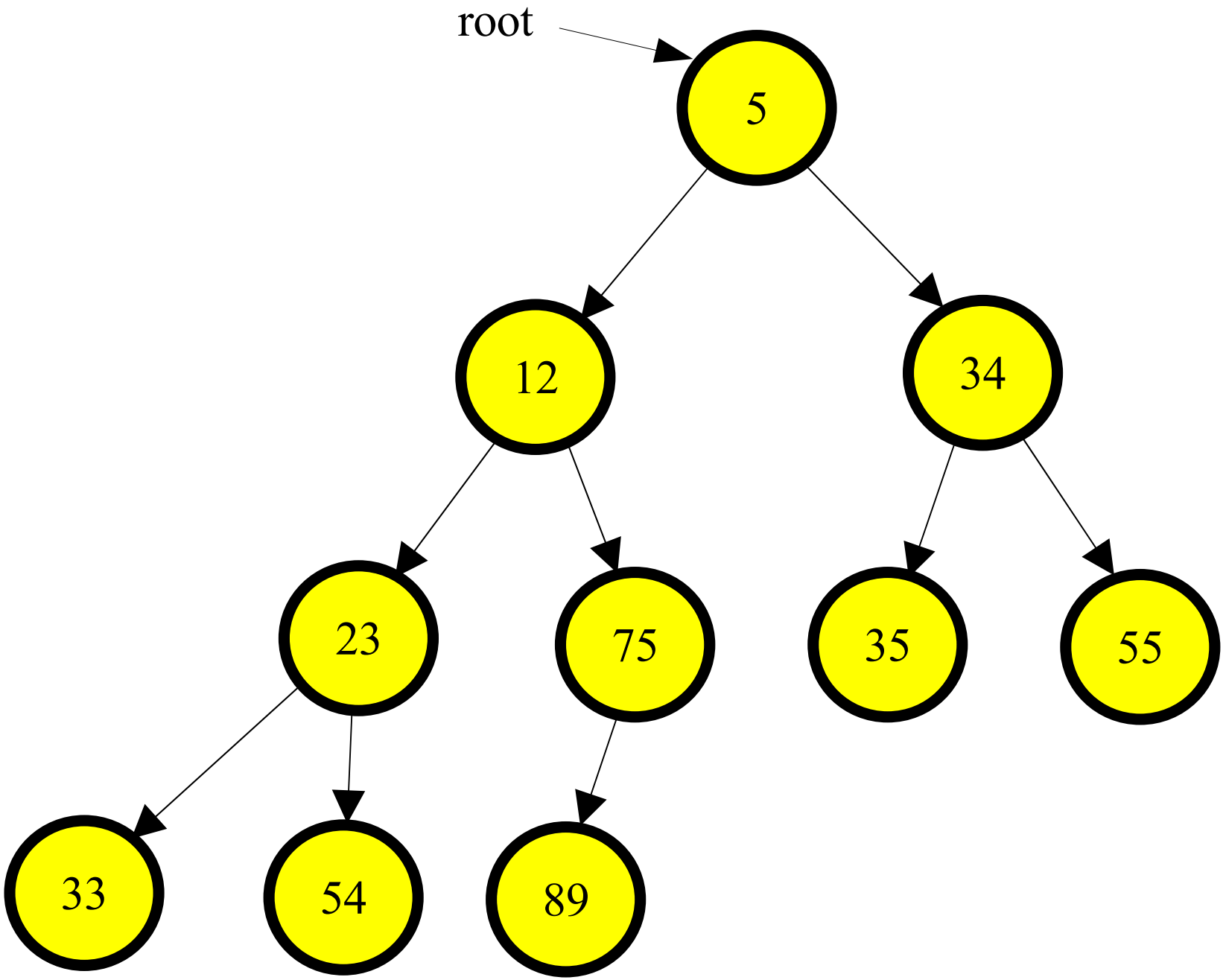
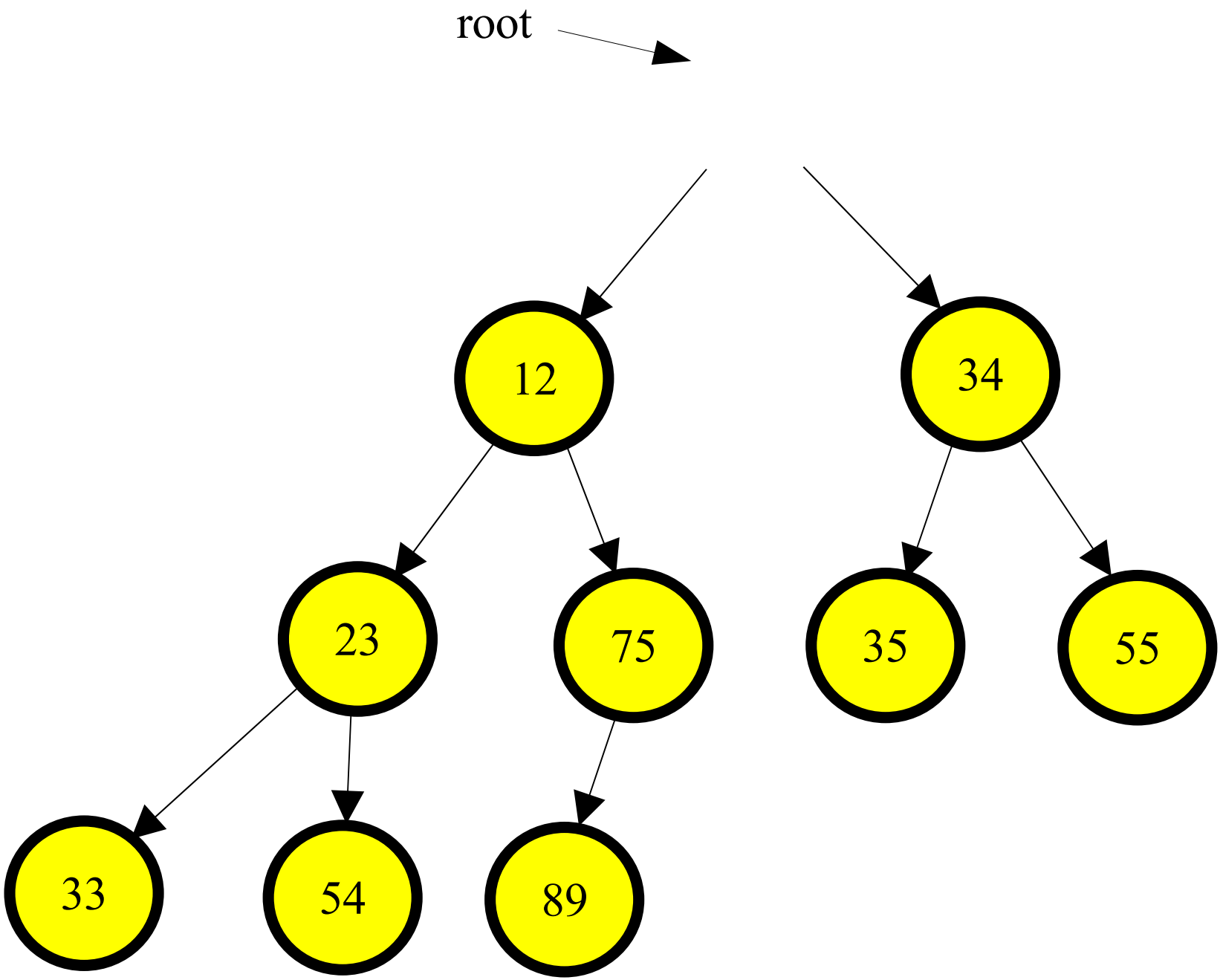
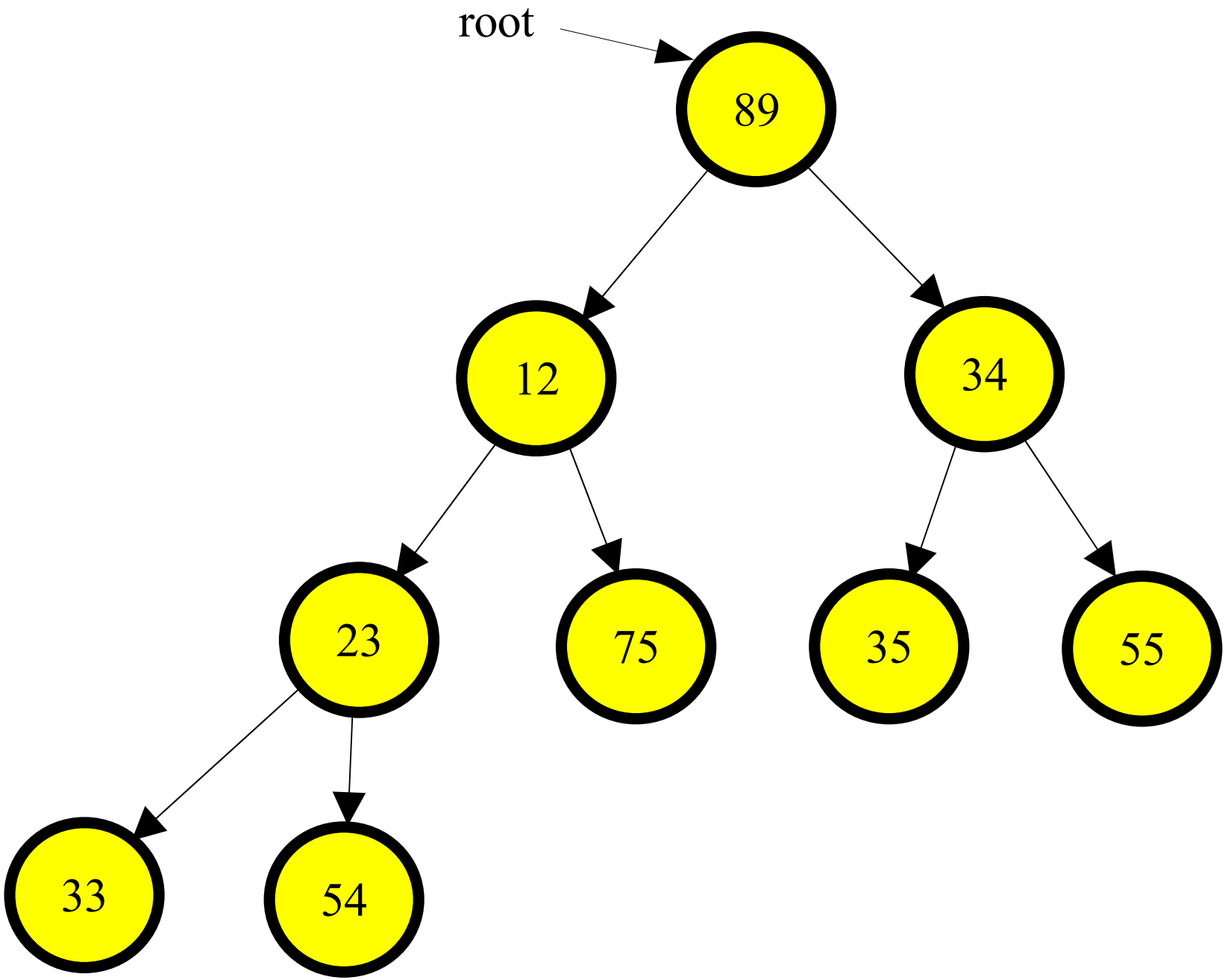
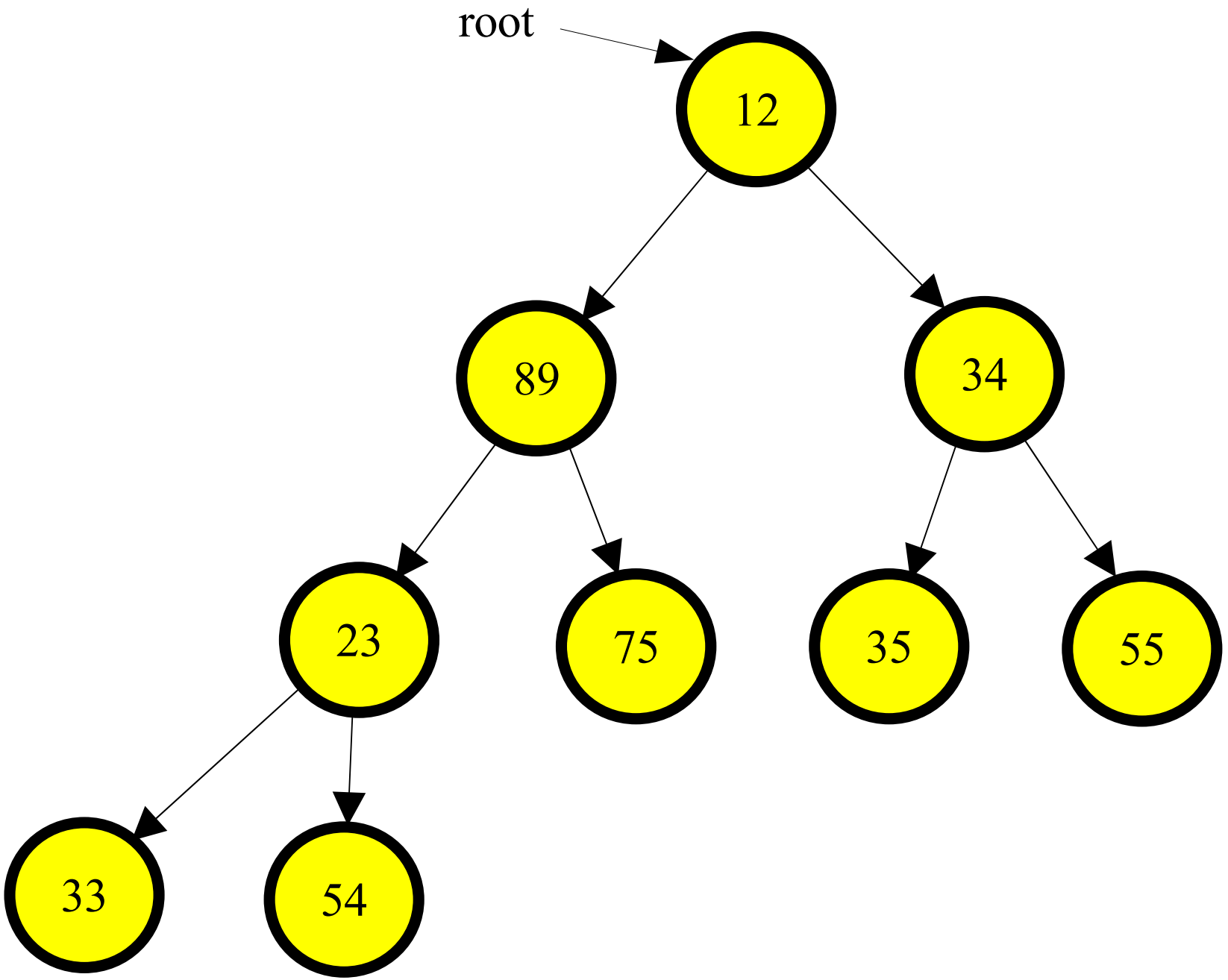
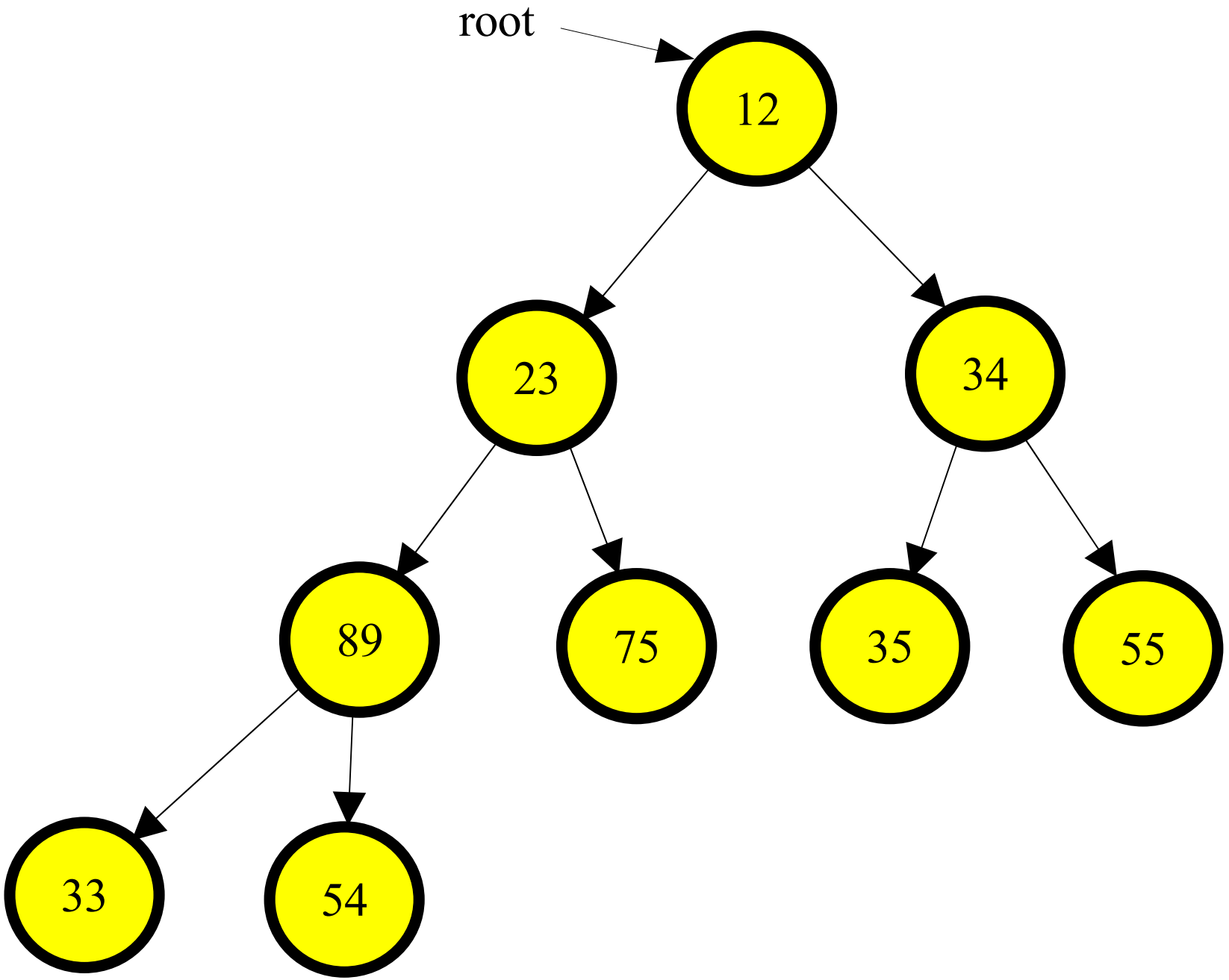
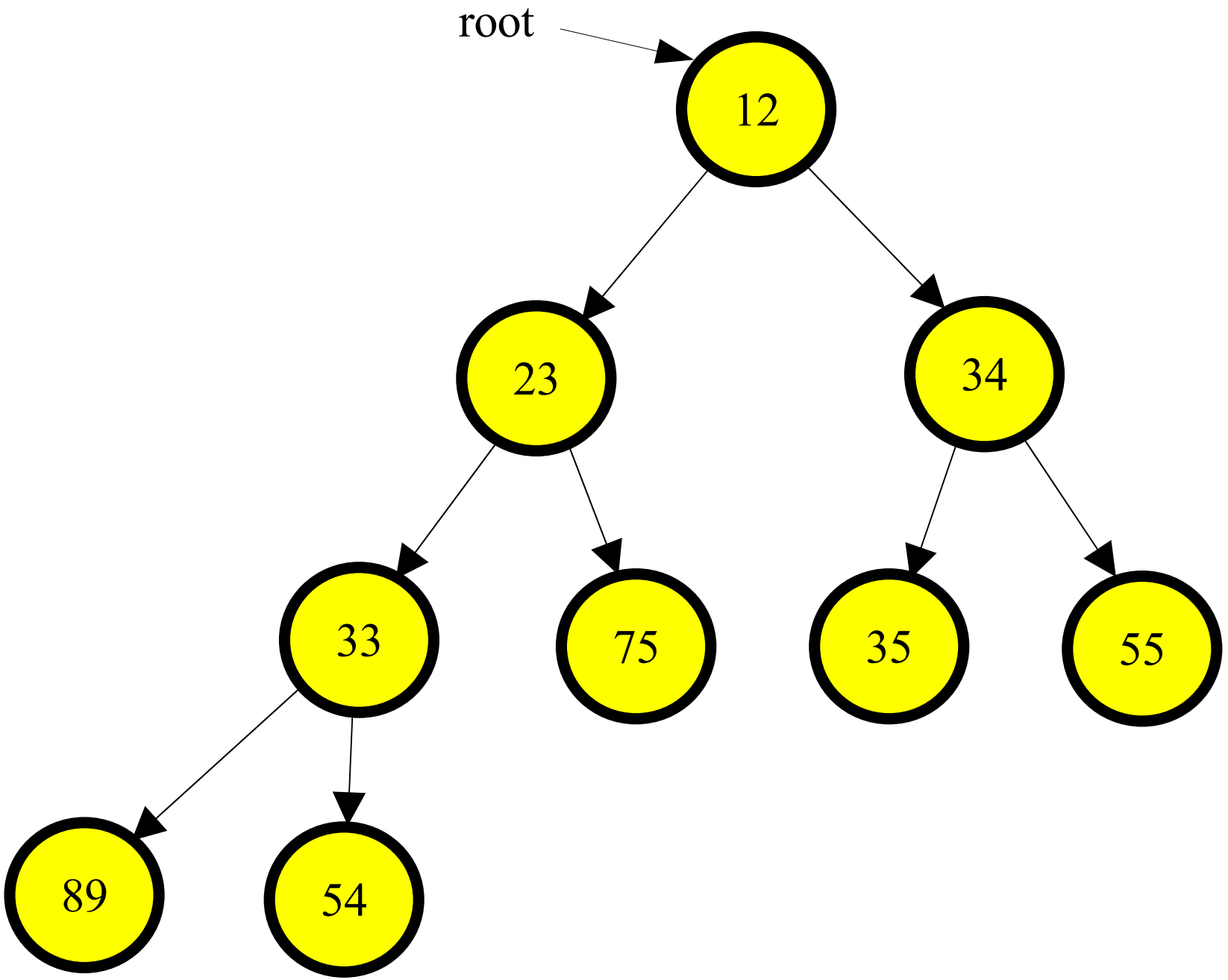

