UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC201 Computer Science I for Majors

Lecture 24 – Algorithmic Analysis

Last Class We Covered

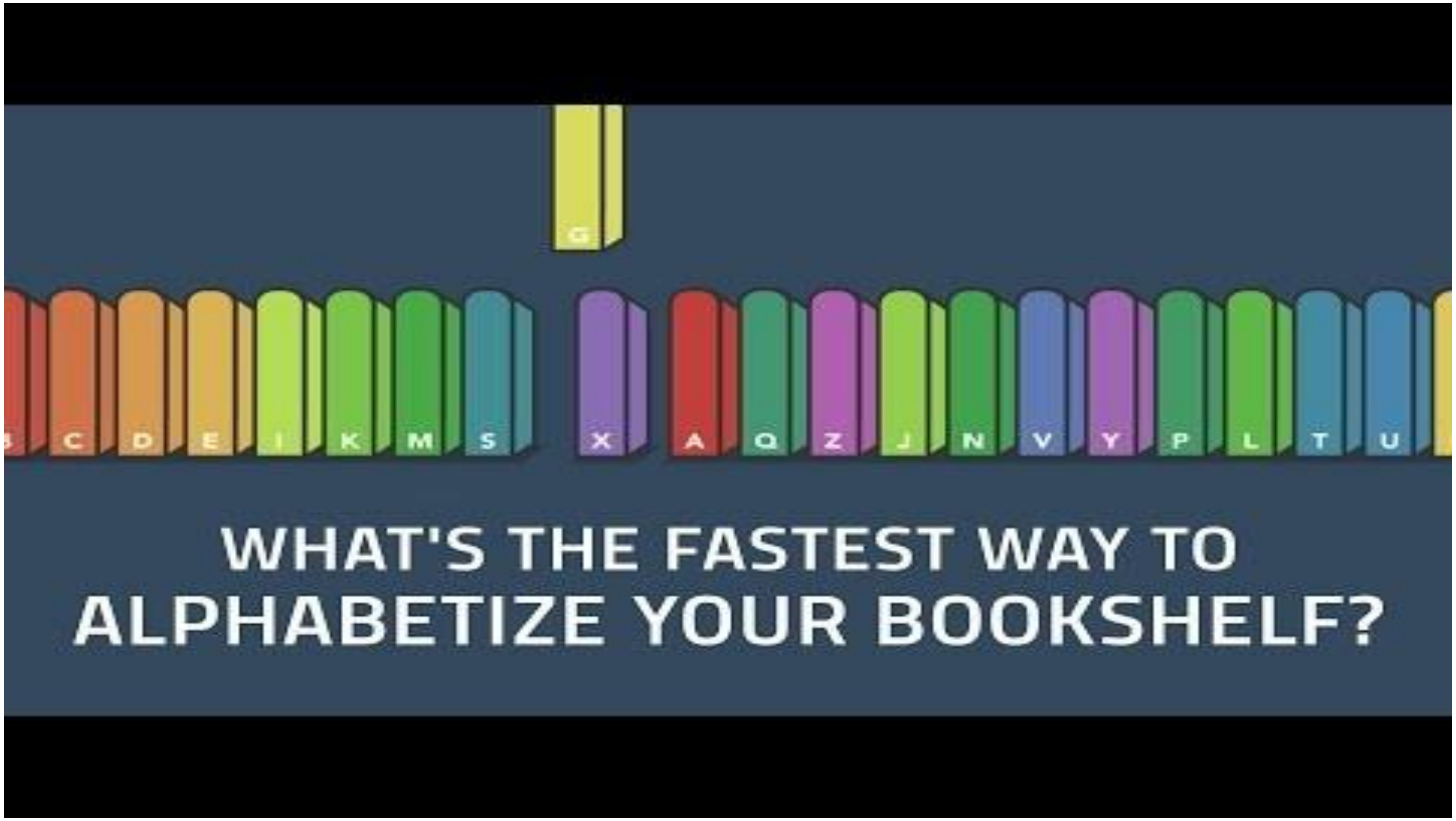
- Sorting algorithms
  - Selection Sort
  - Bubble Sort
  - Quicksort
- “Run” time
  - Why it’s important

Any Questions from Last Time?

