BINARY SEARCH

Based on PPT by David G. Sullivan
For a given task, there may be more than one algorithm that works.

When choosing among algorithms, one important factor is their relative efficiency:

- **Space efficiency:** How much memory an algorithm requires.
- **Time efficiency:** How quickly an algorithm is executed:
  - How many “operations” it performs.
Consider the problem of finding a phone number in a phone book

Let’s informally compare the time efficiency of two algorithms
- Look at every page of the phone book from left to right until number is found

- If there were 1000 pages in the phone book, how many pages would we look at in the worst case scenario?

- What if there were 1,000,000?
ALGORITHM 2

- Divide the book in half
- If the number is in first half, toss out the second half
- Else if number is in second half, toss out the first half
- Repeat above steps until number is found

If there where 1,000 pages in the phonebook, how many pages would we look at in the worst case?

What if there were 1,000,000 pages?
The phonebook problem is one example of a common task: searching for an item in a collection of data.

Algorithm 1 is known as *sequential search*
- Also known as *linear search*

Algorithm 2 is known as *binary search*
- Only works if the items in the data collection are sorted
ANOTHER EXAMPLE: CAN YOU FIND THE NUMBER 38?
ANOTHER EXAMPLE: CAN YOU FIND THE NUMBER 38?
ANOTHER EXAMPLE: CAN YOU FIND THE NUMBER 38?
Y = B^x

is the same as

\log_B Y = X
In the worst case scenario, it takes at most $\log_2 n$ steps to find your item using binary search. Why?

- After every step, we cut in half the size of the list we’re looking at
- $n$, $n/2$, $n/4$
- Divide by 2 each time

This is a lot faster than using sequential search, which takes $n$ steps in the worst case.

Using binary search in the phonebook problem, leads to looking at $\log_2 1000000 = \sim 20$ in the worst case.