Searching and Sorting

A Glimpse of Algorithms

Remember the Game “GuessANum”?

```
import random
def guessANum():
    secret = random.randint(0,99)
    guess = -1
    print('I have a number in mind between 0 and 99')
    while guess != secret:
        guess = int(input('Guess a number: '))
        if guess == secret:
            print('Bingo!!! The answer is {}'.format(secret))
        elif guess < secret:
            print('Your number is too small')
        else:
            print('Your number is too big')

guessANum()
```

```
>>> guessANum()
I have a number in mind between 0 and 99
Guess a number: 66
Your number is too big
Guess a number: 30
Your number is too small
Guess a number: 44
Your number is too small
Guess a number: 55
Your number is too big
Guess a number: 49
Your number is too big
Guess a number: 47
Your number is too big
Guess a number: 46
Your number is too big
Guess a number: 45
Bingo!!! The answer is 45
>>>
```

Strategies

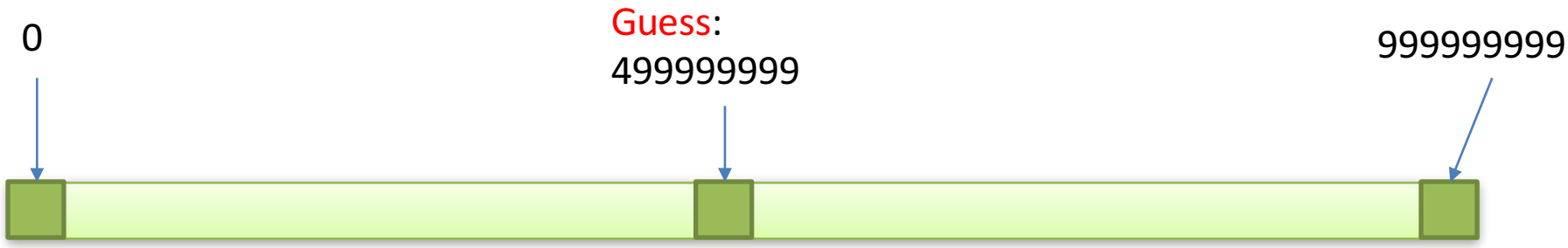
- Strategy 1
  - Randomly guess
    - 44, 31, 89, 90, 54, 31, 2, 5, 89, 10, …
- Strategy 2
  - Guess one by one starting from 1
    - 1, 2, 3, 4, 5, 6, 7, …

What if

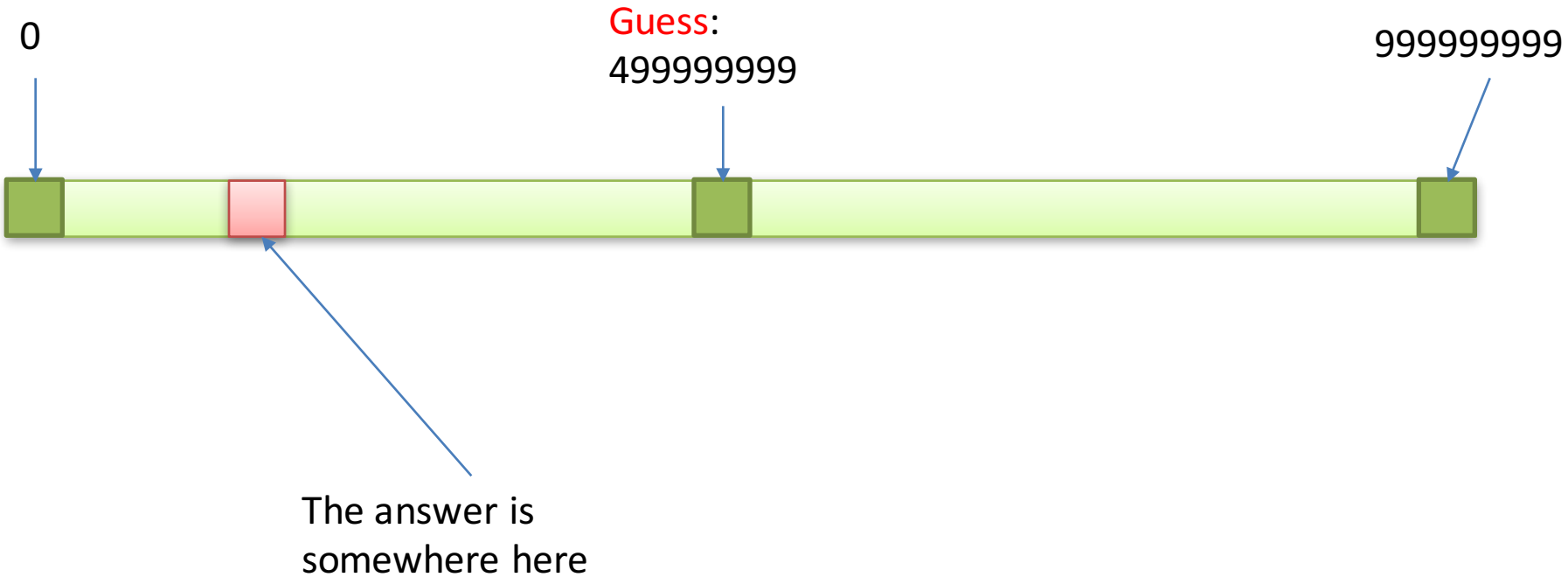
```
I have a number in mind between 0 and 9999999
Guess a number:
```