Today’s Objectives

- To discuss “run time” of algorithms
  - Why one algorithm is “better” than another
- To learn about asymptotic analysis
  - What it is
  - Why it’s important
  - How to calculate it

Alphabetizing a Bookshelf


G
B C D E I K M S X A Q Z J N V Y P L T U
WHAT'S THE FASTEST WAY TO ALPHABETIZE YOUR BOOKSHELF?

Video from https://www.youtube.com/watch?v=WaNLJf8xzC4

Review: Algorithm Run Time

Example: Sum of All Products

- Say we have a list, and we want to find the sum of everything in that list multiplied by everything else in that list
  - So if the list is [1, 2, 3], we want to find the value of:
  - `1*1 + 1*2 + 1*3 + 2*1 + 2*2 + 2*3 + 3*1 + 3*2 + 3*3`
- As an exercise, try writing this function!

```
def sumOfAllProducts(myList):
```

Exercise Solution

```
def sumOfAllProducts(myList):
  result = 0
  for item in myList:
    for item2 in myList:
      result += item * item2
  return result
```

Run Time for Sum of All Products

- How many multiplications does this have to do for a list of 3 things?
    - For 3 things, it does 9 multiplications
  - 8 things?
    - For 8 things, it does 64 multiplications
  - 16 things?
    - For 16 things, it does 256 multiplications
- In general, if you give it a list of size **N**, you'll have to do $\mathbf{N}^2$ multiplications!

Run Time for Search

- If I am looking through a list of **`N`** items…
- How long does linear search take?
  - **`N`**
- How long does binary search take?
  - **`lg(N)`**

Different Run Times

- As our list gets bigger and bigger, which of the search algorithms is faster?
  - Linear or binary search?
- How <u>much</u> faster is binary search?
  - A lot!
  - But exactly how much is "a lot"?

Asymptotic Analysis

What is "Big O" Notation?

- Big O notation is a concept in Computer Science
  - Used to describe the complexity (or performance) of an algorithm
- Big O describes the **worst-case** scenario
  - Big Omega (Ω) describes the best-case
  - Big Theta (Θ) is used when the best and worst case scenarios are the same

Asymptotic Analysis

- For a list of size **`N`**, linear search does **`N`** operations. So we say it is **`O(N)`** (pronounced "big Oh of n")
- For a list of size **`N`**, binary search does **`lg(N)`** operations, so we say it is **`O(lg(N))`**
- For a list of size **`N`**, our sum of products function does N^2 operations, which means it is $O(N^2)$

- The function inside the **`O()`** parentheses indicates how fast the algorithm scales

Example

- What is the big O of the following, given a list of size **N**:

```
for i in myList:
    for j in myList:
        for k in myList:
            print(i*j*k)
```

- This will be **O(N^3)**

Worst-case vs Best-case

- Why differentiate between the two?

- Think back to selection sort
  - What is the best case for run time?
  - What is the worst case for run time?
- They're the same!
  - Always have to find each minimum by looking through the entire list every time – $\Theta(N^2)$

Worst-case vs Best-case

- What about bubble sort?
  - What is the best case for run time?
  - What is the worst case for run time?

