Whiteboard

Wednesday, April 1, 2020 9:02 AM

Example: Guess a number between 1 & 100

3 is too low

50 is too high

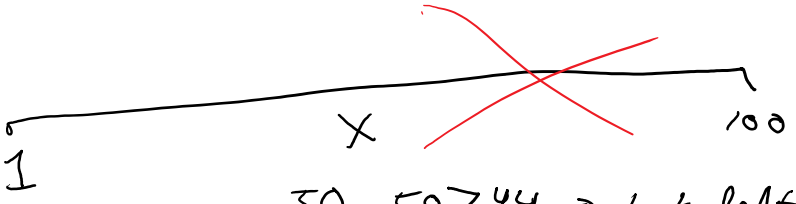
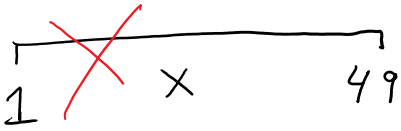
37 is too low

45 is too high

42 is too low

44 is right!

Search for 44

Step	
1	 50, 50 > 44 ⇒ look left
2	 25, 25 < 44 ⇒ look right
3	 37, 37 < 44, look right
4	 43, 43 < 44, look right
5	 46, 46 > 44, look left
6	

44, 44 == 44, done!

⇒ 6 steps using binary search

How many steps would we need for linear search?

⟶ 44

When searching for strings, we compare letter by letter

"apples" < "pear" # Returns True

["apples", "bananas", "pear"] search value: "lime"

How can we compute the midpoint each step?

$$mid = \left\lfloor \frac{a+b}{2} \right\rfloor$$

where a is the interval start
b is the end

floor (truncates to integer)

Pattern at each step

Let low be the start index of interval
high be the end index
mid be the middle index

look left ⇒ 

low stays the same
high = mid − 1

look right ⇒ 

low = mid + 1
high stays the same

What happens when an element is not in the list?

high < low

high < low

Search for 99 in [-20, -4, 44, 58, 99, 145]

L

	0	1	2	3	4	5
	-20	-4	44	58	99	145

Step	low	high	mid	L[mid]
1	0	5	$2 = \frac{5+0}{2} = 2.5 \downarrow = 2$	44
2	3	5	$4 = \frac{5+3}{2} = 4$	99 found it!

Search for 30 in [-20, -4, 44, 58, 99, 145]

	0	1	2	3	4	5
	-20	-4	44	58	99	145

Step	low	high	mid	L[mid]
1	0	5	2	44 ← too high, look left
2	0	1	0	-20 ← too small, right
3	1	1	1	-4 ← too small, right
4	2	1	low > high	

→ item not found!

```
binarySearch(x, L):
    low = 0
    high = len(L) - 1
    while low <= high:
        mid = int((low + high)/2)
        if x < L[mid]:  # look left
            high = mid - 1
        elif x > L[mid]:  # look right
            low = mid + 1
        else:
            return mid
```

return -1

Runtime analysis of binary search

$O(\log N)$

step 1 : $N/2$

step 2 : $(N/2)/2$

step 3 : $((N/2)/2)/2$

⋮

step k : $N/2^k$