- If you anyhow guess, how many times do you need?
  - Strategy 1
    - A lot, can be even infinite
  - Strategy 2
    - Can be as many as N = number of numbers
- Is there any better strategy?

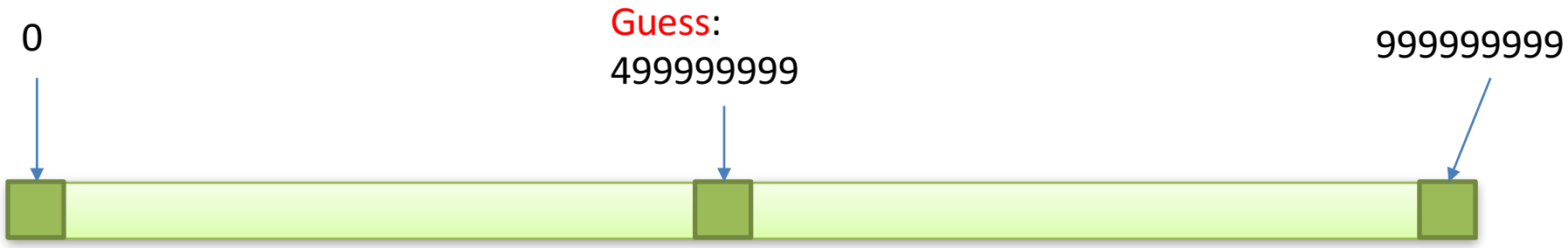
Strategy 3



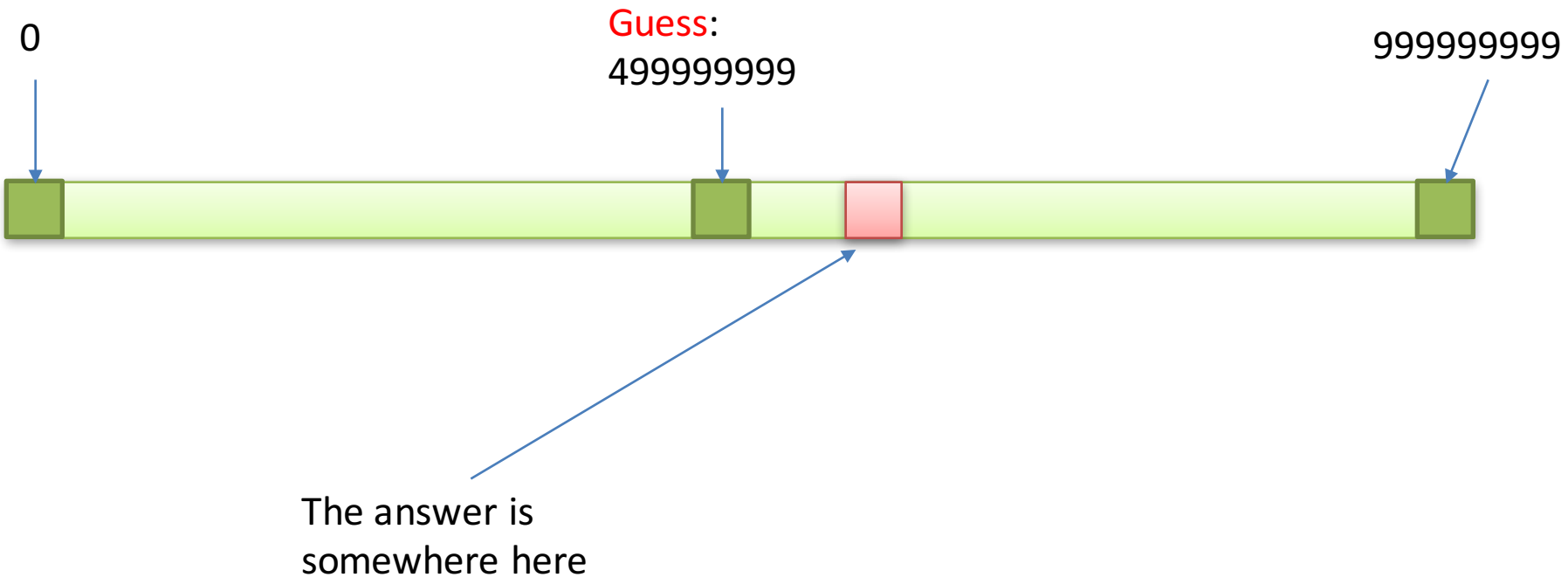
- If the guess is too big?