root
5
12
34
23
75
35
55
33
54
89

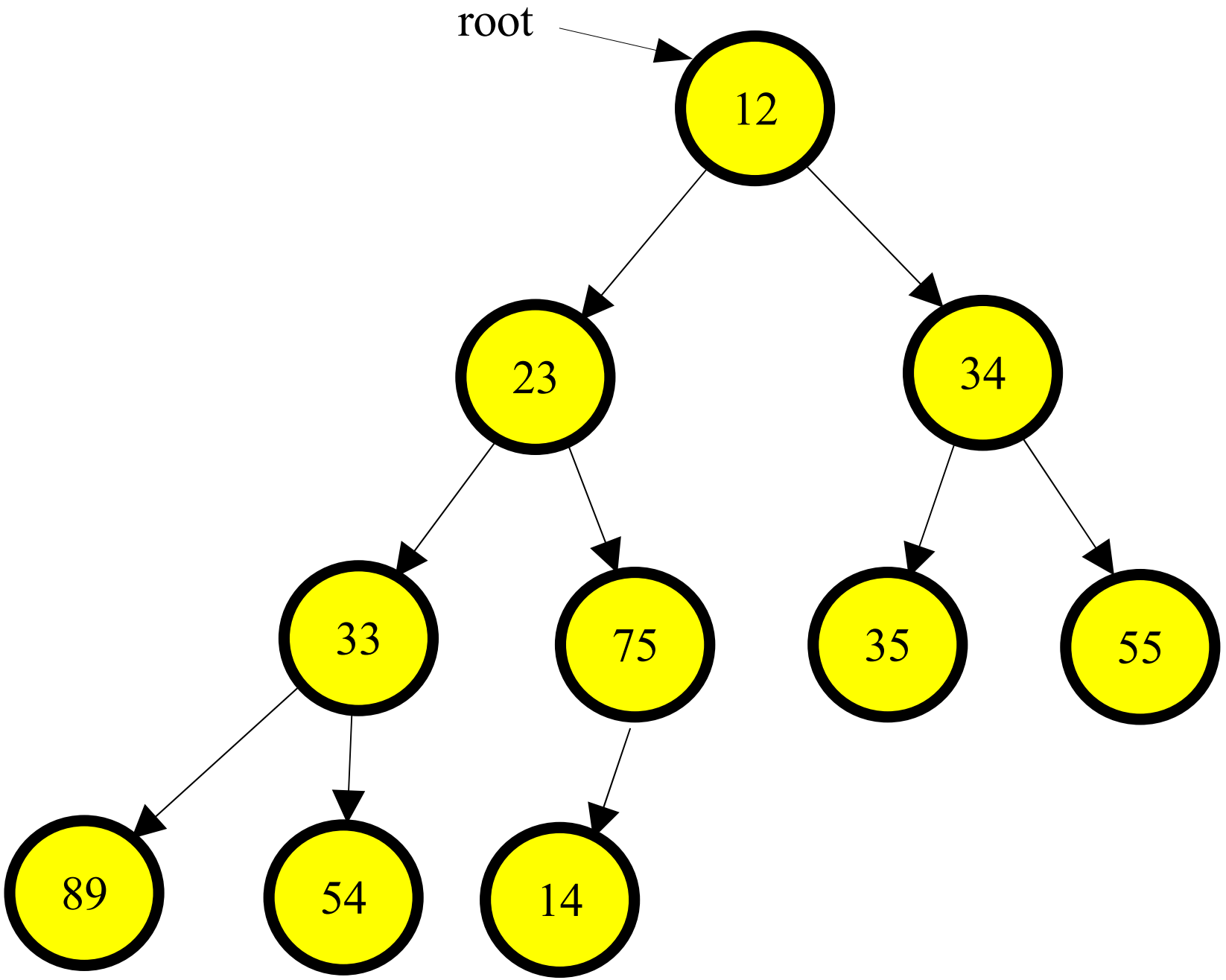

root
12
34
23
75
35
55
33
54
89

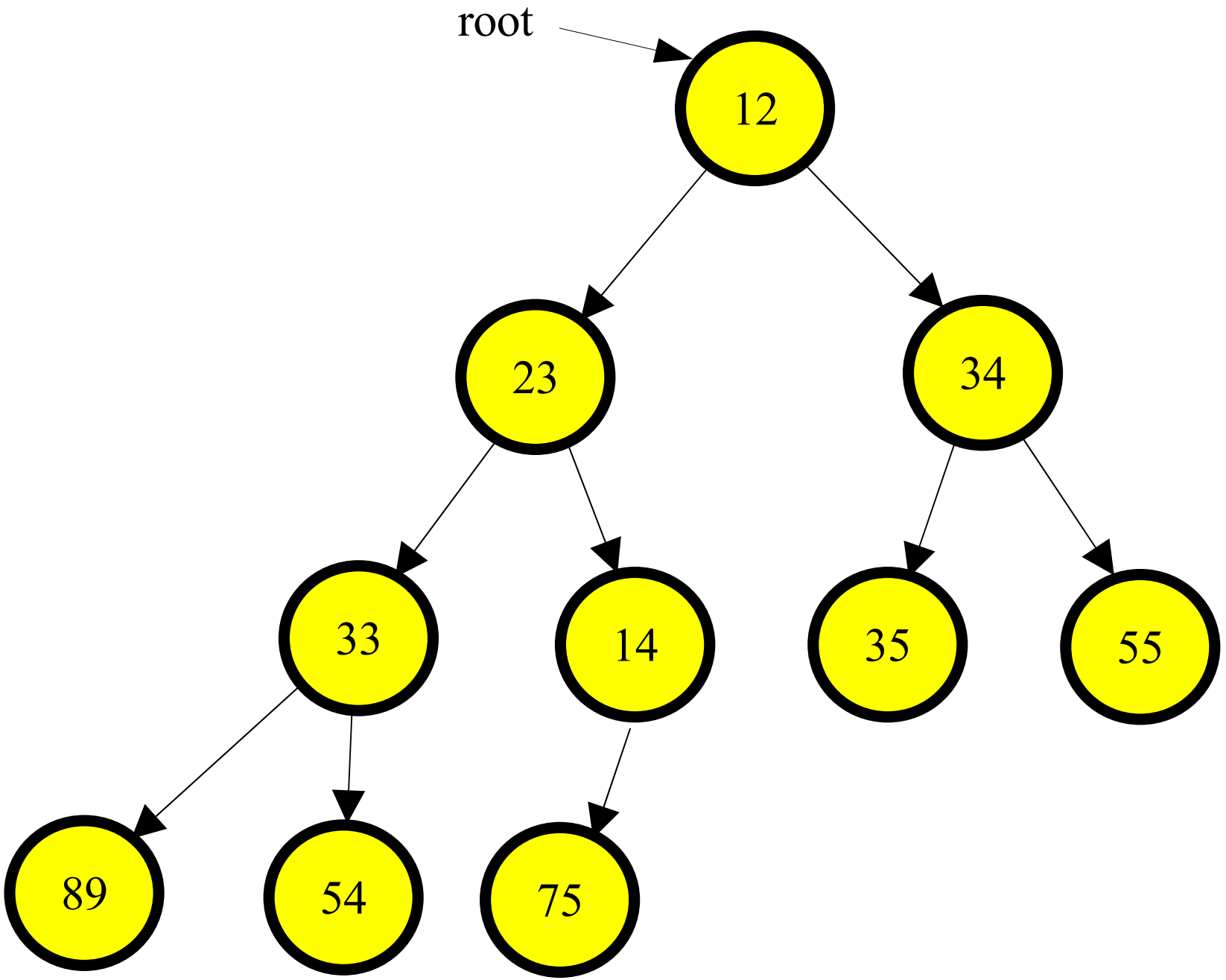

root
89
12
34
23
75
35
55
33
54

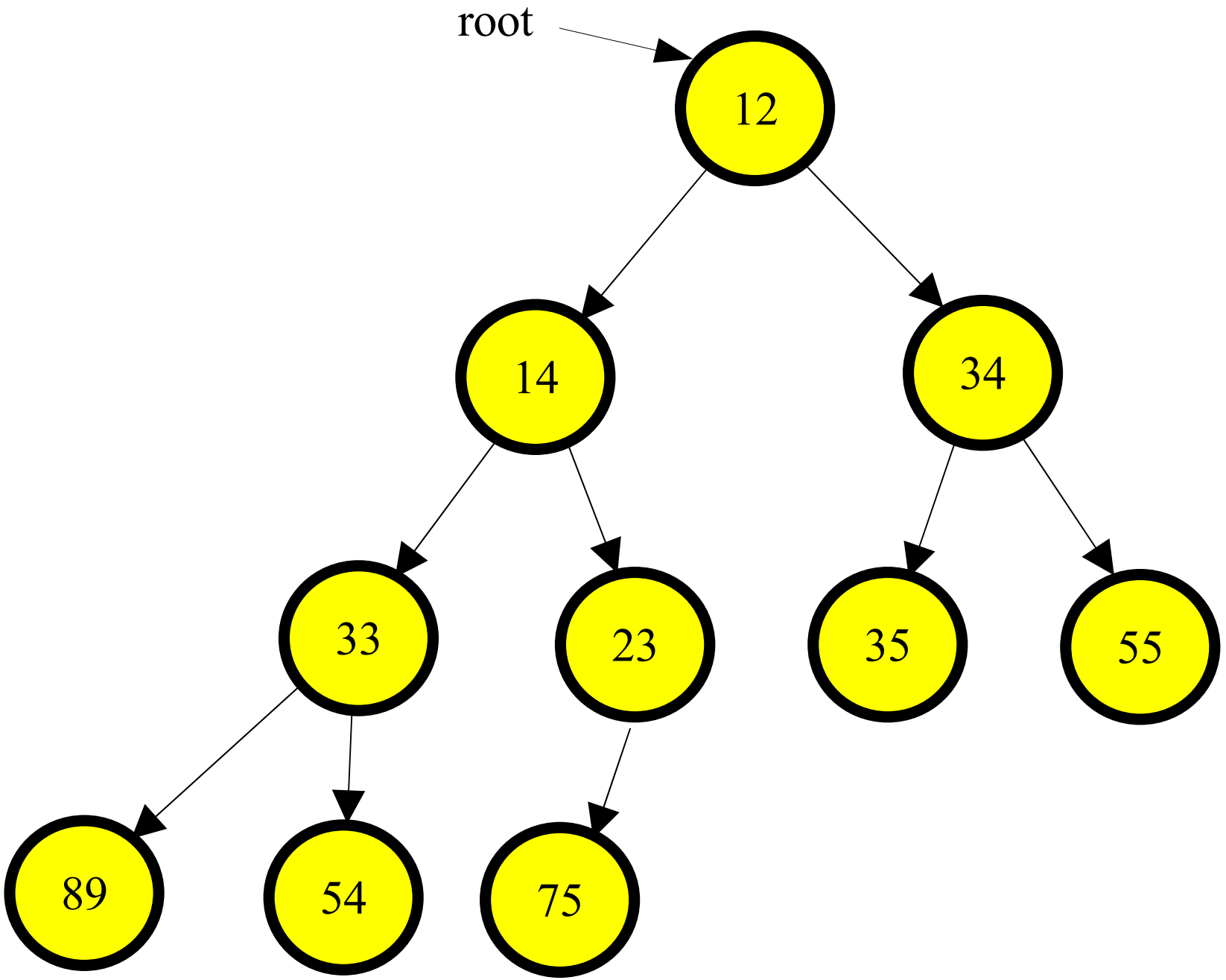

root
12
89
34
23
75
35
55
33
54


root
12
23
34
89
75
35
55
33
54


root
12
23
34
33
75
35
55
89
54

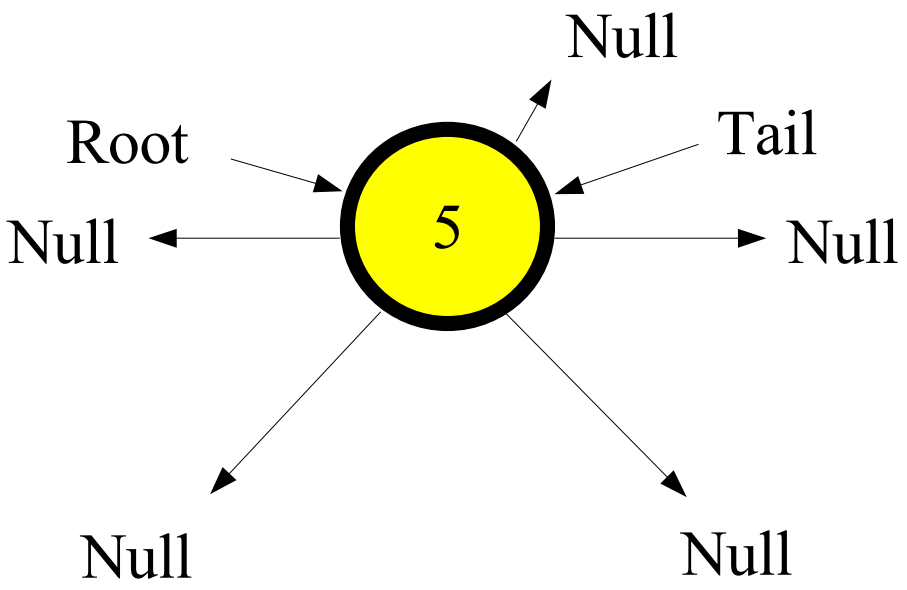

root
12
23
34
33
75
35
55
89
54
14

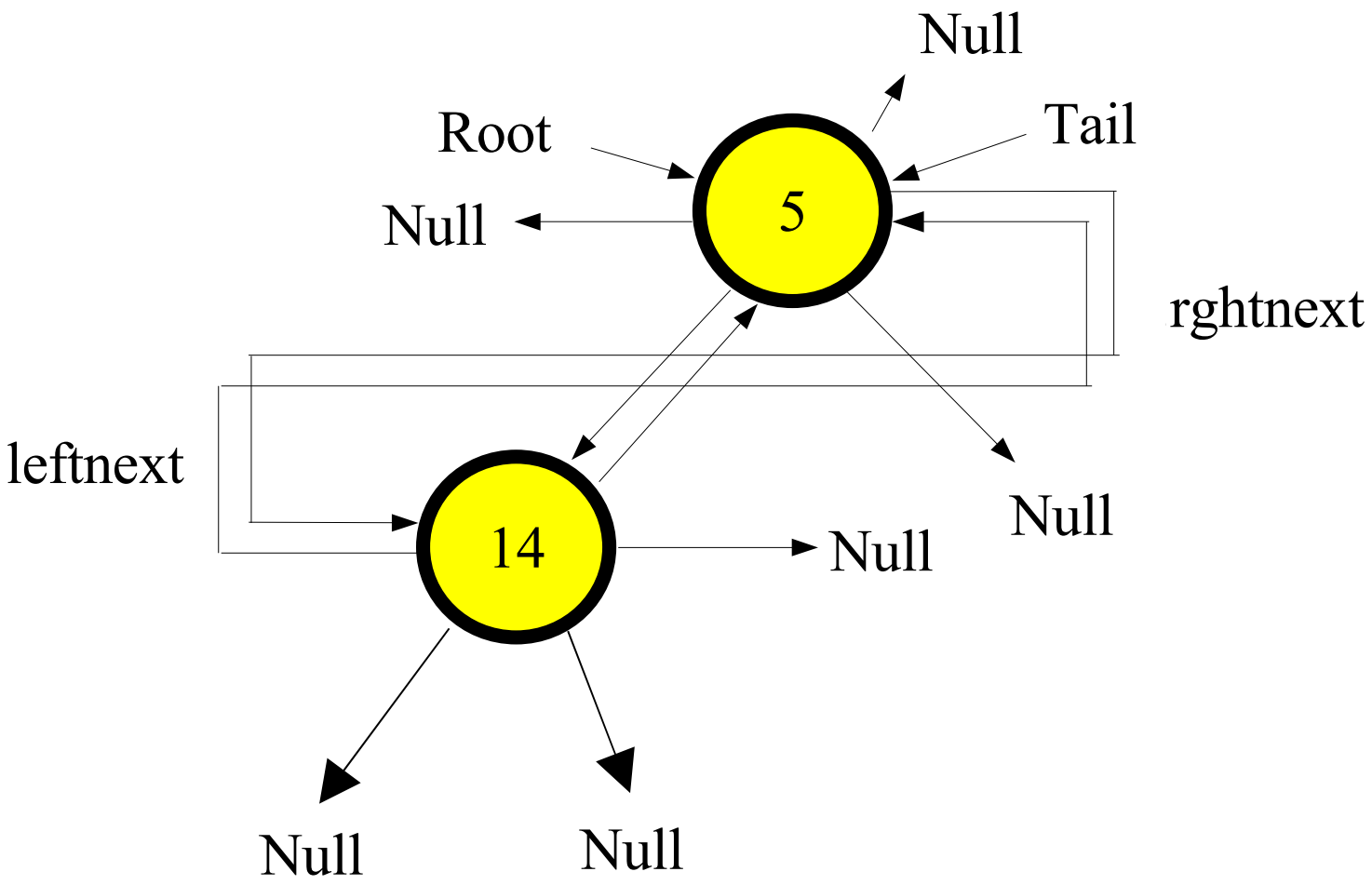

root
12
23
34
33
14
35
55
89
54
75

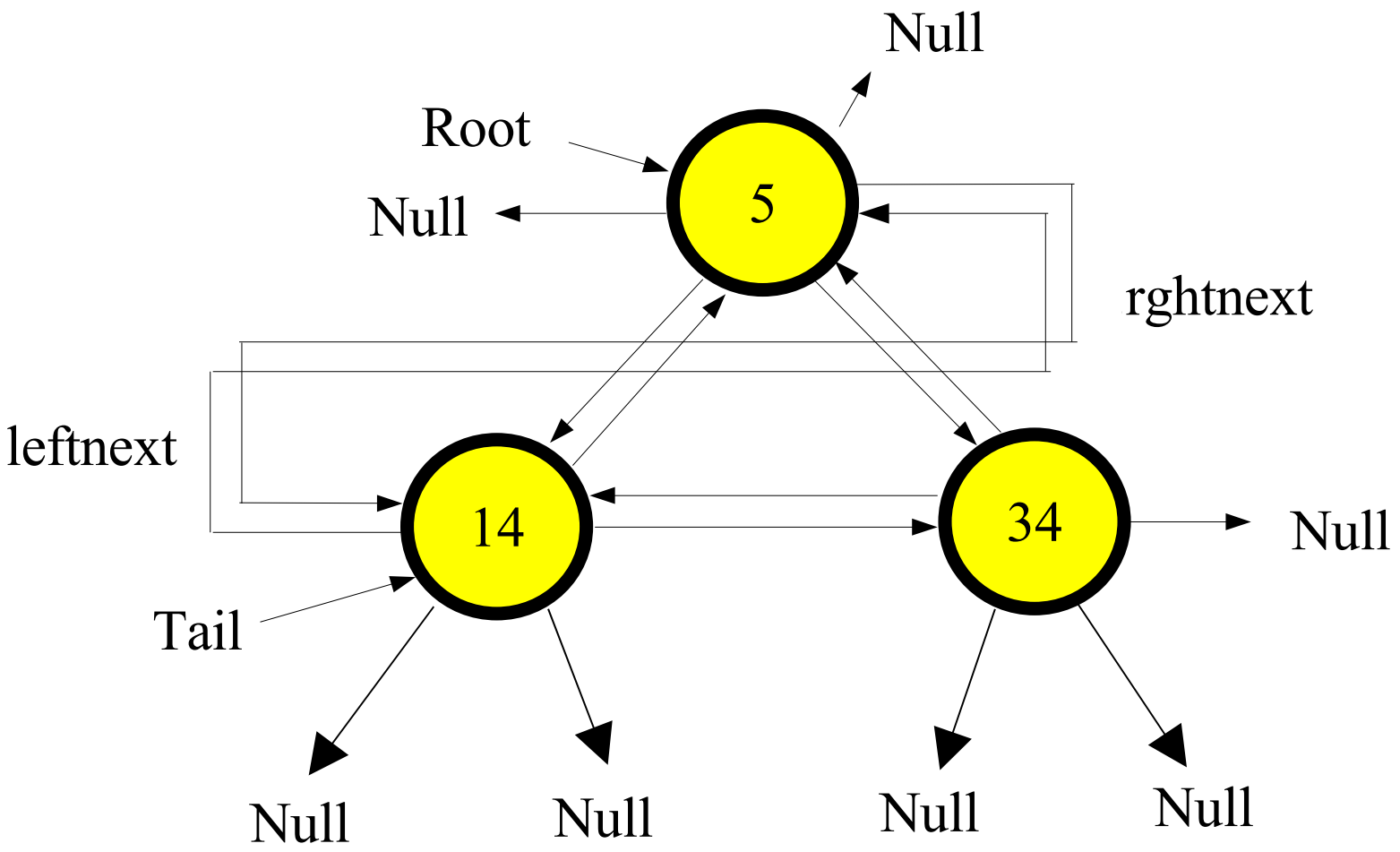

root
12
14
34
33
23
35
55
89
54
75

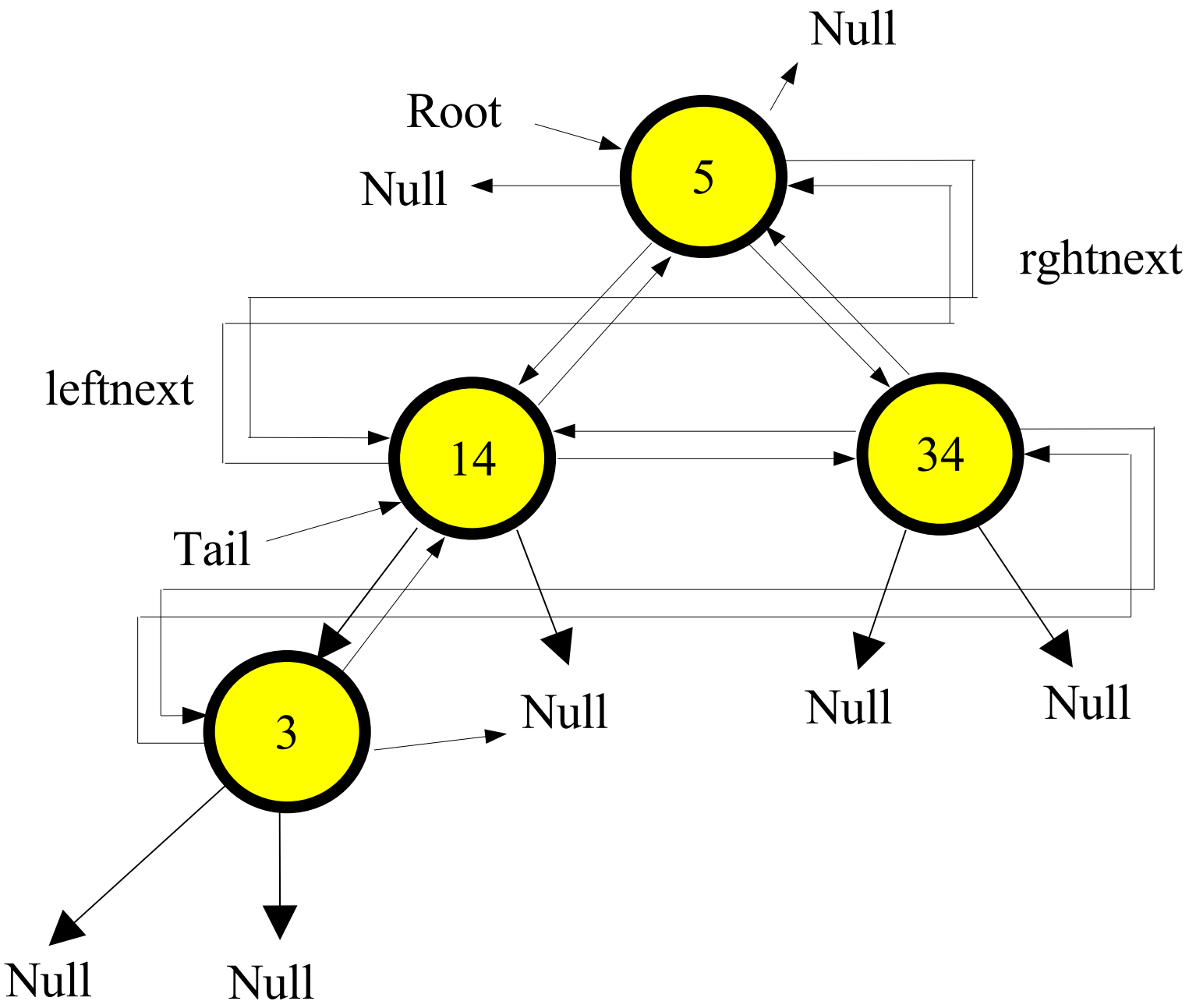
Implementation Details

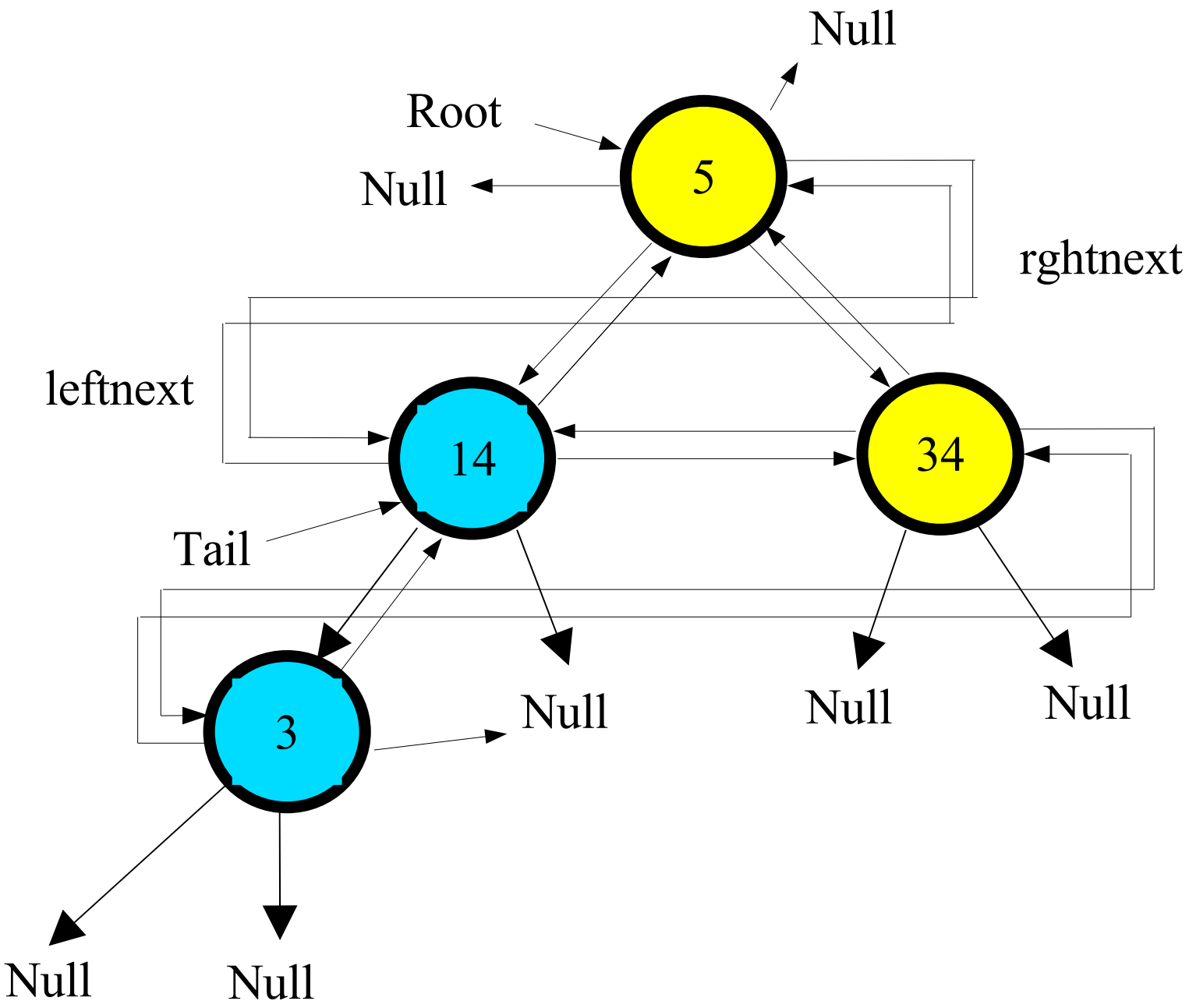

Root
Tail
Null

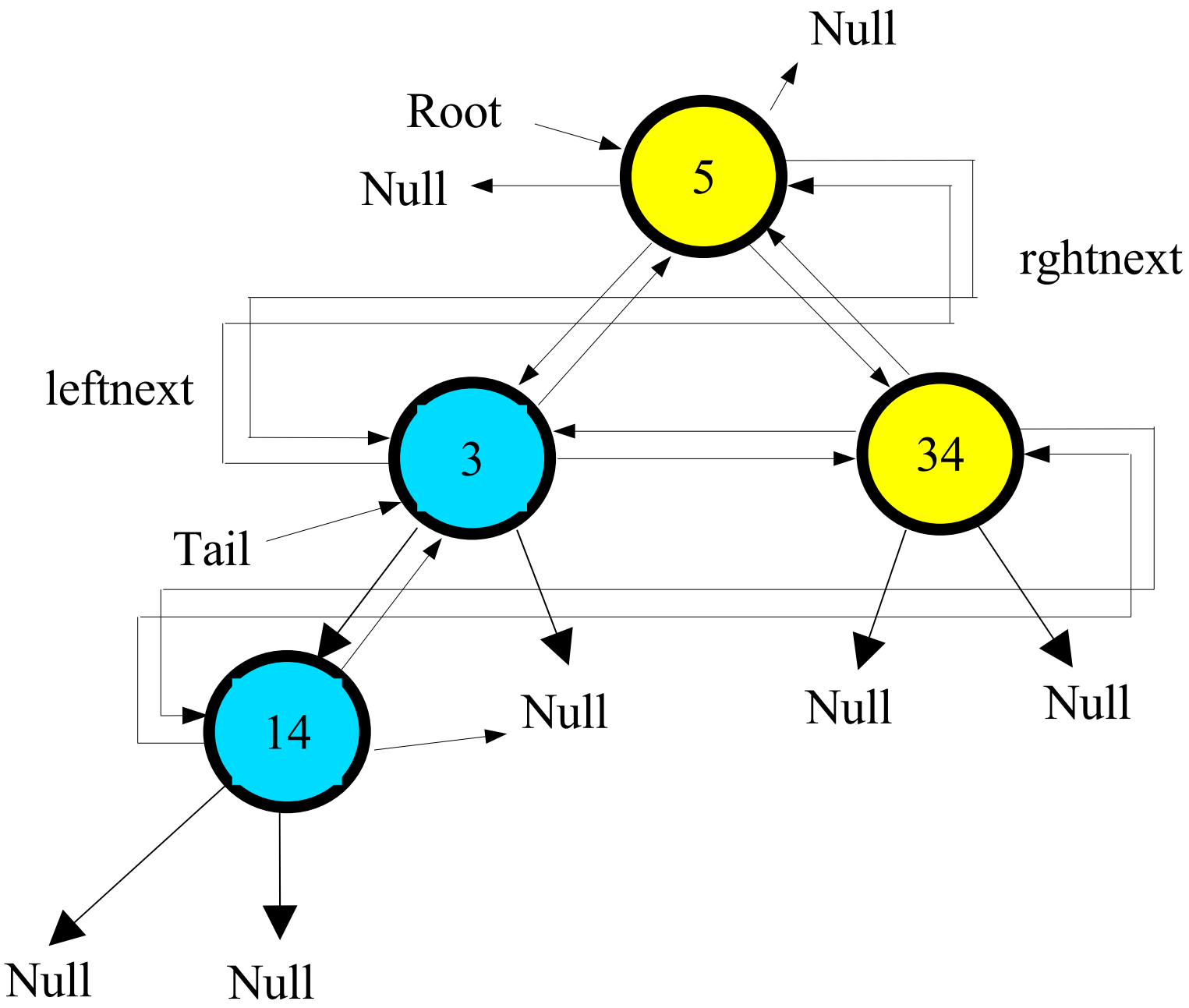

Null
Root
Tail
Null
5
Null
Null
Null

