Algorithms & Data Structures I: 1_Introduction

Mikhail Anukhin

Contact

Mikhail Anukhin
email: anukhinm@gmail.com
telegram: @clumpytuna

Today's Topics

- Why algorithms are important
- Course general information
  - Course program
  - Skills you will learn
  - Assessment: criterias, formats
  - Collaboration policy
  - Supporting Materials
  - Useful links: chats, course page, etc.
- How to measure efficiency of an algorithm?
  - Big O-notation: definition, examples.
  - Master Theorem
- Binary search. Time complexity

Why Algorithms are important

- Important for all other branches of computer science
- Plays a key role in modern technological innovation
- Provides novel “lens” on processes outside of computer science and technology
- Challenging (i.e., good for the brain!)
- Let you go through the job interview

About the course program

- Unit 1: Introduction. Algorithms vocabulary
  - Introduction. Big-O notation. Master Theorem. Binary Search
  - Linked Lists. Stack implementation using a linked list
- Unit 2: Sorting
  - Sorting. Lower bound for comparisons in the sort. Insertion sort. Bubble Sort. Time complexity & space complexity
  - Quick Sort
  - Merge Sort
  - Binary Heap. Sift Up, Sift Down, Insert, GetMin, ExtractMin, DecreaseKey. Heap Sort.

About the course program

- Unit 3: Binary Trees
  - Binary Search Trees. Insert & Delete & BST Sort
  - Balanced Binary search Trees. AVL Tree. Height of AVL Tree on n nodes.
- Unit 4: Hashing
  - Hash Table Chaining. Insert & Delete & Search
  - Hash Table Open Addressing. Insert & Delete & Search
  - Bloom Filter. Insert & Search. Applications. Time complexity & space complexity

Skills you will learn

- Become a better programmer
- Sharpen your mathematical and analytical skills
- Start “thinking algorithmically”
- Prepare for technical interviews

Supporting materials

- Books:
  - Kleinberg/Tardos, *Algorithm Design*, 2005
  - Dasgupta/Papadimitriou/Vazirani, *Algorithms*, 2006.
  - Cormen/Leiserson/Rivest/Stein, *Introduction to Algorithms*, 2009 (3rd edition). → CLRS
  - Mehlhorn/Sanders, *Data Structures and Algorithms: The Basic Toolbox*, 2008.

Supporting materials

- GitHub page of the course: https://github.com/clumpytuna/data-structures-and-algorithms-l-2021
  - program
  - homework & deadlines
  - lectures records & notes & slides
- Chats & Channels: Feel free to ask your questions!
  - chat: https://t.me/joinchat/Hqx22qg99bl-QUS4
  - channel: https://t.me/dsa2021

Assessment

- 11 points in total:
  - 7 points for work during the semester: hw, contests, quizzes
  - 3 points for final exam
  - 1 bonus point for lecture and workshop activity
- homework:
  - every 1-2 week a contest on Y.Contest
  - one random problem from every contest is chosen for code review. You get feedback about your code, and can have 1 submission to improve it.
- contests:
  - After every unit you write a 1.5-2 hours contest based on unit content
- quizzes:
  - Introduction unit + Sorting unit quiz
  - Final quiz

Collaboration Policy

- The goal of homework is to give you practice in mastering the course material.
- You must write up each problem solution by yourself without assistance
- Code you submit must also be written by yourself
- No other student may use your solutions
- Plagiarism and other anti-intellectual behavior cannot be tolerated in any academic environment that prides itself on individual accomplishment
- Read more on the course GitHub page

How to measure efficiency of an algorithm?

$T(n) = \Theta(n)$

if

$\Omega(n) = O(n)$

```
function print_array(array) is
for (i = 0; i < array.size(); ++i) {
 print array[i]
}
```

$2 \cdot n$

$2 \cdot n$

read

print

$T(n) = 2n$

$n + n = 2n$

n

$T(n) = O(n)$ $\exists c = 4 \; T(n) < c \cdot n$

$T(n) < 4 \cdot n$

Big O-notation: Big O

Def: Let $f, g : \mathbb{N} \to \mathbb{N}$, then $f = O(g)$

$[\, f(n) = O(g(n)) \,]$, if $\exists\, C \in \mathbb{N}, N \in \mathbb{N}$ such that