- Very different!
  - Best case, everything is already sorted – **Ω(N)**
  - Worst case, it's completely backwards – **O(N^2)**

Worst-case vs Best-case

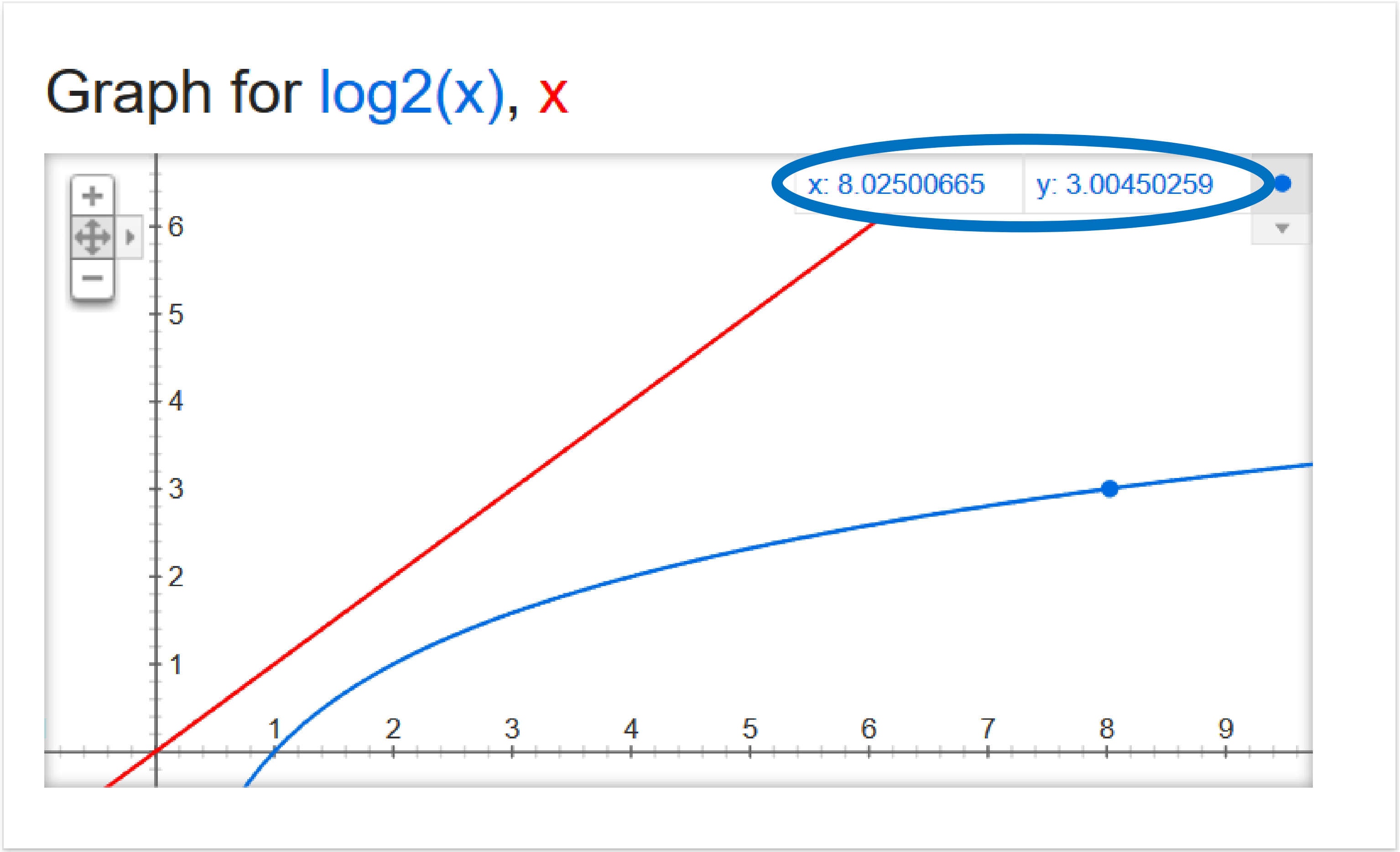
- This is why, even though selection sort and bubble sort have the same run times, bubble sort often runs much faster

- How does this apply to linear search and binary search? What are the best and worst run times for these?