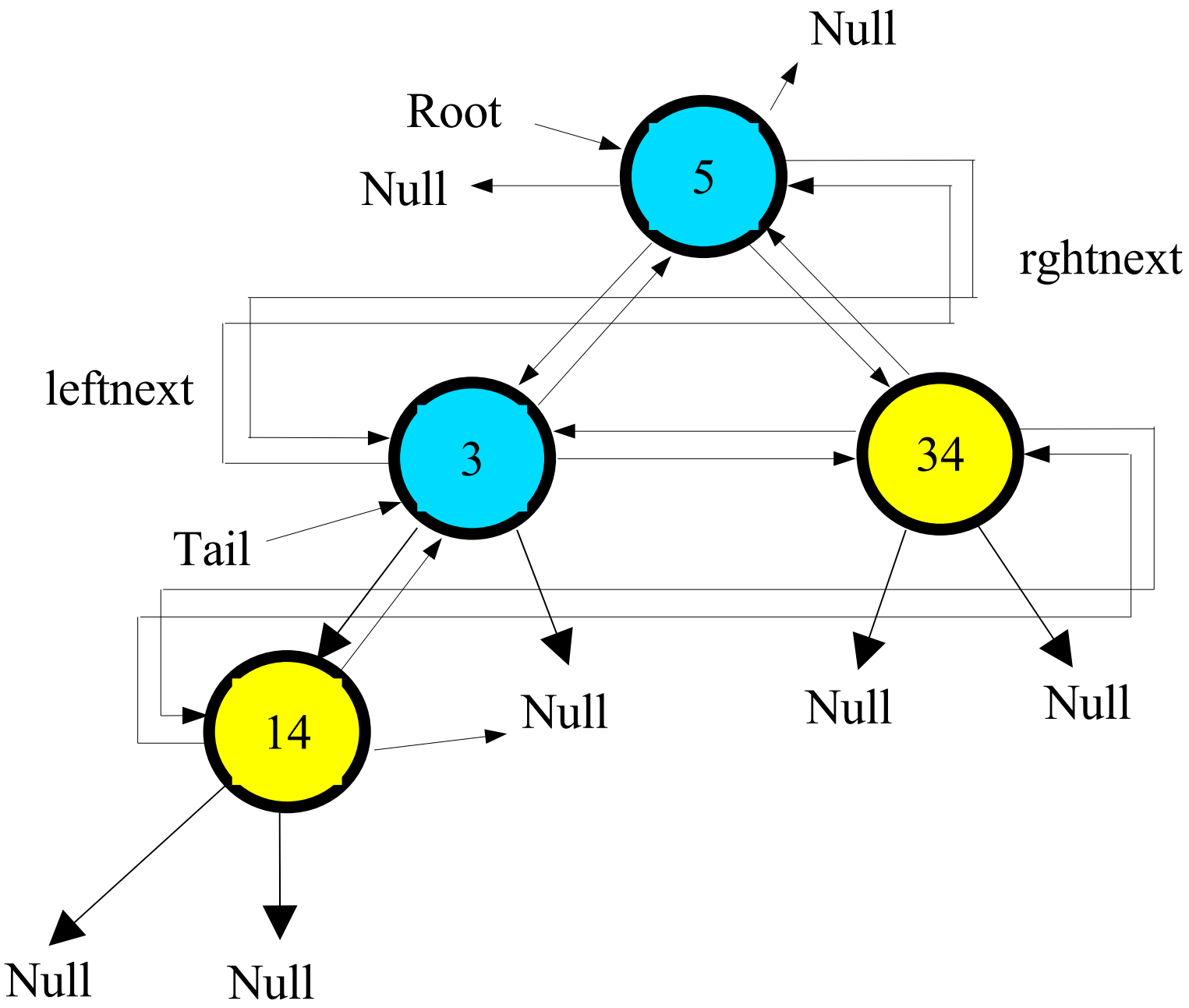

Null
Root
Tail
Null
5
rghtnext
leftnext
Null
14
Null
Null
Null

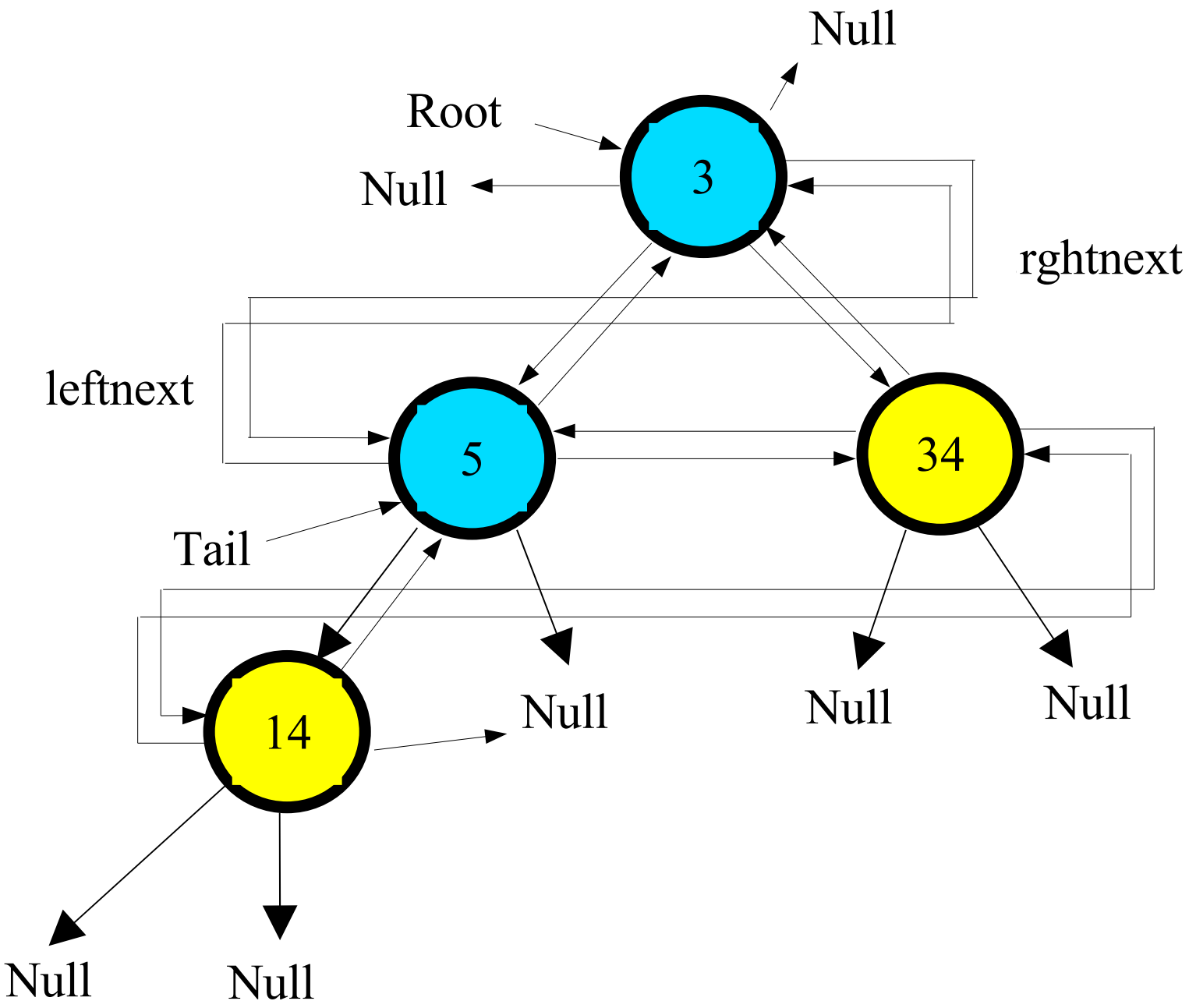

Null
Root
5
Null
rghtnext
leftnext
14
34
Null
Tail
Null
Null
Null
Null

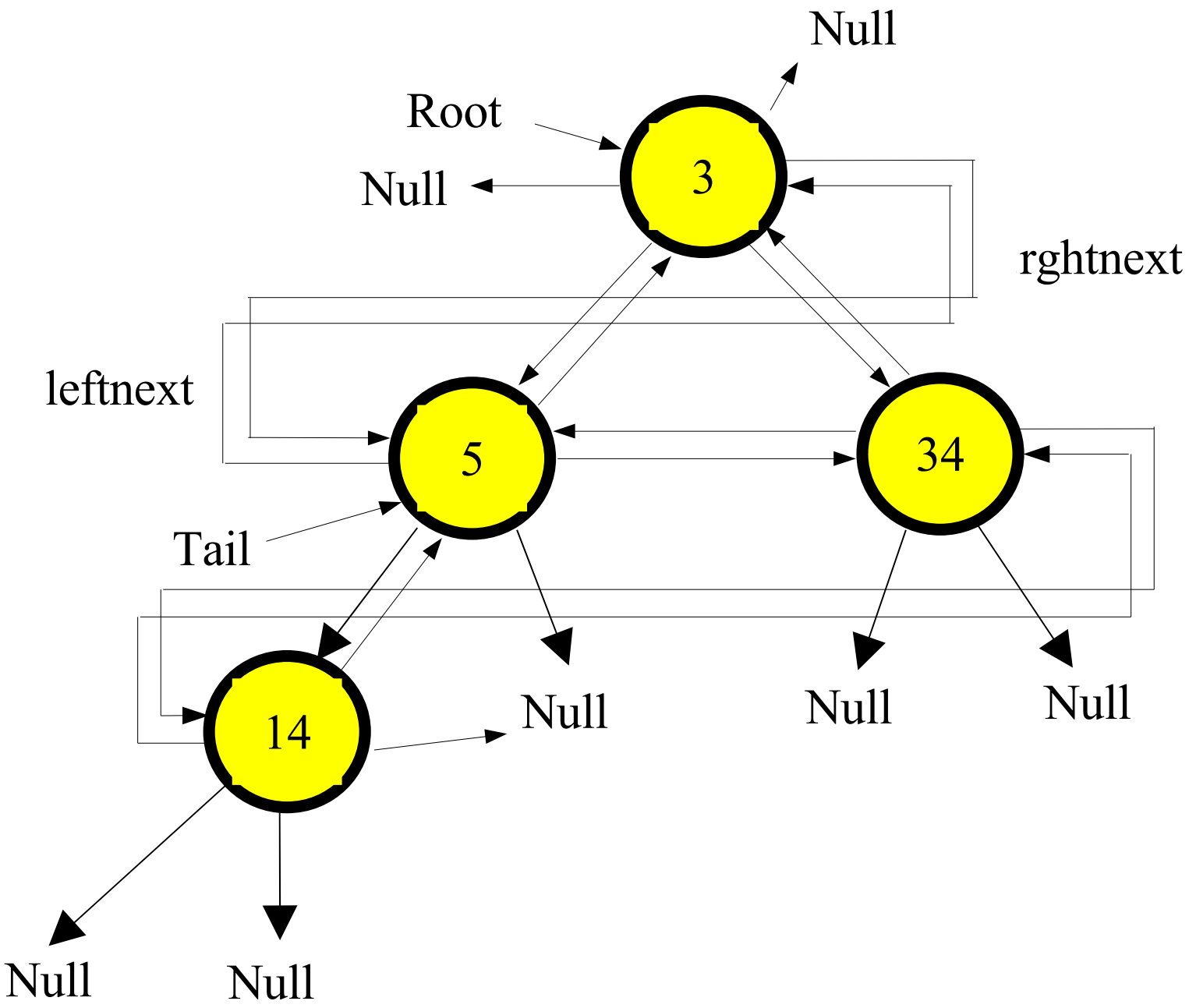

Null
Root
5
Null
rghtnext
leftnext
14
34
Tail
3
Null
Null
Null
Null
Null

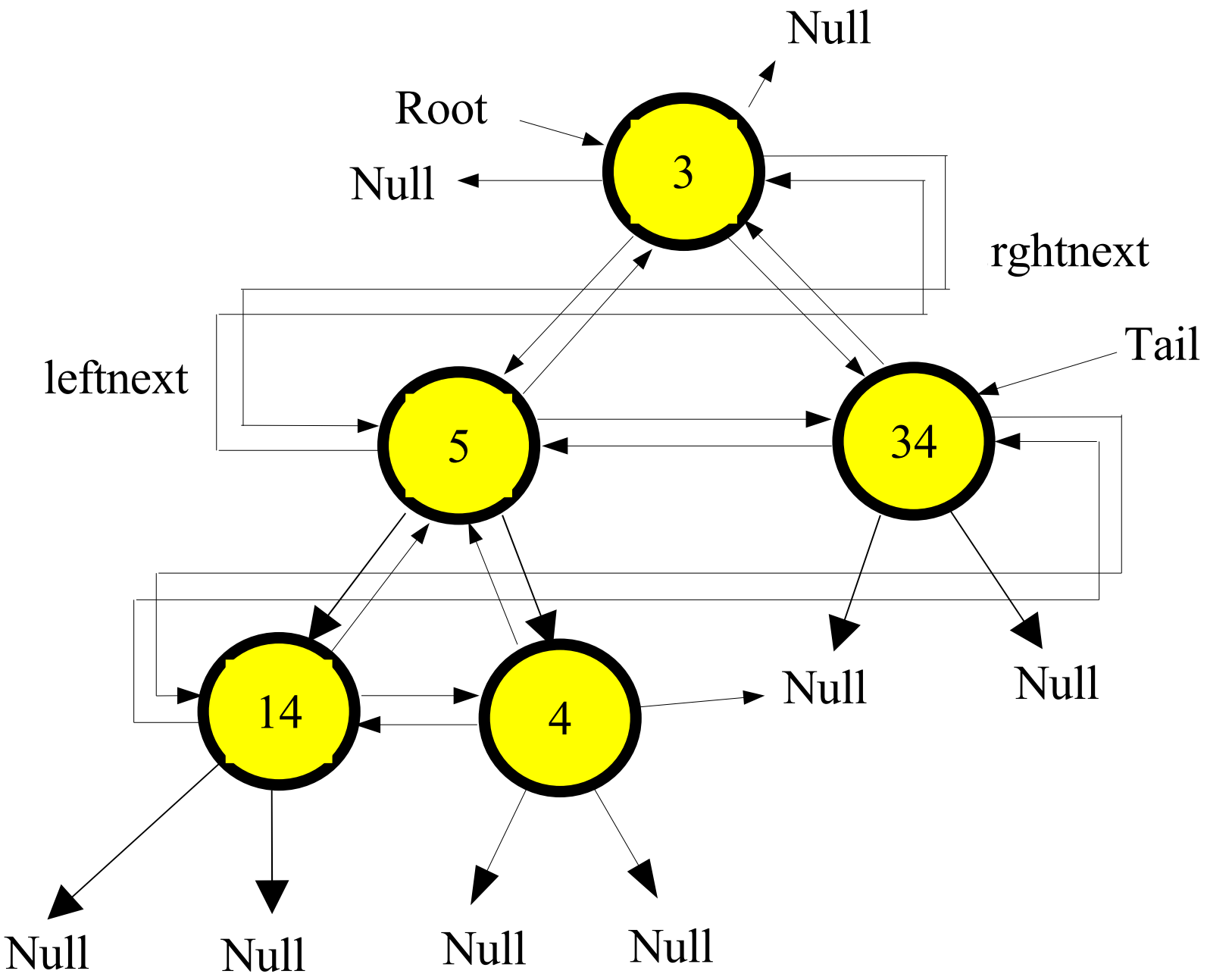

Null
Root
5
Null
rghtnext
leftnext
14
34
Tail
3
Null
Null
Null
Null
Null

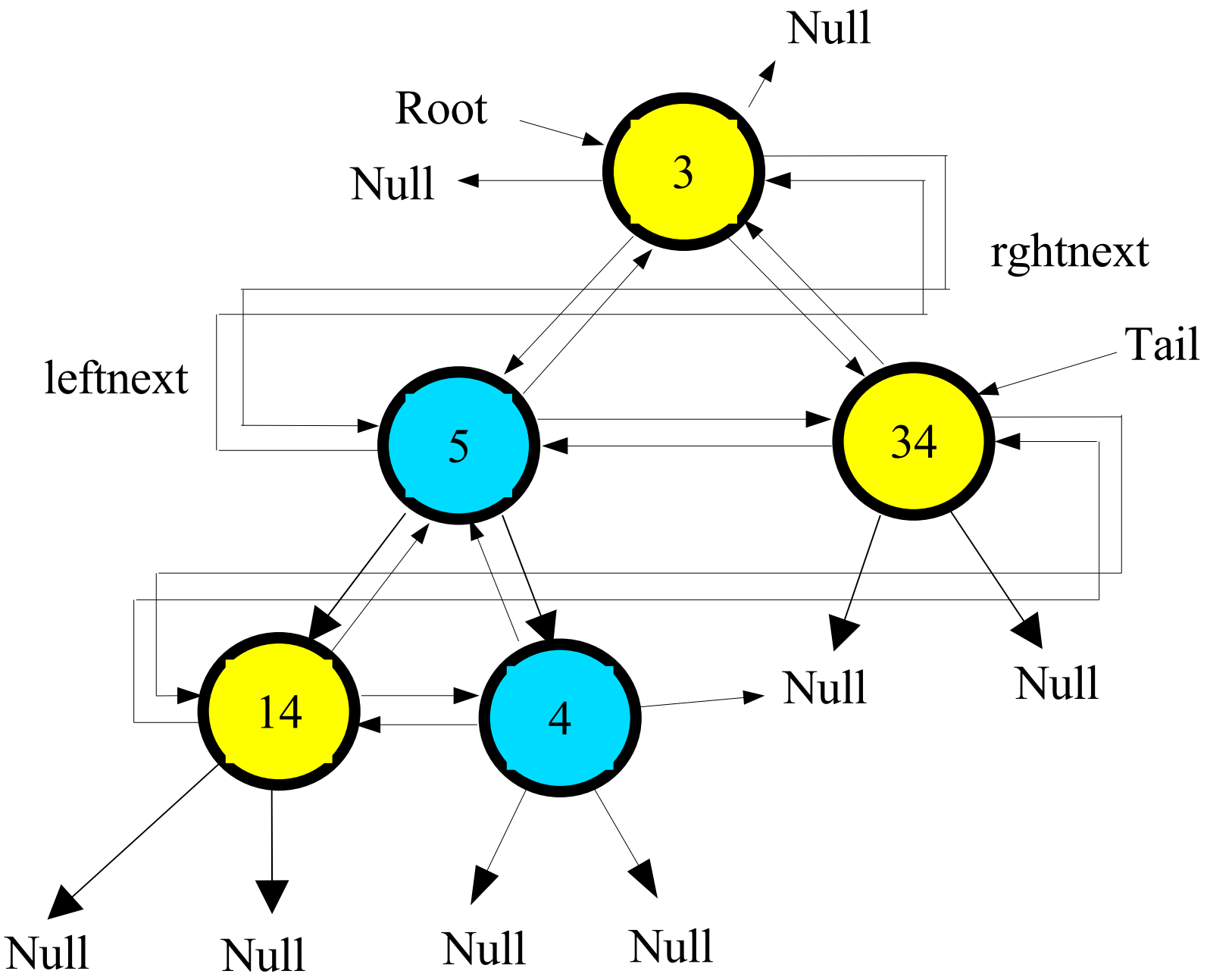

Null
Root
5
Null
rghtnext
leftnext
3
34
Tail
14
Null
Null
Null
Null
Null

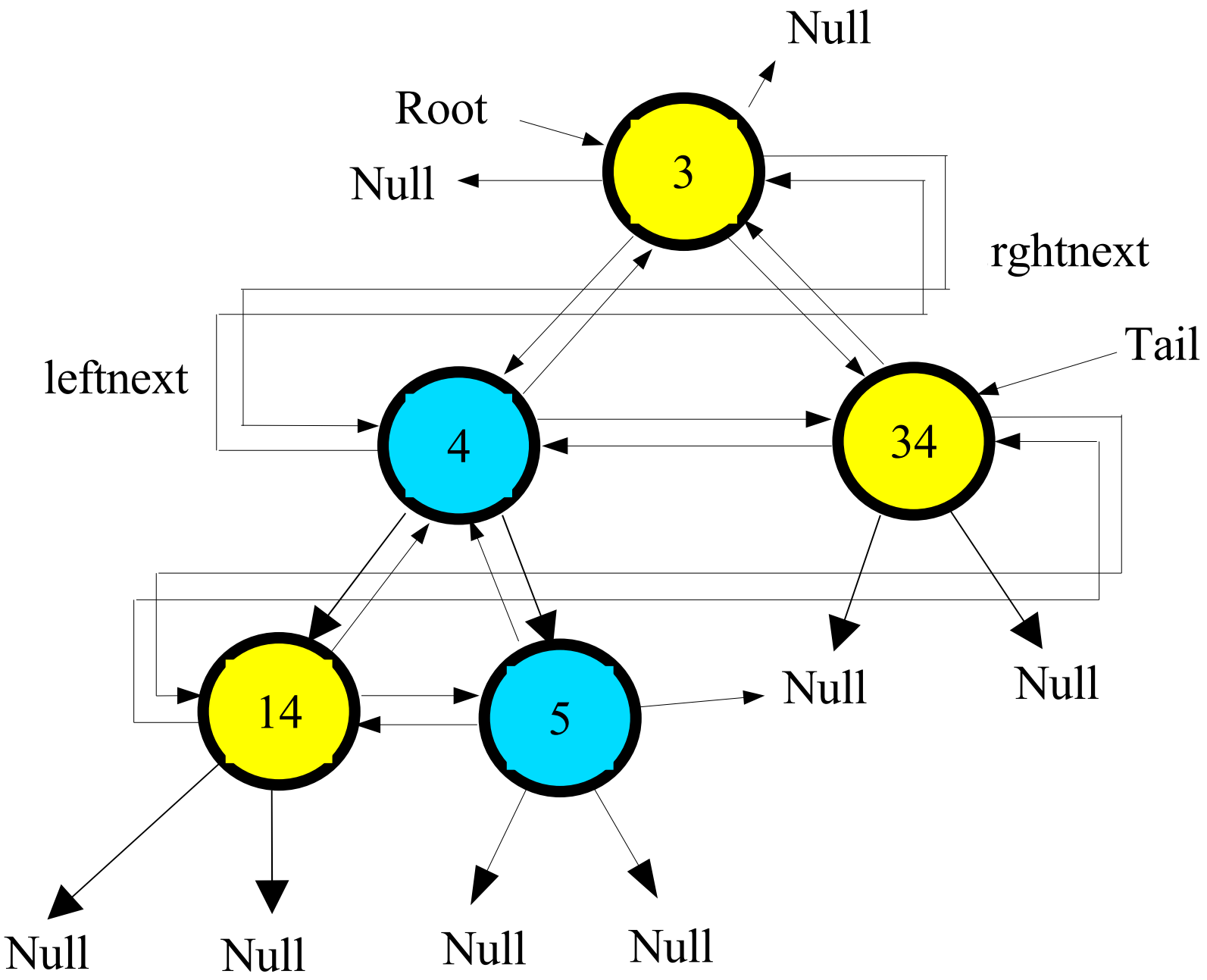

Null
Root
5
Null
rghtnext
leftnext
3
34
Tail
14
Null
Null
Null
Null
Null

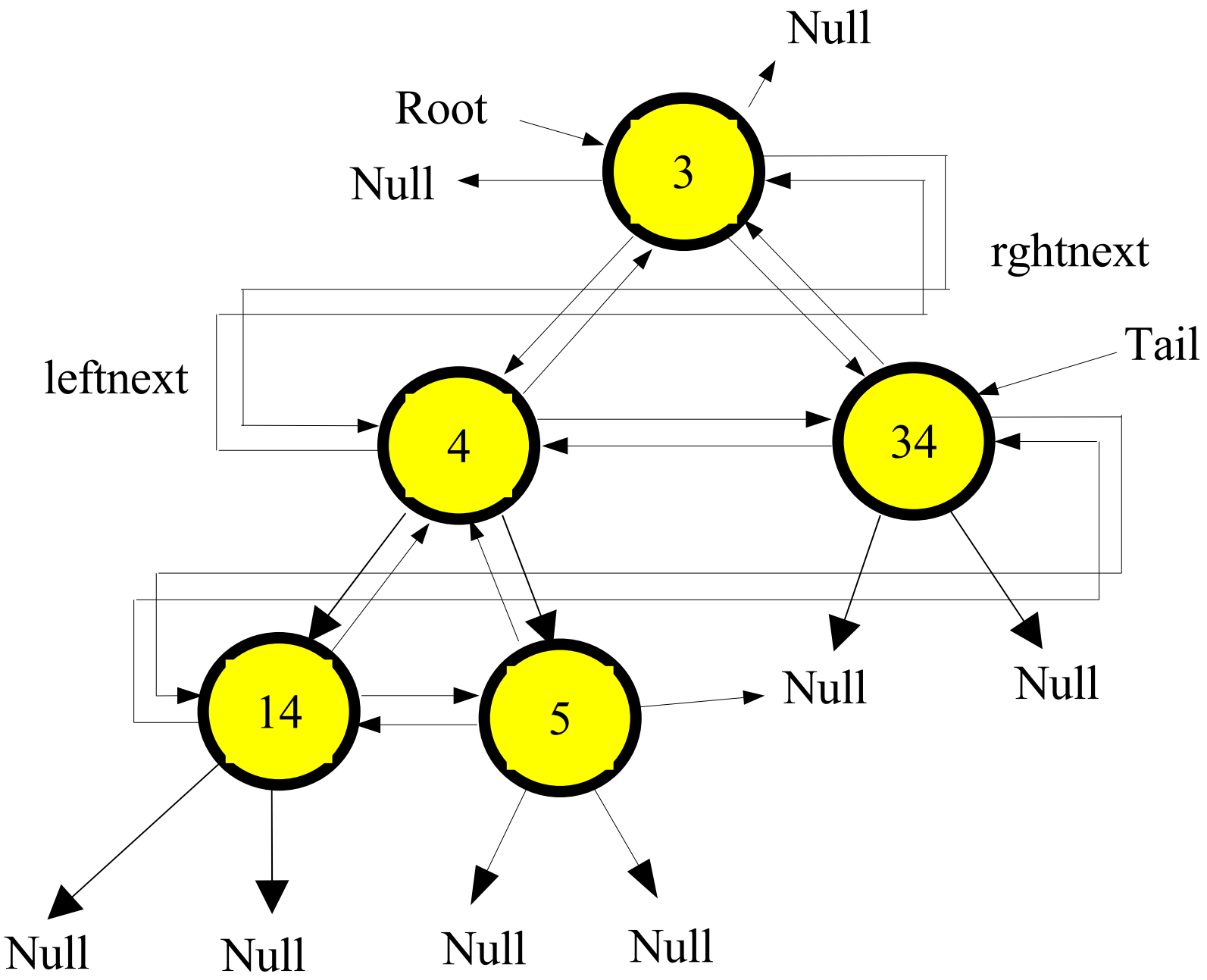

Null
Root
3
Null
rghtnext
leftnext
5
34
Tail
14
Null
Null
Null
Null
Null