Strategy 3

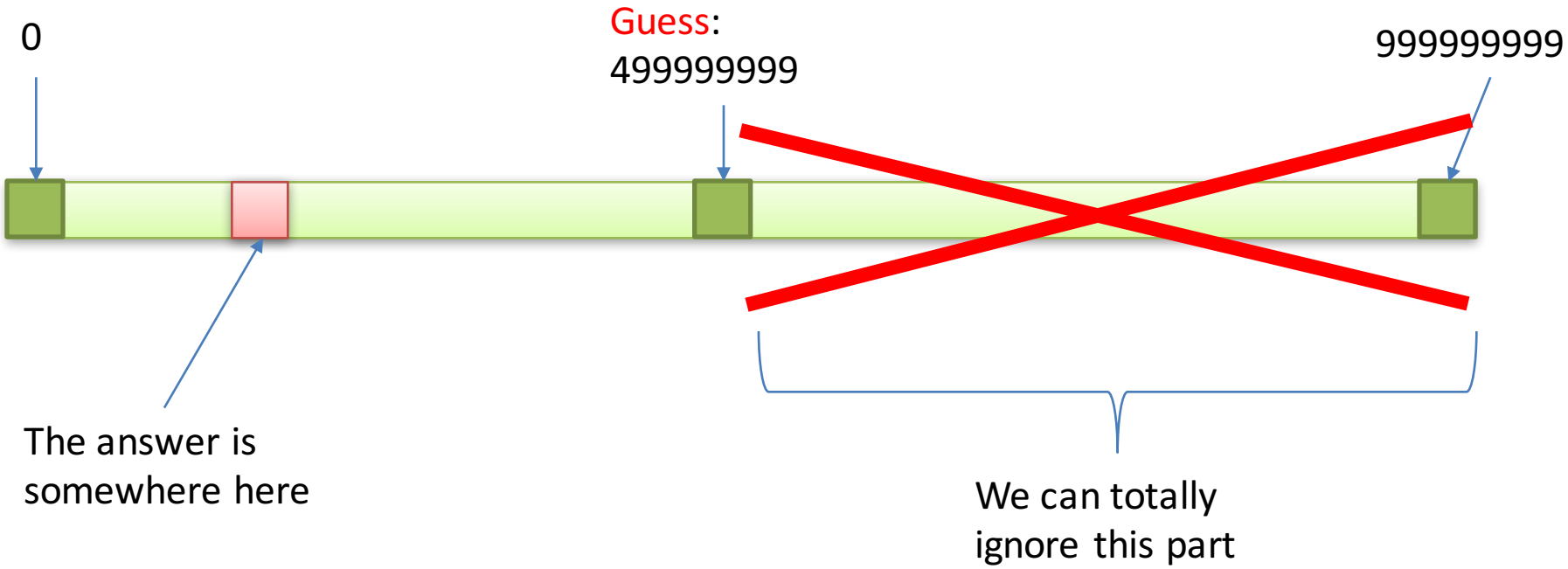


- If the guess is too small?



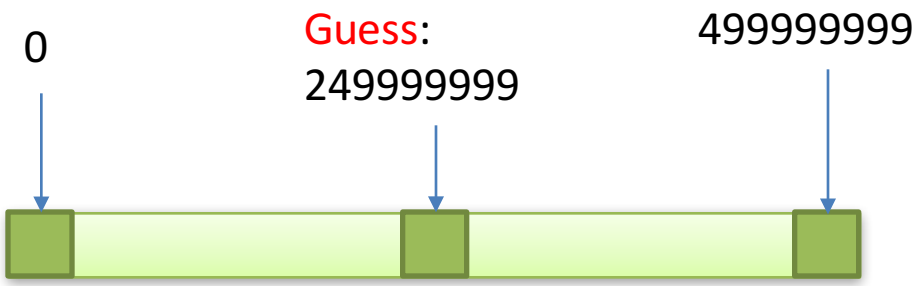
Strategy 3

- If the guess is too big?



Strategy 3

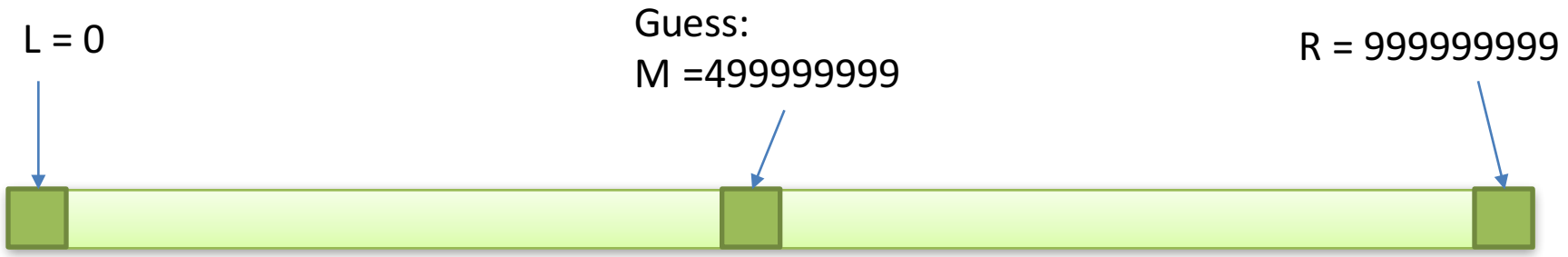
- Repeat!