$\forall n \in \mathbb{N},\ n > N :\ f(n) \leq C \cdot g(n)$

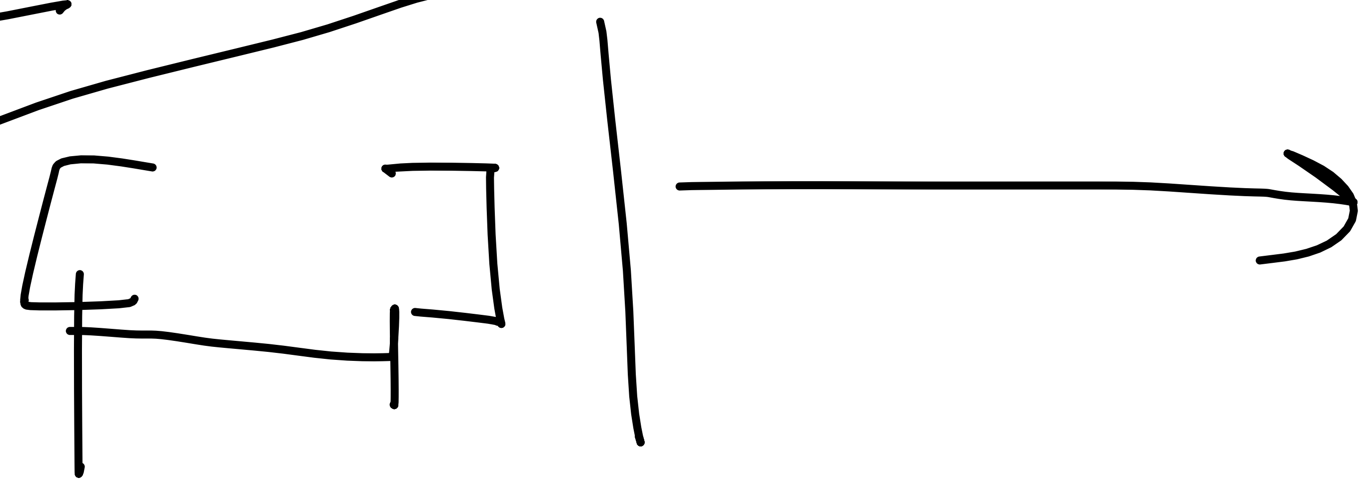
$f = x$

$g = x^2$

$x \leq 1 \cdot x^2$

$x \in (0, 1)$

1