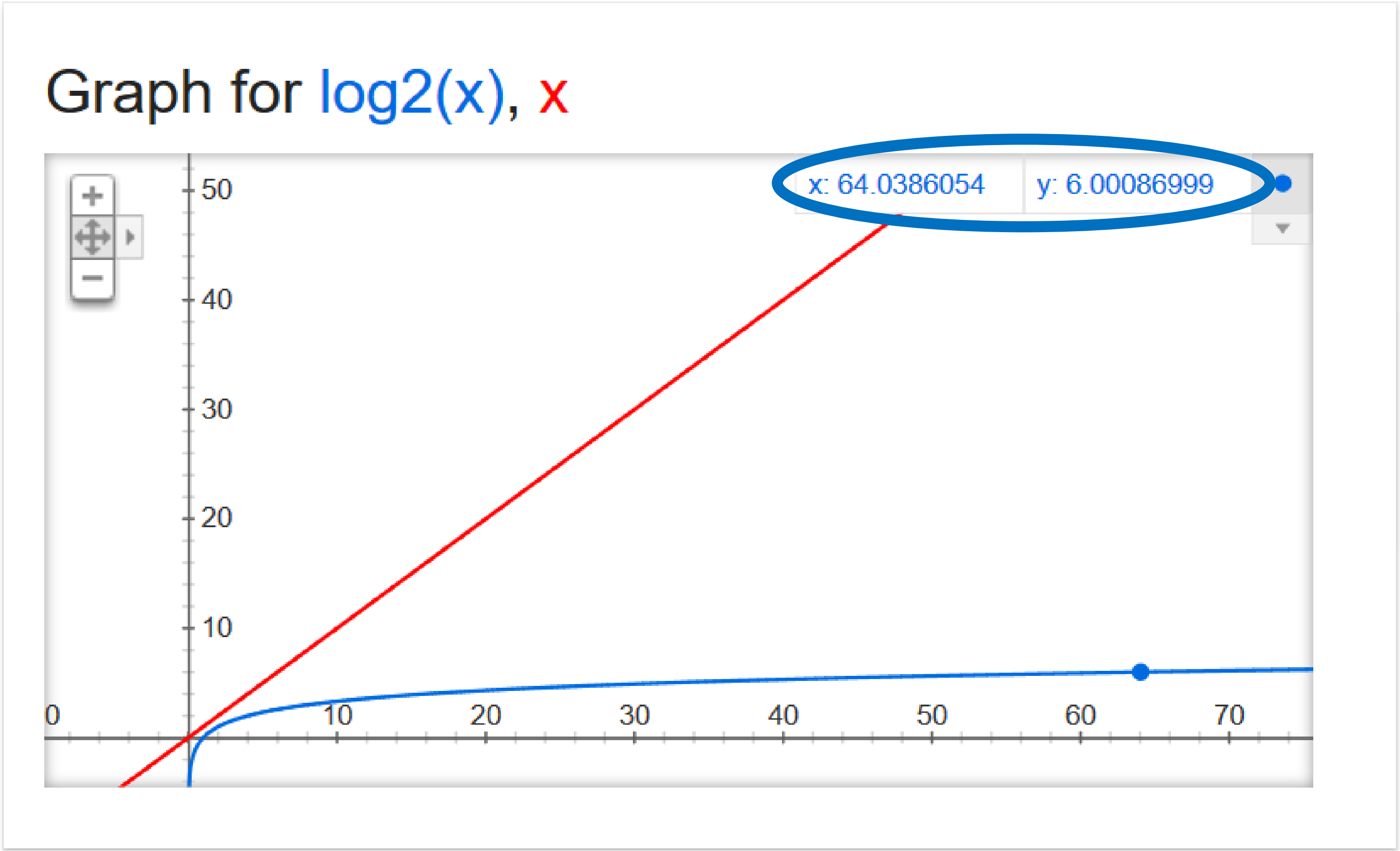
Search Run Times

- Linear search:
  - Best case: $\Omega(1)$
  - Worst case: $O(N)$
- Binary search:
  - Best case: $\Omega(1)$
  - Worst case: $O(\lg(N))$

