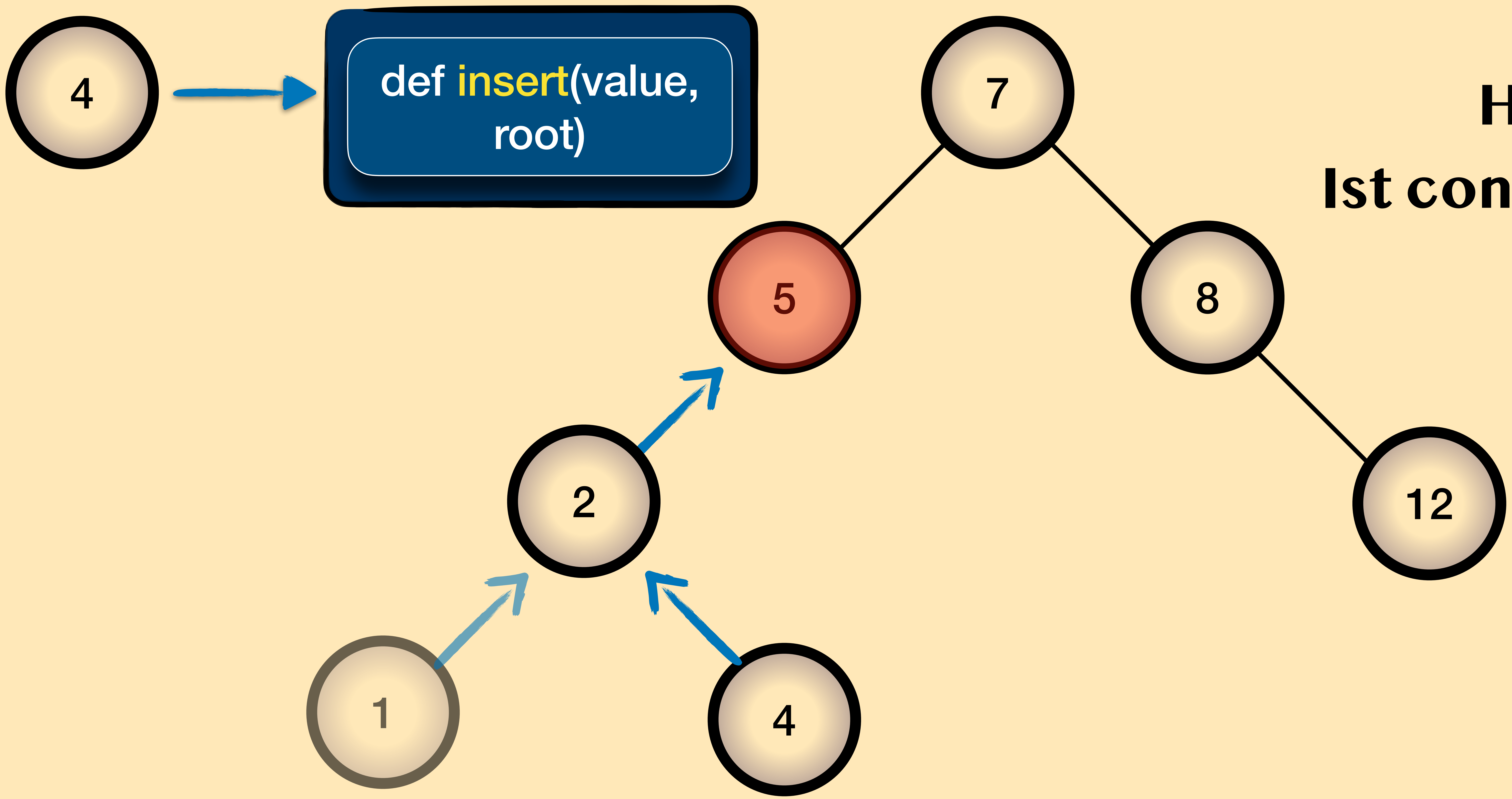
```
def insert(value,
root)
```

How to rotate?

Ist condition:

Balance > 1

AVL: insertion



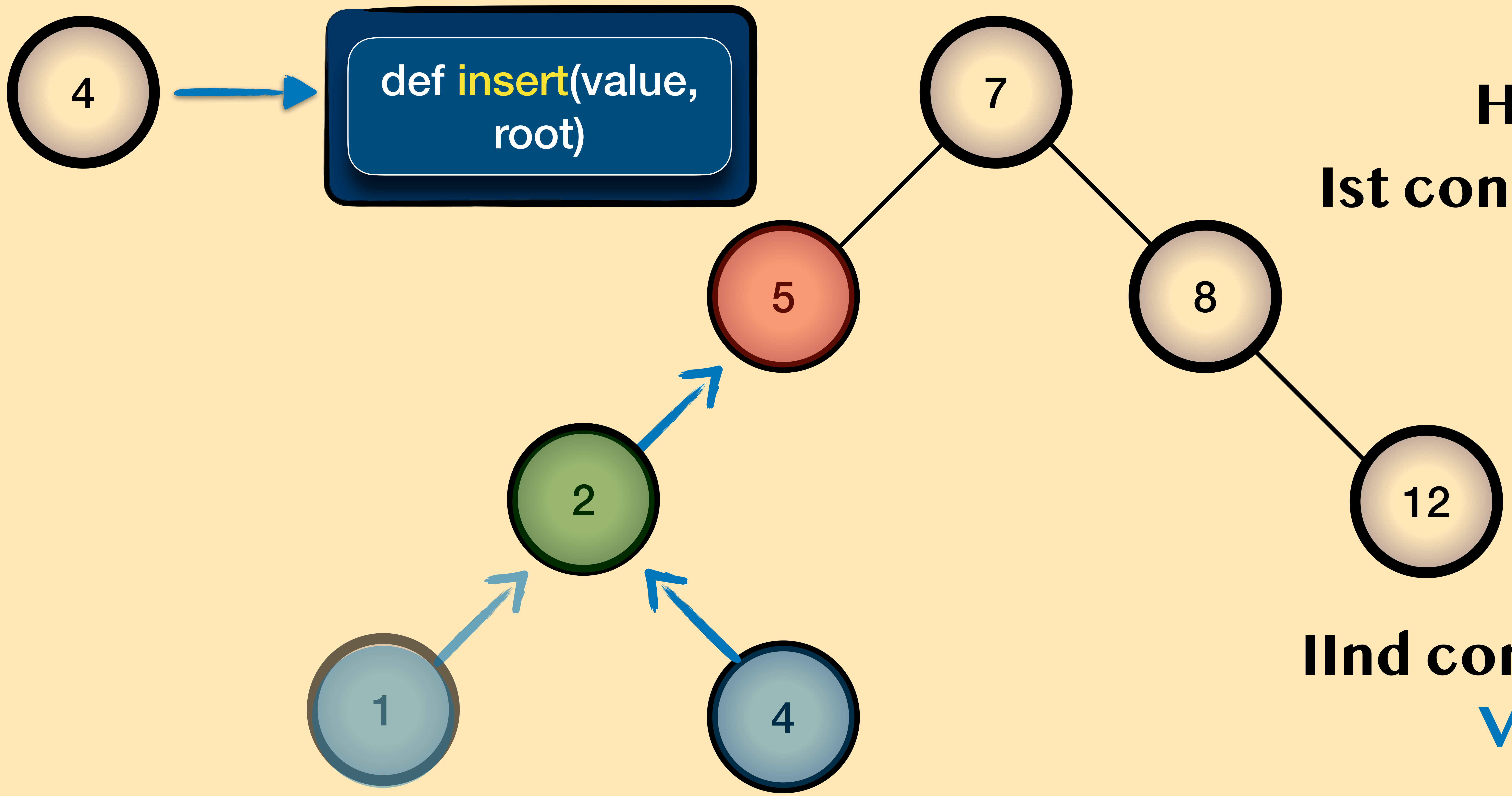
```
def insert(value,
root)
```

How to rotate?

Ist condition:

Balance > 1

AVL: insertion



How to rotate?

Ist condition:

Balance > 1

IInd condition:

Value ? subroot.val

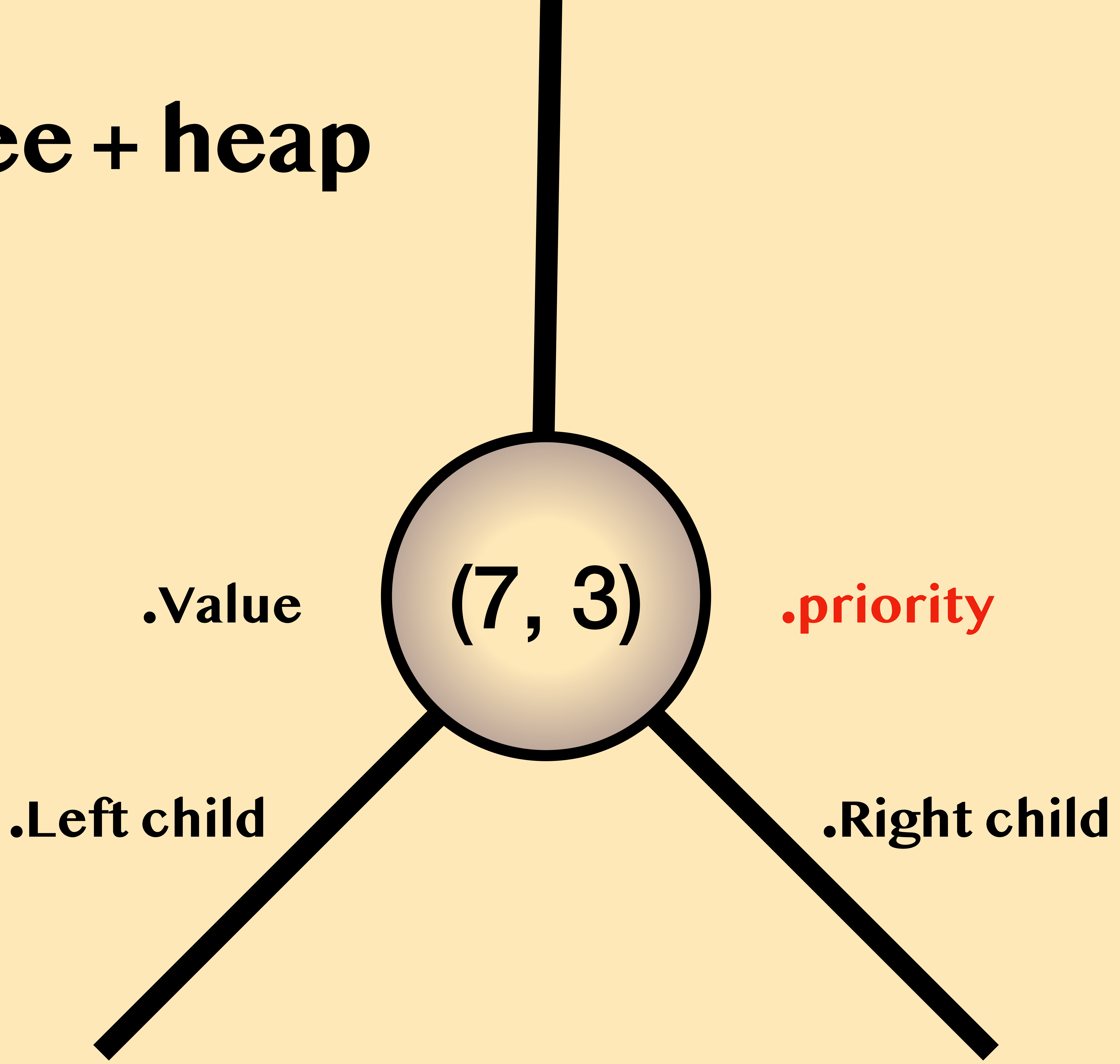
AVL: insertion

	Balance > 1	Balance < -1
left subsubtree		
right subsubtree		

AVL: insertion

	Balance > 1	Balance < -1
left subsubtree	1. Rotate right	1. Rotate right 2. Rotate left
right subsubtree	1. Rotate left 2. Rotate right	1. Rotate left