Mikhail Anukhin

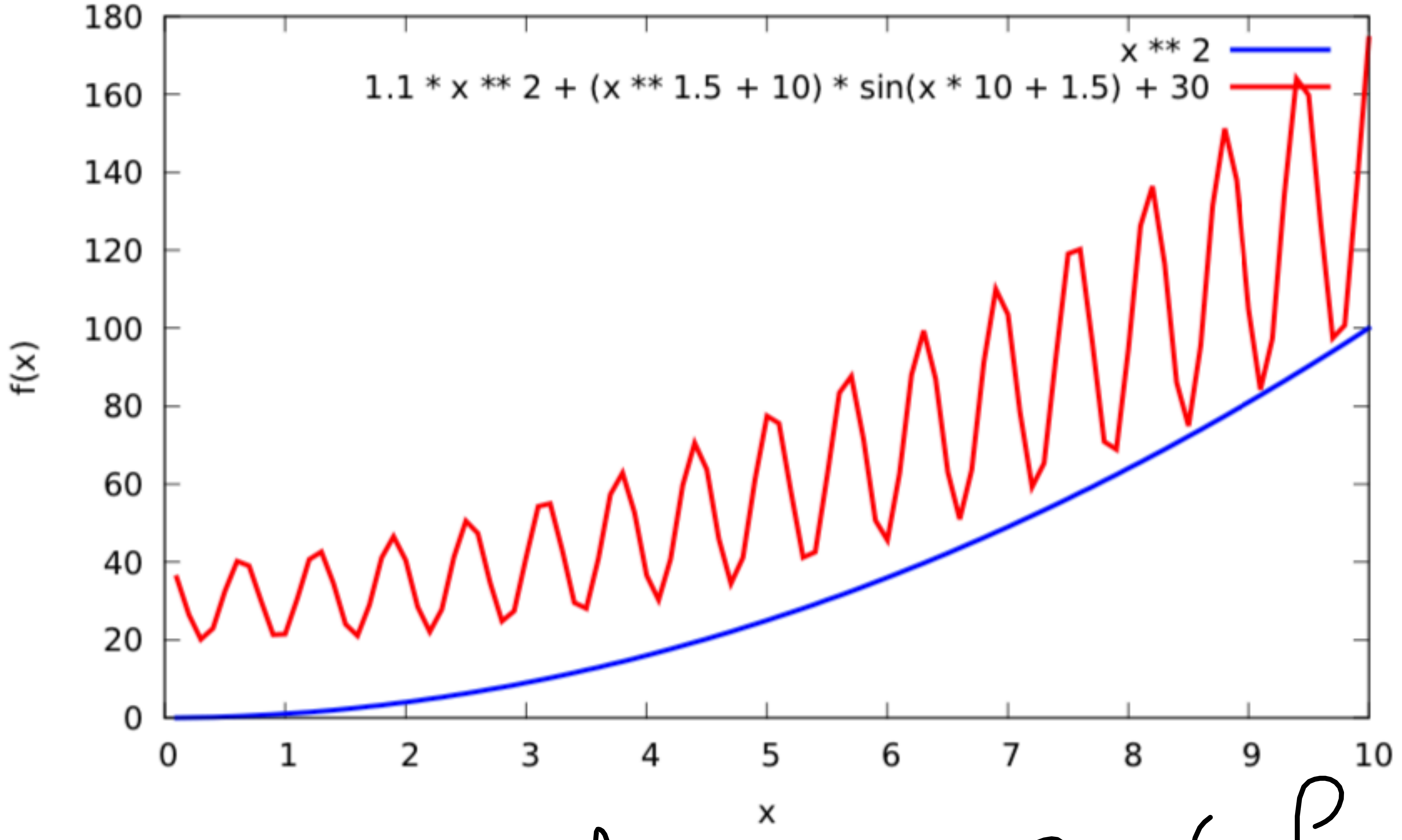
Big O-notation: Big O

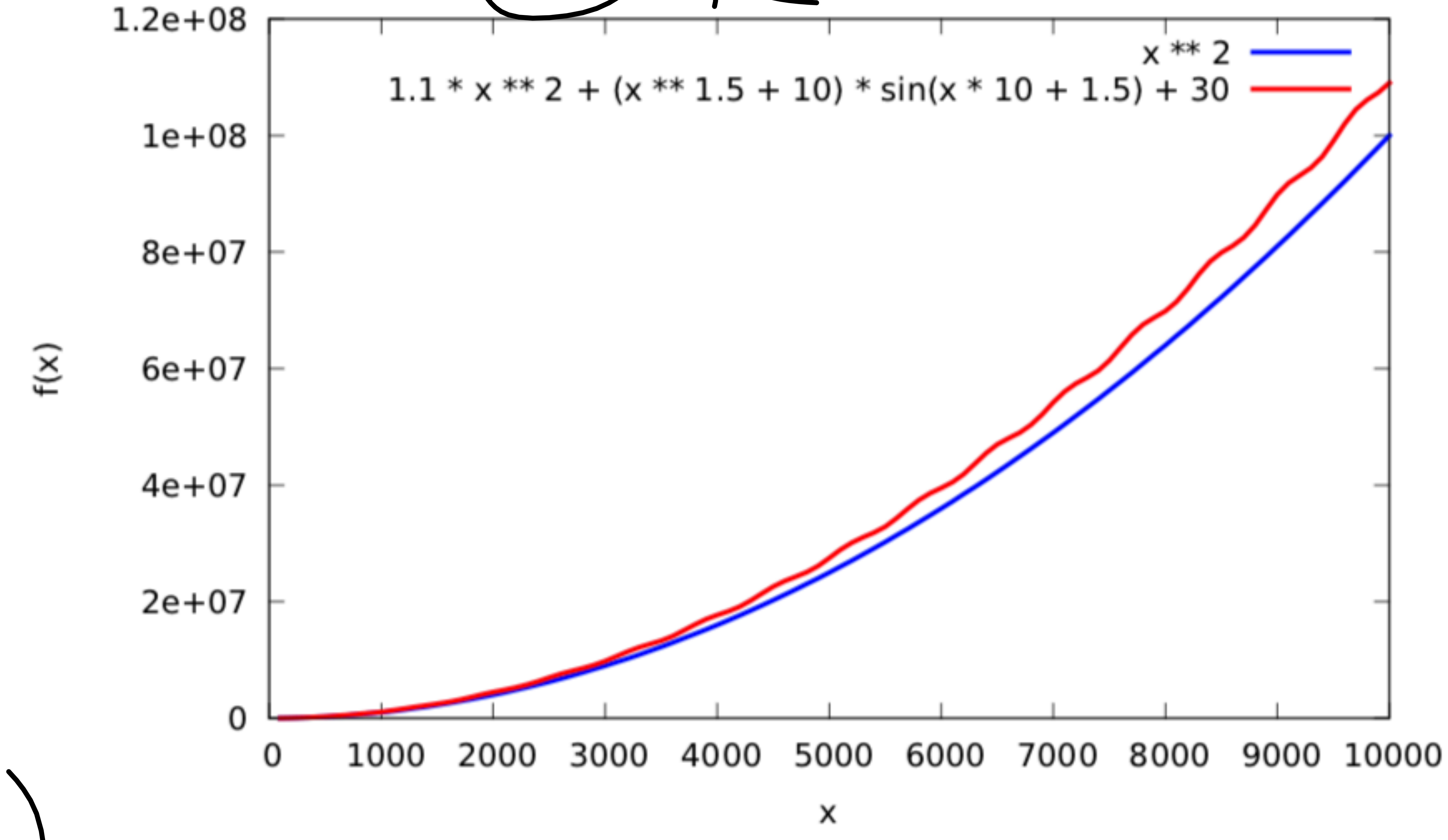
$f_1(x) = x^2$

$f_2(x) = 1.1x^2 + (x^{1.9} + 