Max Heap
based on Arrays

Prepared by Mahdi Ghamkhari

What is a Heap?

What is a Max Heap?

- A binary try which is complete
- Each node has a value greater than the values of its children

What is a Heap?

- Can we have a more general definition for a Max Heap? Yes

  - A max heap is a data structure in which deletion has a time complexity of O(log(n)), and
  - Delete method `return` to the user the largest value stored in the data structure

Heap Implementation

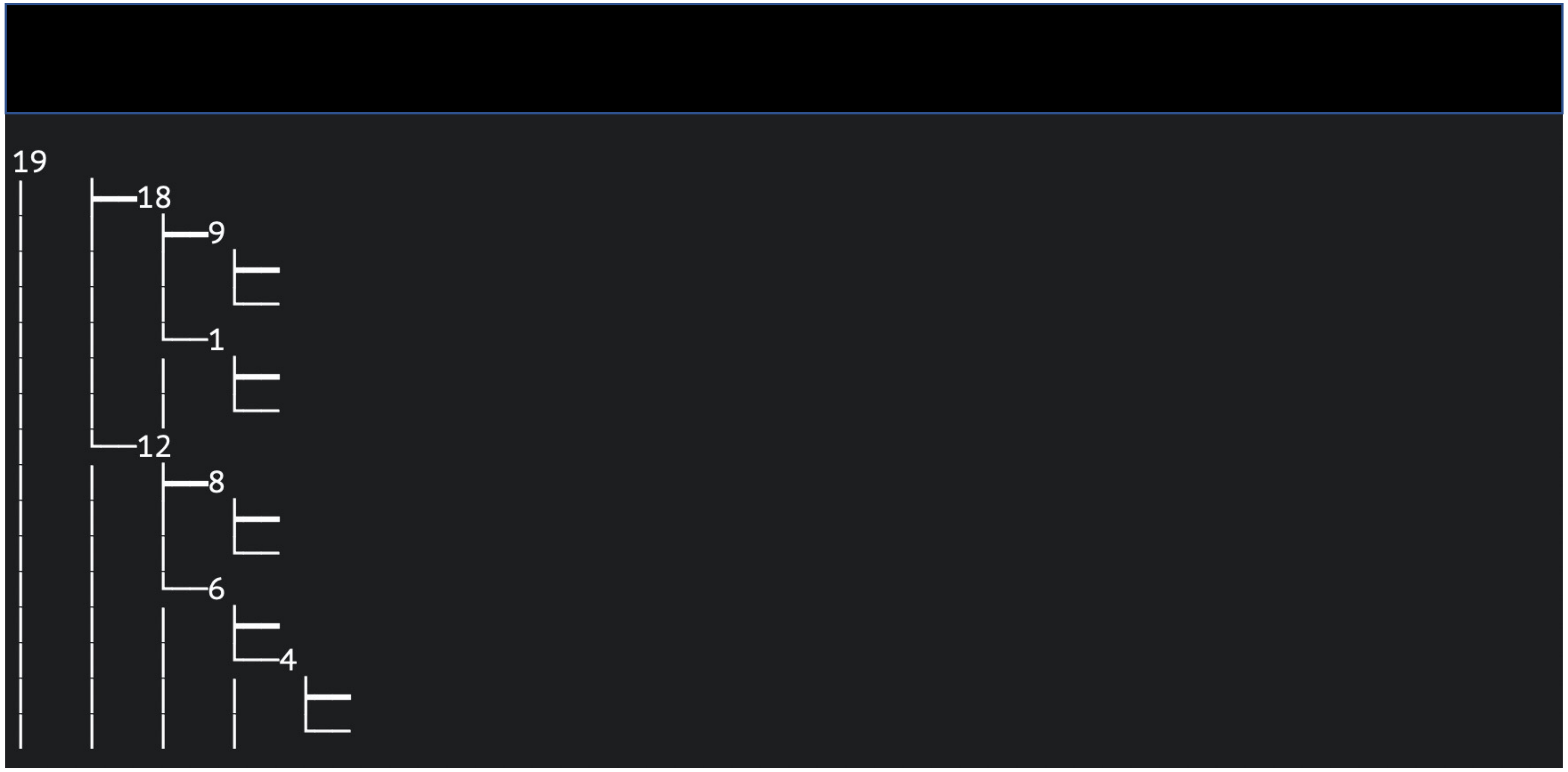
- A Max Heap data structure can be implemented using
  - ➢ Trees
  - ➢ Arrays

We have already seen how Max Heaps are implemented based on trees

Lets’ see how we can implement Max heaps based on arrays

Tree and InternalArray Correspondence

This is a max heap implemented by a tree:

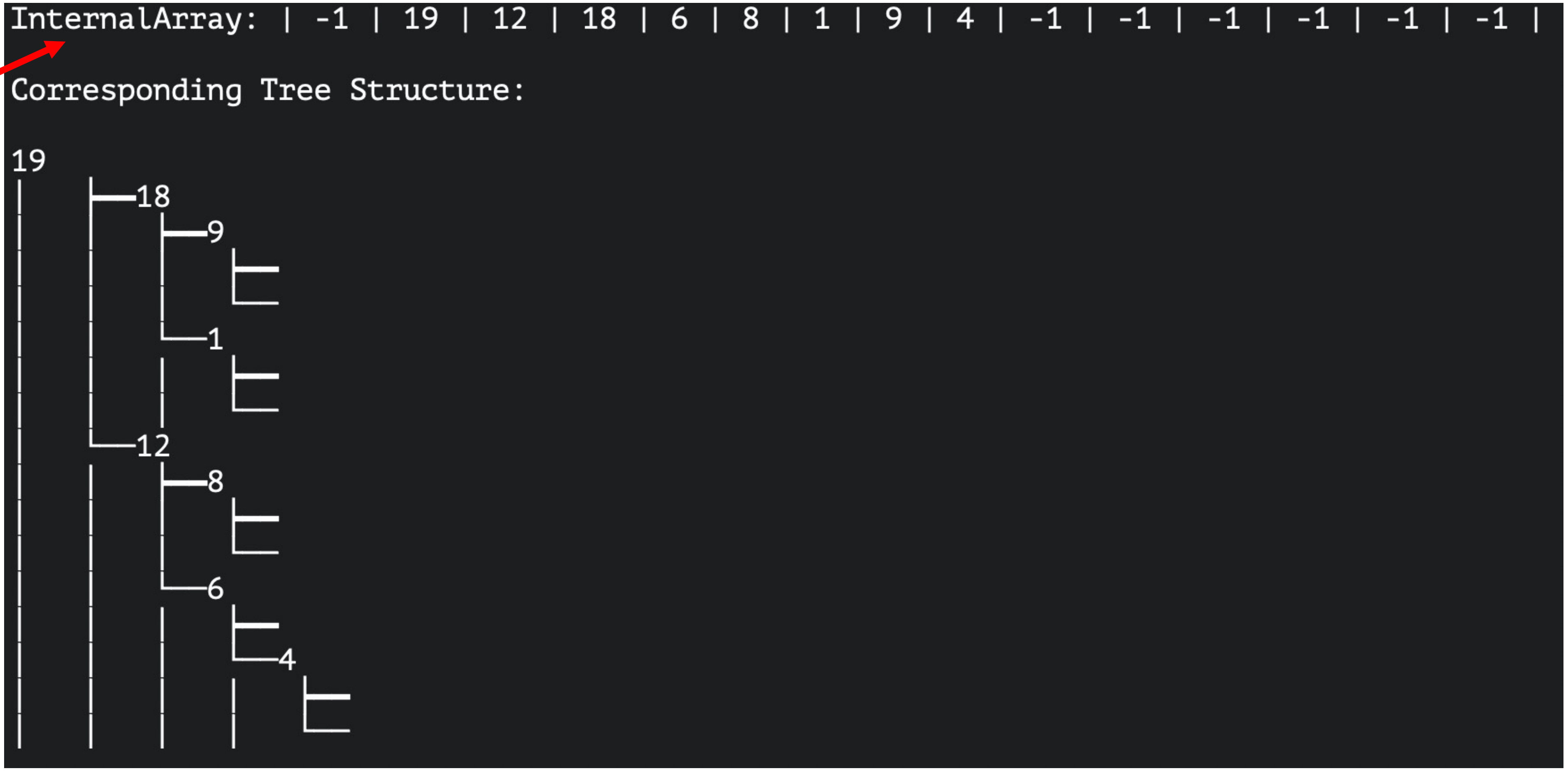


[redacted]

```
19
│    ├──18
│    │    ├──9
│    │    │    ├──
│    │    │    └──
│    │    └──1
│    │    │    ├──
│    │    │    └──
│    └──12
│    │    ├──8
│    │    │    ├──
│    │    │    └──
│    │    └──6
│    │    │    ├──
│    │    │    └──4
│    │    │    │    ├──
│    │    │    │    └──
│    │    │    │
```

Tree and InternalArray Correspondence

The tree is complete. So the values of the tree can be placed side by side in an InternalArray of adequate size