Treap: tree + heap

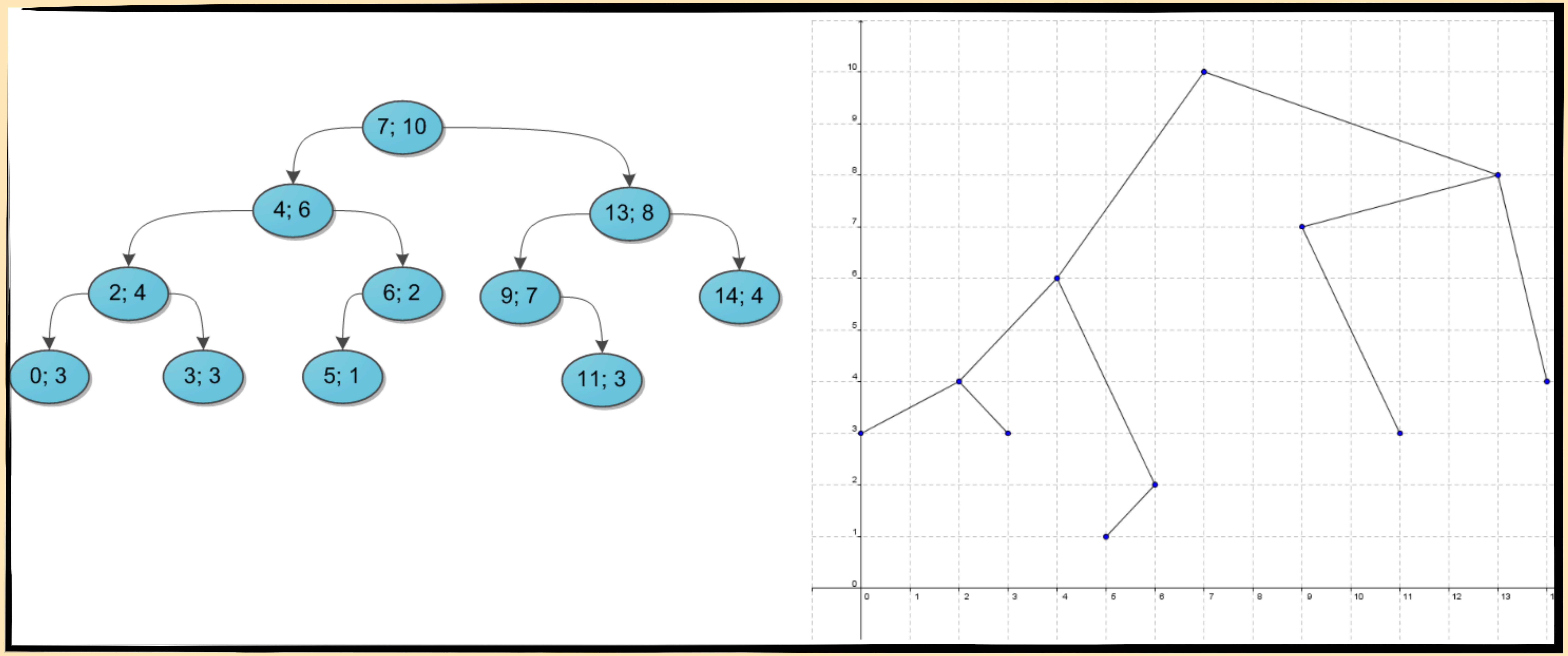


Treap: tree + heap

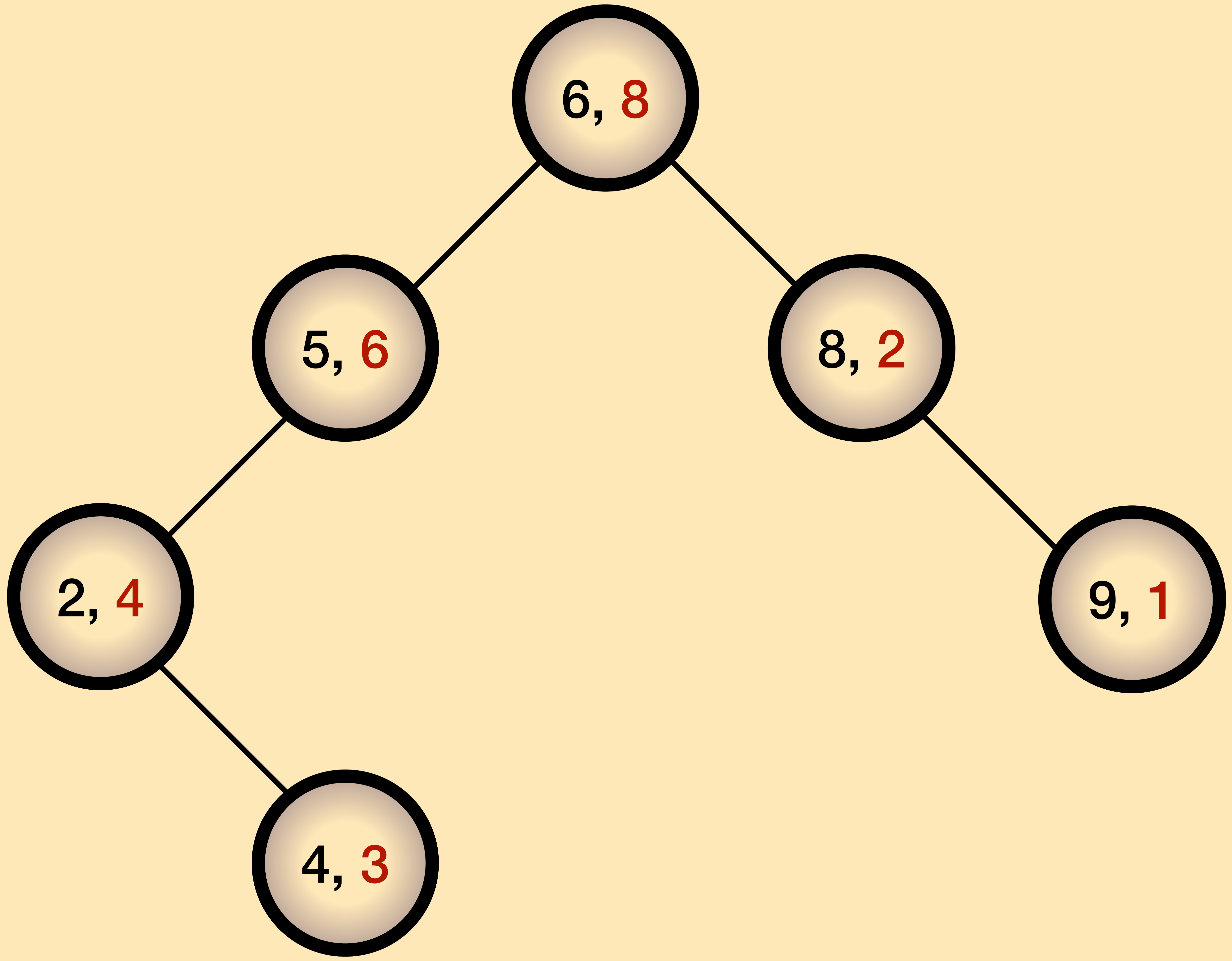
Left_child.value <
Root.value <
Right_child.value