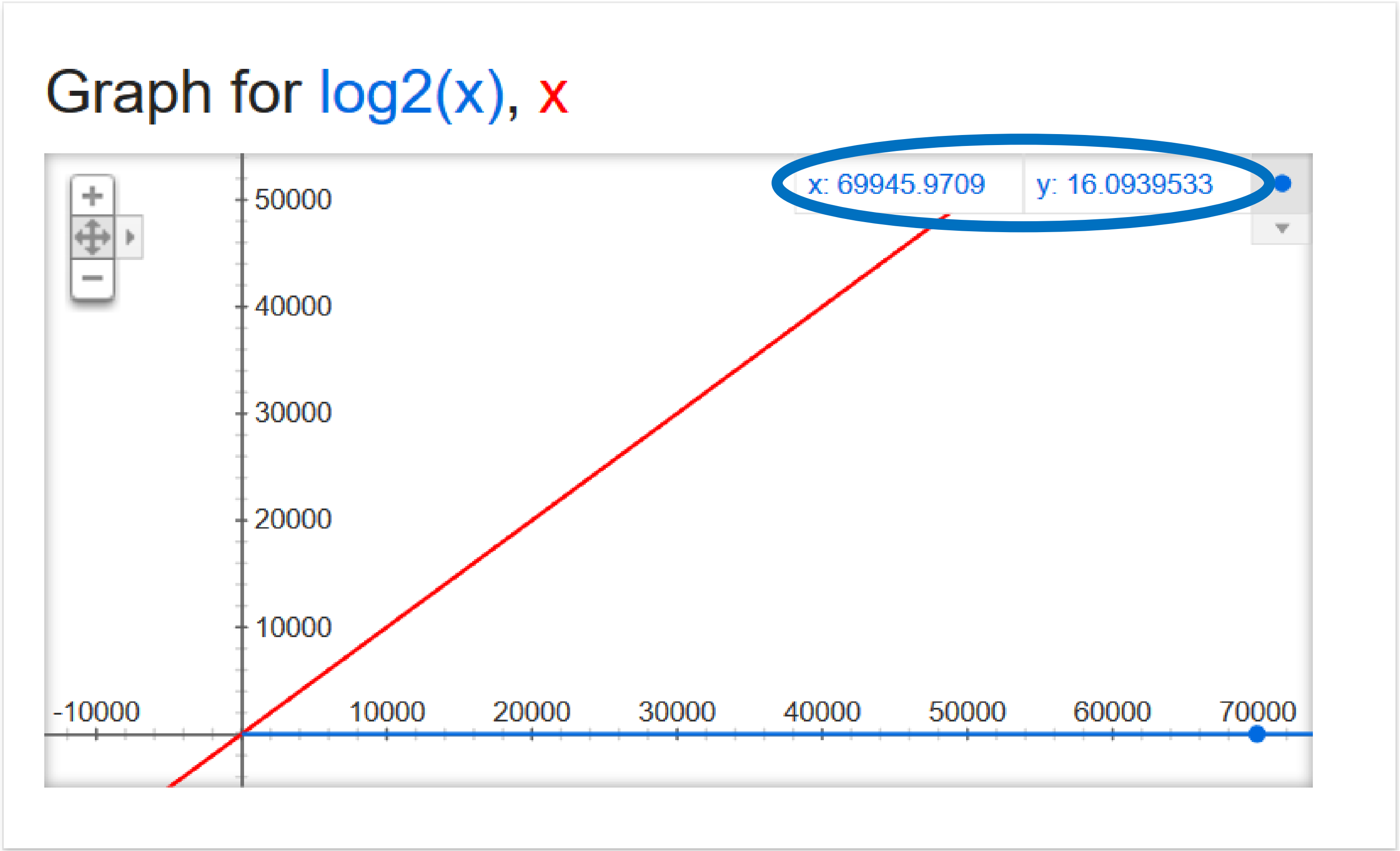
Why Care?