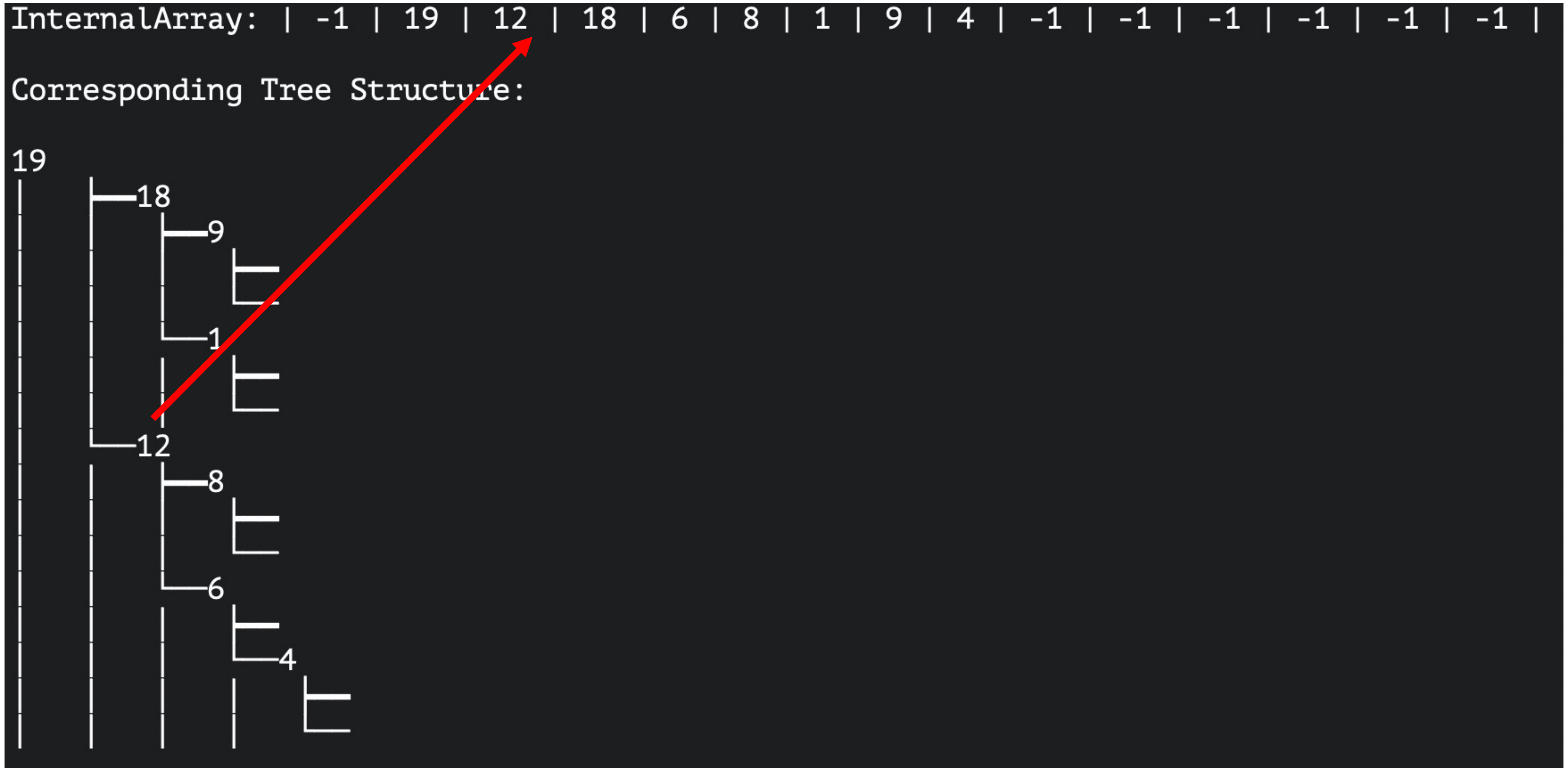
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│   ├──18
│   │   ├──9
│   │   │   ├──
│   │   │   └──
│   │   └──1
│   │   │   ├──
│   │   │   └──
│   └──12
│   │   ├──8
│   │   │   ├──
│   │   │   └──
│   │   └──6
│   │   │   ├──
│   │   │   └──4
│   │   │   │   ├──
│   │   │   │   └──
│   │   │   │
```

Tree and InternalArray Correspondence

Node 2 of the tree

Corresponds to

Index=2 of the InternalArray



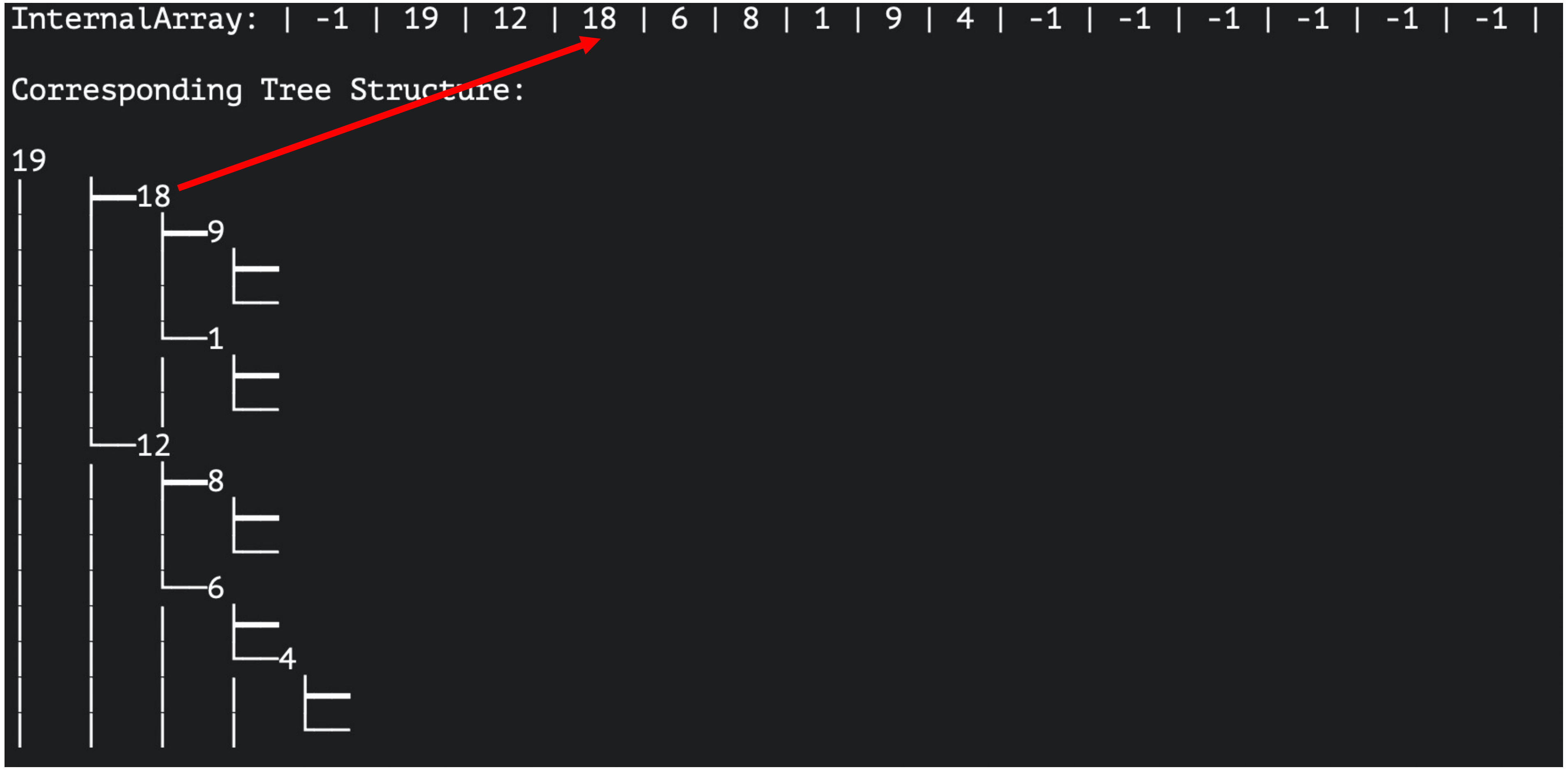
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│   ├──18
│   │   ├──9
│   │   │   ├──
│   │   │   └──
│   │   └──1
│   │   │   ├──
│   │   │   └──
│   └──12
│   │   ├──8
│   │   │   ├──
│   │   │   └──
│   │   └──6
│   │   │   ├──
│   │   │   └──4
│   │   │   │   ├──
│   │   │   │   └──
│   │   │   │
```

Tree and InternalArray Correspondence

Node 3 of the tree

Corresponds to

Index=3 of the InternalArray



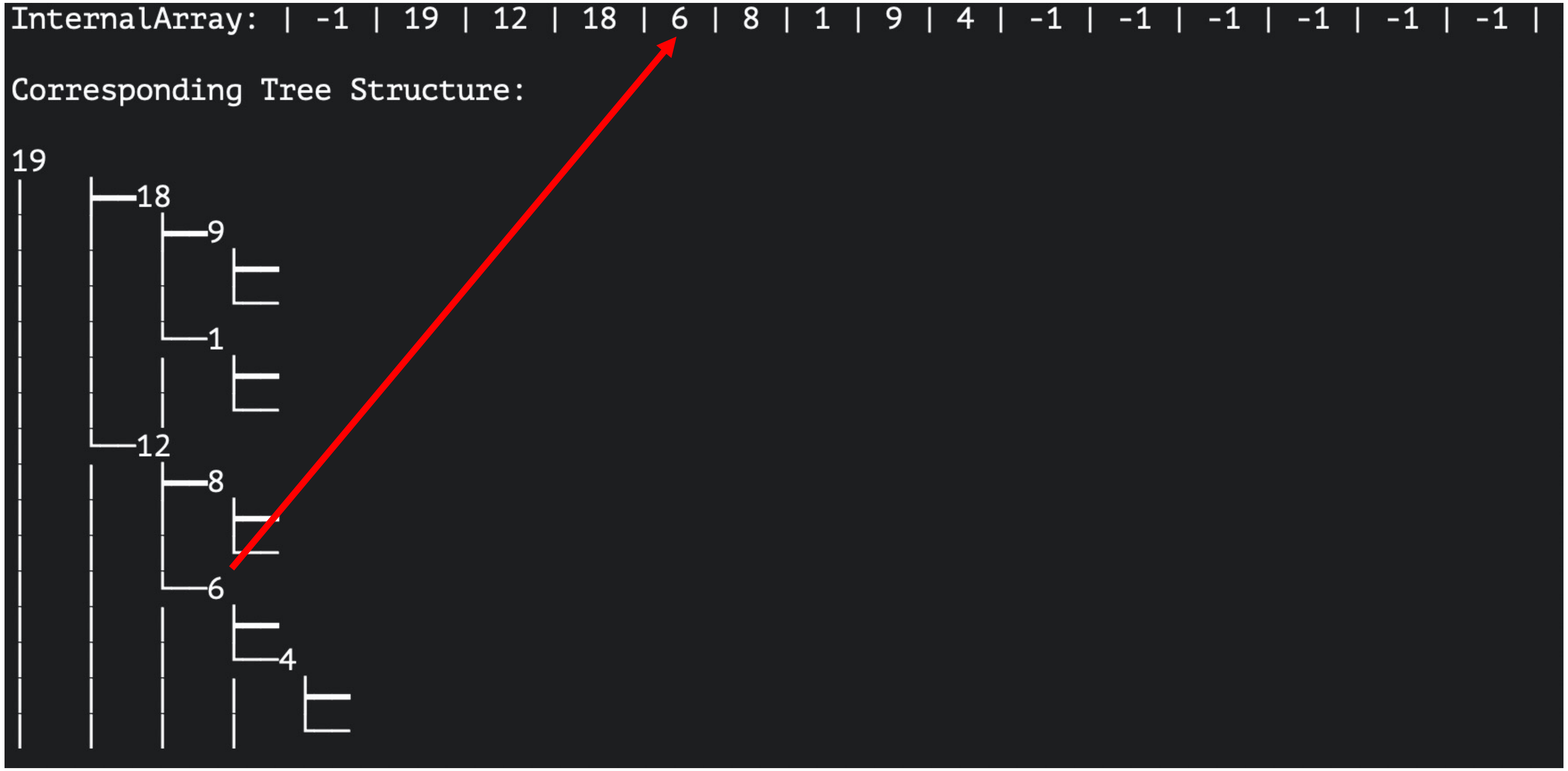
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│    ├──18
│    │    ├──9
│    │    │    ├──
│    │    │    └──
│    │    └──1
│    │    │    ├──
│    │    │    └──
│    └──12
│    │    ├──8
│    │    │    ├──
│    │    │    └──
│    │    └──6
│    │    │    ├──
│    │    │    └──4
│    │    │    │    ├──
│    │    │    │    └──
```

Tree and InternalArray Correspondence

Node 4 of the tree

Corresponds to

Index=4 of the InternalArray



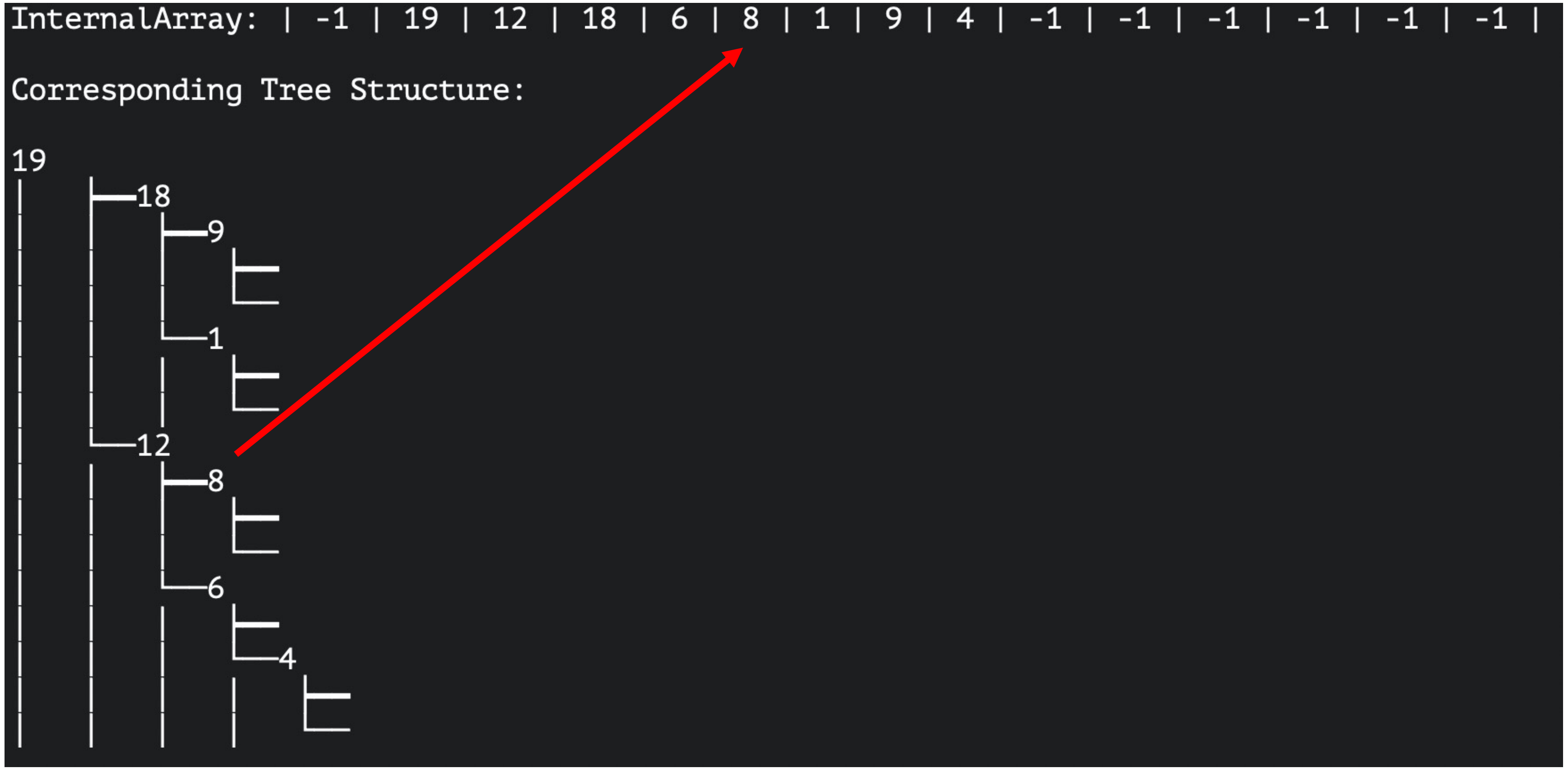
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│    ├──18
│    │    ├──9
│    │    │    ├──
│    │    │    └──
│    │    └──1
│    │    │    ├──
│    │    │    └──
│    └──12
│    │    ├──8
│    │    │    ├──
│    │    │    └──
│    │    └──6
│    │    │    ├──
│    │    │    └──4
│    │    │    │    ├──
│    │    │    │    └──
│    │    │    │
```