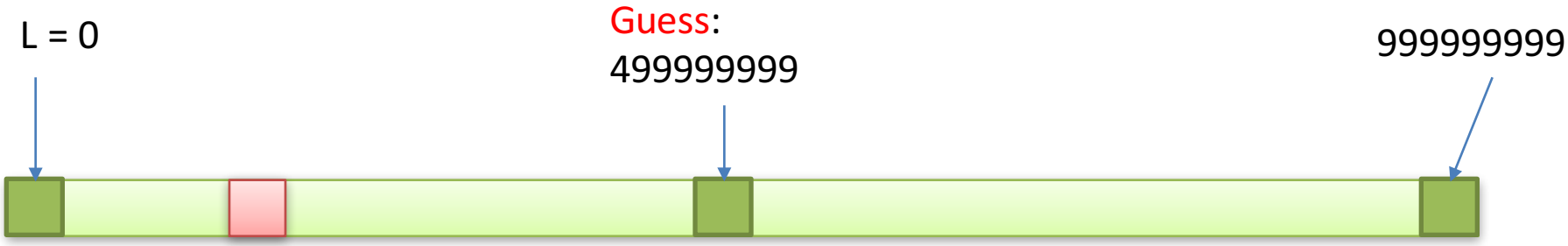
Strategy 3

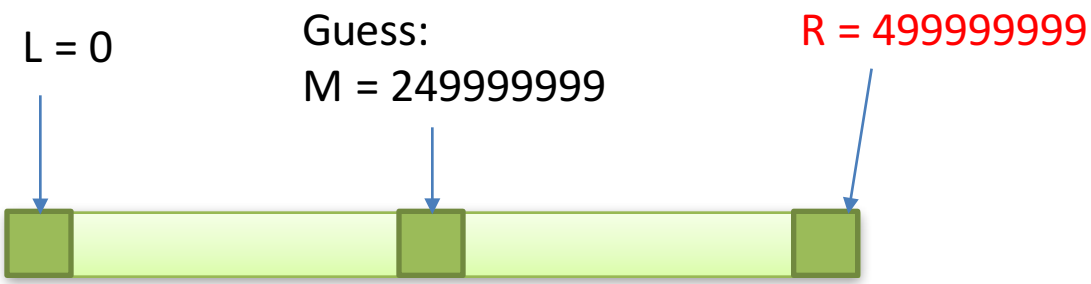
- So the number will be between 0 and 9999999
- Let it be L = 0, R = 9999999
- Repeat
  - Guess M = (L + R) / 2