10)\sin(10x + 1.5) + 30$

$f_1 = O(f_2)$

$\exists c_1 \cdot f_2$





$f_2 = \Omega(f_1)$

Mikhail Anukhin

Big O-notation: Theta

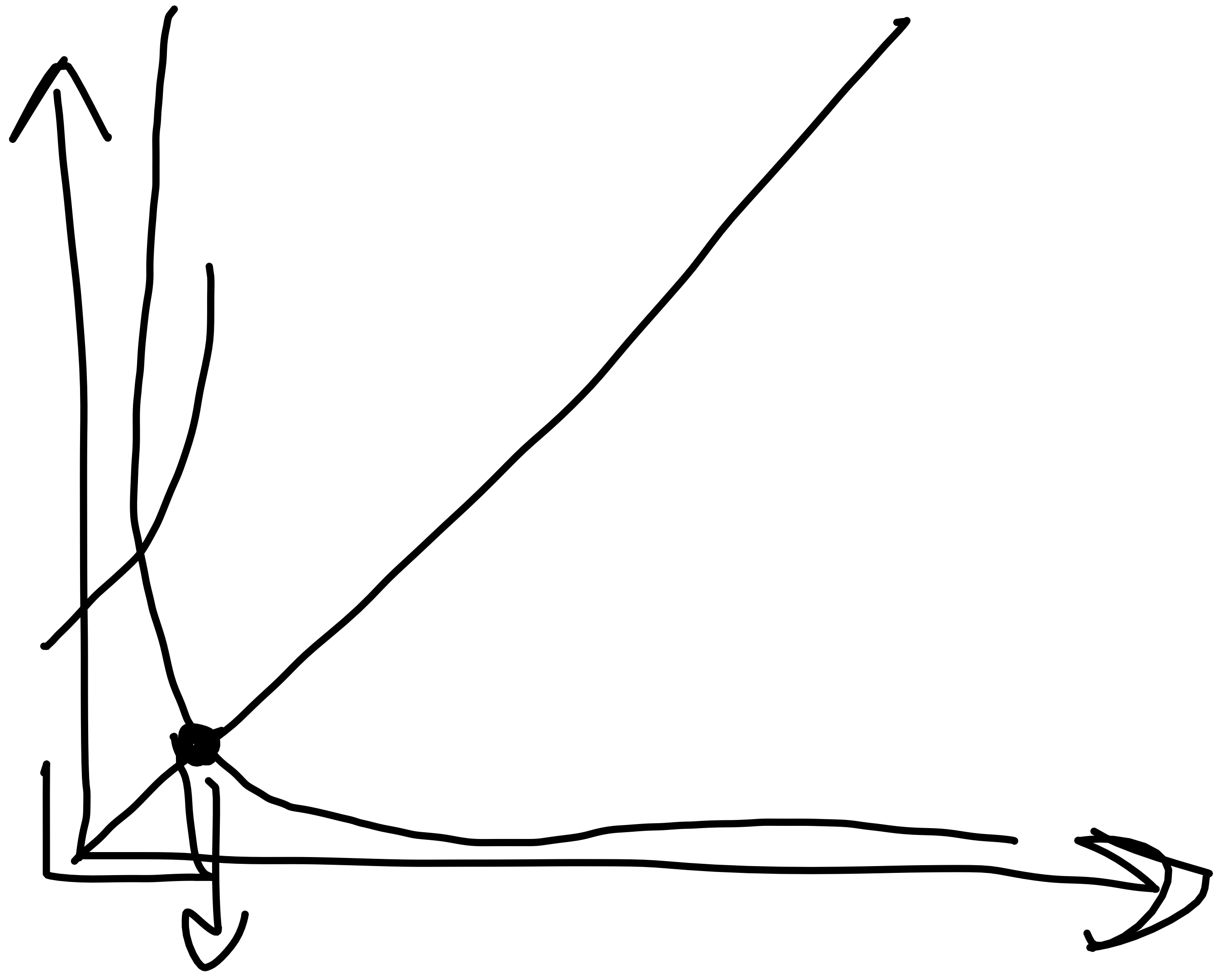