Tree and InternalArray Correspondence

Node 5 of the tree

Corresponds to

Index=5 of the InternalArray



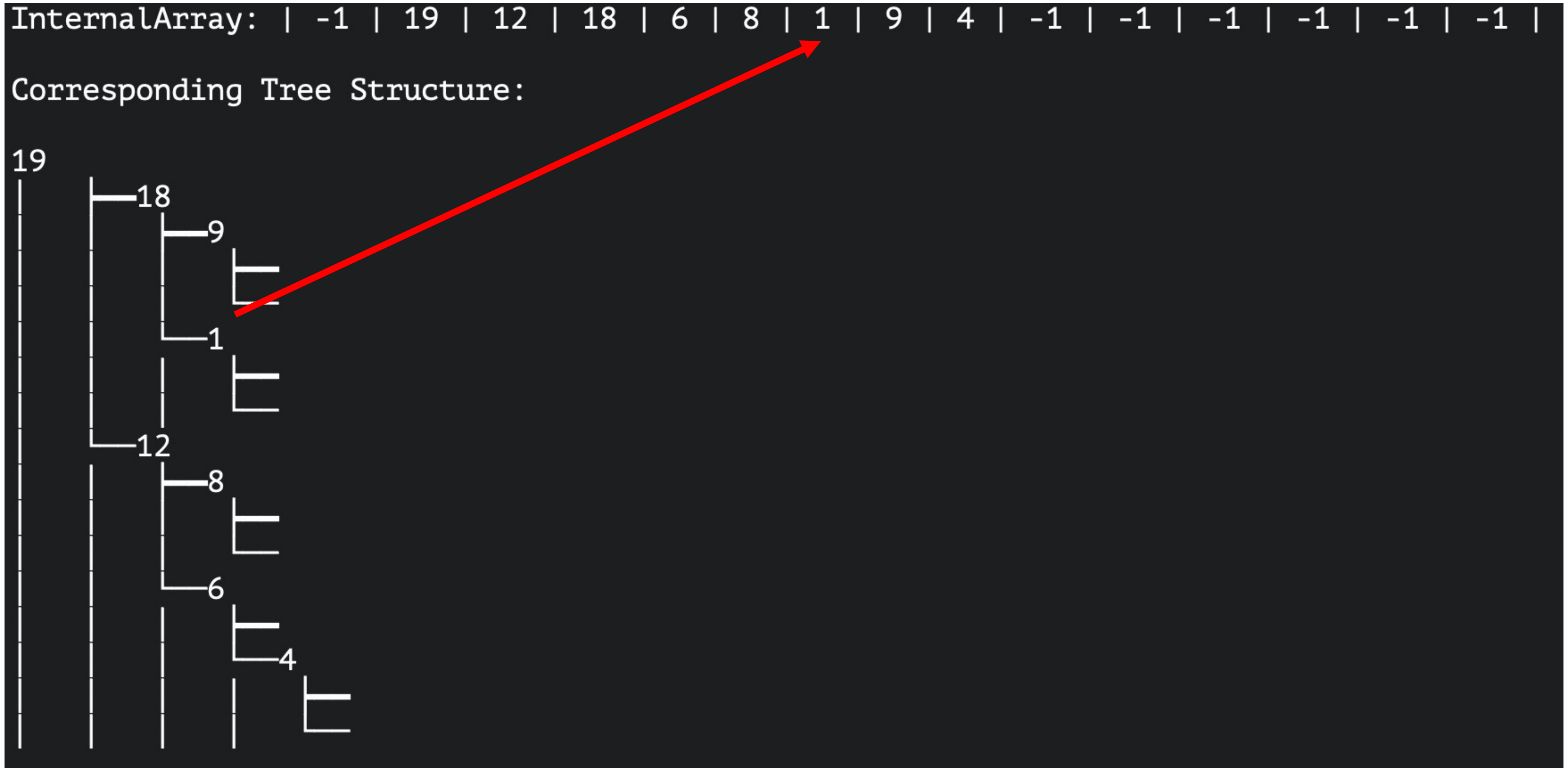
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│   ├──18
│   │   ├──9
│   │   │   ├──
│   │   │   └──
│   │   └──1
│   │   │   ├──
│   │   │   └──
│   └──12
│   │   ├──8
│   │   │   ├──
│   │   │   └──
│   │   └──6
│   │   │   ├──
│   │   │   └──4
│   │   │   │   ├──
│   │   │   │   └──
│   │   │   │
```

Tree and InternalArray Correspondence

Node 6 of the tree

Corresponds to

Index=6 of the InternalArray



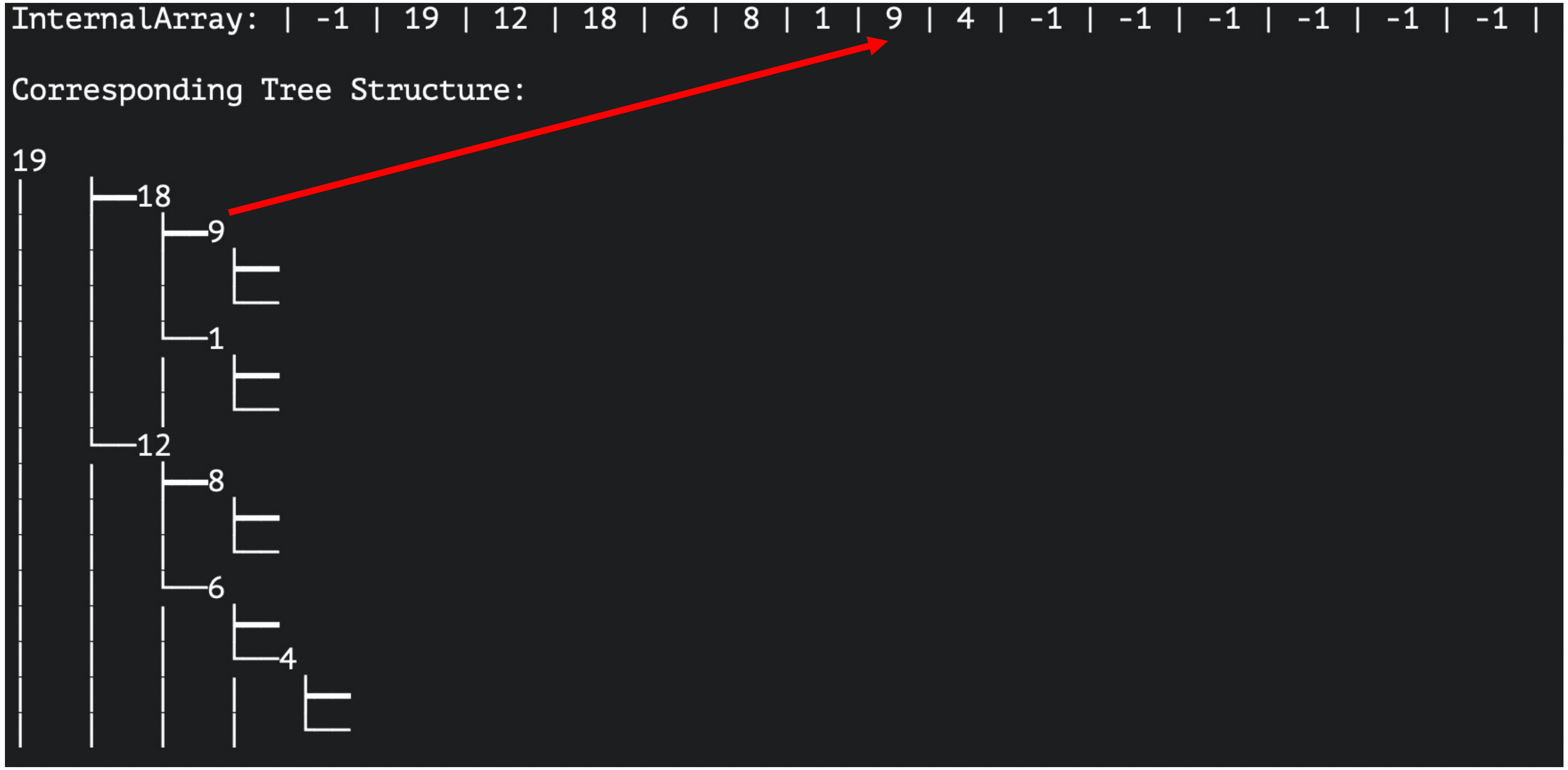
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│   ├──18
│   │   ├──9
│   │   │   ├──
│   │   │   └──
│   │   └──1
│   │   │   ├──
│   │   │   └──
│   └──12
│   │   ├──8
│   │   │   ├──
│   │   │   └──
│   │   └──6
│   │   │   ├──
│   │   │   └──4
│   │   │   │   ├──
│   │   │   │   └──
```

Tree and InternalArray Correspondence

Node 7 of the tree

Corresponds to

Index=7 of the InternalArray

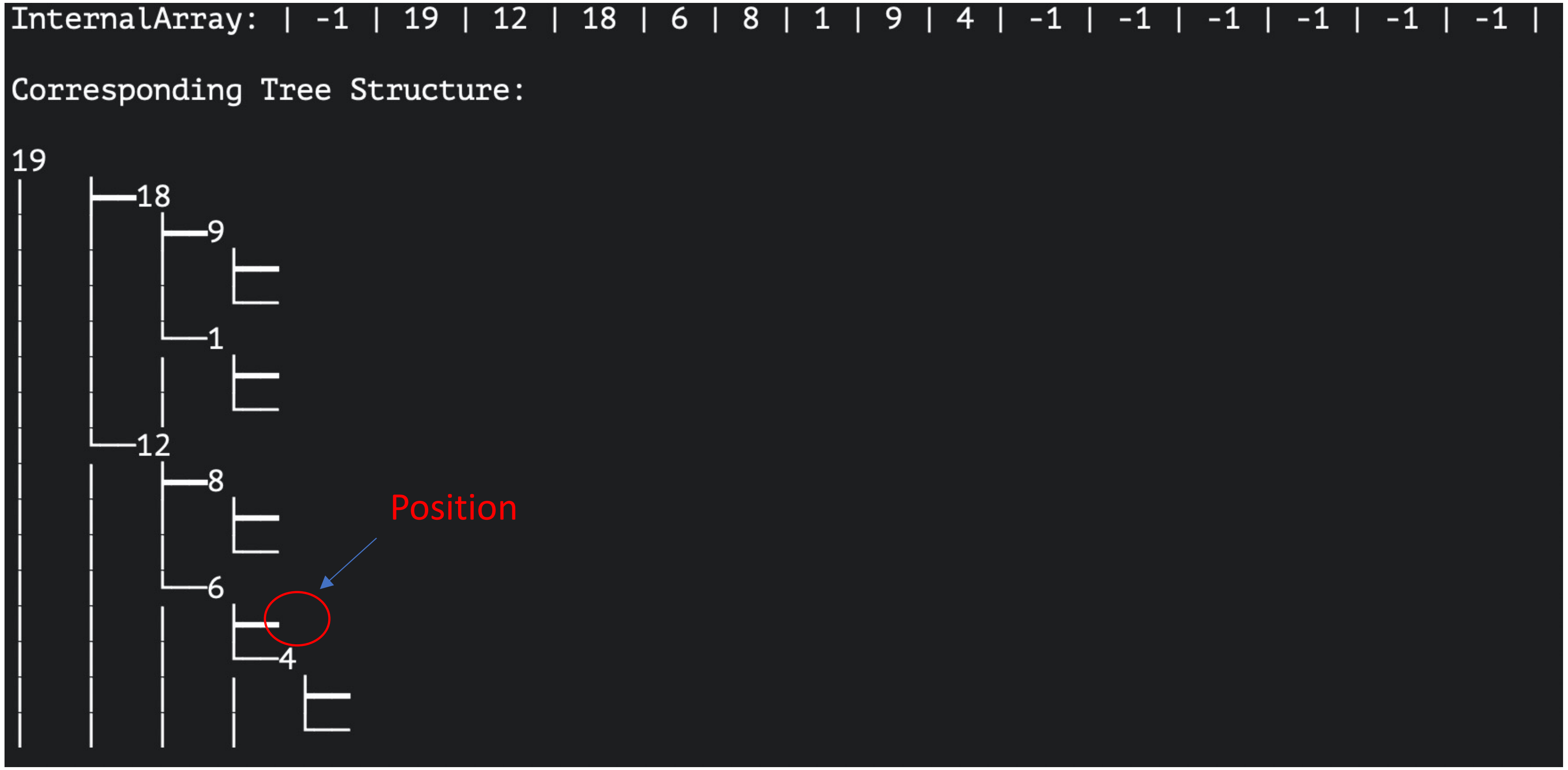


```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│    ├──18
│    │    ├──9
│    │    │    ├──
│    │    │    └──
│    │    └──1
│    │    │    ├──
│    │    │    └──
│    └──12
│    │    ├──8
│    │    │    ├──
│    │    │    └──
│    │    └──6
│    │    │    ├──
│    │    │    └──4
│    │    │    │    ├──
│    │    │    │    └──
│    │    │    │
```