  - If the guess is too big. R = M
    - Else L = M
- Repeat until L == M or the number is found

Strategy 3



- If the guess is too big?





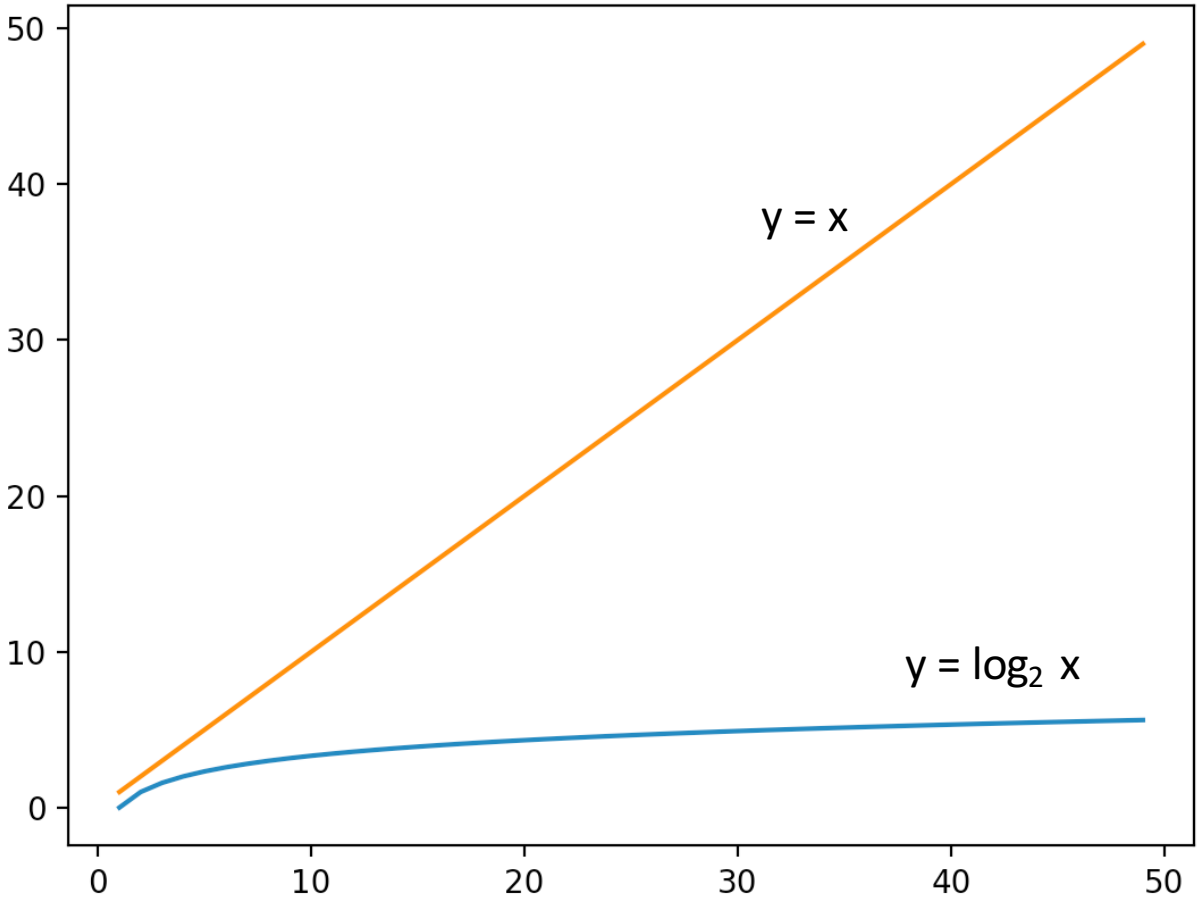
How Many Steps?