Null
Root
3
Null
rghtnext
leftnext
5
34
Tail
14
Null
Null
Null
Null
Null

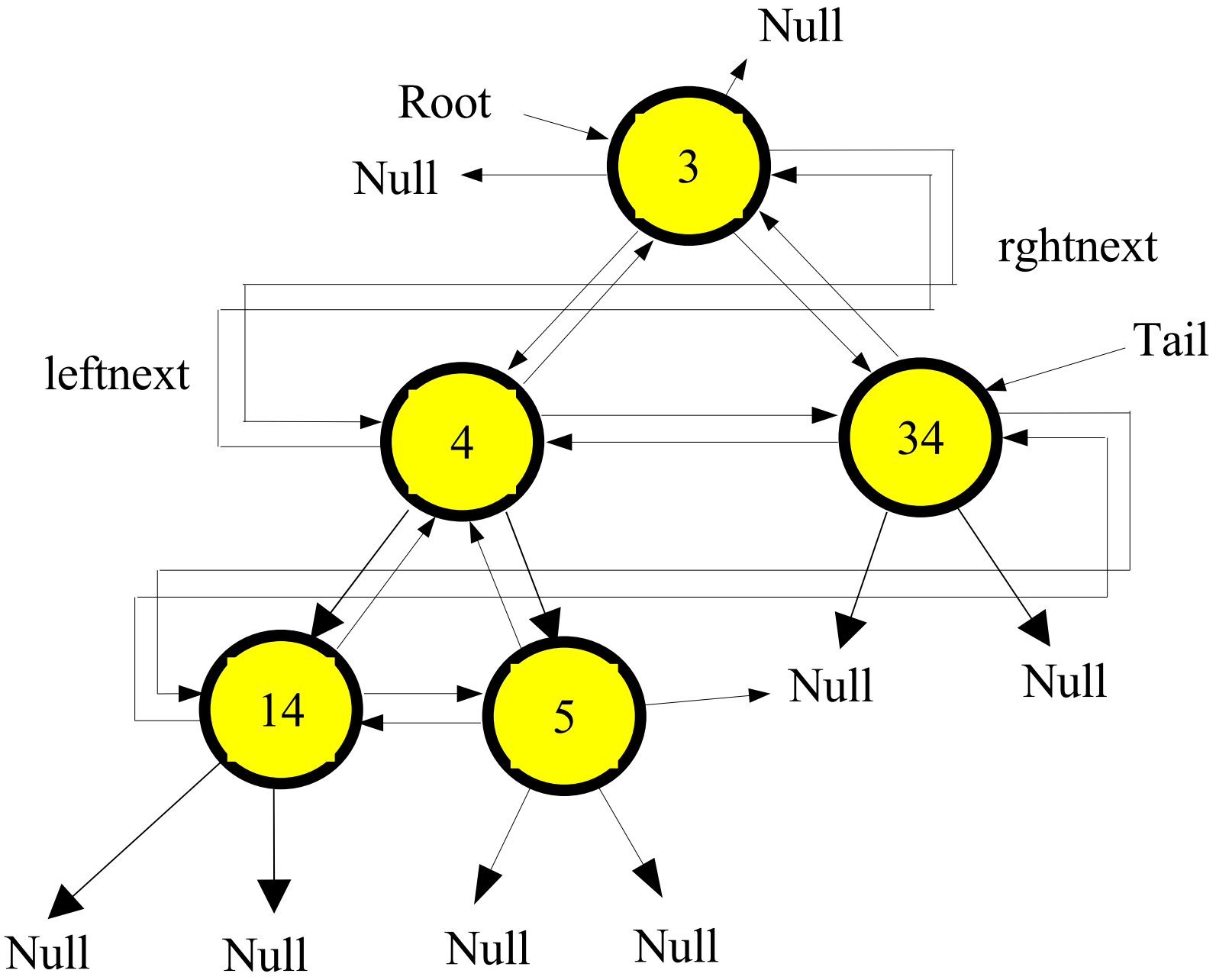

Null
Root
3
Null
rghtnext
Tail
leftnext
5
34
14
4
Null
Null
Null
Null
Null
Null

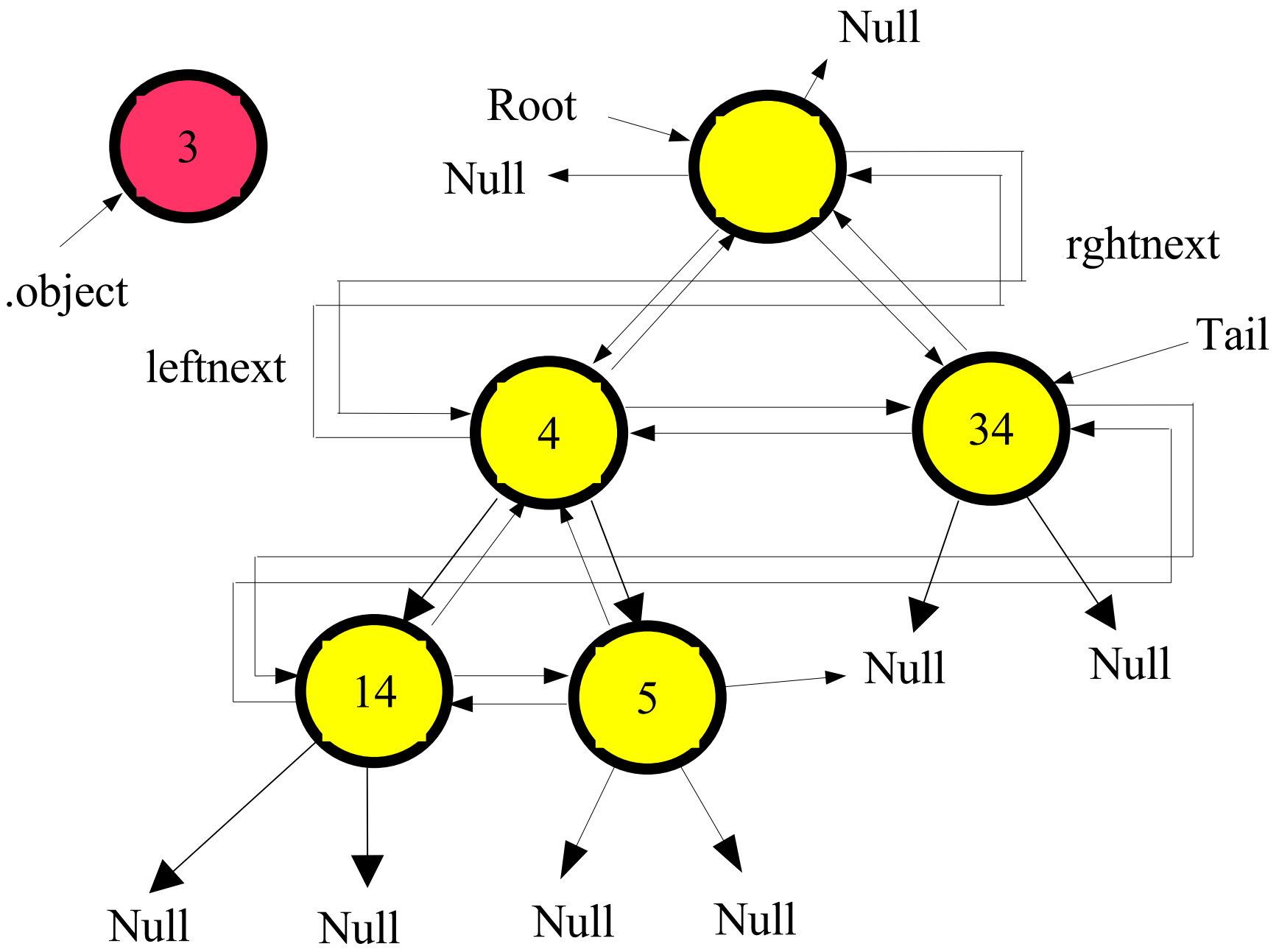

Null
Root
3
Null
rghtnext
Tail
leftnext
5
34
14
4
Null
Null
Null
Null
Null
Null

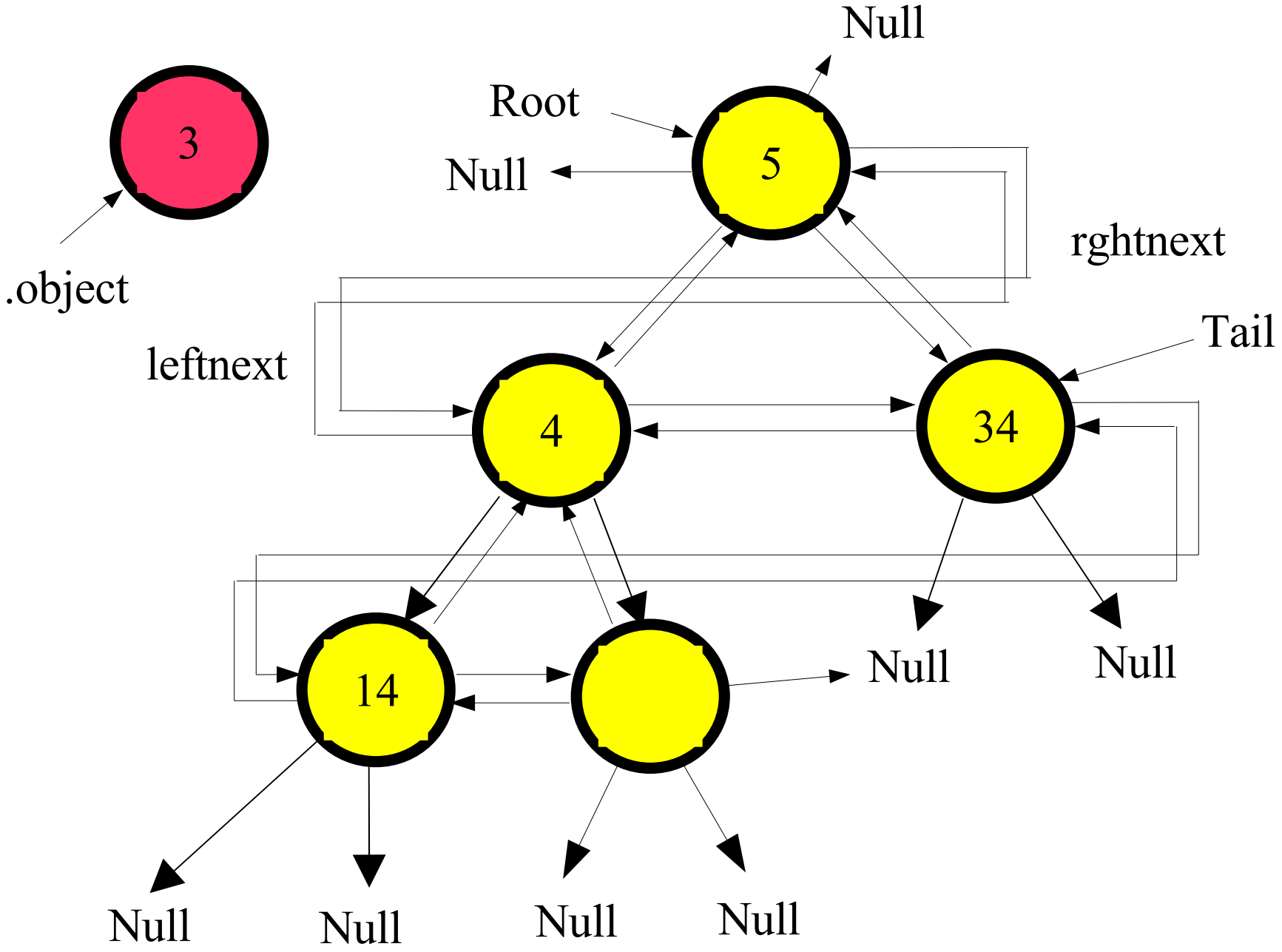

Null
Root
3
Null
rghtnext
Tail
leftnext
4
34
14
5
Null
Null
Null
Null
Null
Null

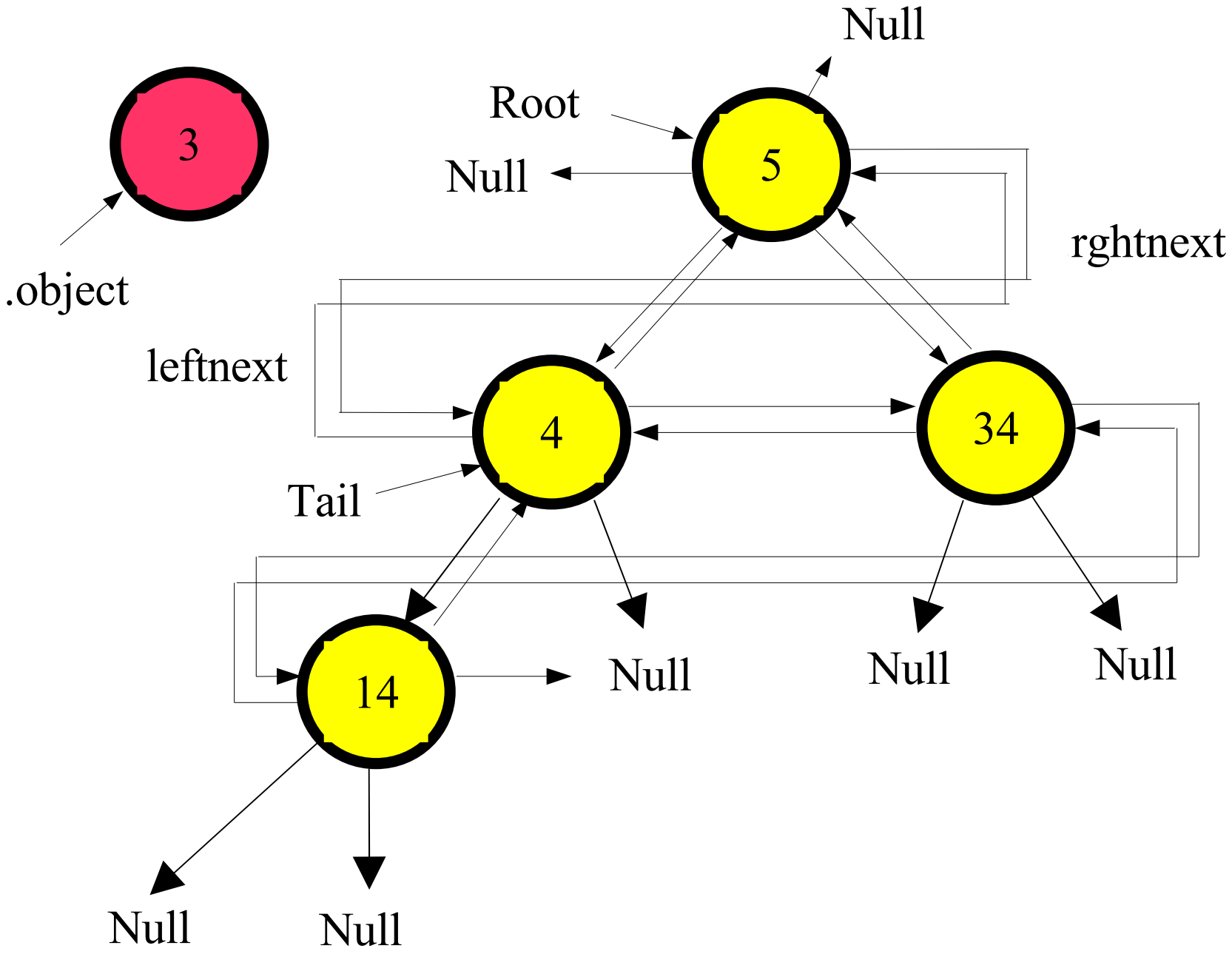

Null
Root
3
Null
rghtnext
leftnext
Tail
4
34
14
5
Null
Null
Null
Null
Null
Null

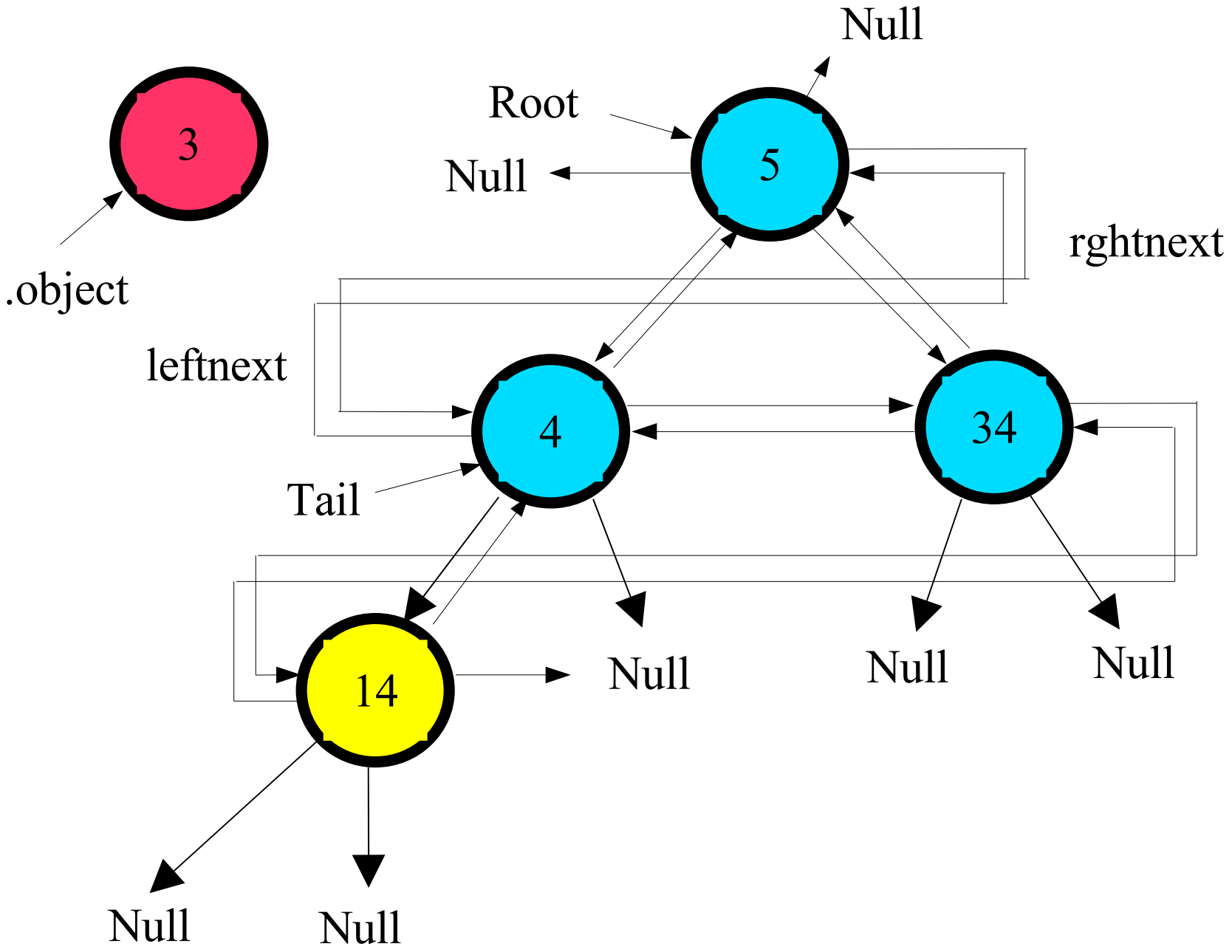
“Pop” an item off the top

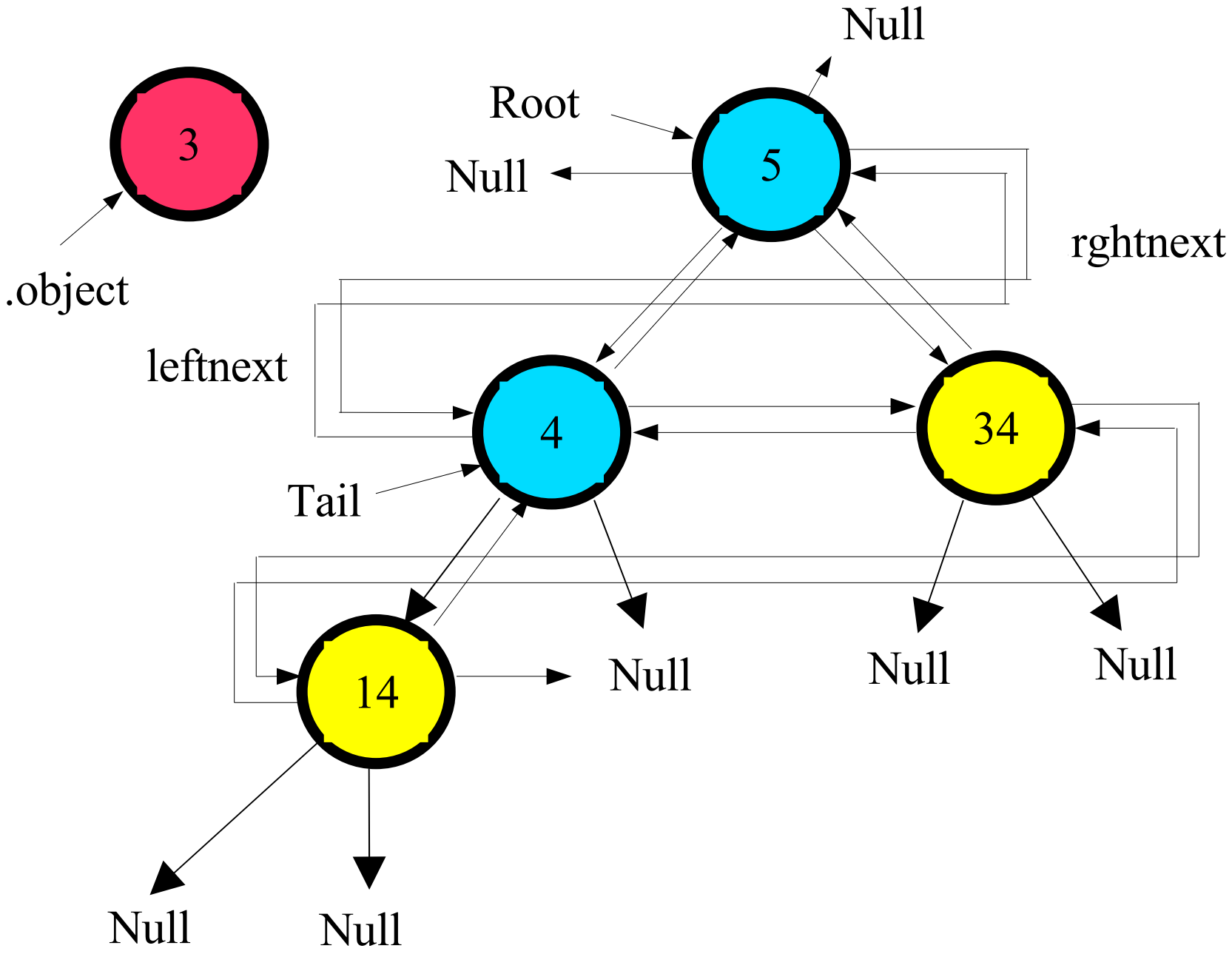

Null
Root
3
Null
rghtnext
Tail
leftnext
4
34
14
5
Null
Null
Null
Null
Null
Null

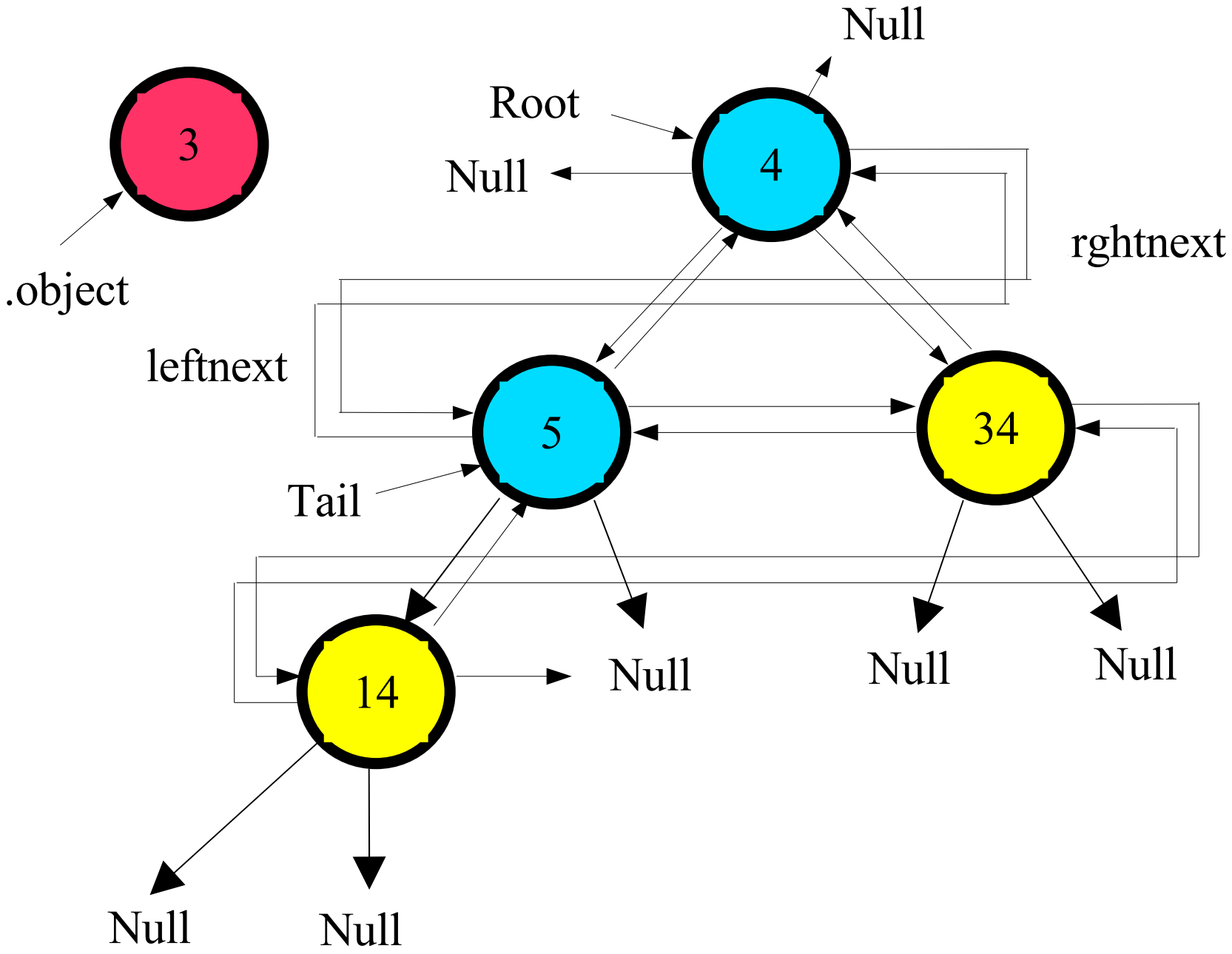