Add() to Tree

myTreeHeap.add(17)

We cannot simply add 17 to this Position

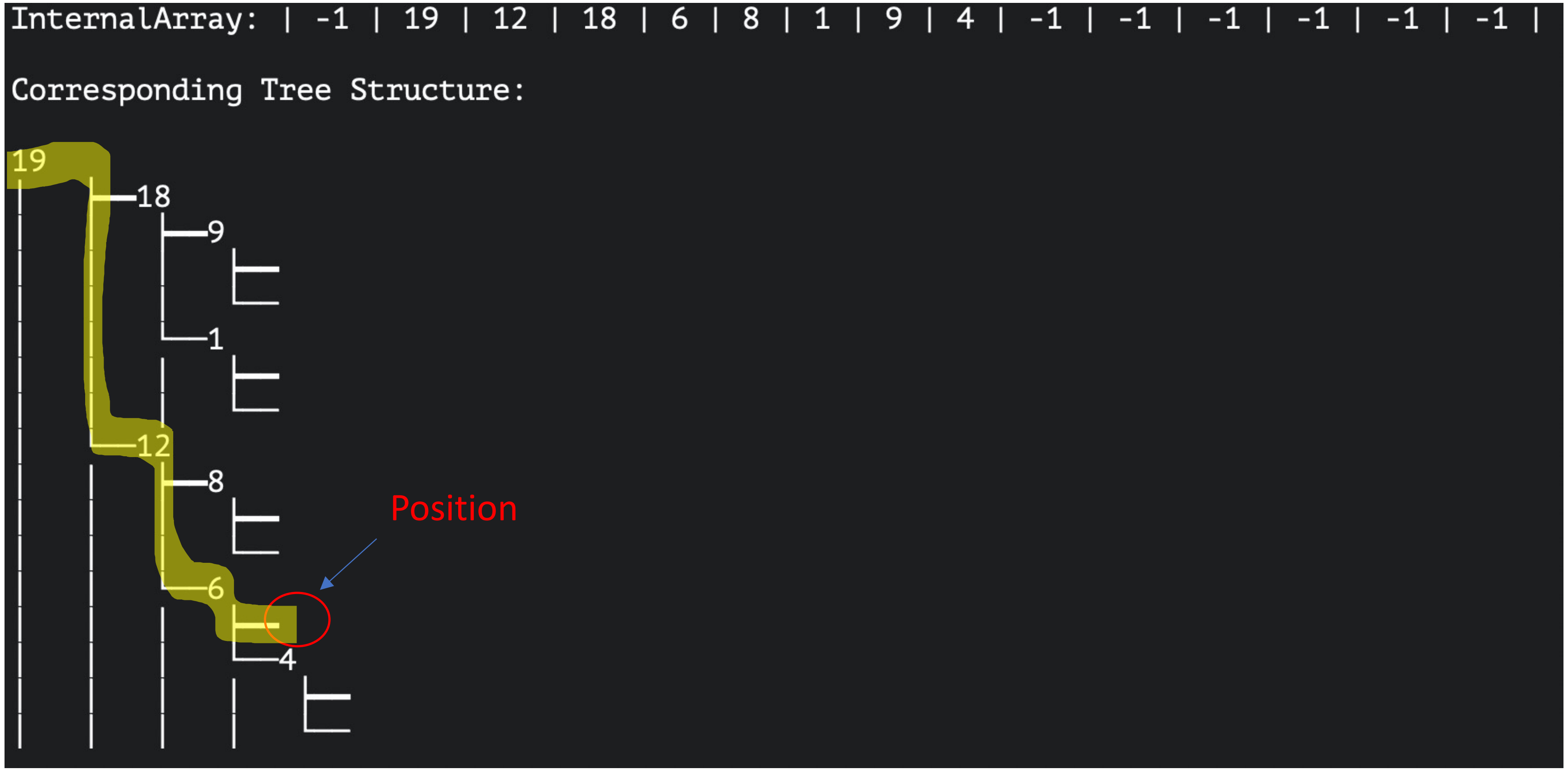


```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│    ├──18
│    │    ├──9
│    │    │    ├──
│    │    │    └──
│    │    └──1
│    │    │    ├──
│    │    │    └──
│    └──12
│    │    ├──8
│    │    │    ├──
│    │    │    └──
│    │    └──6
│    │    │    ├──
│    │    │    └──4
│    │    │    │    ├──
│    │    │    │    └──
```

Add() to Tree

myTreeHeap.add(17)

Certain values in the tree need to be rearranged



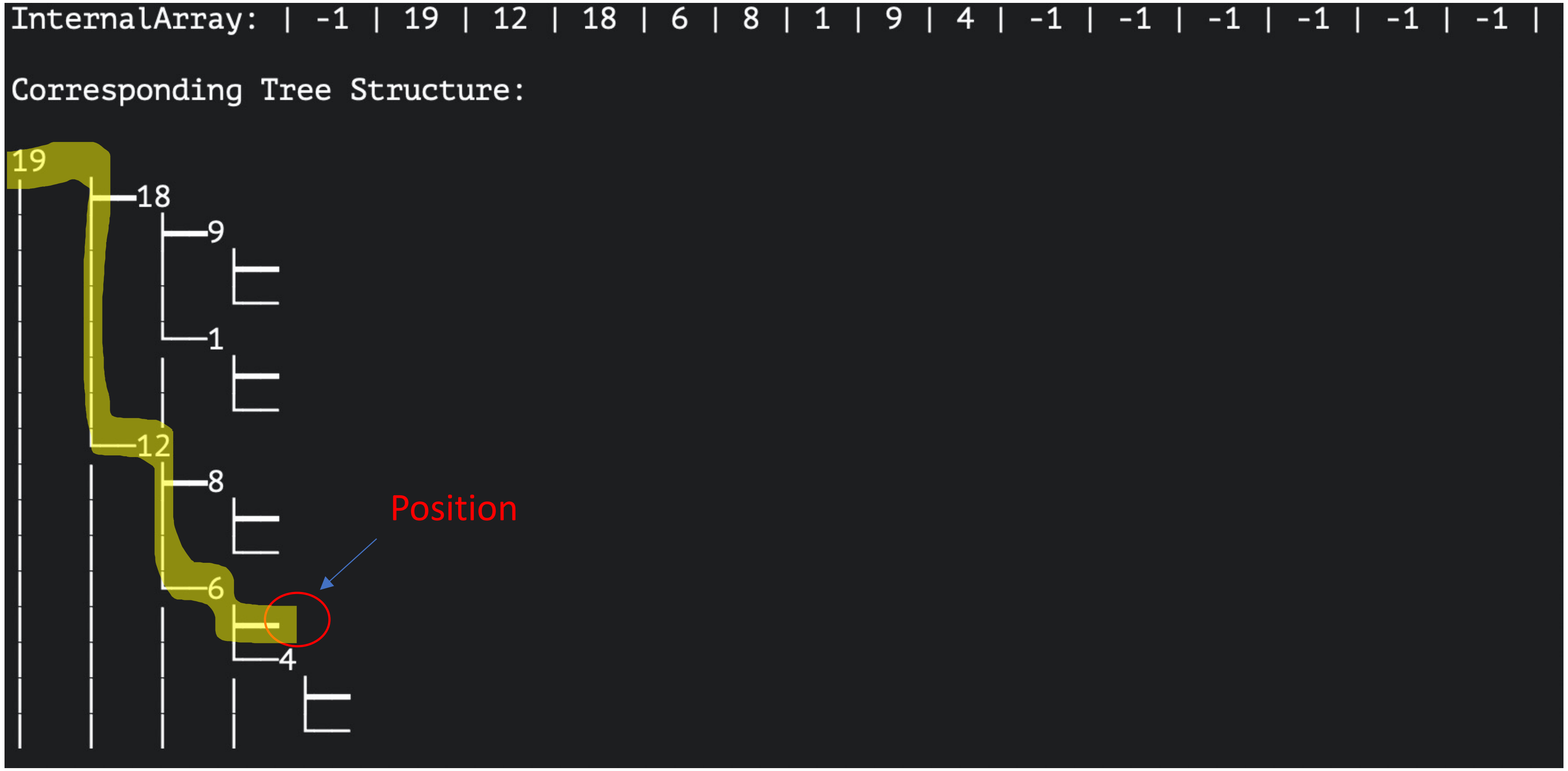
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│  ├──18
│  │  ├──9
│  │  │  ├──
│  │  │  └──
│  │  └──1
│  │  │  ├──
│  │  │  └──
│  └──12
│  │  ├──8
│  │  │  ├──
│  │  │  └──
│  │  └──6
│  │  │  ├──
│  │  │  └──4
│  │  │  │  ├──
│  │  │  │  └──
```

Add() to Tree

myTreeHeap.add(17)

Certain values in the tree need to be rearranged

Values on the route to Position need to be rearranged



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│   ├──18
│   │   ├──9
│   │   │   ├──
│   │   │   └──
│   │   └──1
│   │       ├──
│   │       └──
│   └──12
│       ├──8
│       │   ├──
│       │   └──
│       └──6
│           ├──
│           └──4
│               ├──
│               └──
```

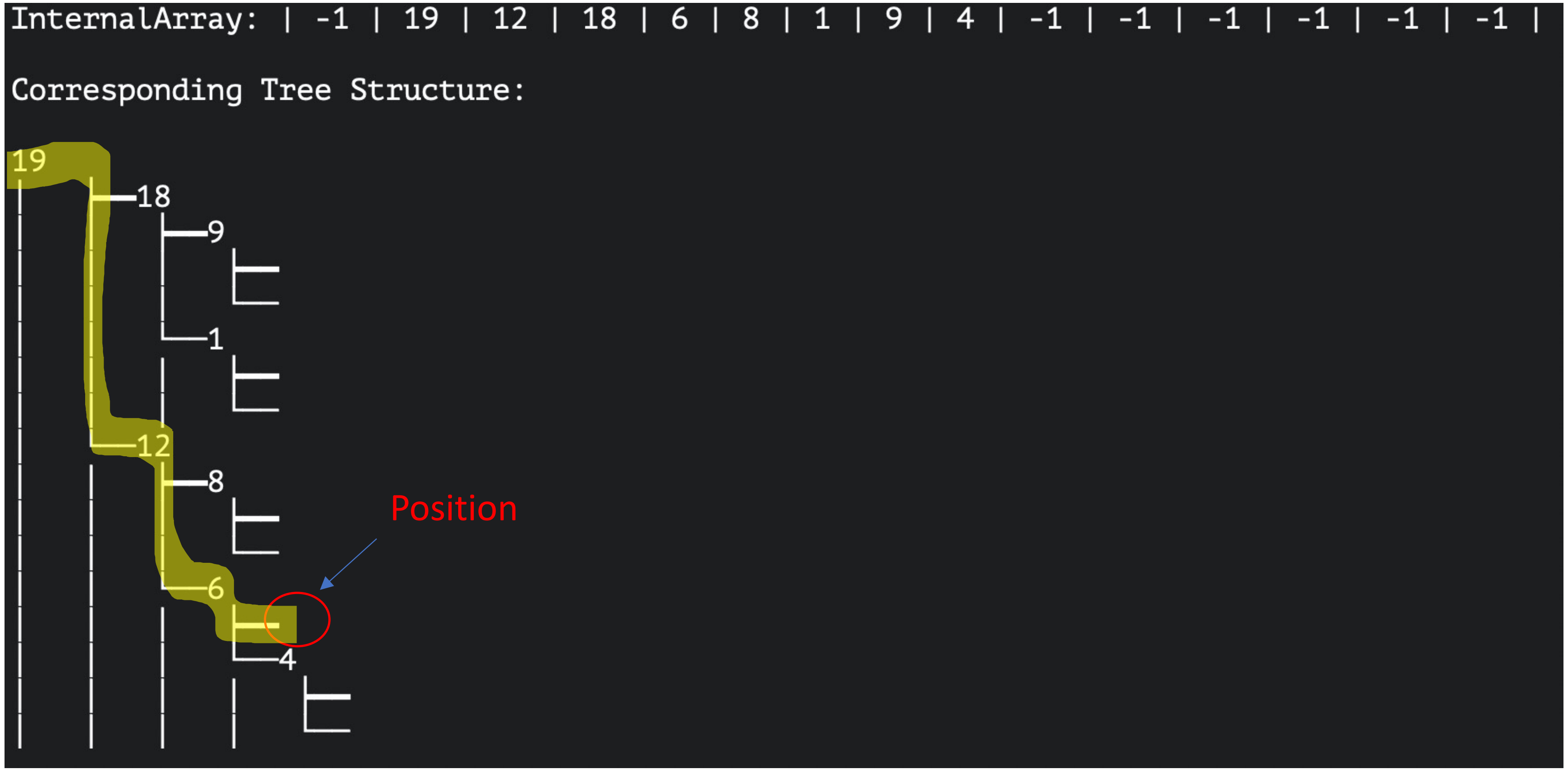
Add() to Tree

myTreeHeap.add(17)