- If the guess is between 0 to N
- Strategy 1 (guess randomly)
  - Worst case: infinite
- Strategy 2 (linear)
  - Worst case: N
- Strategy 3
  - Binary search
  - Log_2 N

Why log N?

- If you have N items, and each time you reduce the search space by half
- How many steps do you need to half until the search space is only 1?
  - Let S be the number of steps
    - $N \times (\frac{1}{2})^S = 1$
    - $N = 2^S$
    - $\log N = S \log 2$
    - $S = \log N / \log 2$

Order of Growth



Ok, now I know how to guess a number quickly

Computation

- Usually we rely on the computer to do tasks that are
  - repetitive
    - e.g. computing a function with series like cosine

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$
$$= \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$$

  - involving a large volume of data
    - Searching for a NRIC in the many millions of Singaporeans

Search?

- Simple problem
- Let's say we have a file containing all the names of Singaporeans
  - 3.4millions in 2015
- You want to check
  - Is "Alan Cheng Ho Lun" a Singaporean?

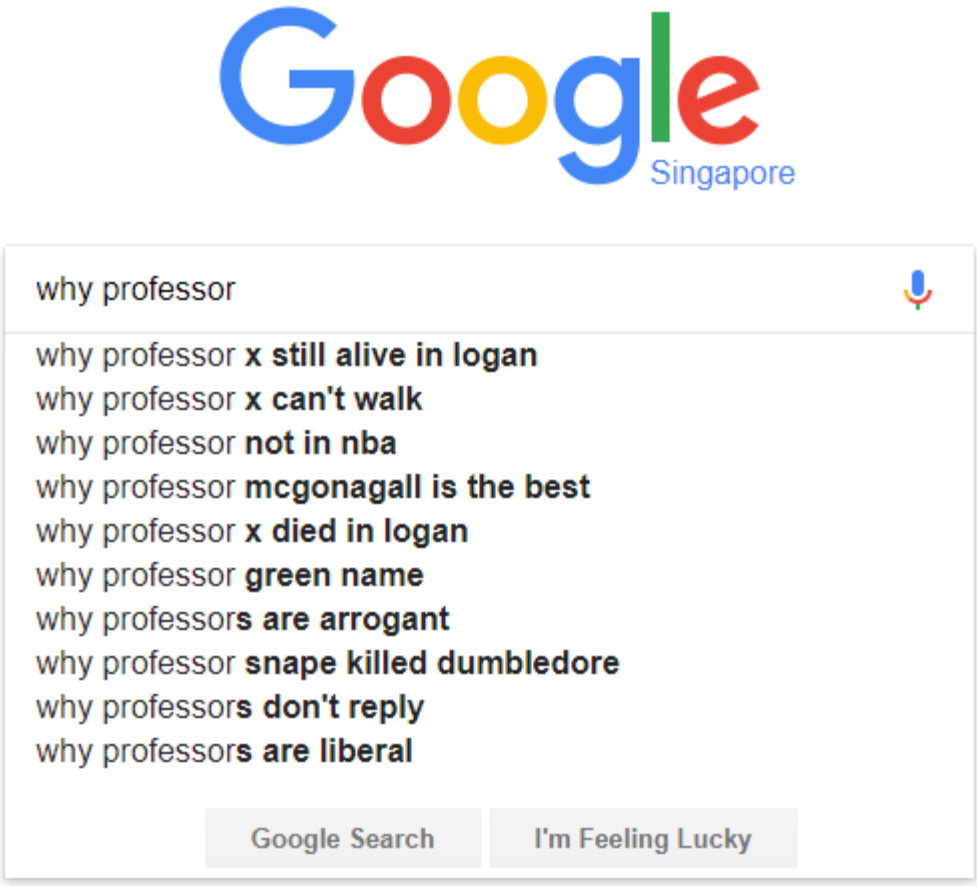
Linear Search

```
all_SG = ['Jeanette Aw', 'Vivian Balakrishnan',
          'Tharman Shanmugaratnam', 'Taufik Batisah',
           'Daniel Bennett', 'Yusof bin Ishak',
           'Agu Casmir', 'Chan Heng Chee' 'Chan Chun Sing',
           'Kit Chan']

def linear_search(list,name):
    for i in list:
        if i == name:
            print(name + " Found!")
            return
    print(name + " Not Found!")


linear_search(all_SG,'Kit Chan')
linear_search(all_SG,'Alan Cheng')
```