Graph for log2(x), x

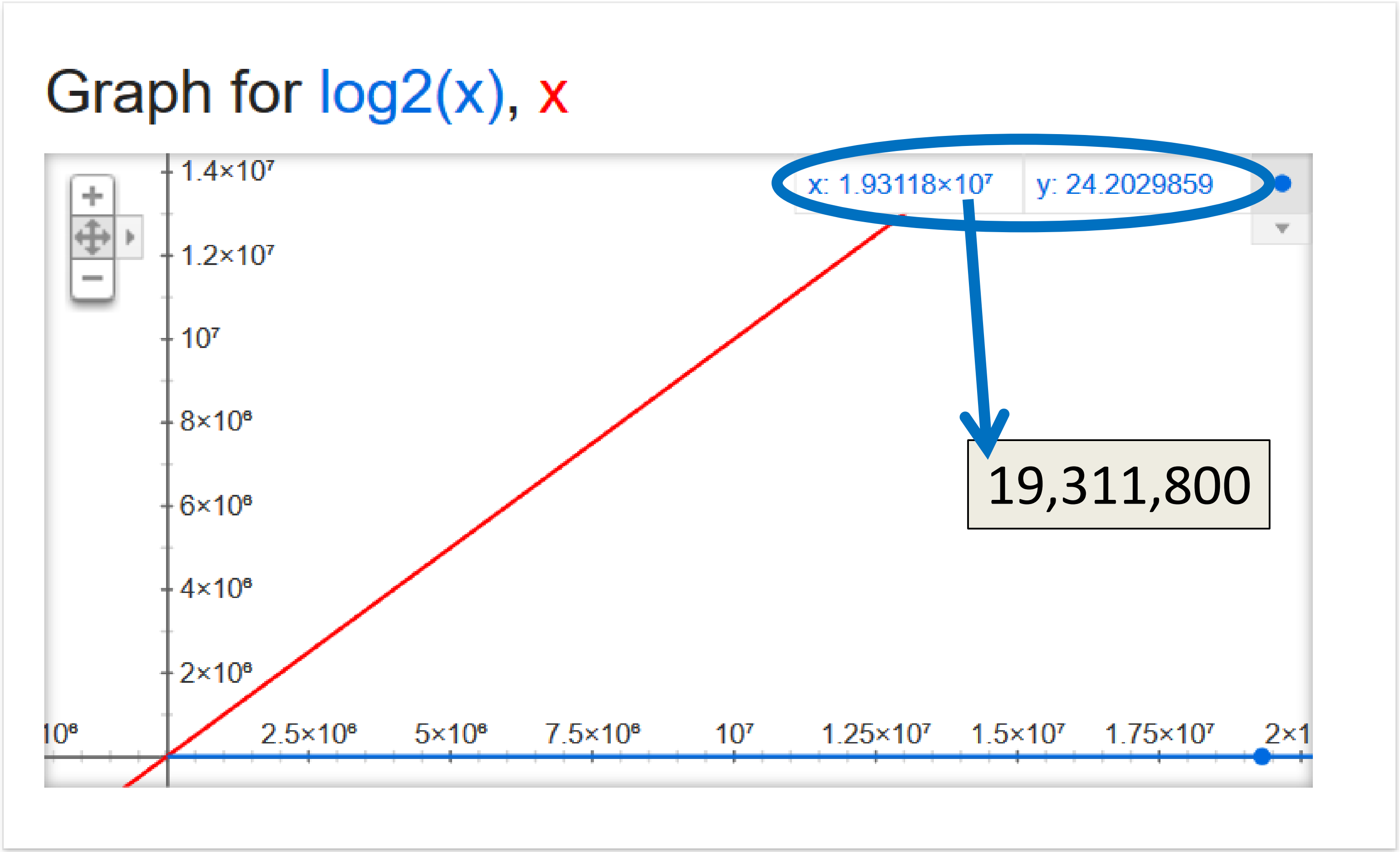
Why Care?