Certain values in the tree need to be rearranged

Values on the route to Position need to be rearranged

route(Position) provides a string of “Downward” and “Upward” keywords to show the route/path



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
|   ├──18
|   |   ├──9
|   |   |   ├──
|   |   |   └──
|   |   └──1
|   |       ├──
|   |       └──
|   └──12
|       ├──8
|       |   ├──
|       |   └──
|       └──6
|           ├──
|           └──4
|               ├──
|               └──
```

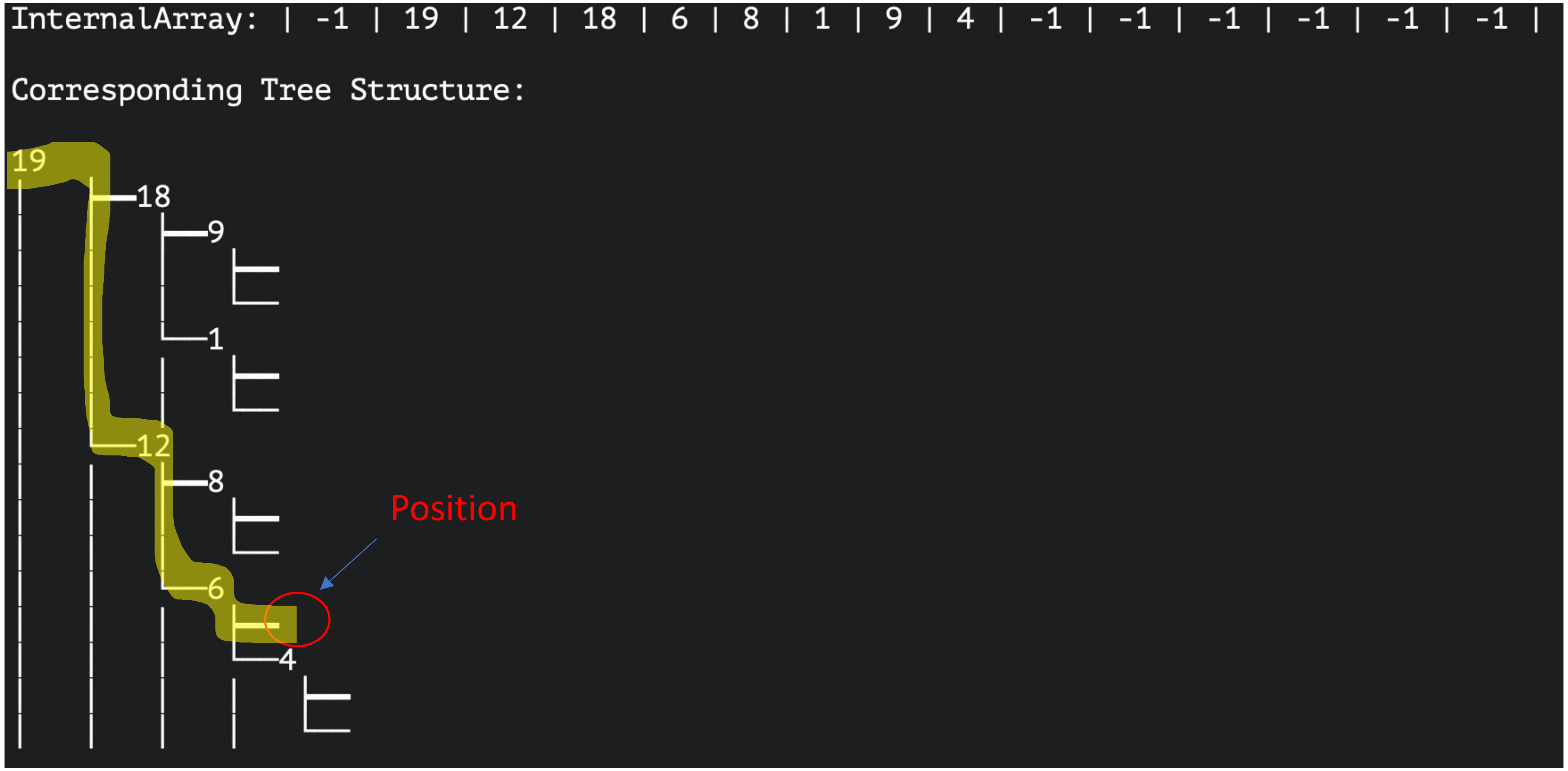
Add() to Tree

myTreeHeap.add(17)

Position=9

route(9) gives:

DownWard
DownWard
UpWard

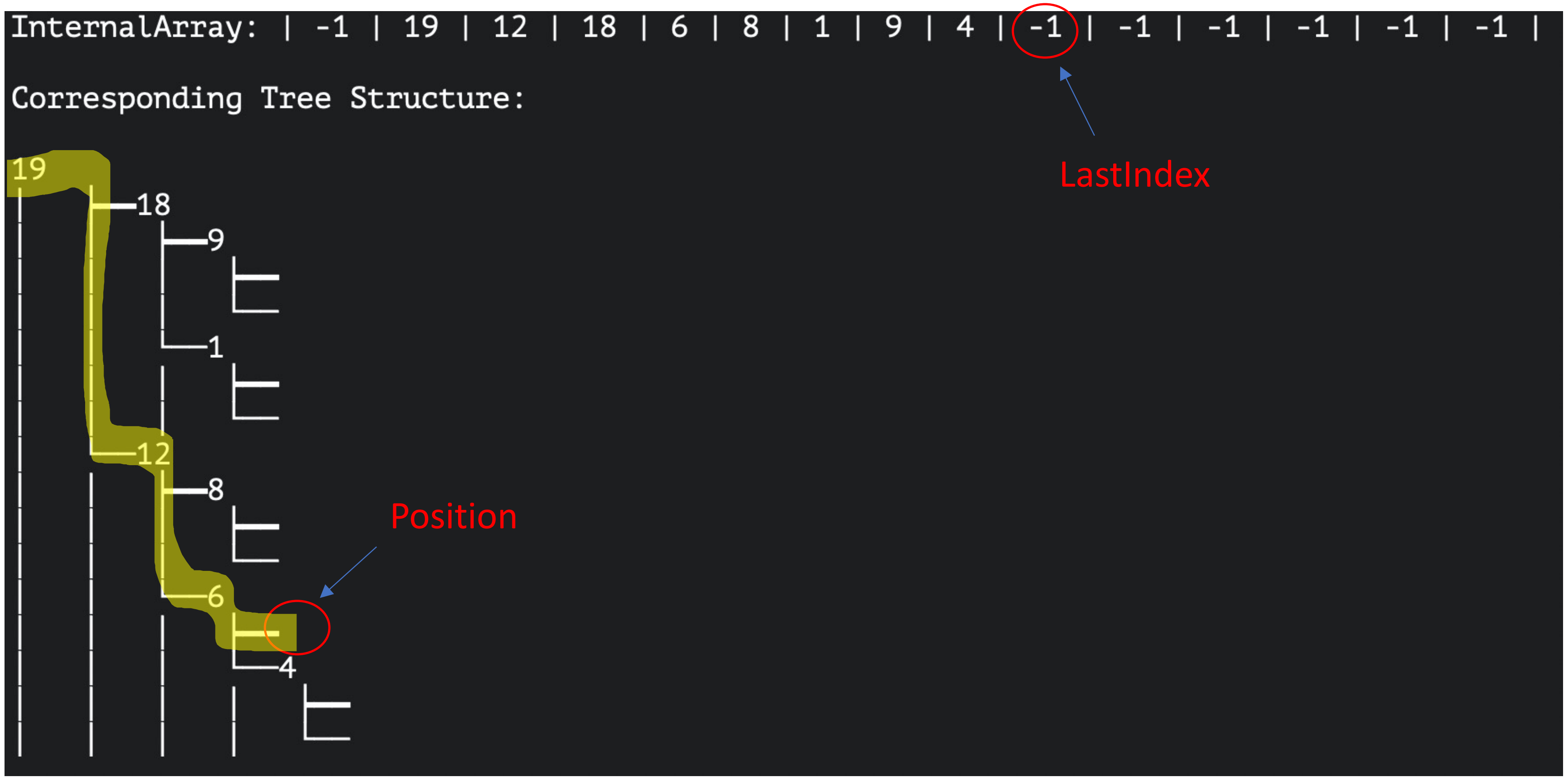


```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
├──18
│  ├──9
│  │  ├──
│  │  └──
│  └──1
│     ├──
│     └──
└──12
   ├──8
   │  ├──
   │  └──
   └──6
      ├──
      └──4
         ├──
         └──
```

Add() to InternalArray

myArrayHeap.add(17)

To add a new value 17 to the InternalArray, we cannot place 17 at Index=LastIndex



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
│  ├─18
│  │  ├─9
│  │  │  ├─
│  │  │  └─
│  │  └─1
│  │     ├─
│  │     └─
│  └─12
│     ├─8
│     │  ├─
│     │  └─
│     └─6
│        ├─
│        └─4
│           ├─
│           └─
```

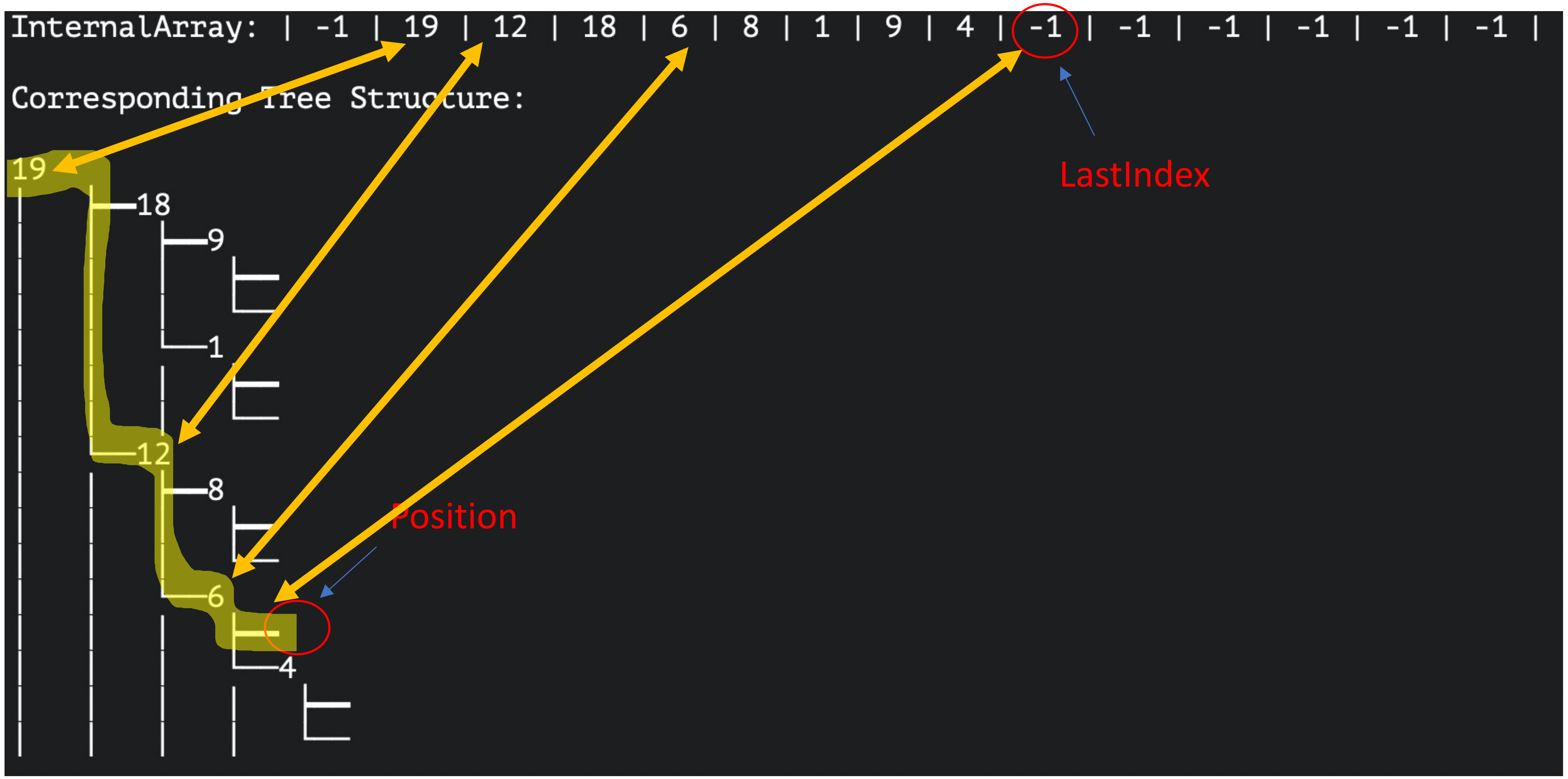
Add() to InternalArray

myArrayHeap.add(17)