Google?


Google
Singapore
why professor
why professor x still alive in logan
why professor x can't walk
why professor not in nba
why professor mcgonagall is the best
why professor x died in logan
why professor green name
why professors are arrogant
why professor snape killed dumbledore
why professors don't reply
why professors are liberal
Google Search
I'm Feeling Lucky

How much data does Google handle?

- About 10 to 15 Exabyte of data
  - 1 Exabyte(EB)= 1024 Petabyte(PB)
  - 1 Petabyte(PB) = 1024 Terabytes(TB)
  - 1 Terabyte(PB) = 1024 Gigabytes(TB)
    - = 4 X 256GB iPhone
- So Google is handling about 60 millions of iPhones

How fast is my desktop?

```
import time

def create_list(n):
    start_time = time.time()
    l = [i for i in range(n)]
    end_time = time.time()
    print('Duration = ' +
          str(end_time-start_time) + ' s')

create_list(1000)
create_list(1000*1000)
create_list(1000*1000*100)
```

Return the time in seconds since the epoch (1970/1/1 00:00) as a floating point number.

Create 100M of numbers, estimated to be 400MB of data

Output:

```
Duration = 0.0 s
Duration = 0.047691822052001 95 s
Duration = 6.026865720748901 s
```

Let's calculate

- 400M of data needs 6 seconds
- 15 Exabyte of data needs how long?
  - 15 EB = 15 x 1024 x 1024 x 1024 x 1024 MB
  - To search through 15EB of data…..
    - 7845 years…..
- If we do it with Binary Search
  - $\log_2$ (15EB) = 43 steps!!!!!

Binary Search

- But we cannot do a binary search on this list
- Why?

```
all_SG = ['Jeanette Aw', 'Vivian Balakrishnan',
          'Tharman Shanmugaratnam', 'Taufik Batisah',
           'Daniel Bennett', 'Yusof bin Ishak',
           'Agu Casmir', 'Chan Heng Chee' 'Chan Chun Sing',
           'Kit Chan']
```

- Because it’s not sorted

Binary Search

- In Python, you can sort a list by

```
>>> all_SG.sort()
>>> all_SG
['Agu Casmir', 'Chan Chun Sing', 'Chan Heng Chee', 'Dani
el Bennett', 'Jeanette Aw', 'Kit Chan', 'Taufik Batisah'
, 'Tharman Shanmugaratnam', 'Vivian Balakrishnan', 'Yuso
f bin Ishak']
```

- Now we can do a binary search

Searching

```
all_SG.sort()

def binary_search(data,item):
    L = 0
    R = len(data)-1
    while(L!=R):
        M = int((L+R)/2)
        if data[M] == item:
            print('Found at position ' + str(M))
            return
        elif data[M] > item:
            R = M
        else:
            L = M + 1
    print('Item not found')
```