Null
Root
3
Null
rghtnext
.object
Tail
leftnext
4
34
14
5
Null
Null
Null
Null
Null
Null

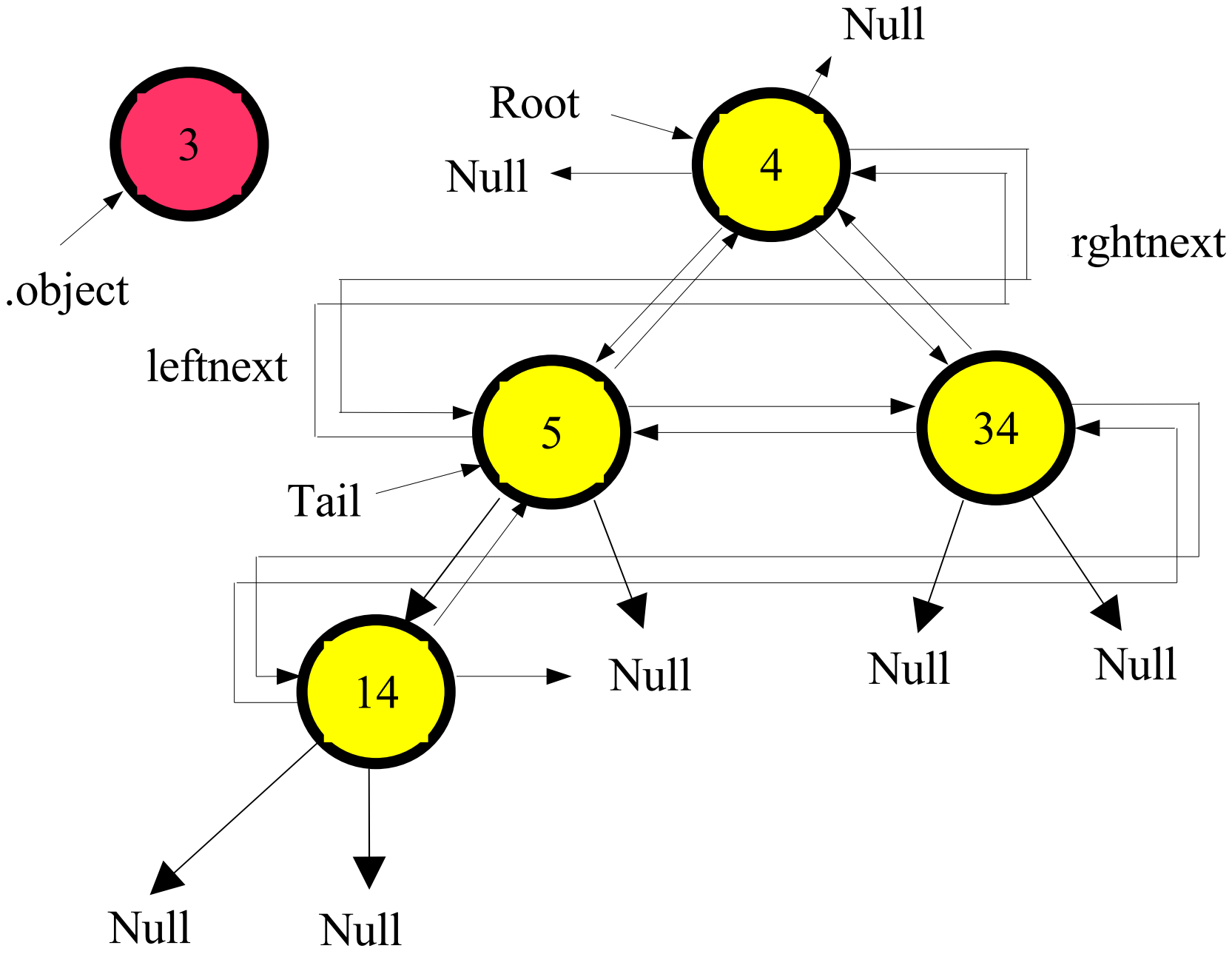

Null
Root
3
5
Null
rghtnext
.object
Tail
leftnext
4
34
14
Null
Null
Null
Null
Null
Null

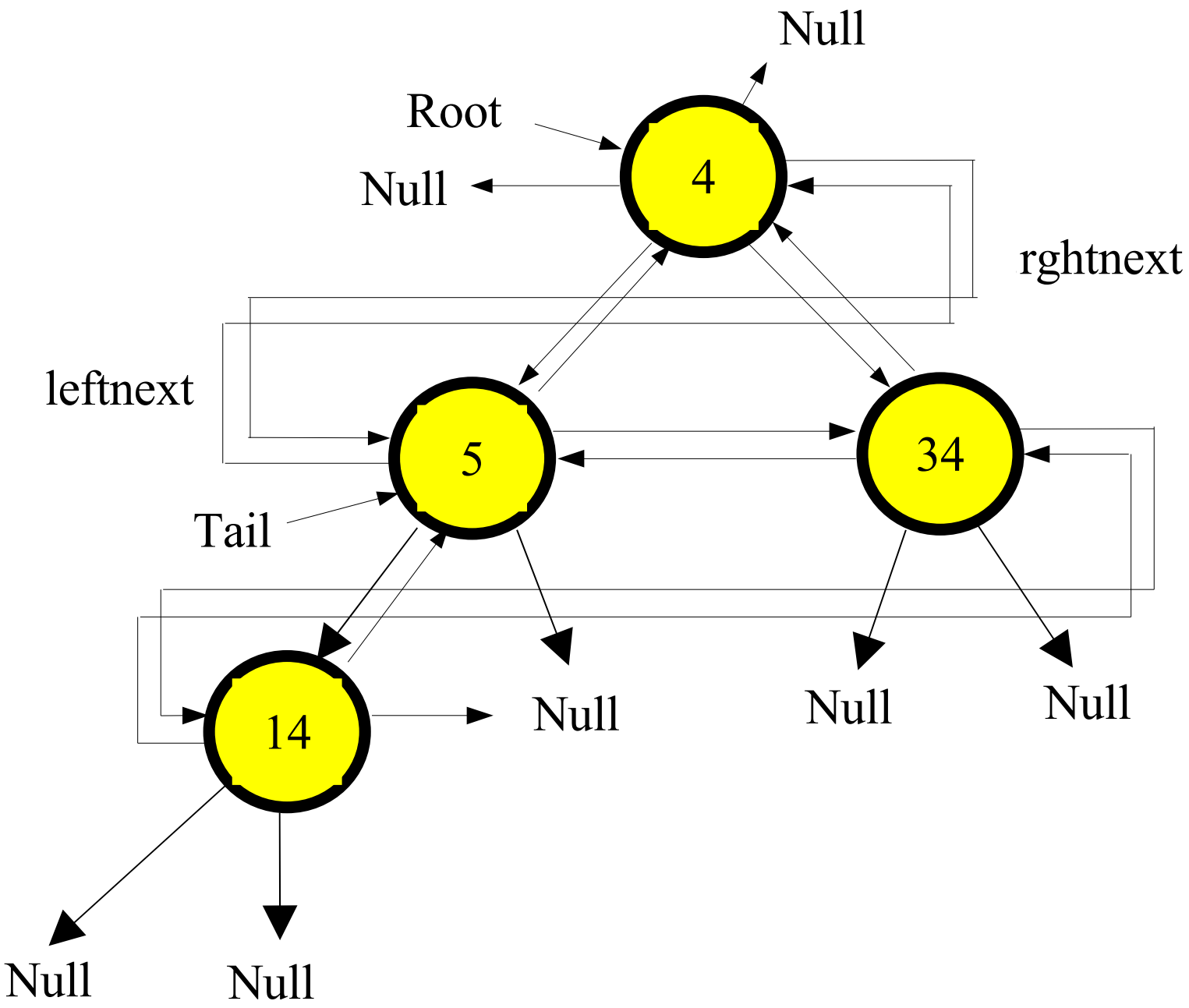

Null
Root
3
5
Null
rghtnext
.object
leftnext
4
34
Tail
14
Null
Null
Null
Null
Null


Null
Root
3
5
Null
rghtnext
.object
leftnext
4
34
Tail
14
Null
Null
Null
Null
Null

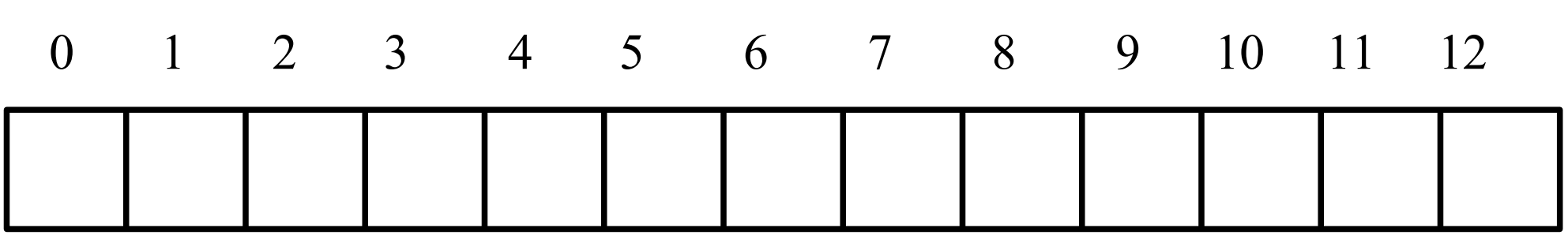

Null
Root
3
5
Null
rghtnext
.object
leftnext
4
34
Tail
14
Null
Null
Null
Null
Null


Null
Root
3
4
Null
rghtnext
.object
leftnext
5
34
Tail
14
Null
Null
Null
Null
Null

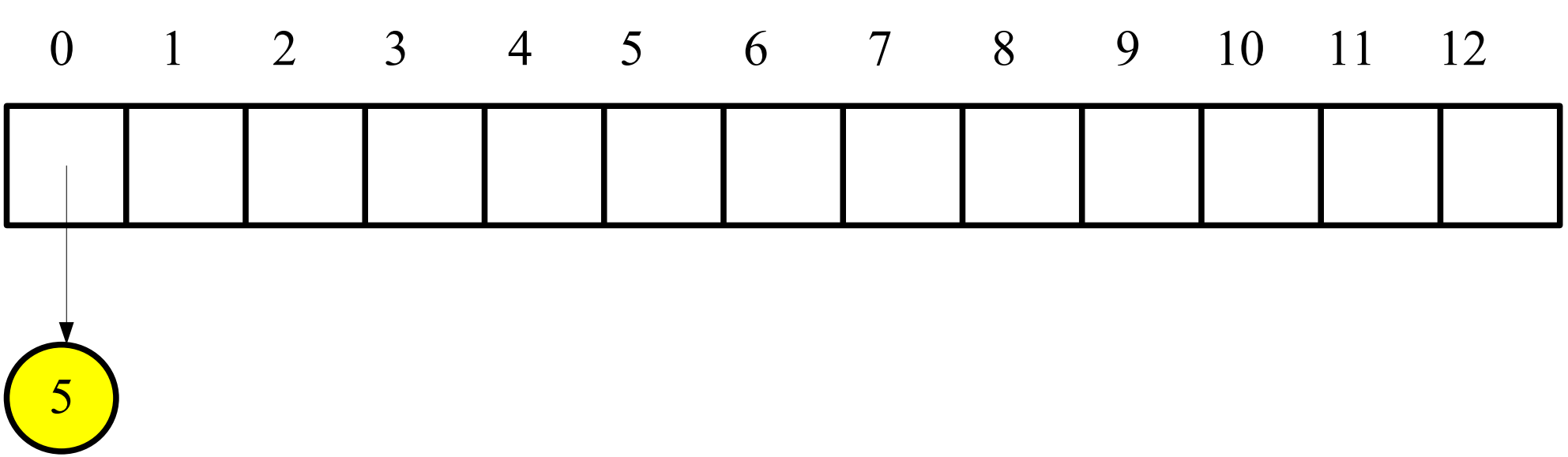

Null
Root
3
4
Null
rghtnext
.object
leftnext
5
34
Tail
14
Null
Null
Null
Null
Null