To add a new value 17 to the InternalArray, we cannot place 17 at Index=LastIndex

Certain indexes of InternalArray should be rearranged

Those indexes that are corresponding to the nodes in the route



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |

Corresponding Tree Structure:

19
 |  ├─18
 |  |  ├─9
 |  |  |  ├─
 |  |  |  └─
 |  |  └─1
 |  |     ├─
 |  |     └─
 |  └─12
 |     ├─8
 |     |  ├─
 |     |  └─
 |     └─6
 |        ├─
 |        └─4
 |           ├─
 |           └─
```

Add() to InternalArray

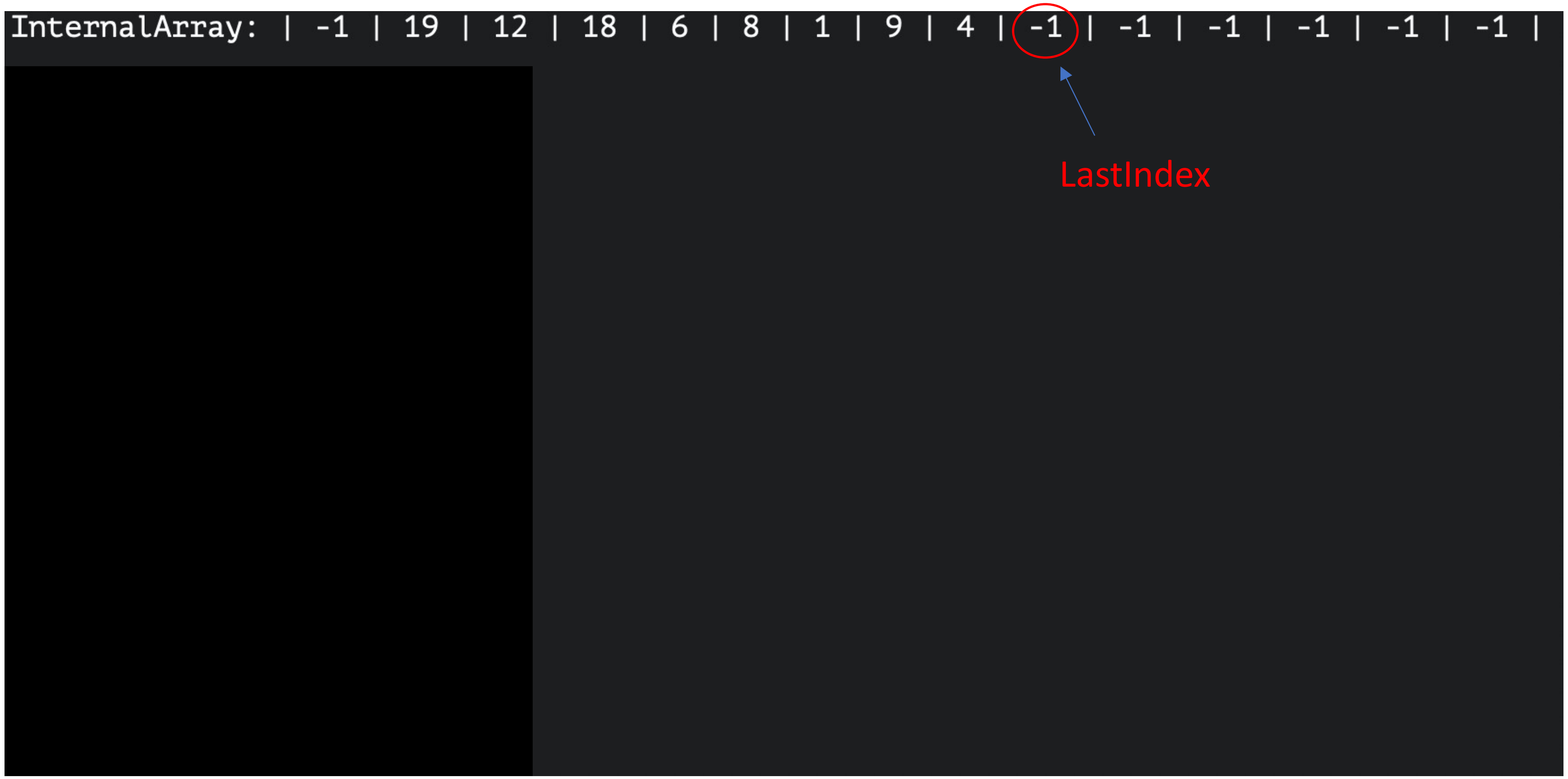
myArrayHeap.add(17)

Those indexes can be found as follows:

LastIndex=9

route(9) gives:

DownWard
DownWard
UpWard



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

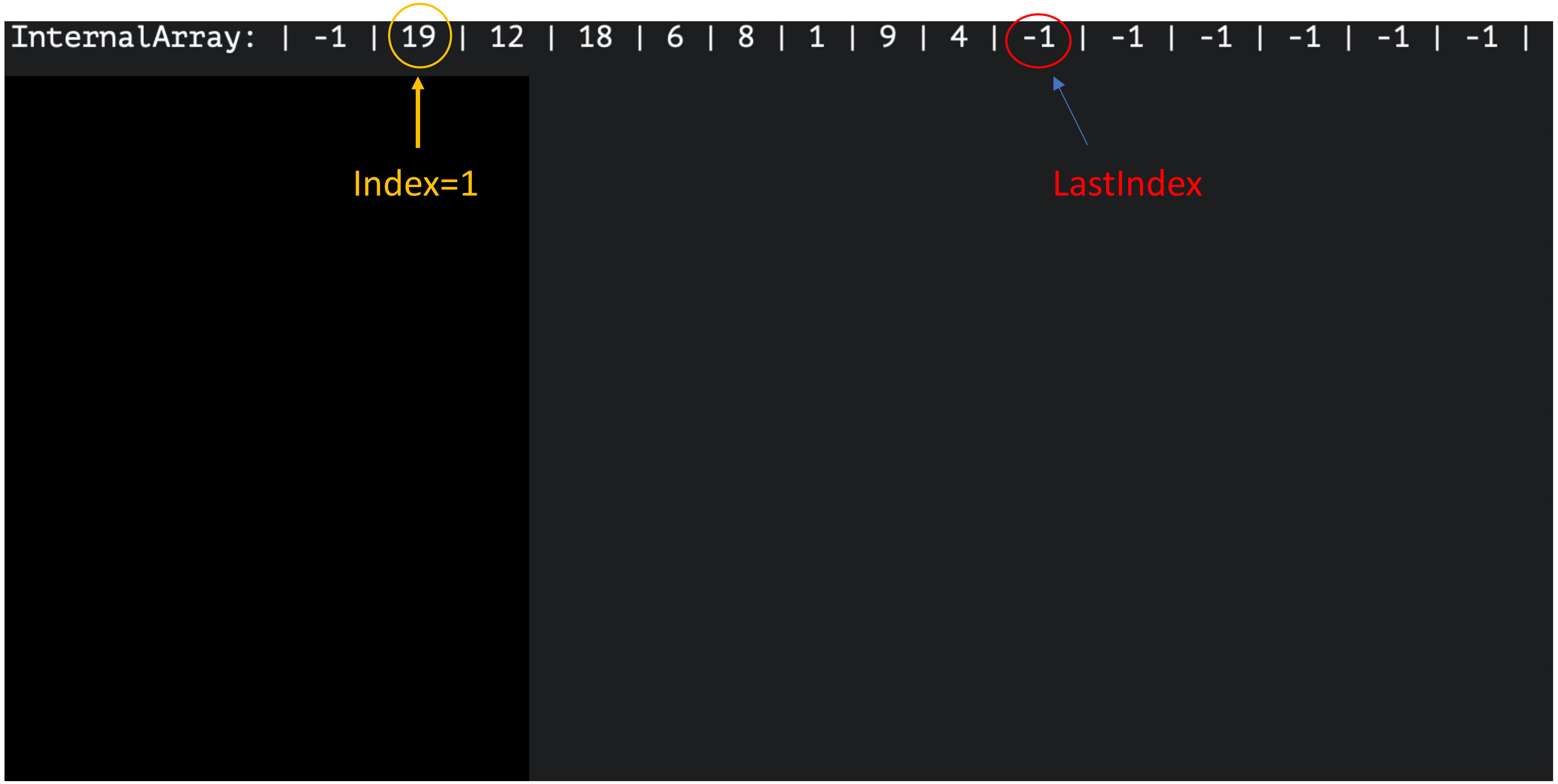
myArrayHeap.add(17)

Those indexes can be found as follows:

LastIndex=9

route(9) gives:

DownWard
DownWard
UpWard