Root.priority >
Child.prioriy

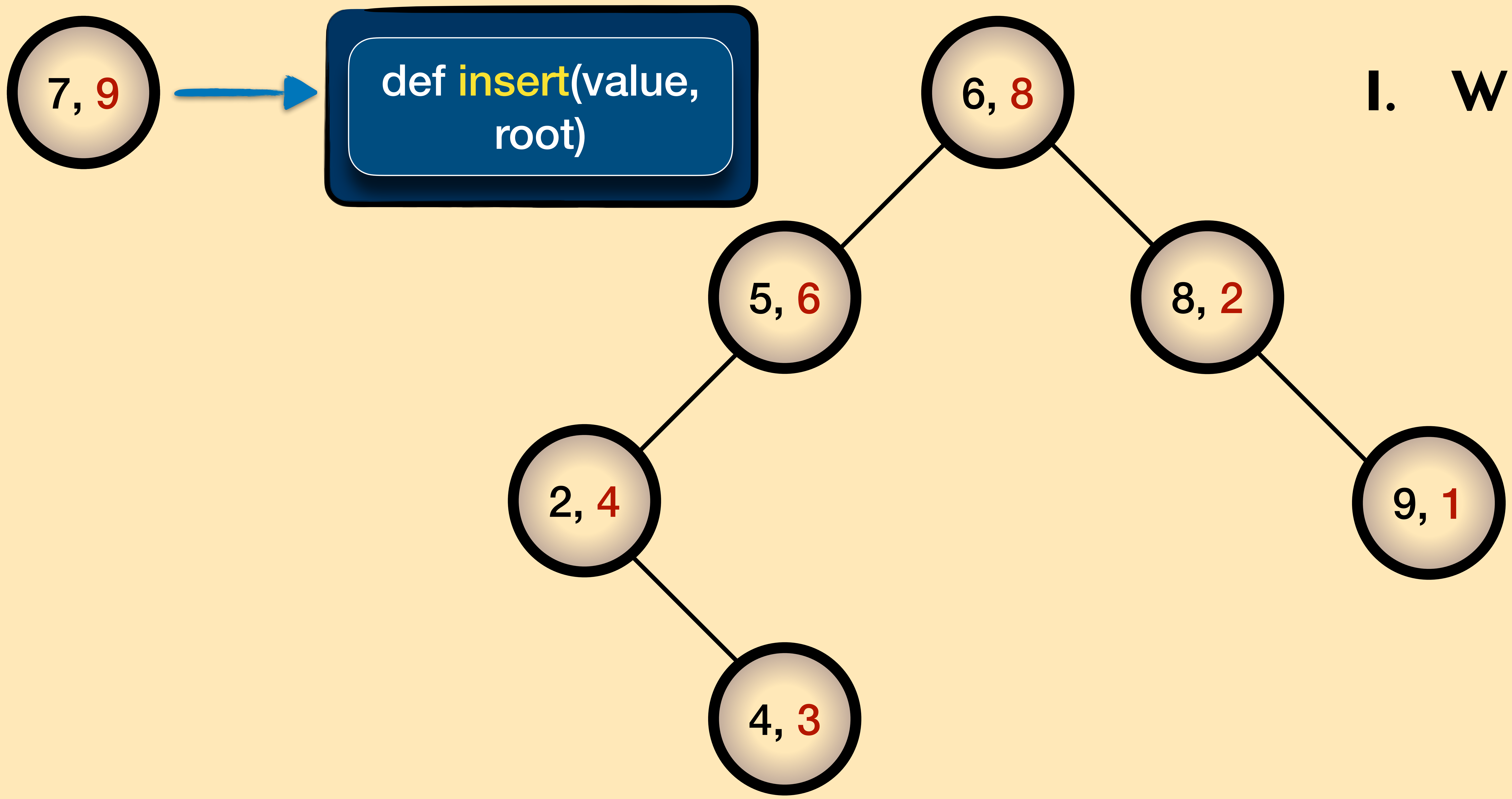
Treap: tree + heap



Treap: tree + heap