Null
Root
4
Null
rghtnext
leftnext
5
34
Tail
14
Null
Null
Null
Null
Null

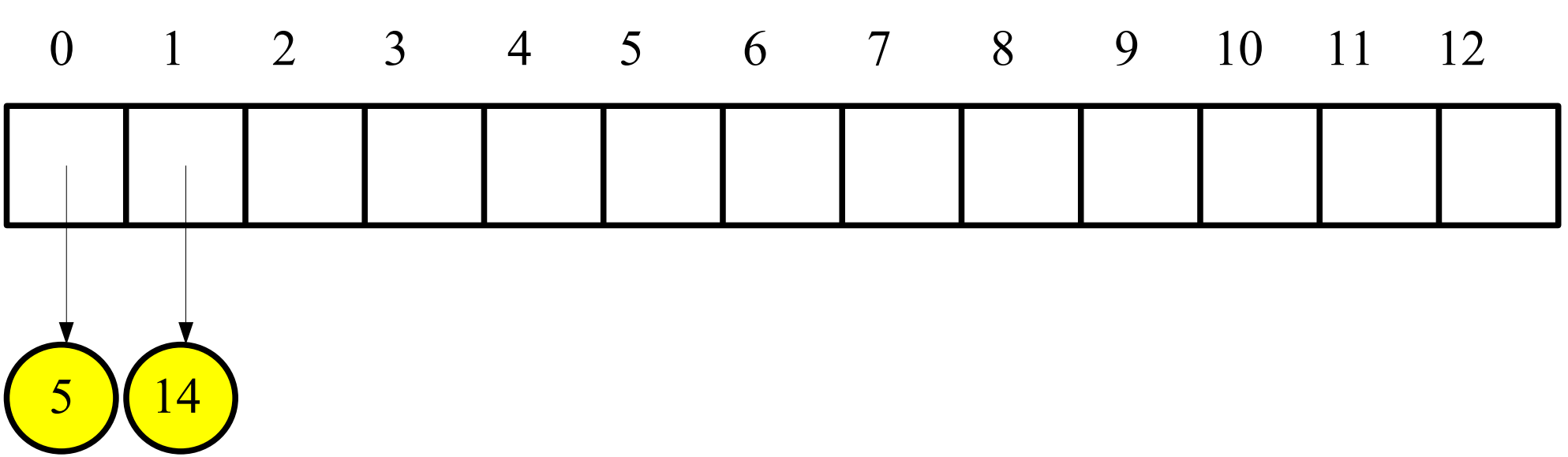
Implementation using an array of pointers



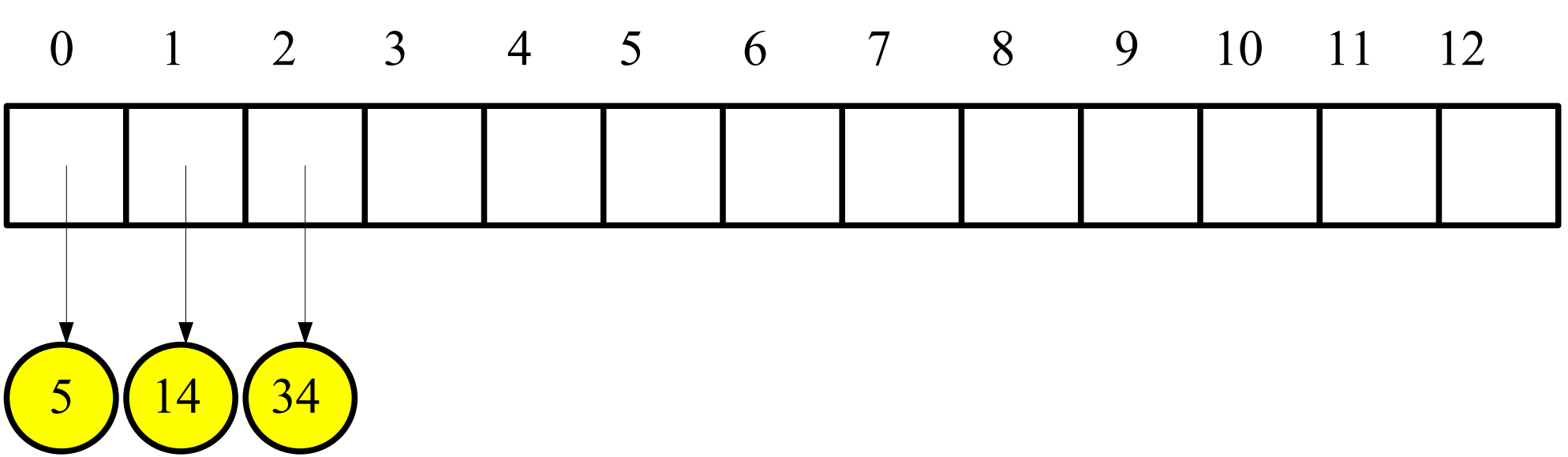
root = 0
tail = 0



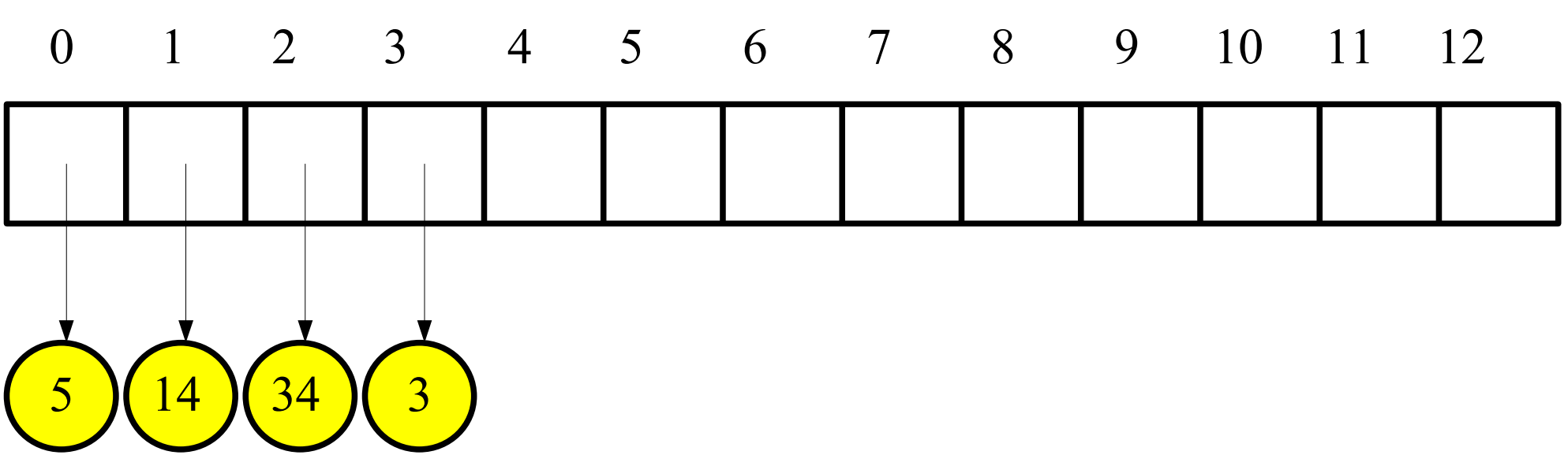
root = 0
tail = 1



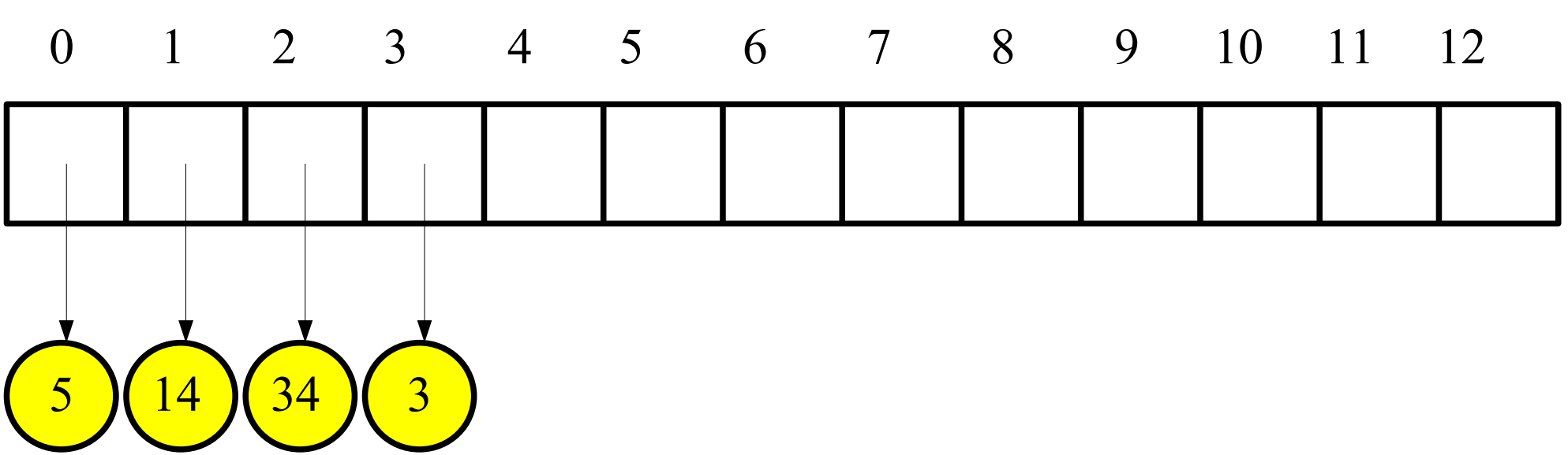
root = 0
tail = 2



root = 0
tail = 3



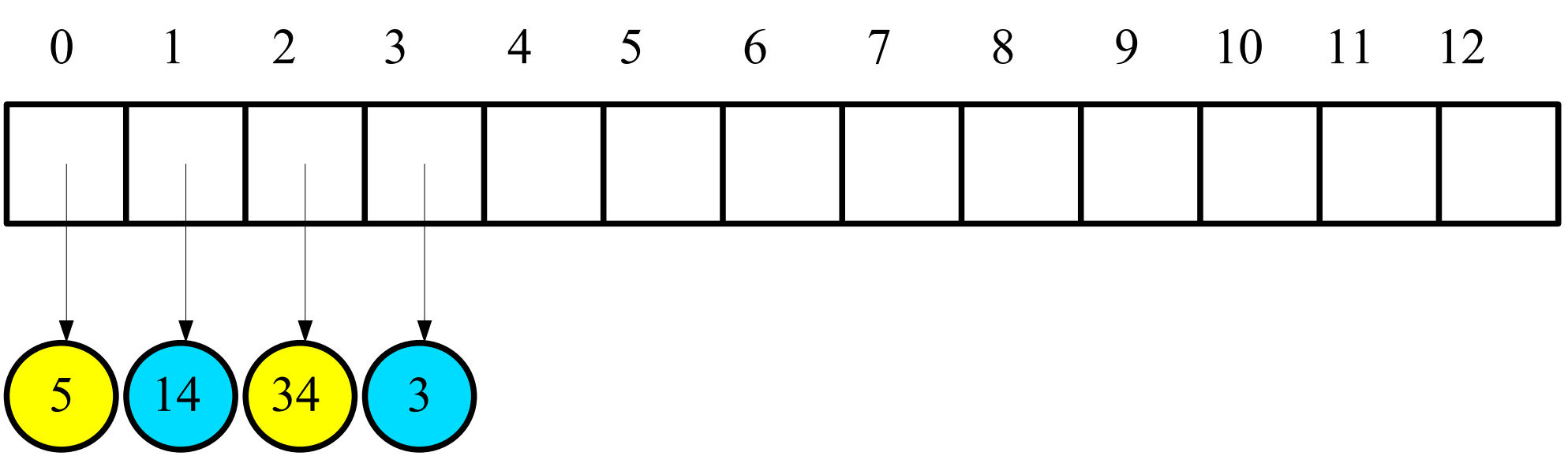
root = 0
tail = 4



root = 0
tail = 4

Parent of “3” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(3-1)/2 = 1

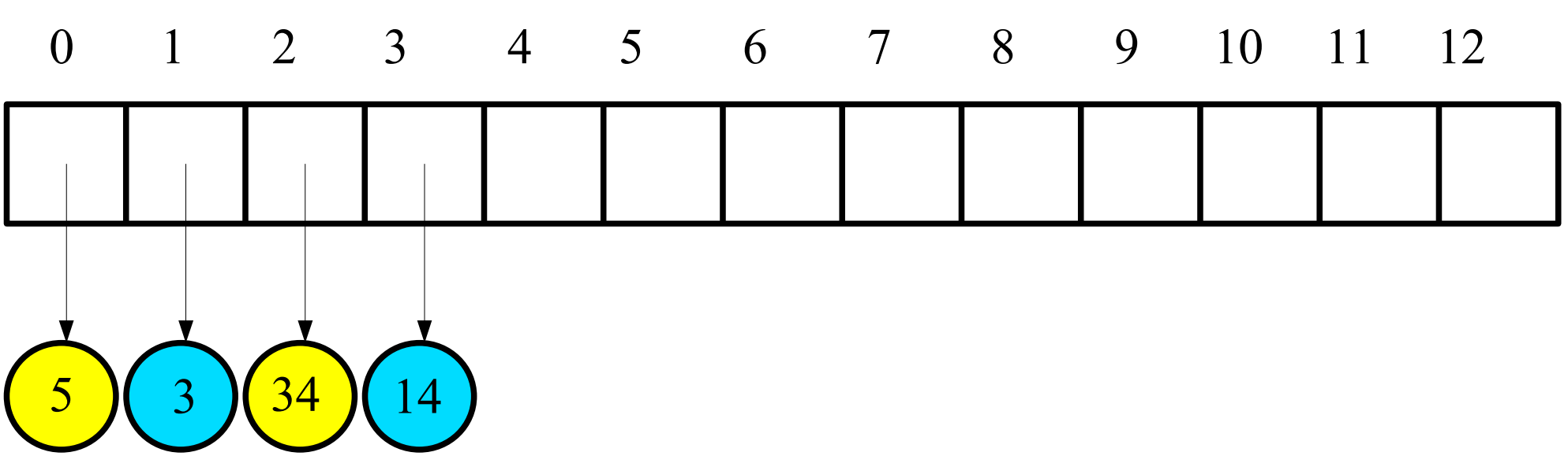


root = 0
tail = 4

Parent of “3” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(3-1)/2 = 1

Compare parent with child.

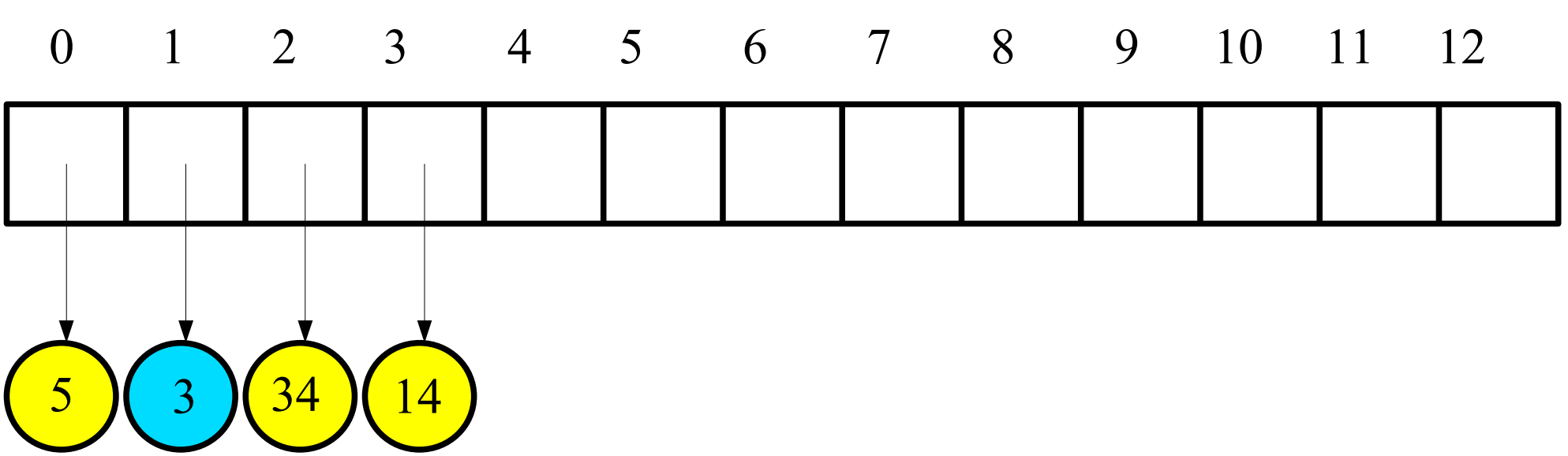


root = 0
tail = 4

Parent of “3” node is determined by subtracting the index of that node by 1 and dividing by 2.
In this case its

(3-1)/2 = 1

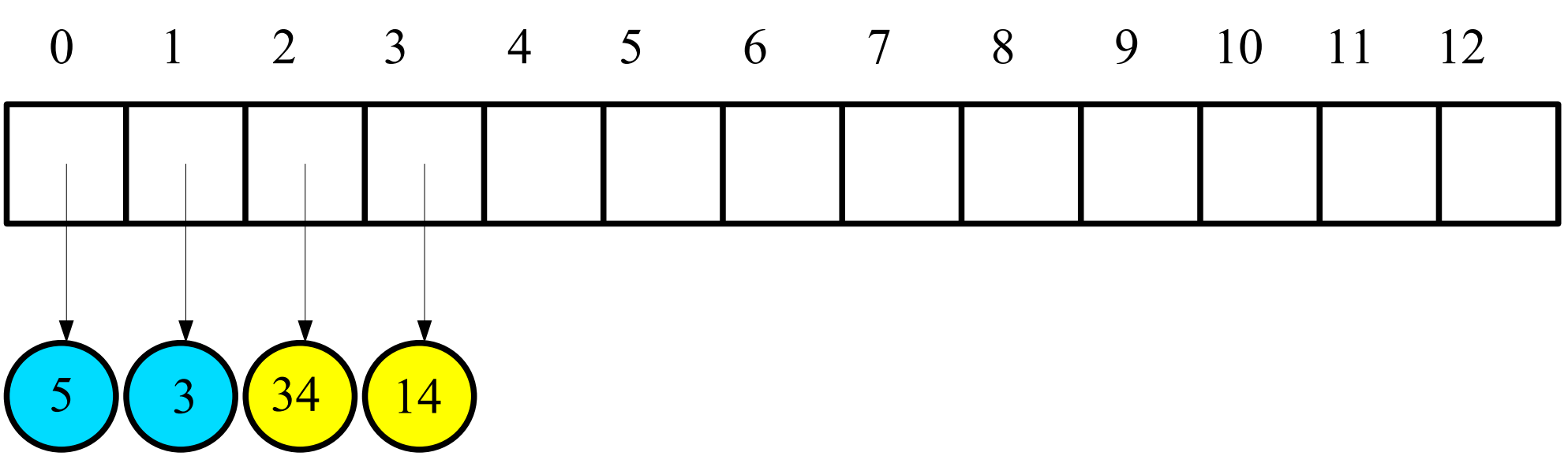
Compare parent with child.
Move the child up by one level.



root = 0
tail = 4

Parent of “3” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(1-1)/2 = 0

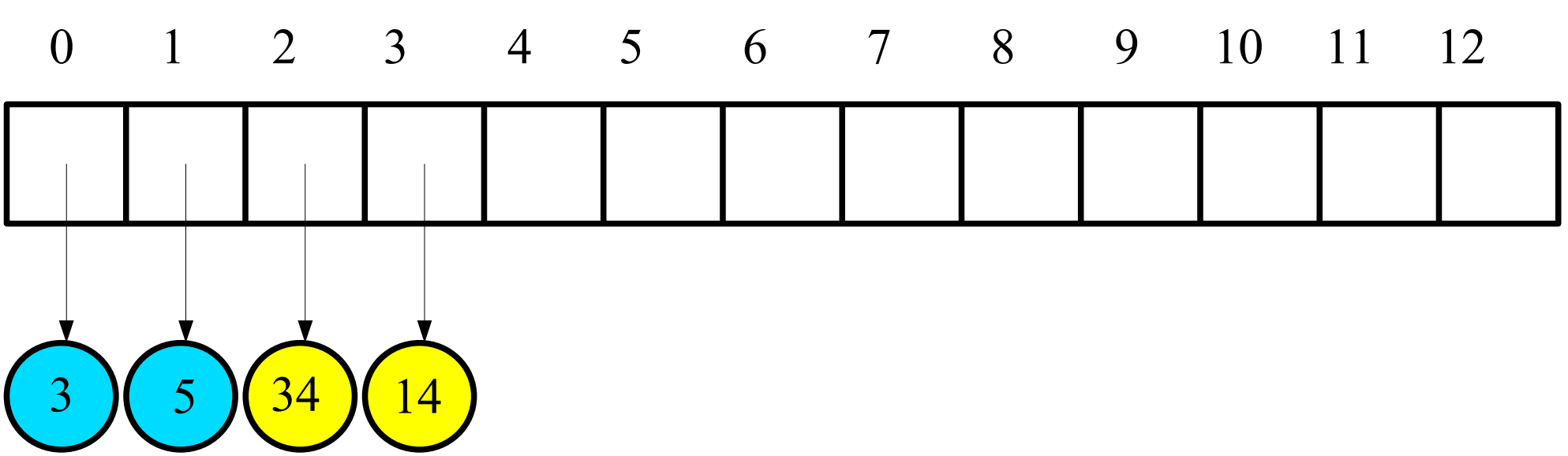


root = 0
tail = 4

Parent of “3” node is determined by subtracting the index of that node by 1 and dividing by 2.
In this case its

(1-1)/2 = 0

Compare parent and child.