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

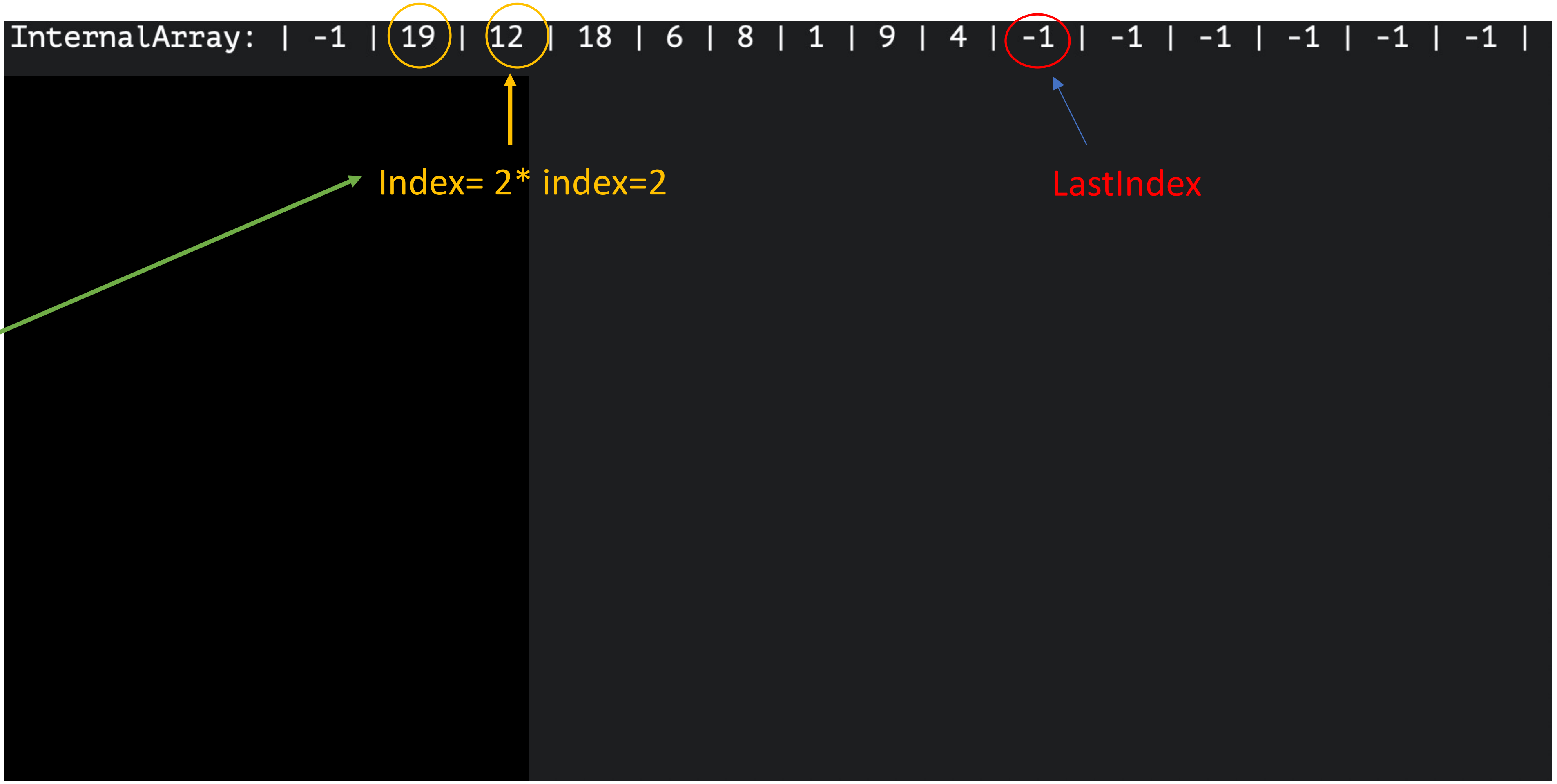
myArrayHeap.add(17)

Those indexes can be found as follows:

LastIndex=9

route(9) gives:

DownWard
DownWard
UpWard



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

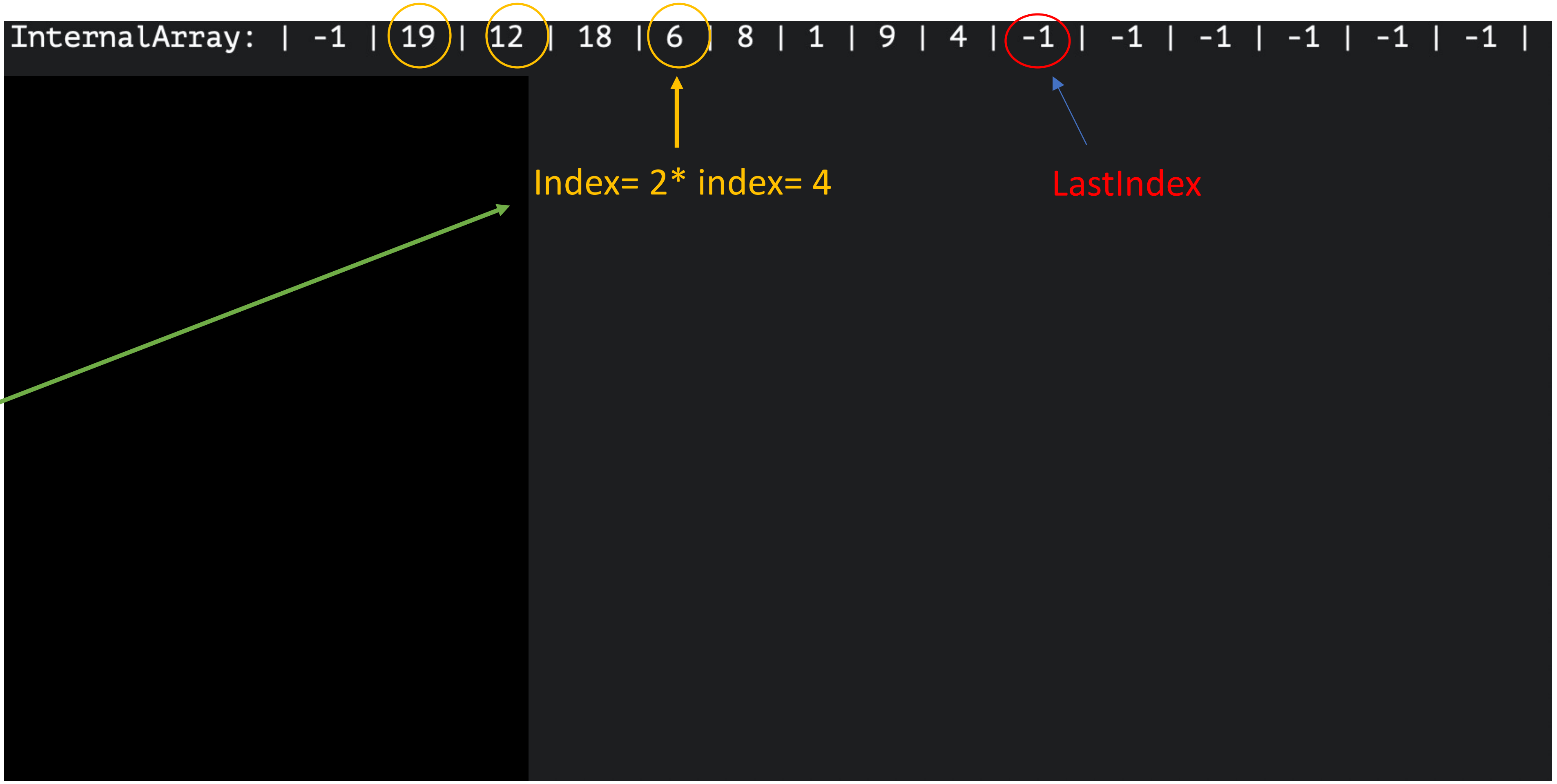
myArrayHeap.add(17)

Those indexes can be found as follows:

LastIndex=9

route(9) gives:

DownWard
DownWard
UpWard



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

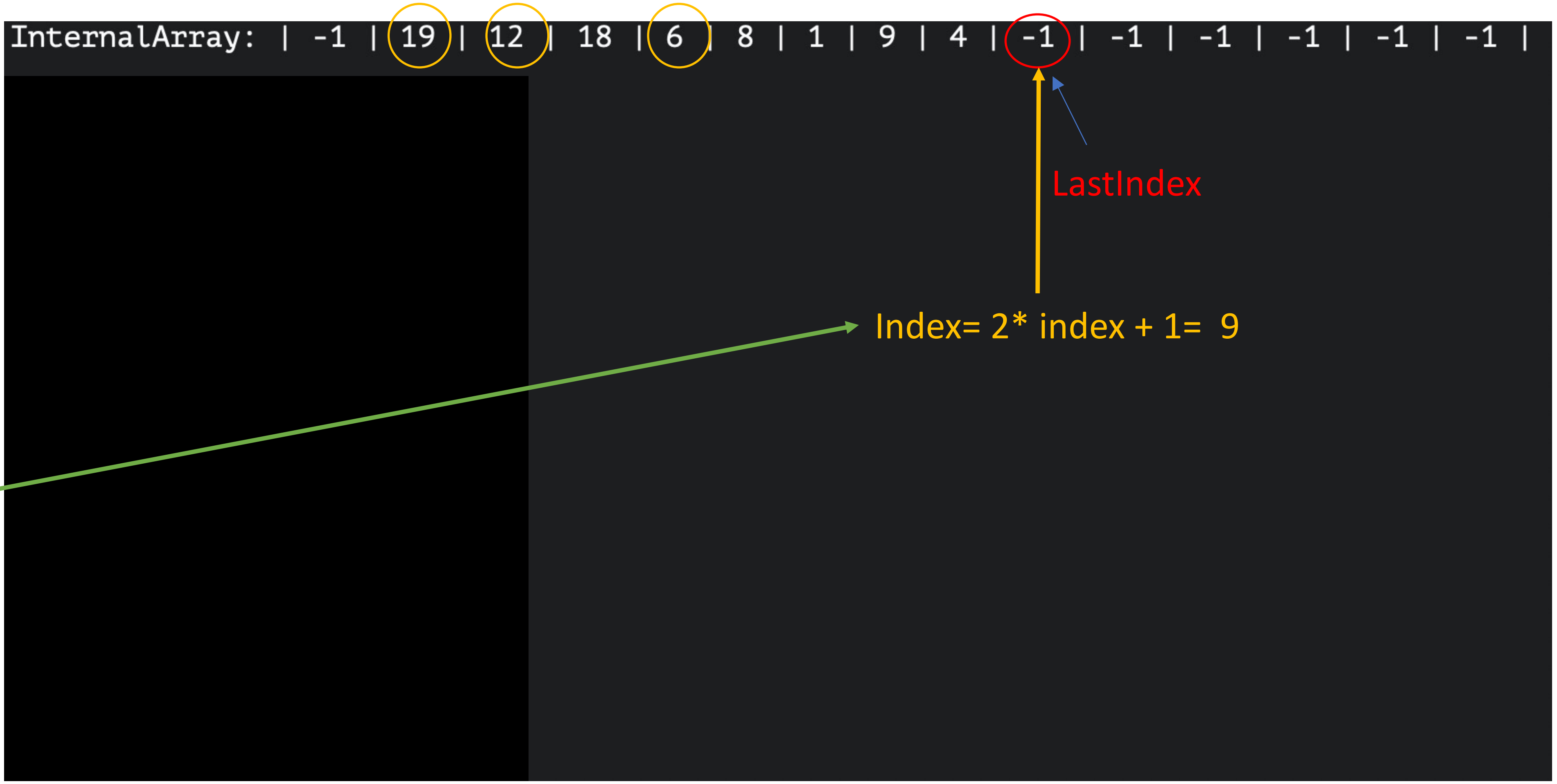
myArrayHeap.add(17)

Those indexes can be found as follows:

LastIndex=9

route(9) gives:

DownWard
DownWard
UpWard



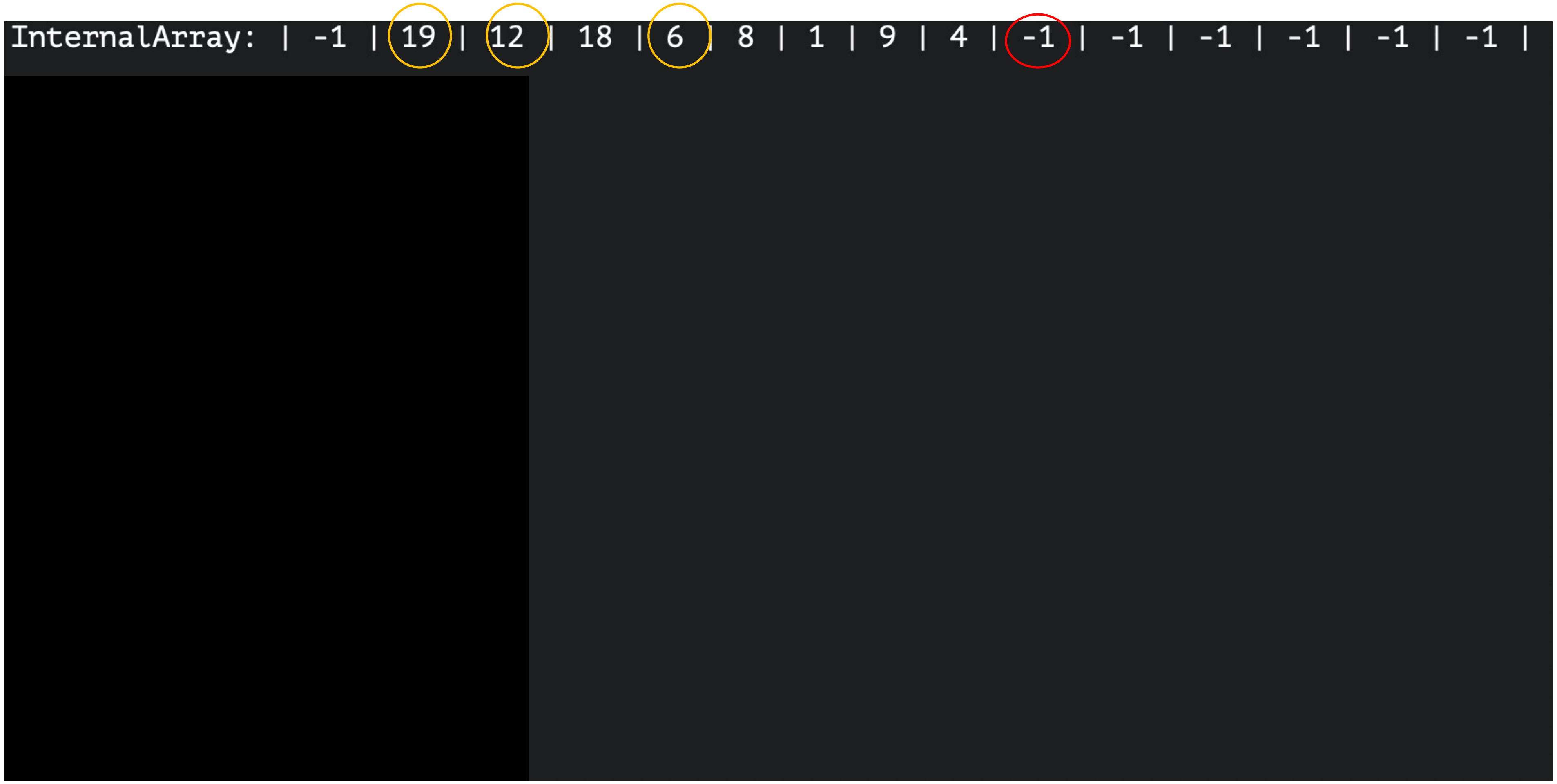
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

myArrayHeap.add(17)

Values to be rearranged

19 12 6 17



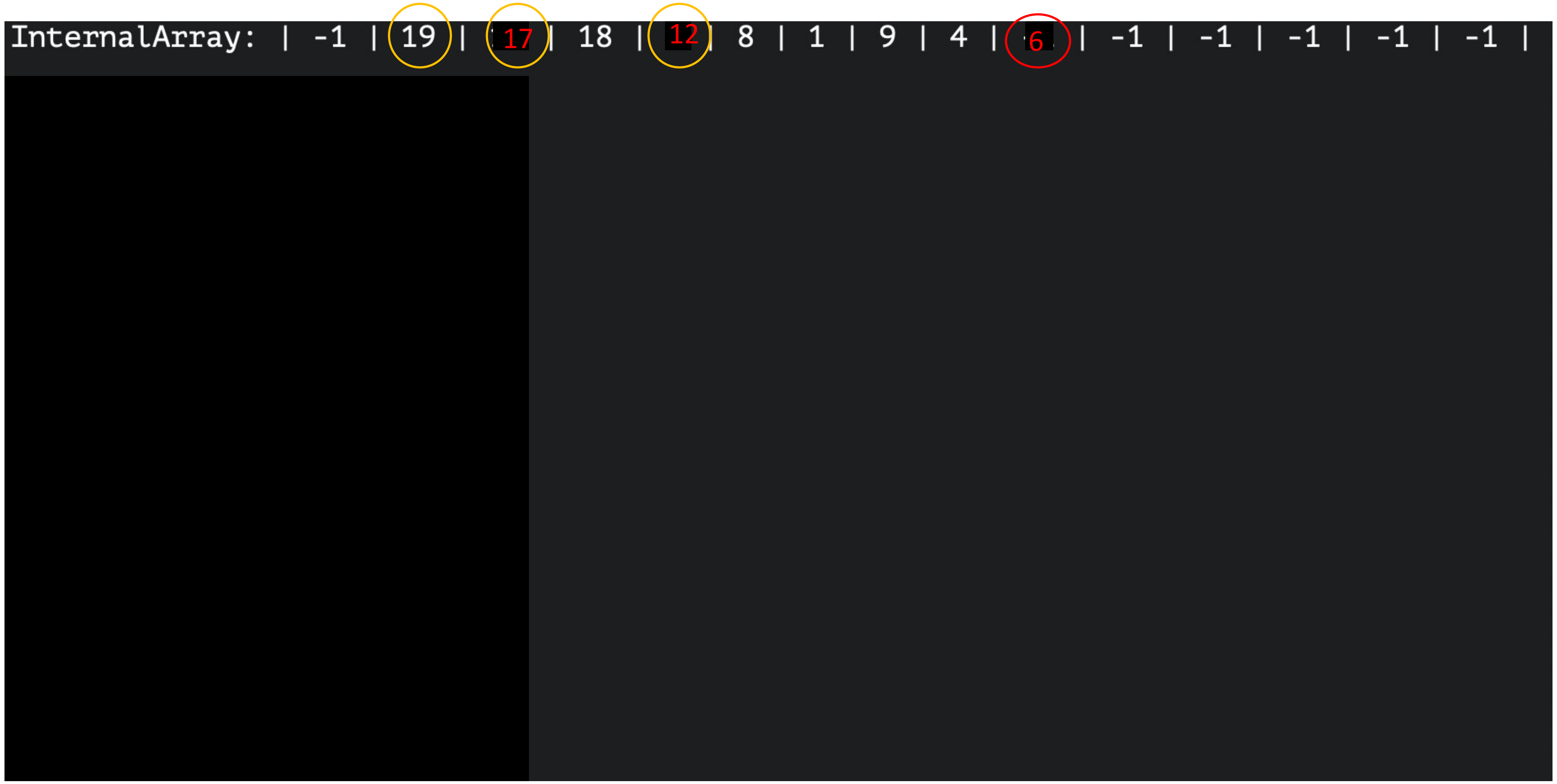
```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