Graph for log2(x), x
x: 64.0386054
y: 6.00086999
50
40
30
20
10
0
10
20
30
40
50
60
70

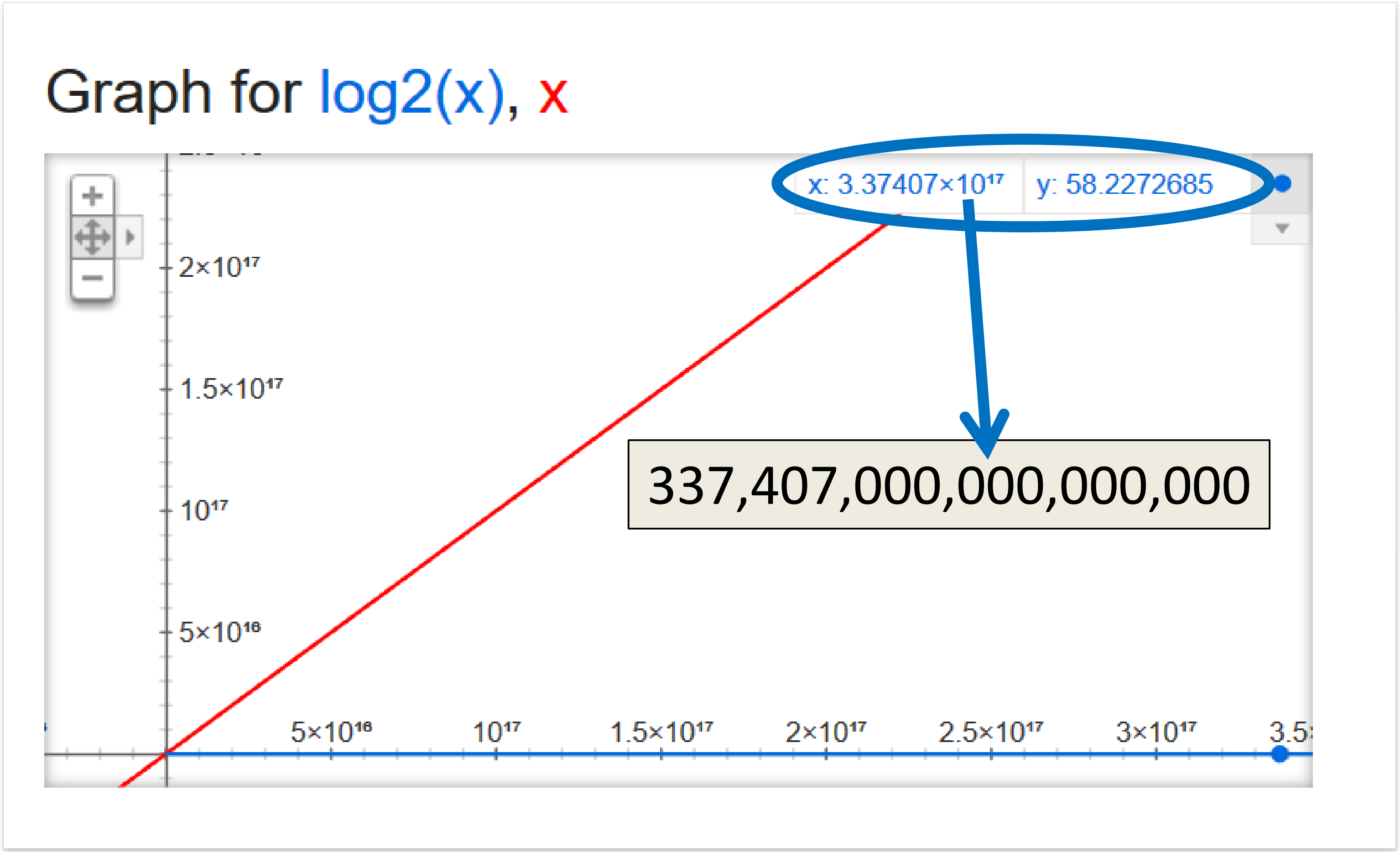
Why Care?


Graph for log2(x), x

Why Care?


