Binary Search

Imagine you're looking for a word in a dictionary.

If you start flipping from page 1 and check every page one by one, it could take forever, especially if the word starts with "Z"!
In the worst case, you'd have to turn through *all* the pages.

A smarter way is to open the dictionary around the middle.
Suppose you land at "Mango."

- If your word is *before* "Mango," you know you only need to look in the first half.
- If it's *after* "Mango," you only need the second half.

Each time you open to the middle of the remaining pages and compare, cutting the possibilities in half.

By repeating this "middle-check" strategy, you find the word *much faster* — even in a giant dictionary, it would only take a handful of page flips to find your word.

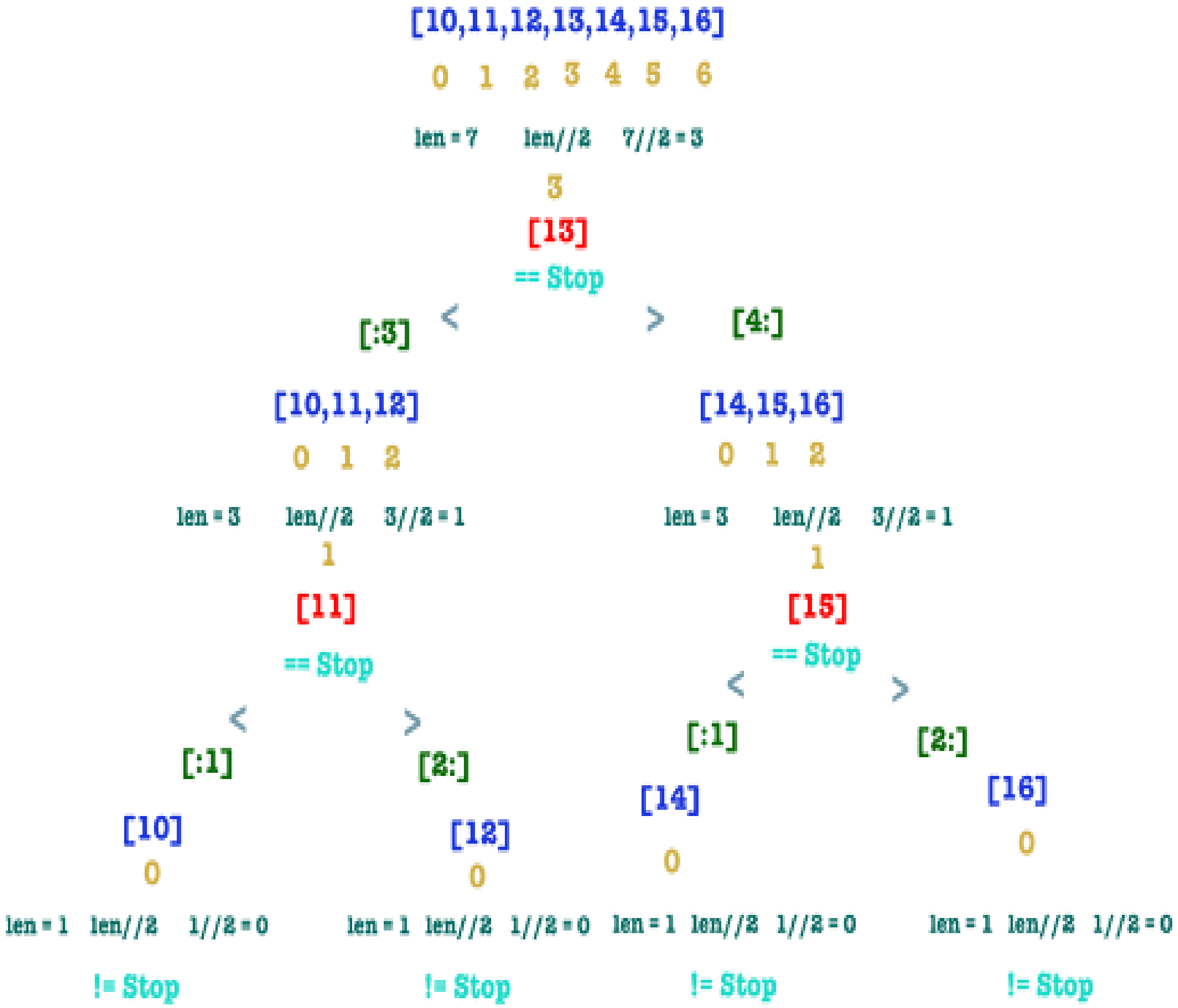
This smart searching method is called binary search.

Here's how it works:

Start by comparing the target value with the middle element of the current search range.
To find the middle, use the formula: middle = (low + high) / 2, where low is the index of the first element and high is the index of the last element in the current range.

- If the middle element matches the target, the search is successful.
- If the target is smaller than the middle element, continue searching in the **left half** of the range.
- If the target is larger, focus on the **right half**.

Repeat this process: keep recalculating the middle and narrowing down the search range by half each time.
Continue until the target is found or until there are no more elements left to check.


[10,11,12,13,14,15,16]
0 1 2 3 4 5 6
len = 7
len//2
7//2 = 3
3
[13]
== Stop
[:3]
<
>
[4:]
[10,11,12]
0 1 2
len = 3
len//2
3//2 = 1
1
[11]
== Stop
[14,15,16]
0 1 2
len = 3
len//2
3//2 = 1
1
[15]
== Stop
<
>
[:1]
[2:]
<
>
[:1]
[2:]
[10]
0
[12]
0
[14]
0
[16]
0
len = 1
len//2
1//2 = 0
!= Stop
len = 1
len//2
1//2 = 0
!= Stop
len = 1
len//2
1//2 = 0
!= Stop
len = 1
len//2
1//2 = 0
!= Stop