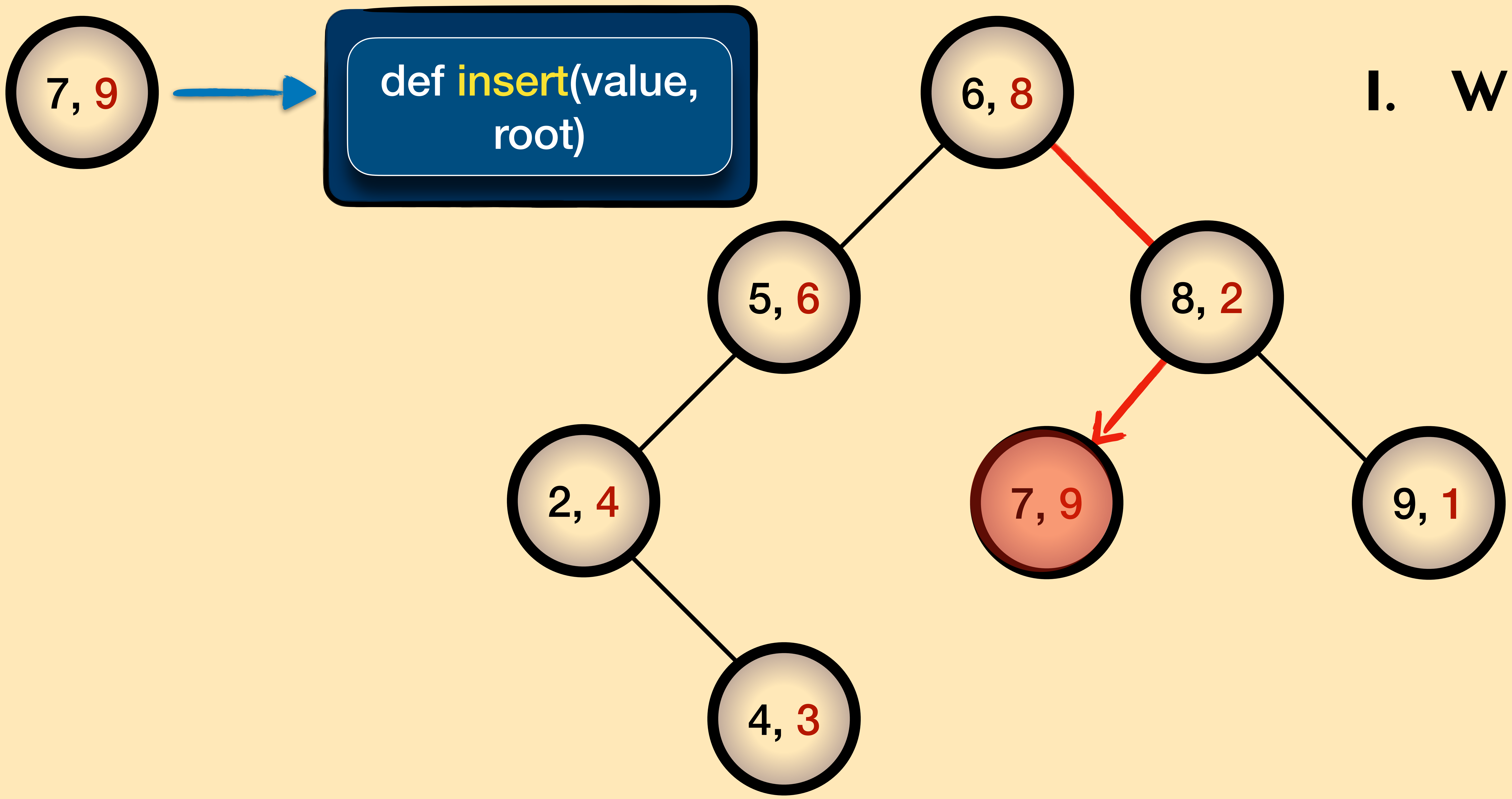


Treap: tree + heap



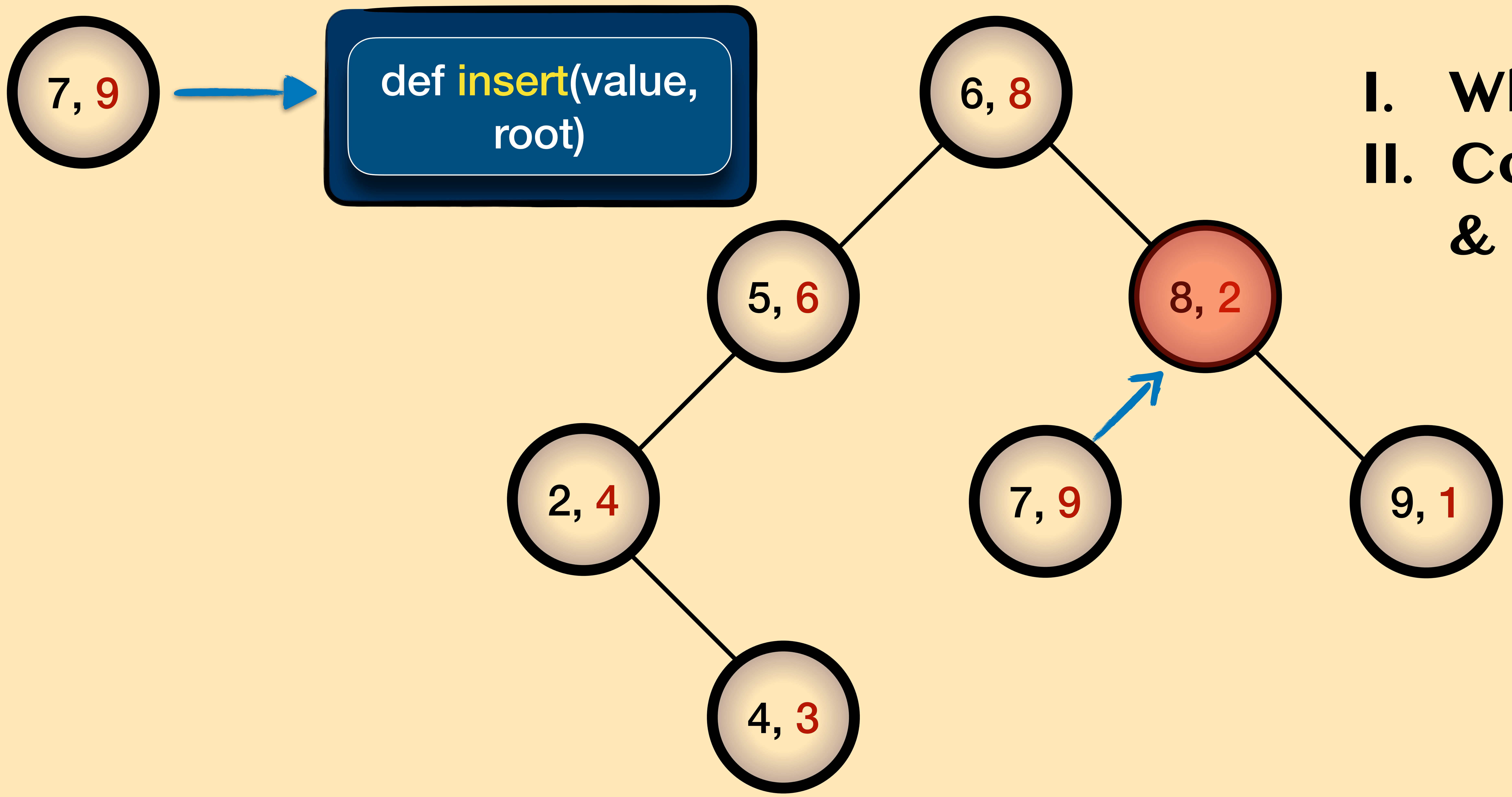
1. Where to insert?