Graph for log2(x), x
x: 1.93118×10⁷
y: 24.2029859
1.4×10⁷
1.2×10⁷
10⁷
8×10⁶
6×10⁶
4×10⁶
2×10⁶
2.5×10⁶
5×10⁶
7.5×10⁶
10⁷
1.25×10⁷
1.5×10⁷
1.75×10⁷
19,311,800

Why Care?


Graph for log2(x), x

337,407,000,000,000,000

Why Care?

- For large problems, there's a *huge* difference!
- If we can do 1,000,000 operations per second, and the list is 337.4 <u>quadrillion</u> items
  - Binary search takes 0.000058 seconds
  - Linear search takes 337,407,000,000 seconds
    5,623,450,000 minutes
    93,724,166 hours
    3,905,173 days
    10,699 years

Announcements

- Final is Thursday, December 15th (3:30 – 5:30)
  - Review worksheet will be released online on Friday, so you can start working on it over the weekend
- Project 2 out now
  - Due on ***<u>Tuesday</u>***, December 13th
- Survey #3 also out – follow link in announcement
- Now we'll talk about SEEQs

SEEQs

The Student Evaluation of Educational Quality (SEEQ) is a standardized course evaluation instrument used to provide measures of an instructor's teaching effectiveness. The results of this questionnaire will be used by promotion and tenure committees as part of the instructor's evaluation.

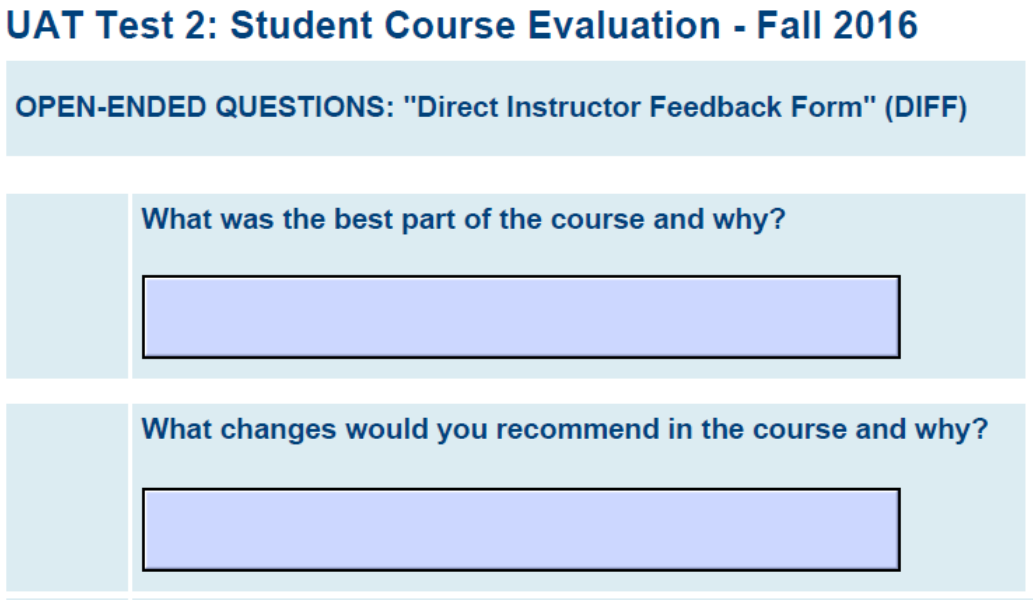
The Direct Instructor Feedback Forms (DIFFs) were designed to provide feedback to instructors and they are not intended for use by promotion and tenure committees.

The responses to the SEEQ and the DIFFs will be kept confidential and will not be distributed until final grades are in.

Completing SEEQs

- Please take the time now, if you haven't already, to complete the SEEQ online
- You can access it via the link in your email, or via Blackboard

This is the part →
I will get to see


UAT Test 2: Student Course Evaluation - Fall 2016

OPEN-ENDED QUESTIONS: "Direct Instructor Feedback Form" (DIFF)

What was the best part of the course and why?

What changes would you recommend in the course and why?