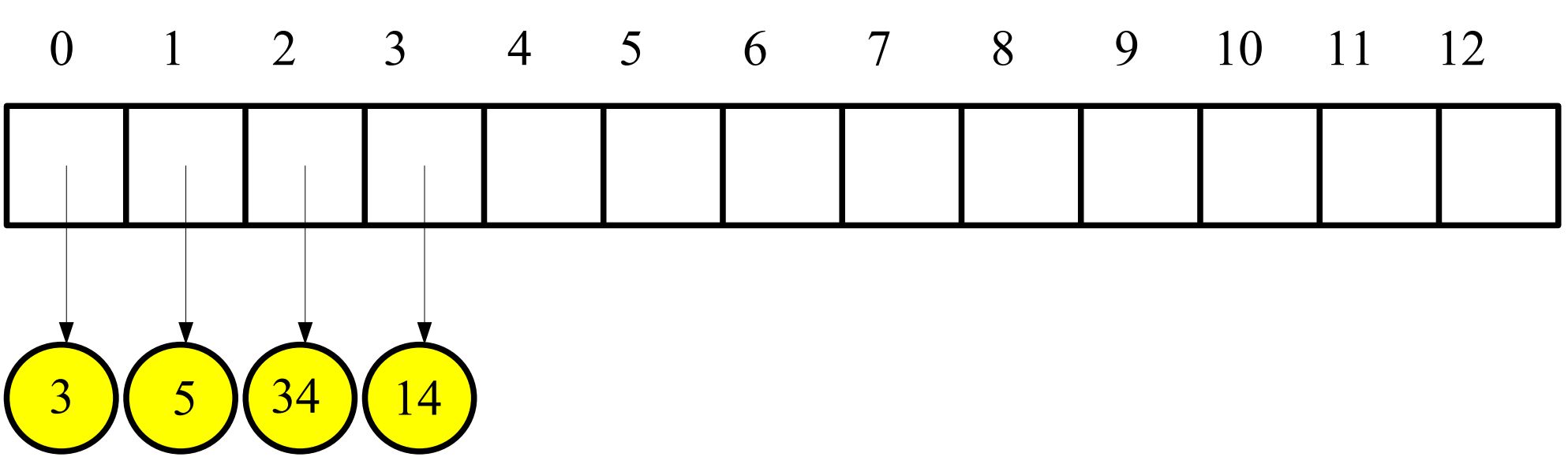


root = 0
tail = 4

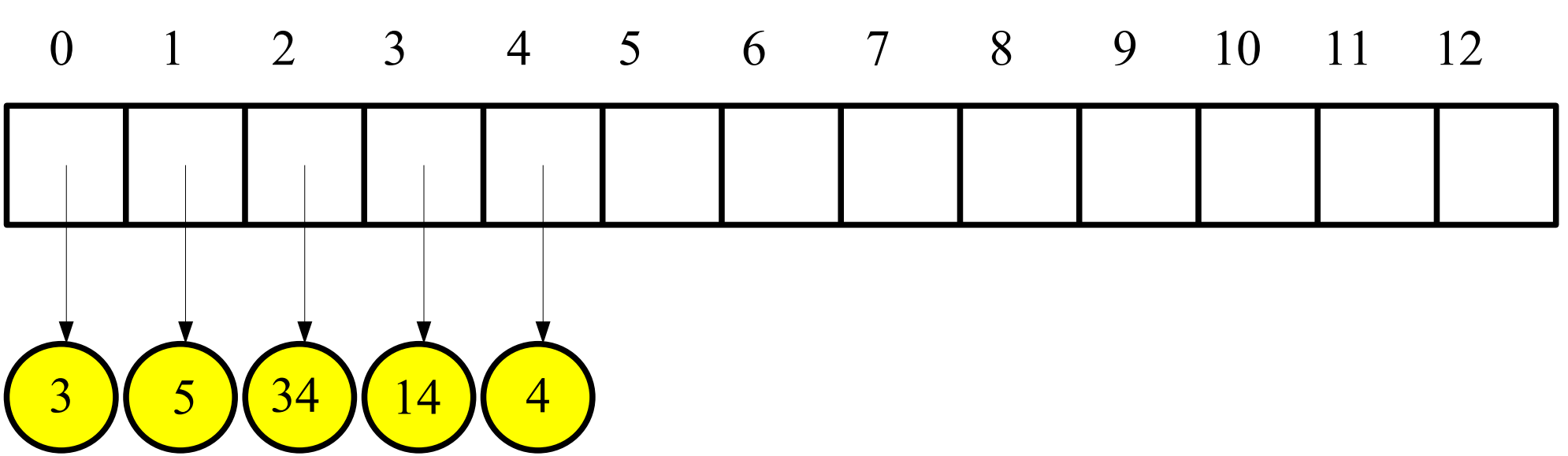
Parent of “3” node is determined by subtracting the index of that node by 1 and dividing by 2.
In this case its

(1-1)/2 = 0

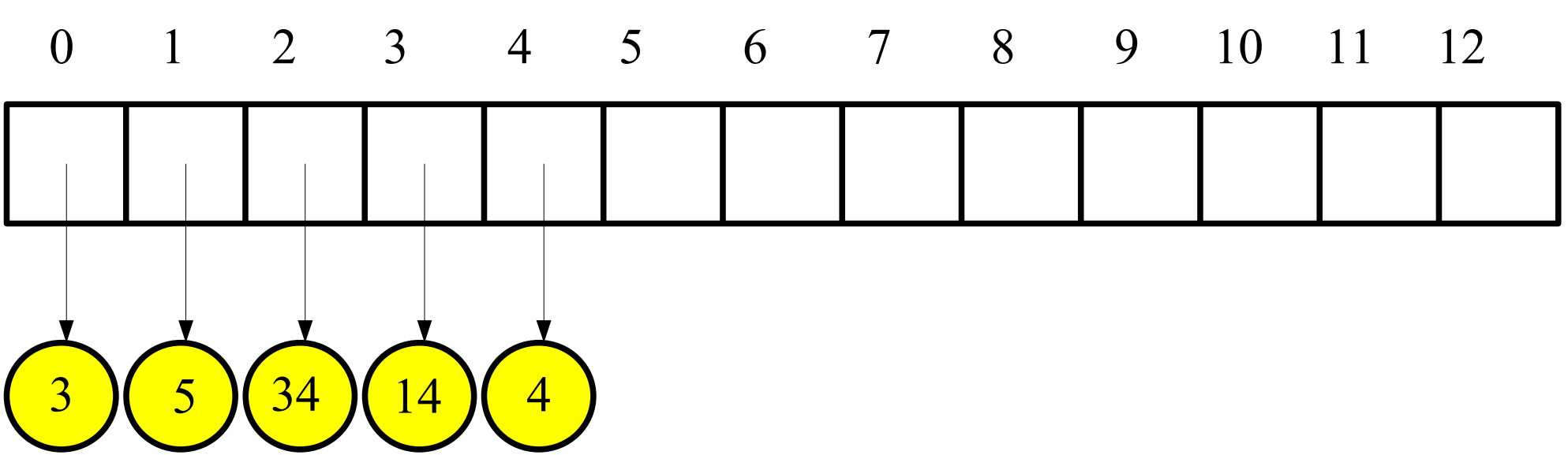
Compare parent and child.
Move child up one level.



root = 0
tail = 4



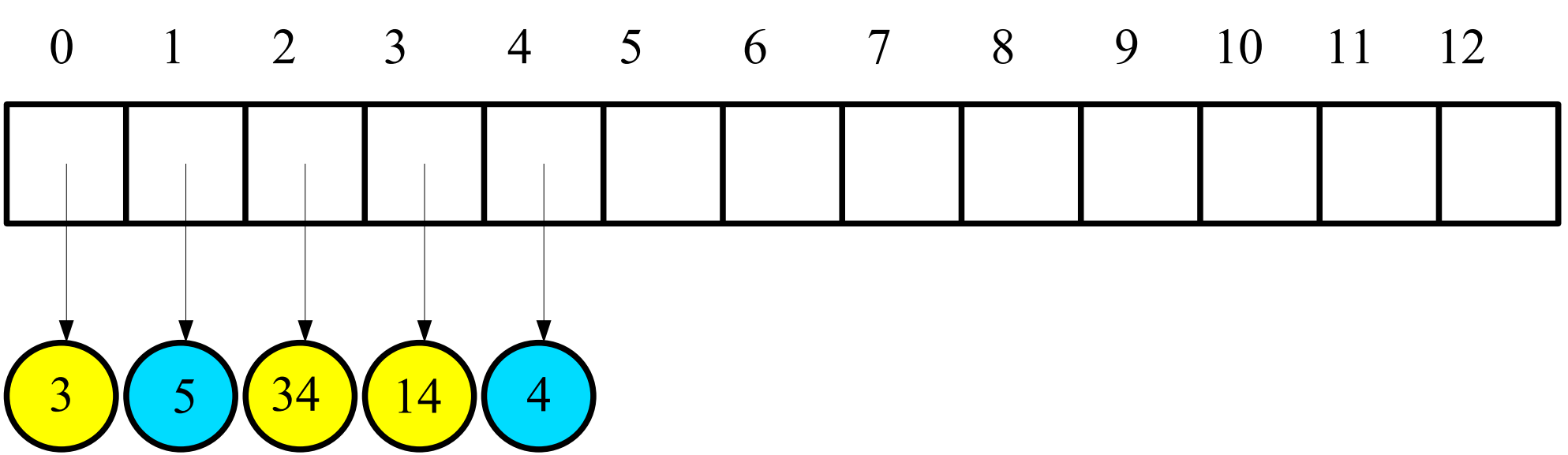
root = 0
tail = 5



root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(4-1)/2 = 1

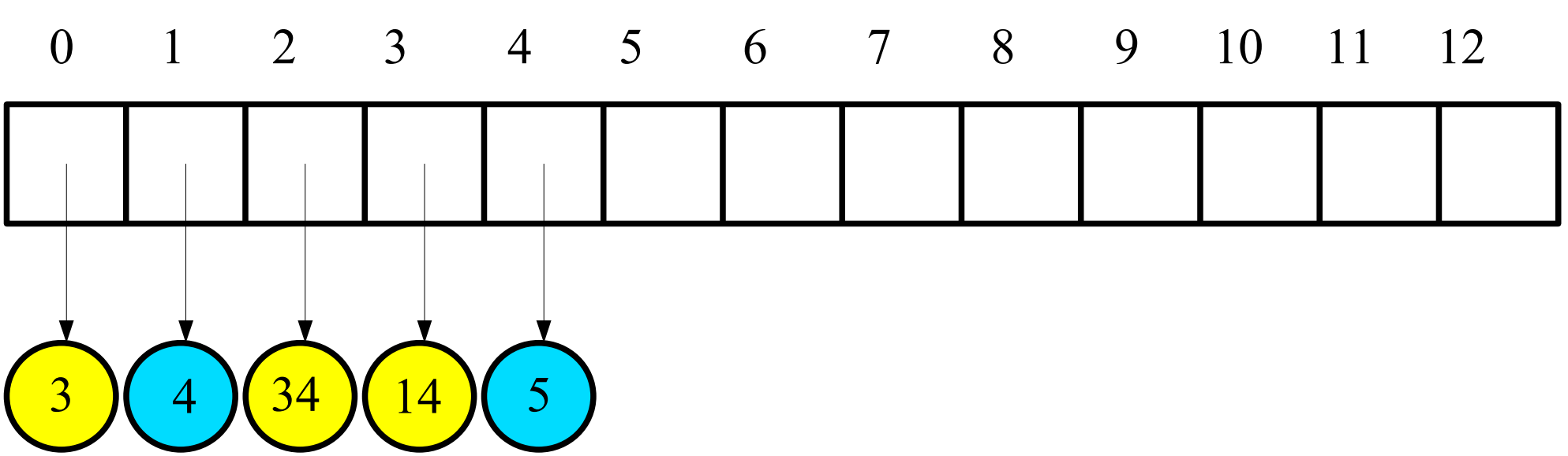


root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(4-1)/2 = 1

Compare parent and child.

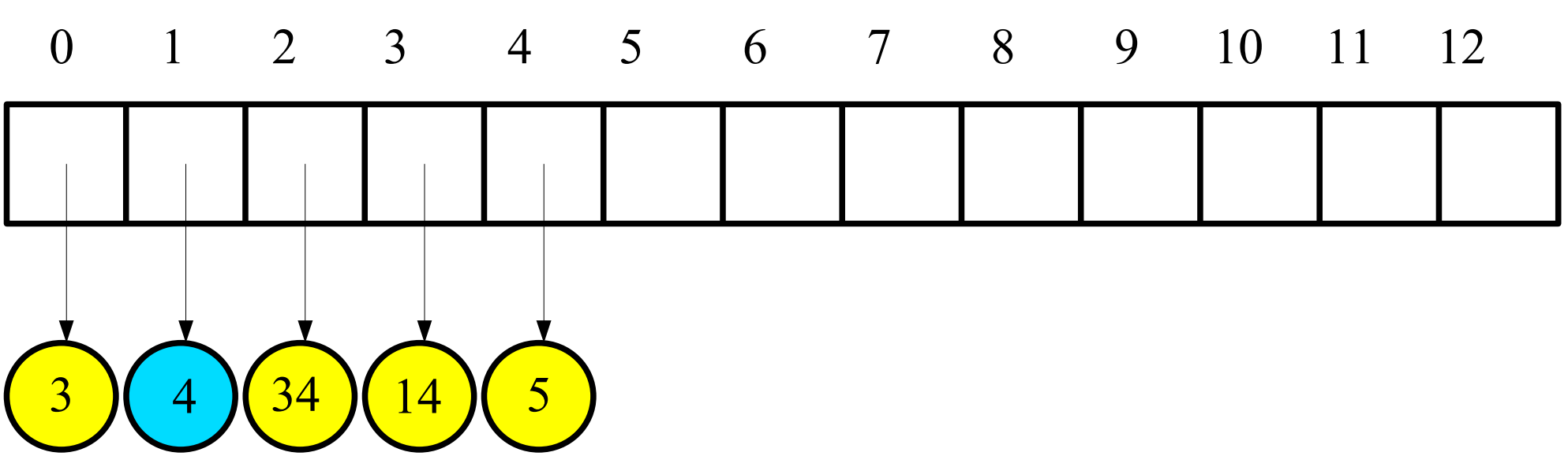


root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2.
In this case its

(4-1)/2 = 1

Compare parent and child.
Move child up one level.

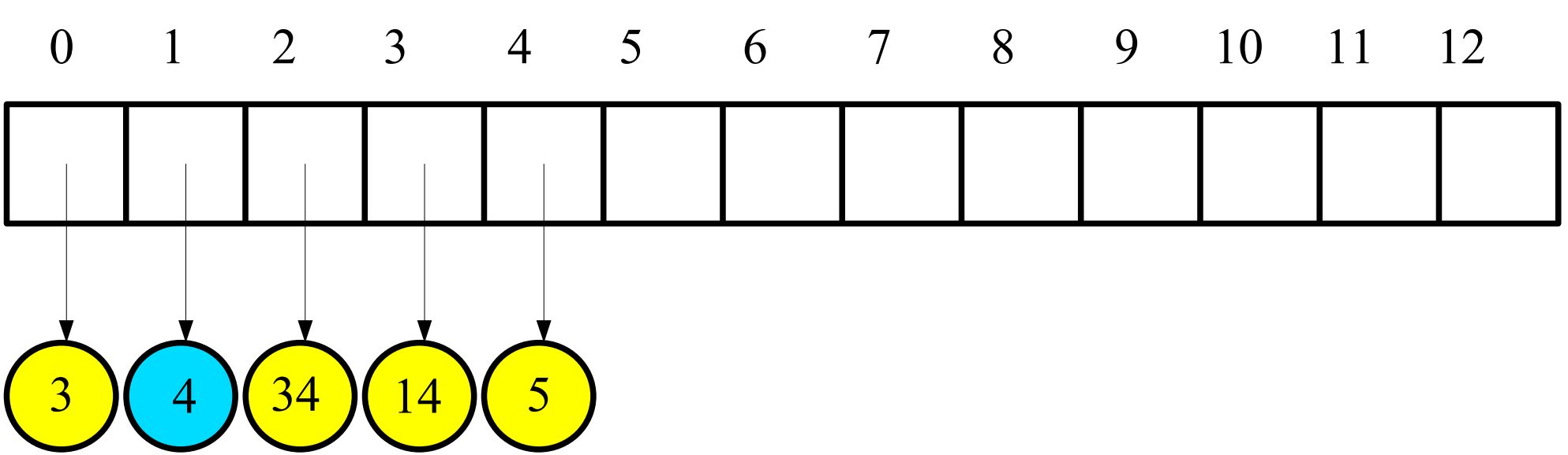


root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2.
In this case its

(4-1)/2 = 1

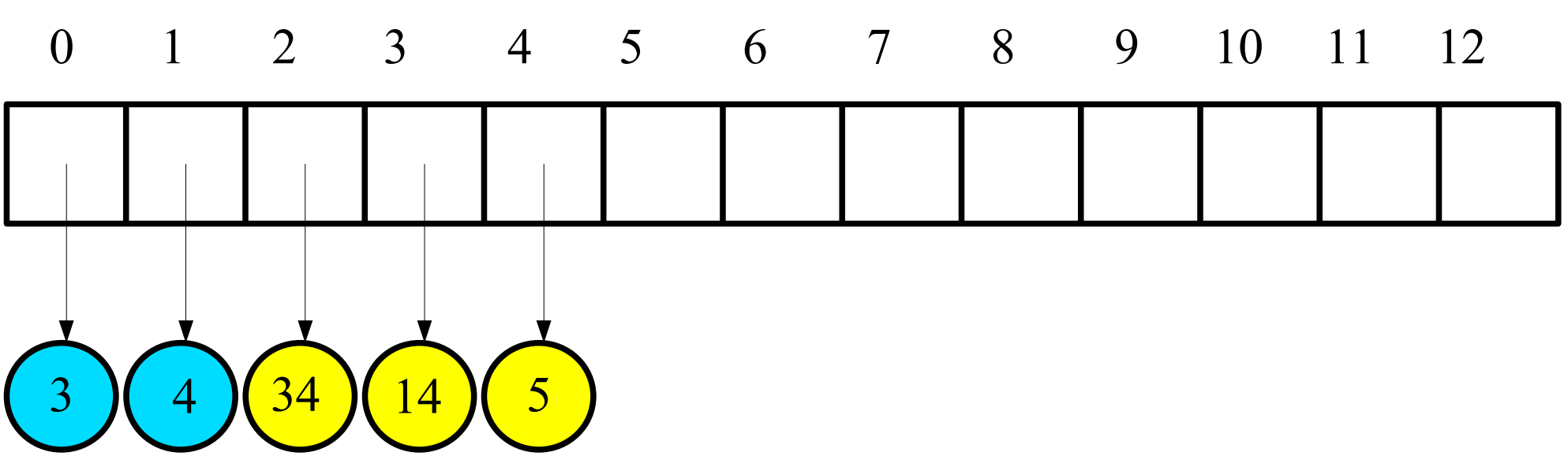
Compare parent and child.
Move child up one level.



root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(1-1)/2 = 0

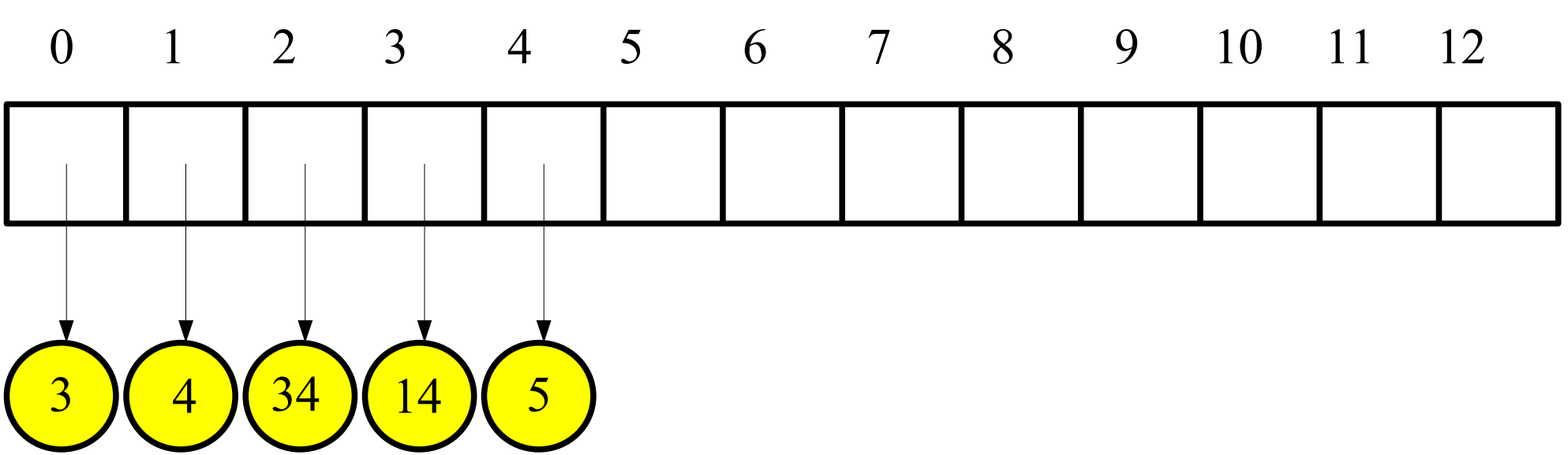


root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2. In this case its

(1-1)/2 = 0

Compare parent and child.



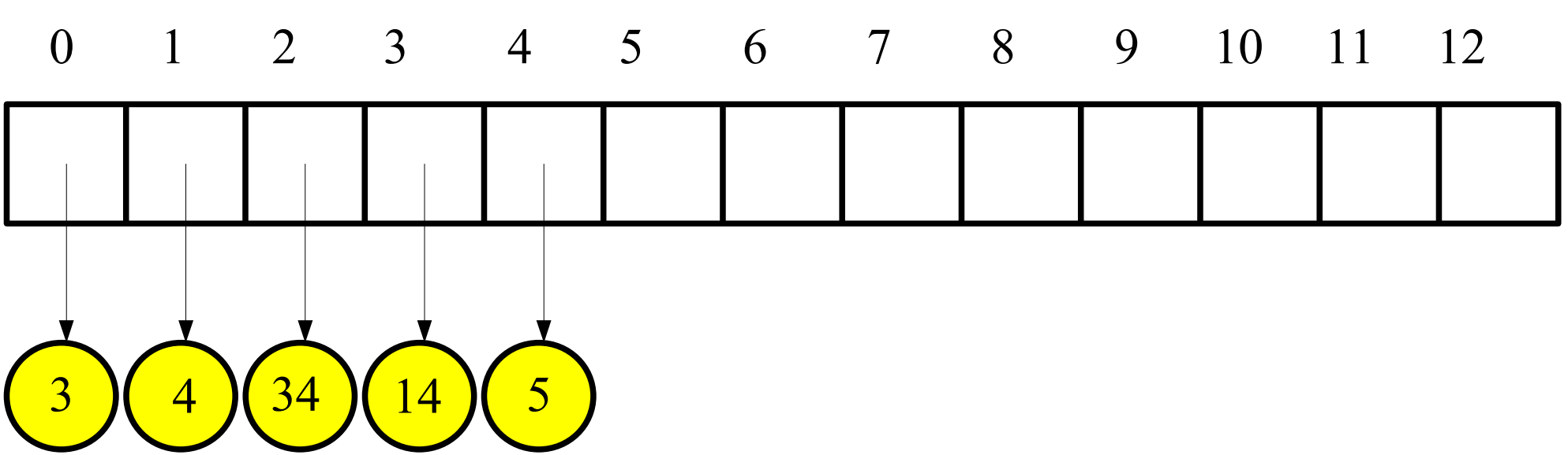
root = 0
tail = 5

Parent of “4” node is determined by subtracting the index of that node by 1 and dividing by 2.
In this case its

(1-1)/2 = 0

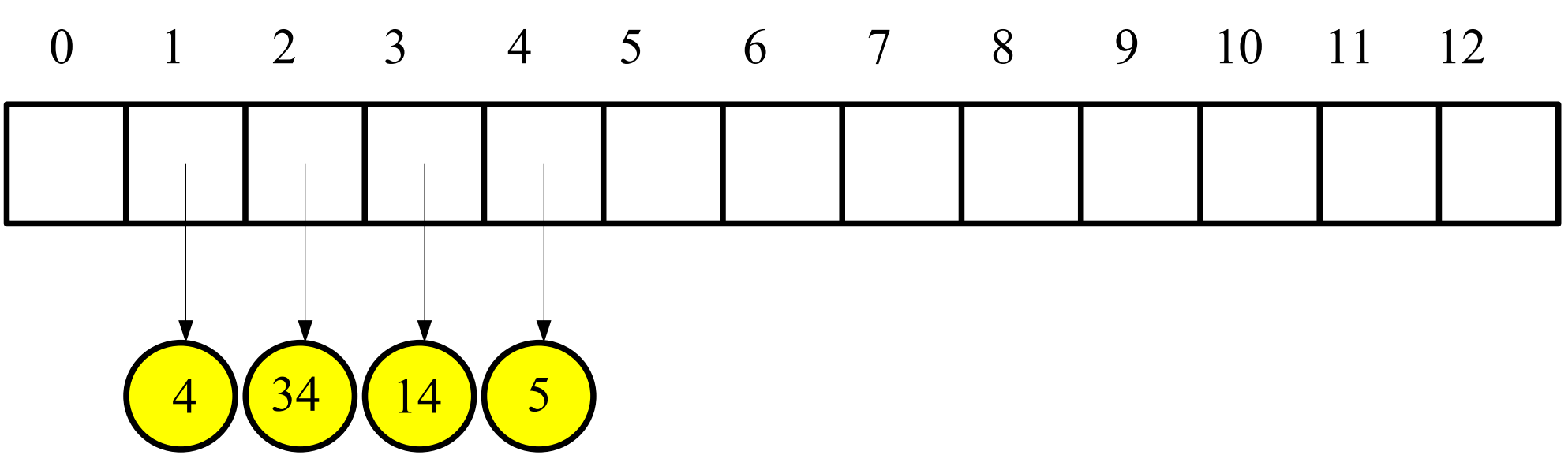
Compare parent and child.
Leave child alone.