Treap: tree + heap



1. **Where to insert?**

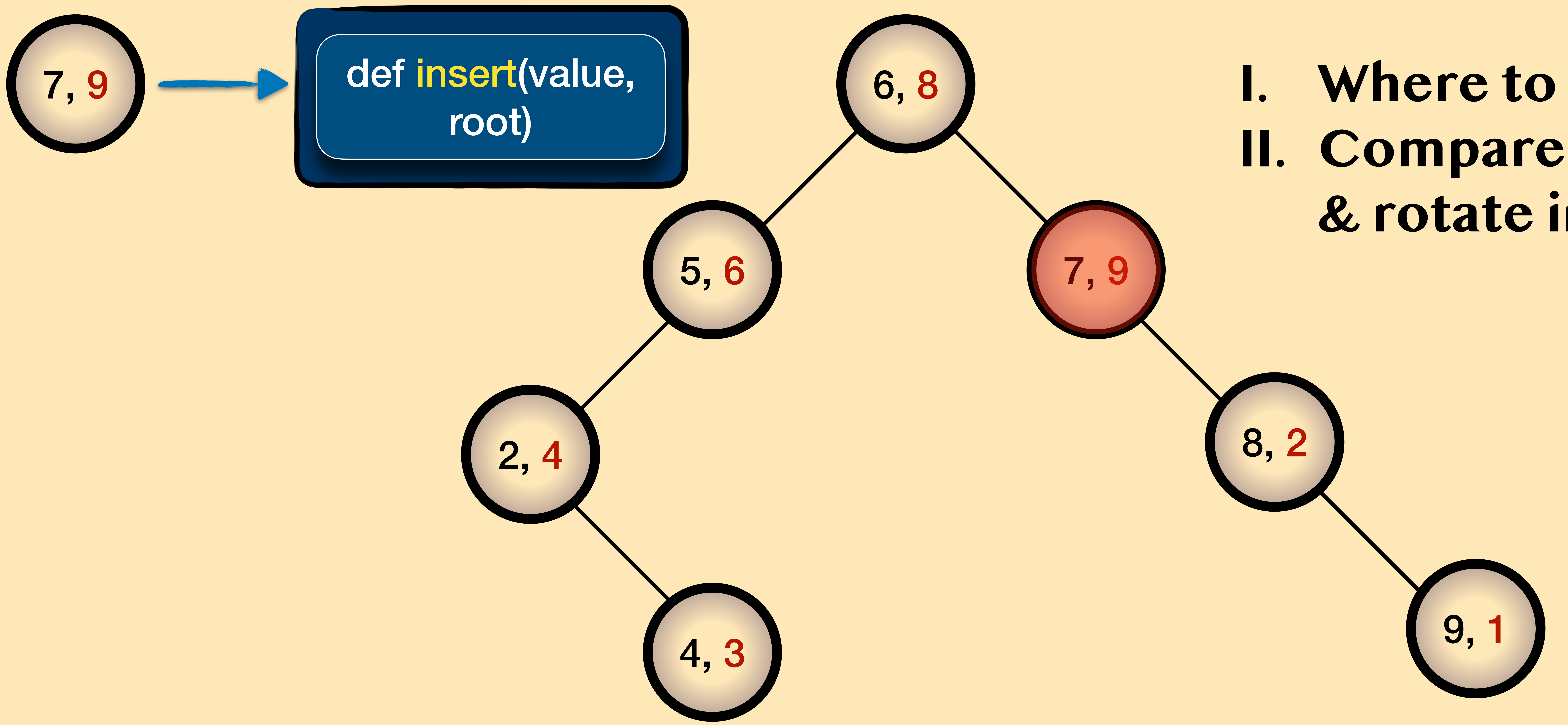
Treap: tree + heap



```
def insert(value,
root)
```

