Recursion??

Asymptotic Notations

Is often used to describe how the size of the input data affects an algorithm’s usage of computational resource.

Asymptotic Notations

Algorithms perform f(n) basic operations to accomplish tasks.

- ✓ Identify that function
- ✓ Identify the size of problem (n)
- ✓ count number of operations in terms on n

Big-O

- Big - O notation is an approximate mathematical formula to determine how many operations are necessary to perform the search or sort.

O(1)	“constant”
O(log n)	“logarithmic”
O(n)	“linear”
O(n log n)	“supralinear”
O(n^2)	“quadratic”
O(n^c)	“polynomial”

A buggy code:

```
>>> def f():
...     print s
...     s = "Me too."
...     print s
... s = "I hate spam."
... f()
... print s
UnboundLocalError: local variable 's' referenced
before assignment
```

Never to be used

“global”

Correction with global:

```
>>> def f():
...     global s
...     print s
...     s = "That’s clear."
...     print s
... s = "Python is great!"
... f()
... print s
Python is great!
That’s clear.
That’s clear.
```

Search Algorithms

Linear Search

Binary Search

Linear Search

Example:

- List numlist contains:

17	23	5	11	2	29	3

- Searching for the value 11, linear search examines 17, 23, 5, and 11
- Searching for the value 7, linear search examines 17, 23, 5, 11, 2, 29, and 3

Linear Search

Pseudocode:

```
        set found to false; set position to -1; set index to 0
        while index < number of elements. and found is false
              if list[index] is equal to search value
                    found = true
                 position = index
              end if
              add 1 to index
end while
return position
```

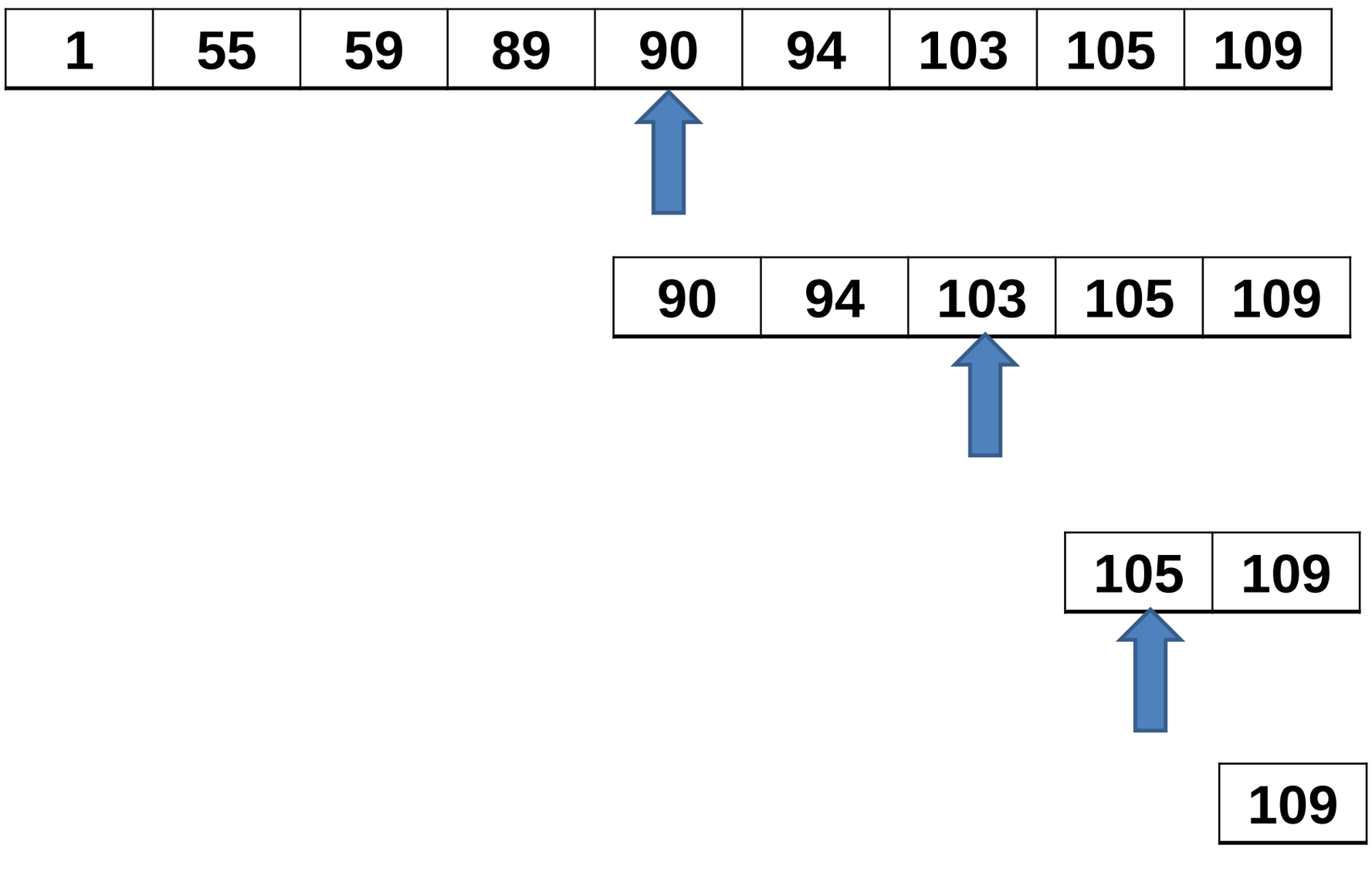
Binary Search

Requires list elements to be in order

1. Divides the list into three sections:
    - middle element
    - elements on one side of the middle element.
    - elements on the other side of the middle element
2. If the middle element is the correct value, done. Otherwise, go to step 1. using only the half of the list that may contain the correct value.
3. Continue steps 1. and 2. until either the value is found or there are no more elements to examine

Binary Search

Target is 109



Binary Search

Pseudocode:

```
If there are no more items to consider then
    Return -1
Else
    Set midpoint to (last + first) / 2
    If the item at index midpoint = = the target item then
        Return midpoint
    Else if the item at index midpoint > the target item then
        Return search the left half of the list (from indices
        first to midpoint - 1)
    Else
        Return search the right half of the list (from indices
        midpoint + 1 to last)
```

Binary Search

- How long (worst case) will it take to find an item in a list 30,000 items long?

$2^{10} = 1024$ $2^{13} = 8192$

$2^{11} = 2048$ $2^{14} = 16384$

$2^{12} = 4096$ $2^{15} = 32768$

- So, it will take only 15 tries!
- Binary search runs in log(n) time.

Binary Search

Tradeoffs:

- Benefits:
  Much more efficient than linear search. For list of n elements, performs at most log (n) comparisons
- Disadvantage:
  Requires the list elements to be sorted.

to be continued...