Now “pop” the “lowest” valued object



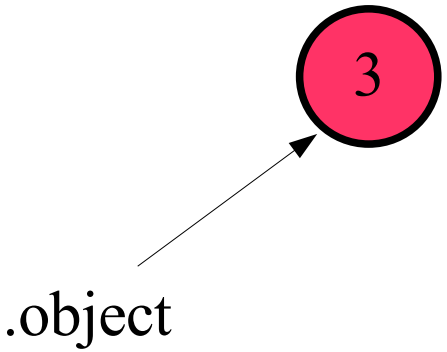
root = 0
tail = 5

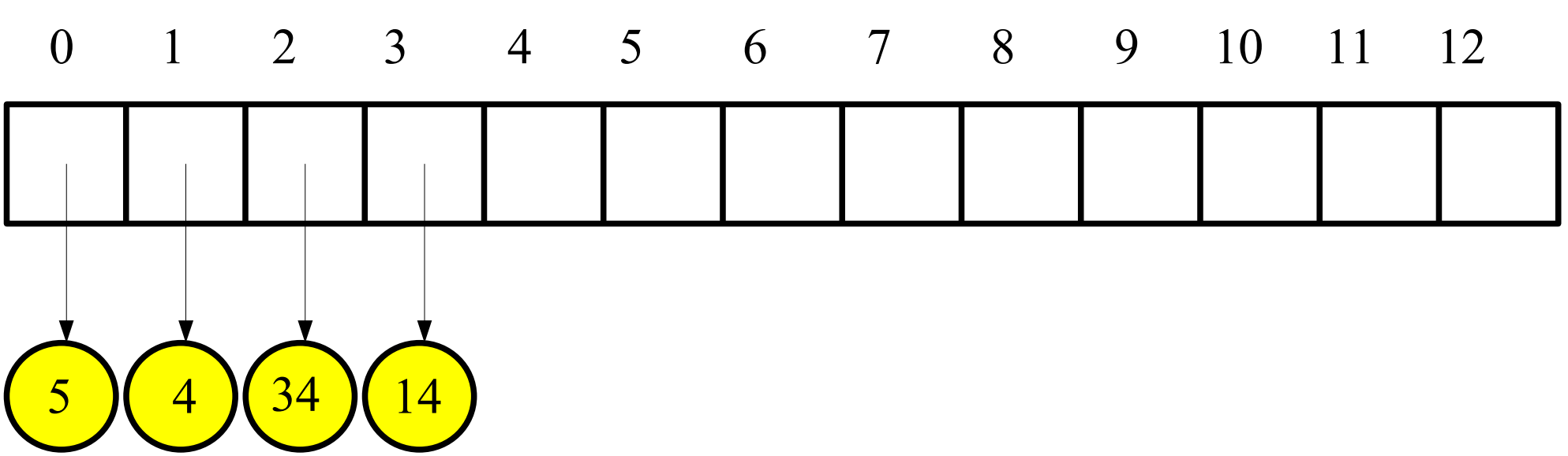
Now to “pop” the root node



root = 0
tail = 5

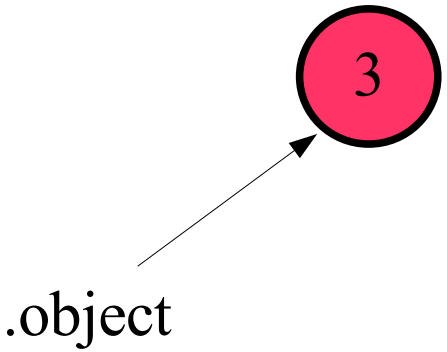
Now to “pop” the root node.

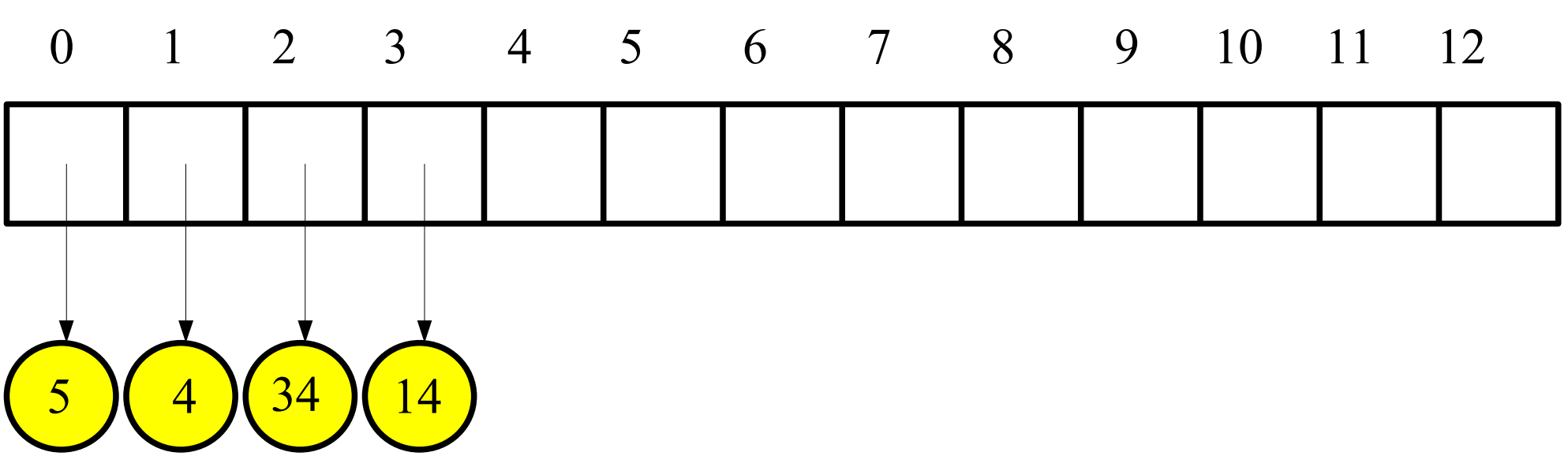




root = 0
tail = 5

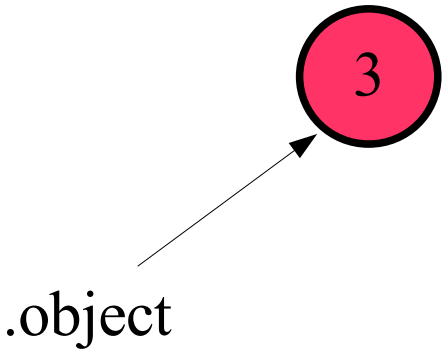
Now to “pop” the root node.
Move “tail” object to root.

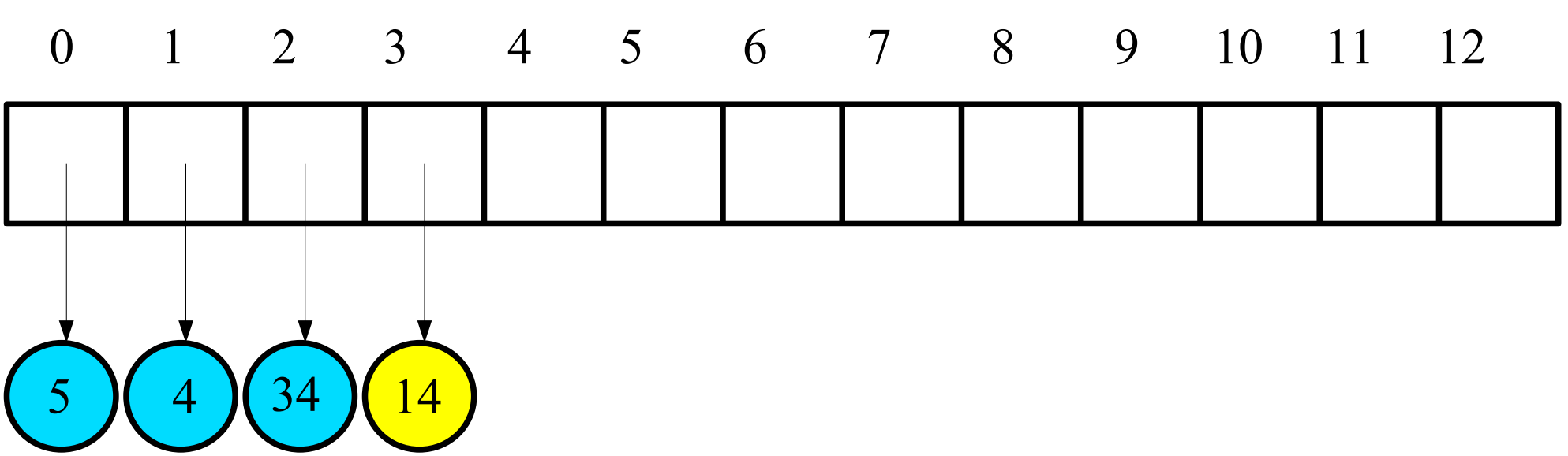




root = 0
tail = 4

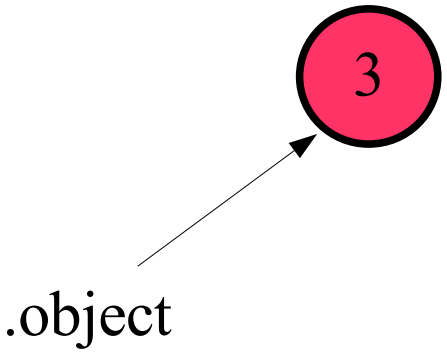
Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index

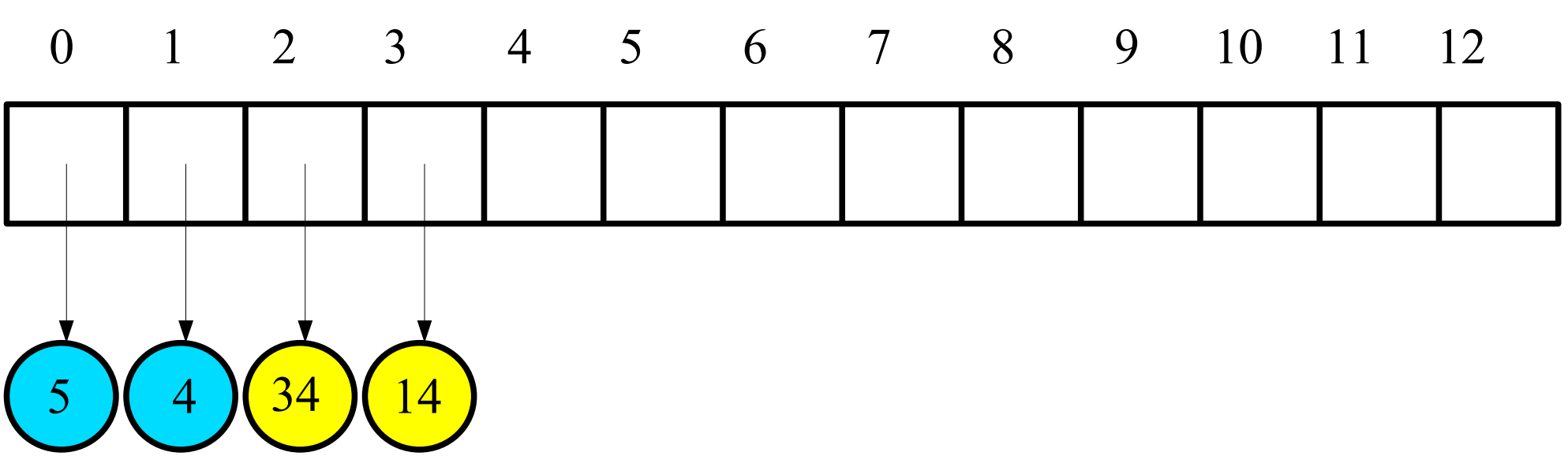




root = 0
tail = 4

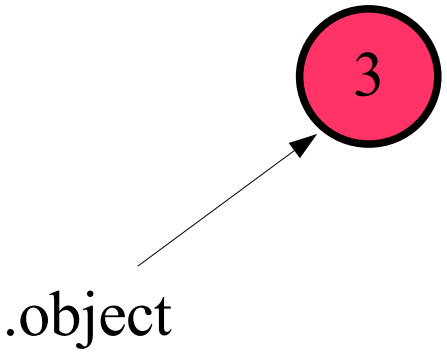
Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index
Compare root with two children

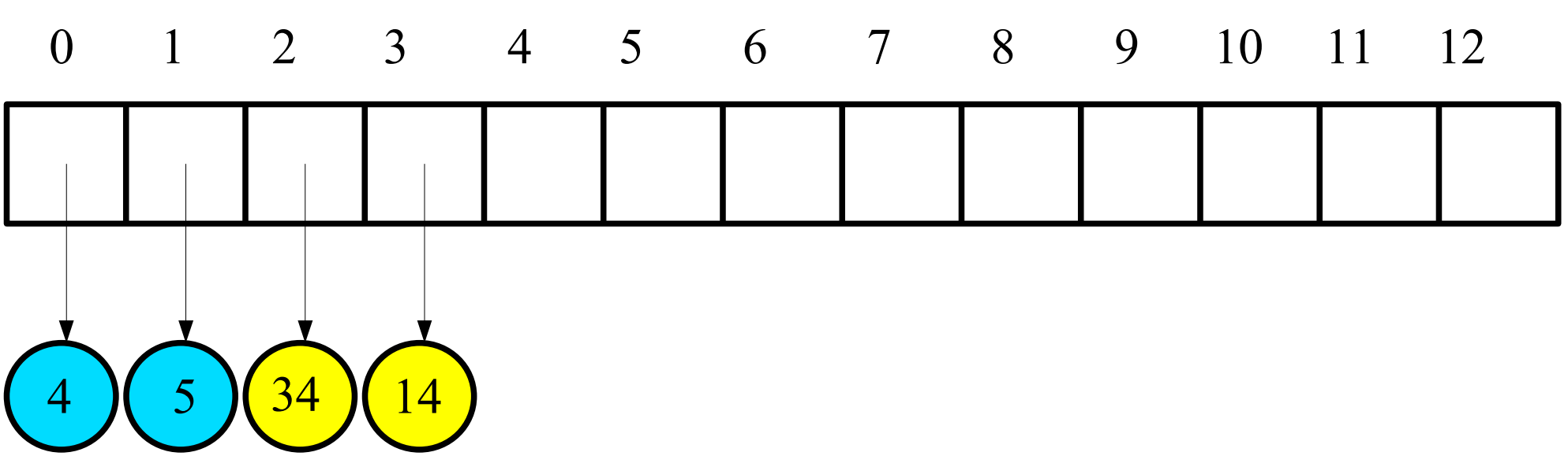




root = 0
tail = 4

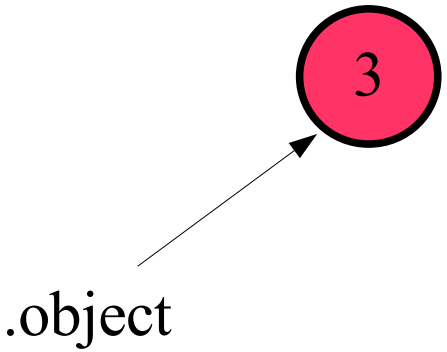
Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index
Compare root with two children
If greater, swap with smallest of the children

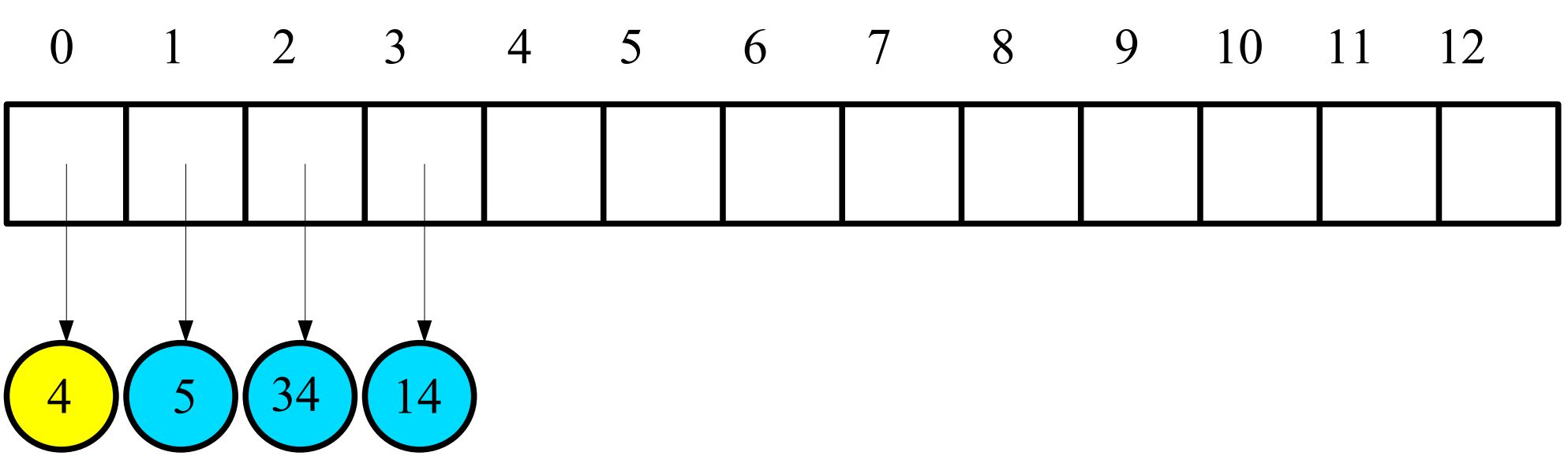




root = 0
tail = 4

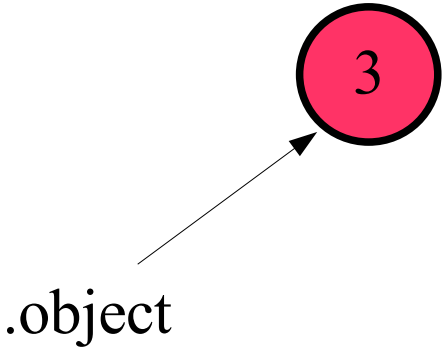
Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index
Compare root with two children
If greater, swap with smallest of the children

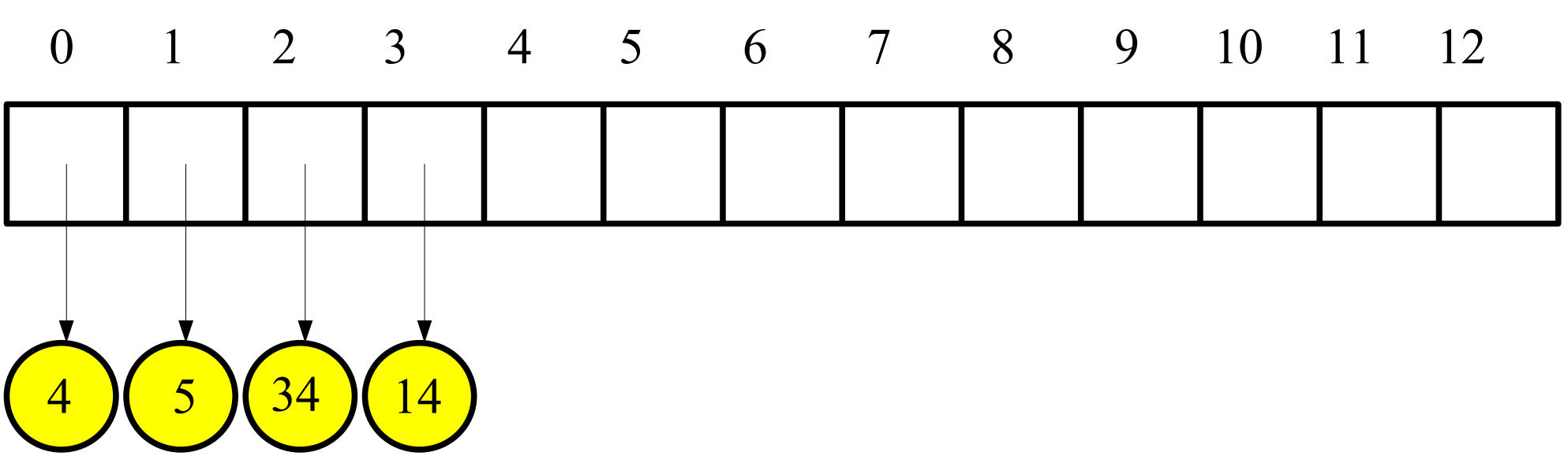




root = 0
tail = 4

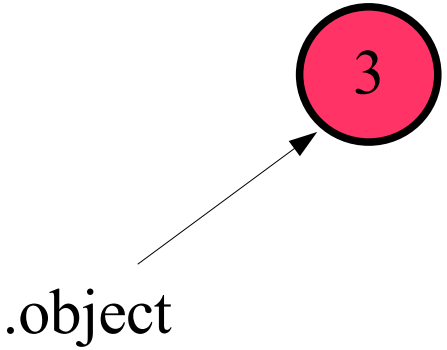
Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index
Compare root with two children
If greater, swap with smallest of the children
Compare swapped object with children
at indices 1*2+1=3 and 1*2+2=4

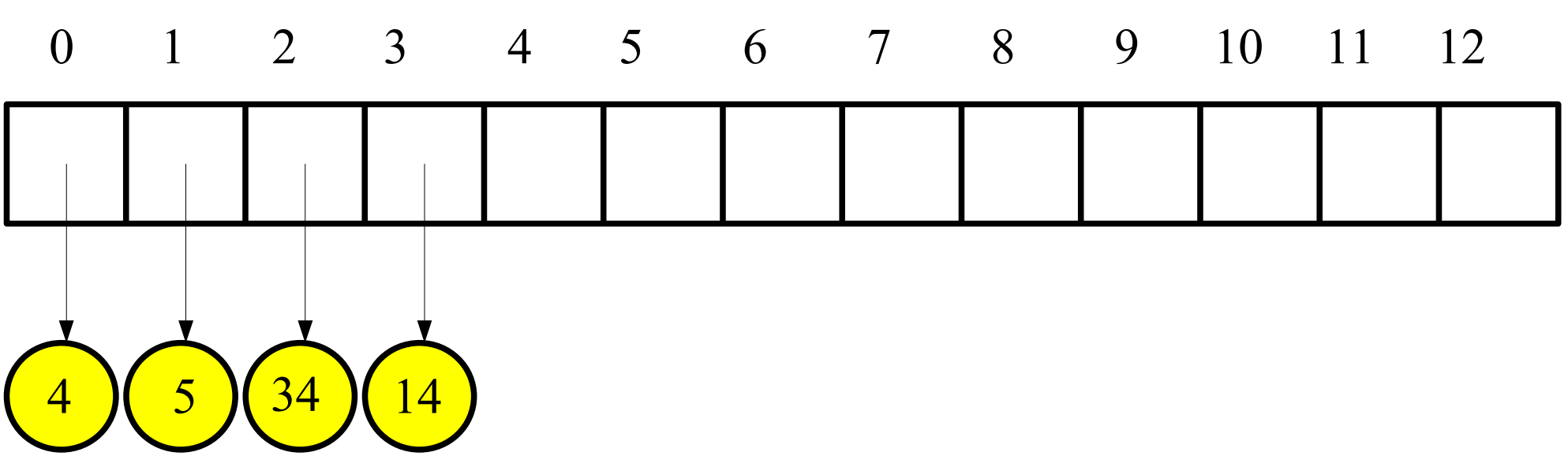




root = 0
tail = 4

Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index
Compare root with two children
If greater, swap with smallest of the children
Compare swapped object with children
at indices 1*2+1=3 and 1*2+2=4
Node is smaller than both: stop





root = 0
tail = 4

Now to “pop” the root node.
Move “tail” object to root.
Change the “tail” index
Compare root with two children
If greater, swap with smallest of the children
Compare swapped object with children
at indices 1*2+1=3 and 1*2+2=4
Node is smaller than both: stop
Return the object