myArrayHeap.add(17)

Rearrangement of values:

19 17 12 6



```
InternalArray: | -1 | 19 | 17 | 18 | 12 | 8 | 1 | 9 | 4 | 6 | -1 | -1 | -1 | -1 | -1 |
```

Add() to InternalArray

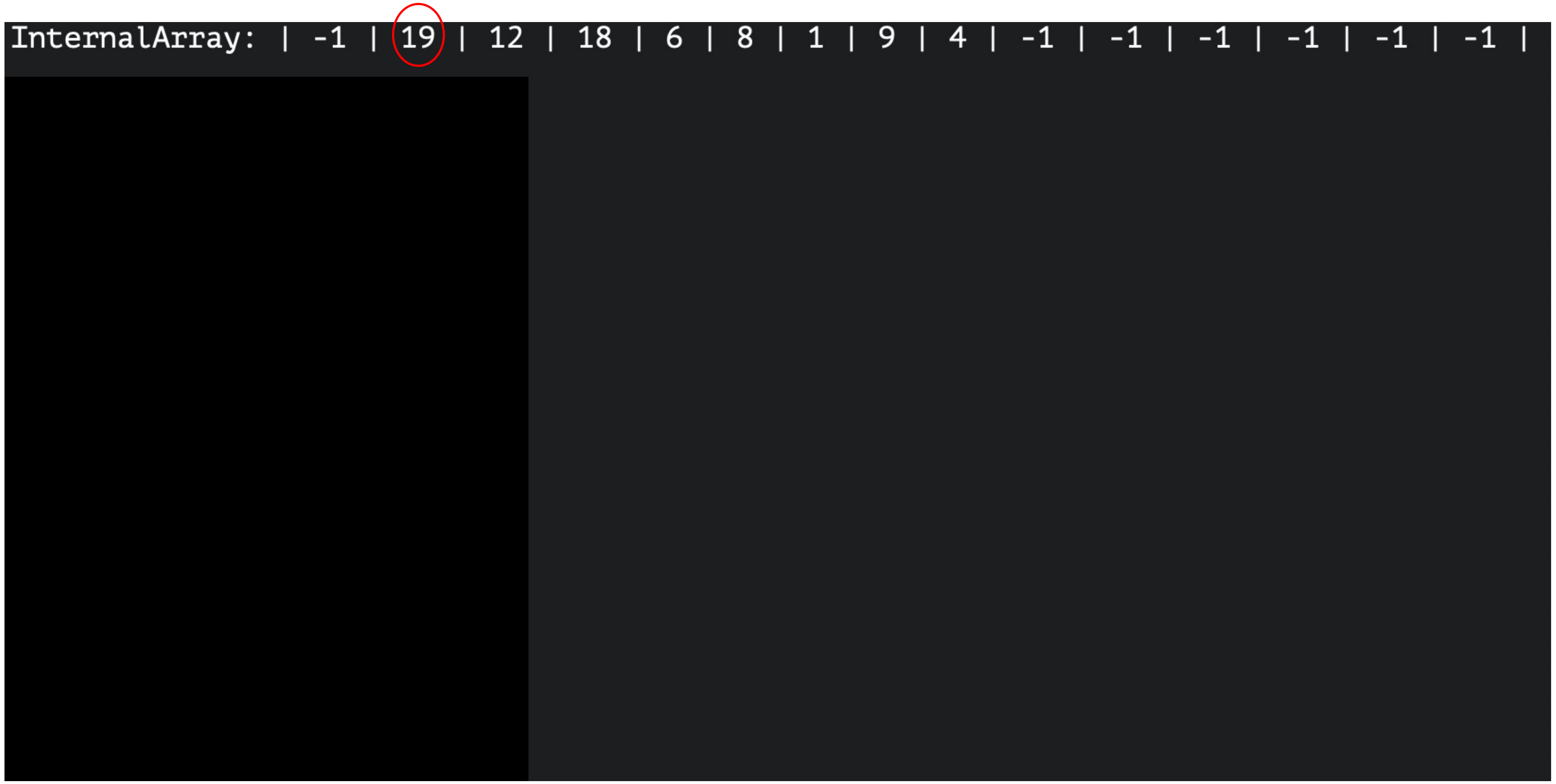
- Insertion to InternalArray when there are zero, one, or two values in the InternalArray are special cases

delete() from InternalArray

- The algorithm for deleting from InternalArray is similar to the algorithm we discussed for deleting from a Heap based on Trees

delete() from InternalArray

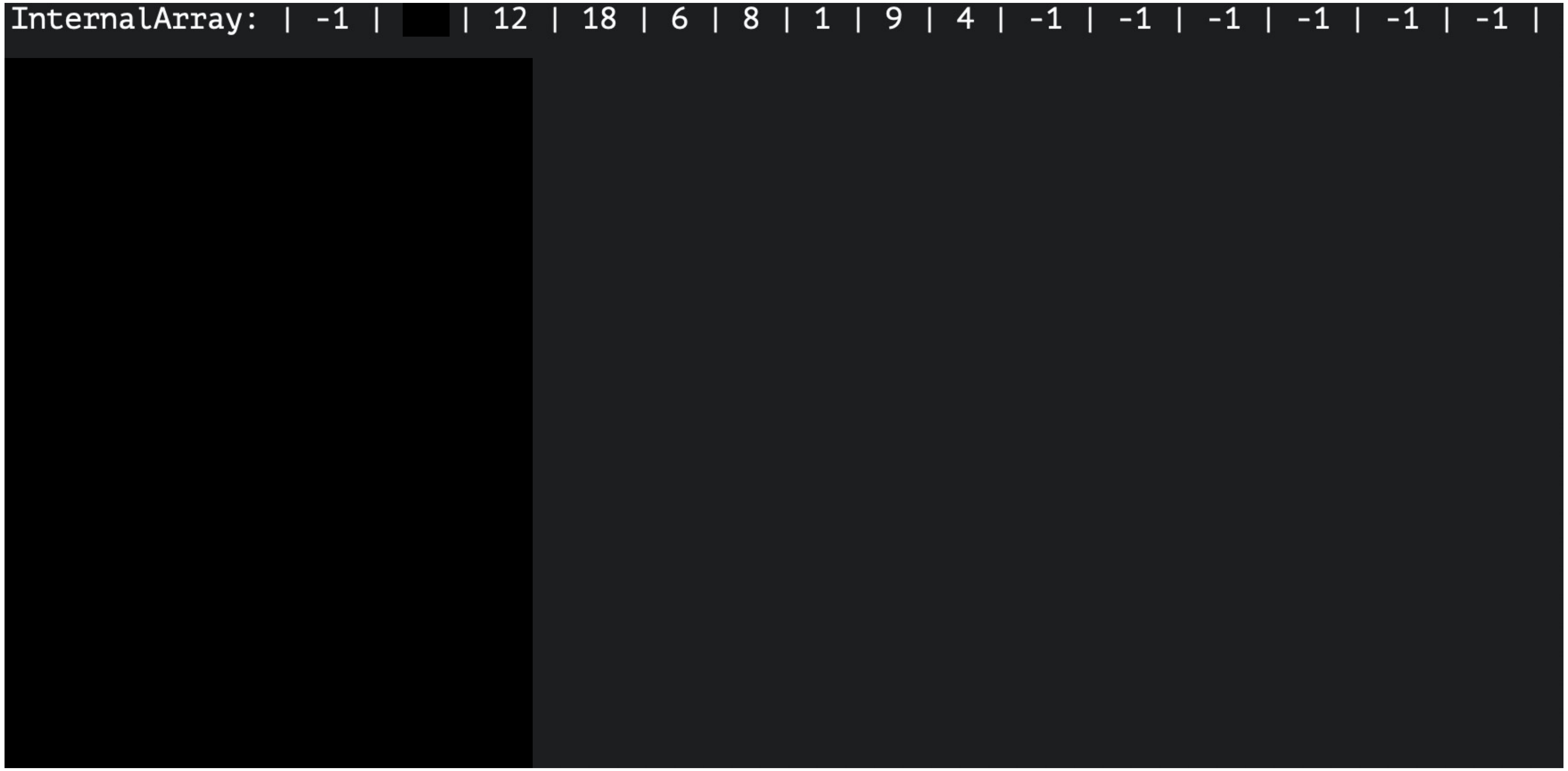
The delete method returns the value at index=1



```
InternalArray: | -1 | 19 | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

delete() from InternalArray

Values at certain indexes should be rearranged



```
InternalArray: | -1 | [redacted] | 12 | 18 | 6 | 8 | 1 | 9 | 4 | -1 | -1 | -1 | -1 | -1 | -1 |
```

delete() from InternalArray

- Deleting from InternalArray when there are only 1, 2 or 3 values existing in the InternalArray are special cases.

delete() from InternalArray

```
int index=1
int lastValue=InternalArray[LastIndex]

while(true)

      if (2*index+1> LastIndex)
            InternalArray[index]=lastValue
            break

      if (InternalArray[2*index+1]>= InternalArray[2*index])
            InternalArray[index]=InternalArray[2*index+1]
            if (InternalArray[index]<lastValue)
                  Swap InternalArray[index] and lastValue
            index=2*index+1
            continue

      if (InternalArray[2*index+1]< InternalArray[2*index])
            InternalArray[index]=InternalArray[2*index]
            if (InternalArray[index]<lastValue)
                  Swap InternalArray[index] and lastValue
            index=2*index
            continue

InternalArray[LastIndex]=-1;
LastIndex=LastIndex-1
return DeletedValue
```