Let's Create a Large Data Set to Try

```
N = 10000000

list_of_num = [random.randint(1,N*2) for i in range(N)]

list_of_num.sort()
```

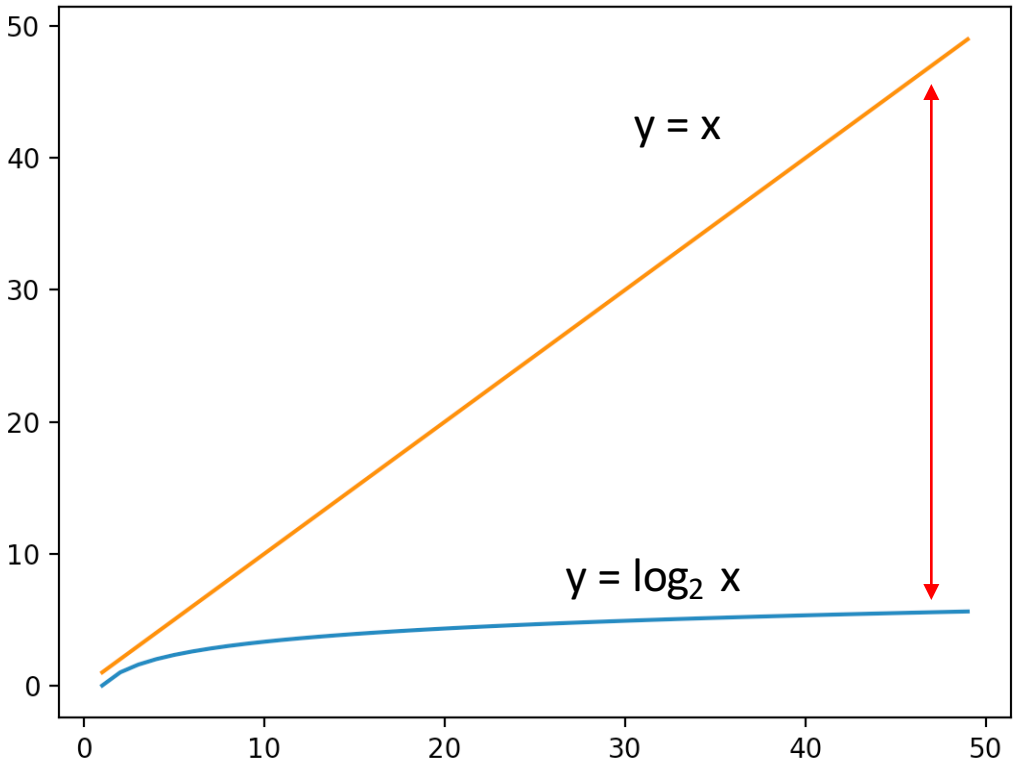
- Try it on the two searches, with durations

```
33 Not Found!
Linear search duration = 1.7579751014709473 s
Item not found
Linear search duration = 0.03387212753295898 s
```

- 52 times faster!

Speed Improvement

- If the number of data N is even larger, the improvement in speed is even greater



The Magic?

- Binary search vs Linear search
- However, one step back
  - The data has to be sorted
  - The time needed for sorting is $n \log n$
  - Slower than linear search
  - But we only have to do it once
  - Is it?
    - Can we assume that we can sort it once for all?

Data Structure and Algorithm

- But what if the data is dynamic?
- Even after sorted, there will be new items added or old items removed from the list?
- How do we maintain a dynamic list that is sorted all the time?
- This is the study of Data Structure
  - Basically, how fast we can compute if the data size N is very large

Data Structure

- How do you organize your data to improve your performance
  - Sorted list, trees, graphs, heaps, hash tables, etc.

- British Library
  - 170m+
- Library of Congress
  - 164m+

Algorithm

- What is the way I do my computation?
- Named for al-Khwarizmi (780-850)
  - Persian mathematician

Conclusion

- Algorithm and Data Structure
  - How to organize data
  - What is the way to compute
  - When the data size N is very large
- If you understand lecture today, you are a computer scientist