I. Where to insert?
II. Compare priority & rotate in case

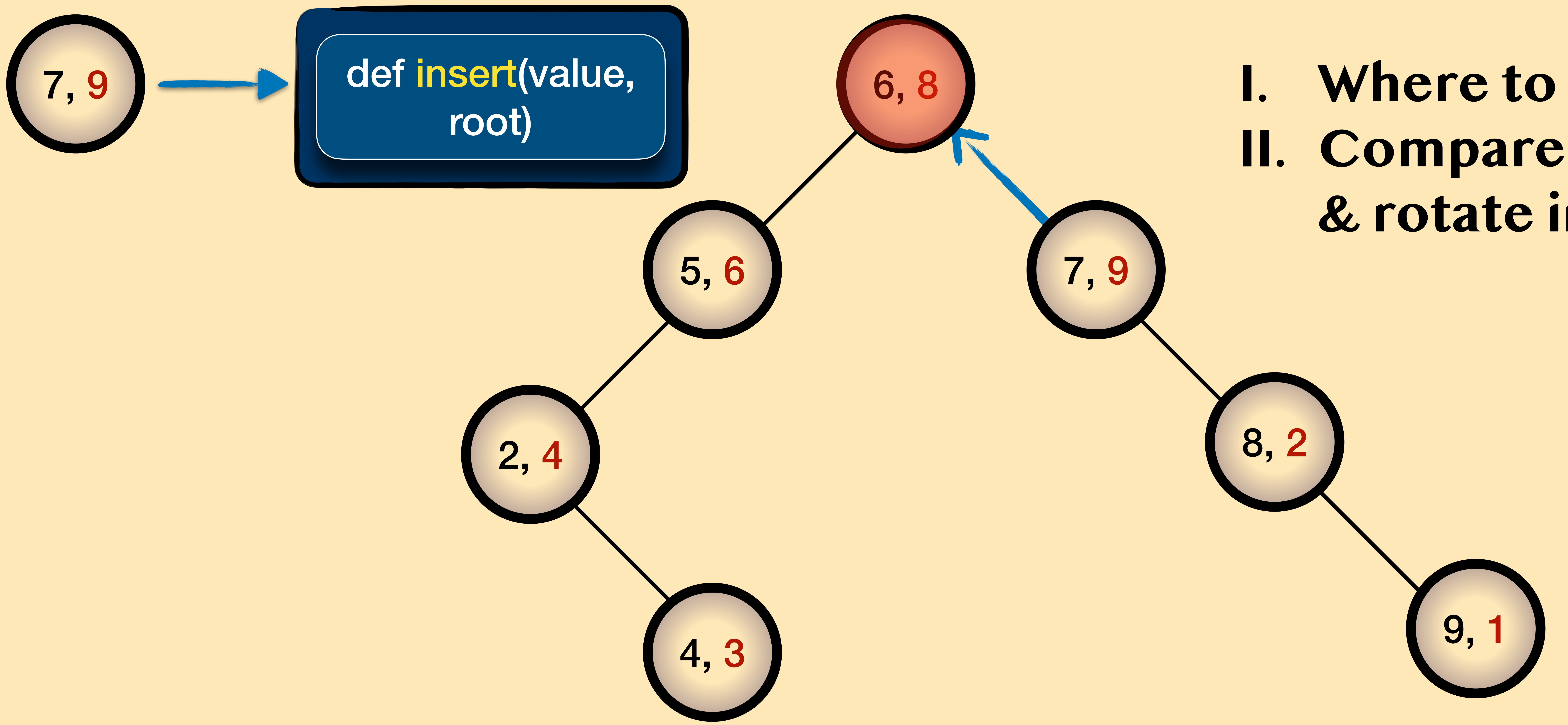
Treap: tree + heap



I. Where to insert?

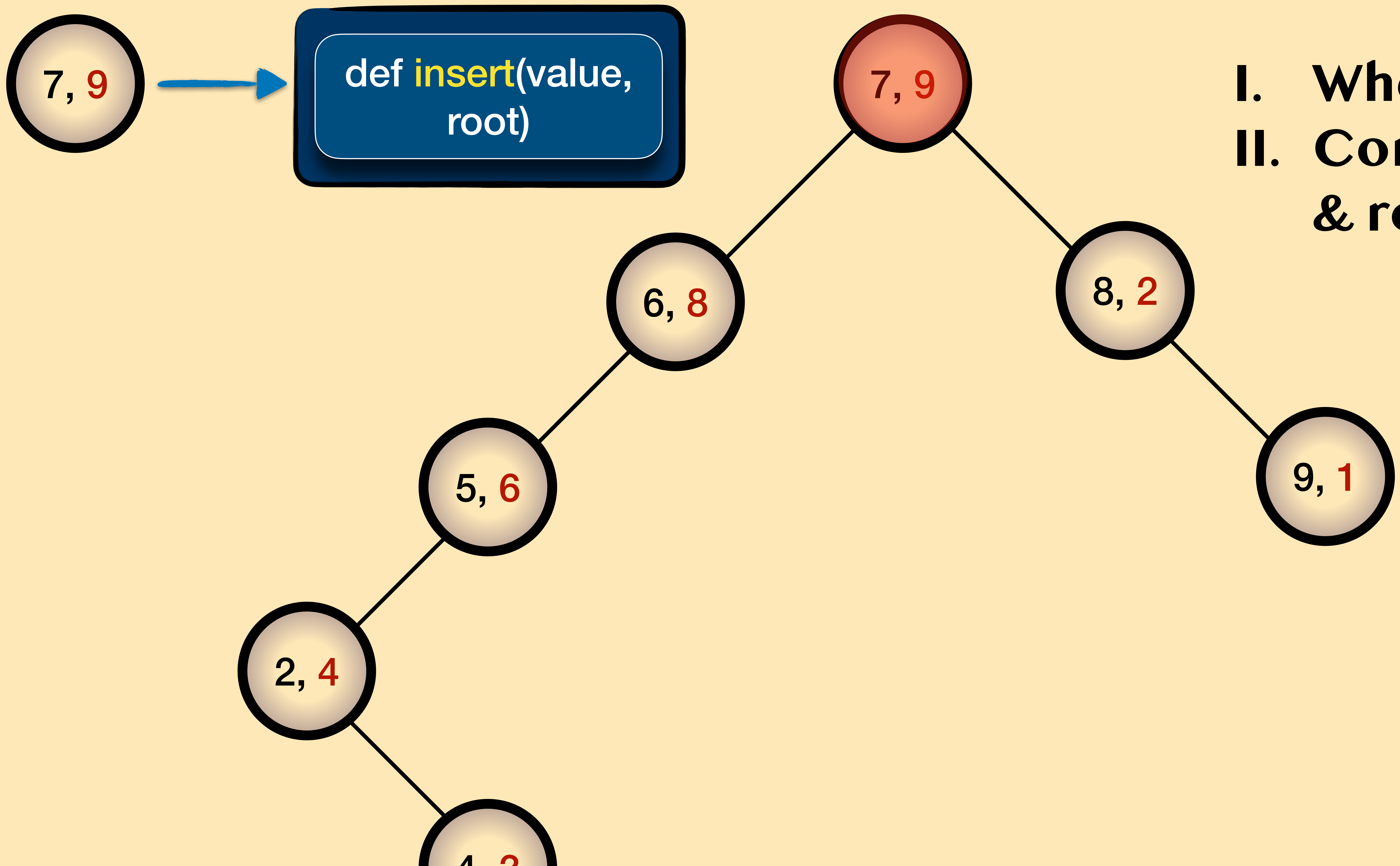
II. Compare priority & rotate in case

Treap: tree + heap



I. Where to insert?
II. Compare priority & rotate in case

Treap: tree + heap



```
def insert(value,
root)
```

I. Where to insert?
II. Compare priority & rotate in case