lower bound

def Let $f, g \in \mathbb{N} \to \mathbb{N}$, then

$f = .$

$f = \frac{1}{x}$

$g = x$



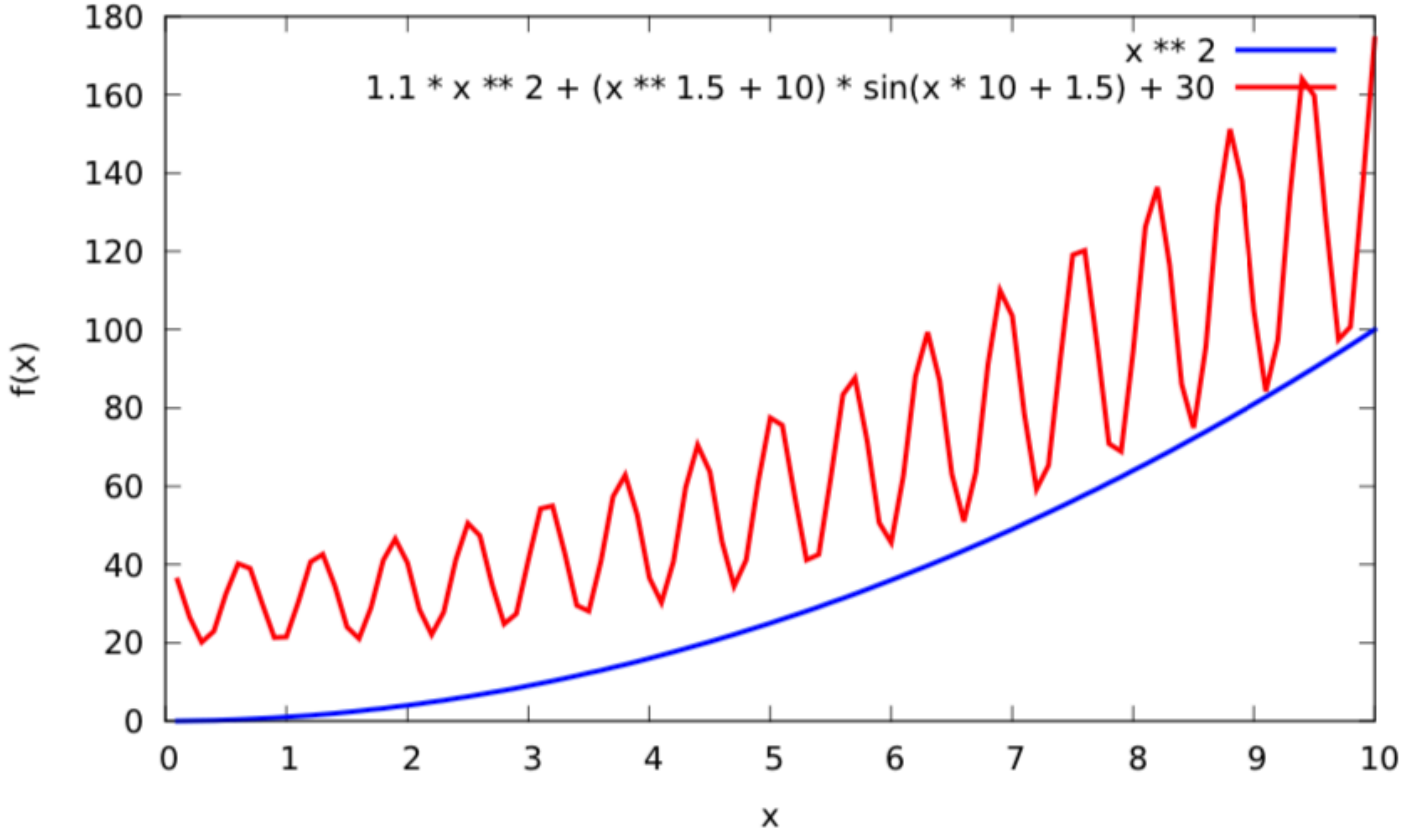
$f = O(g)$

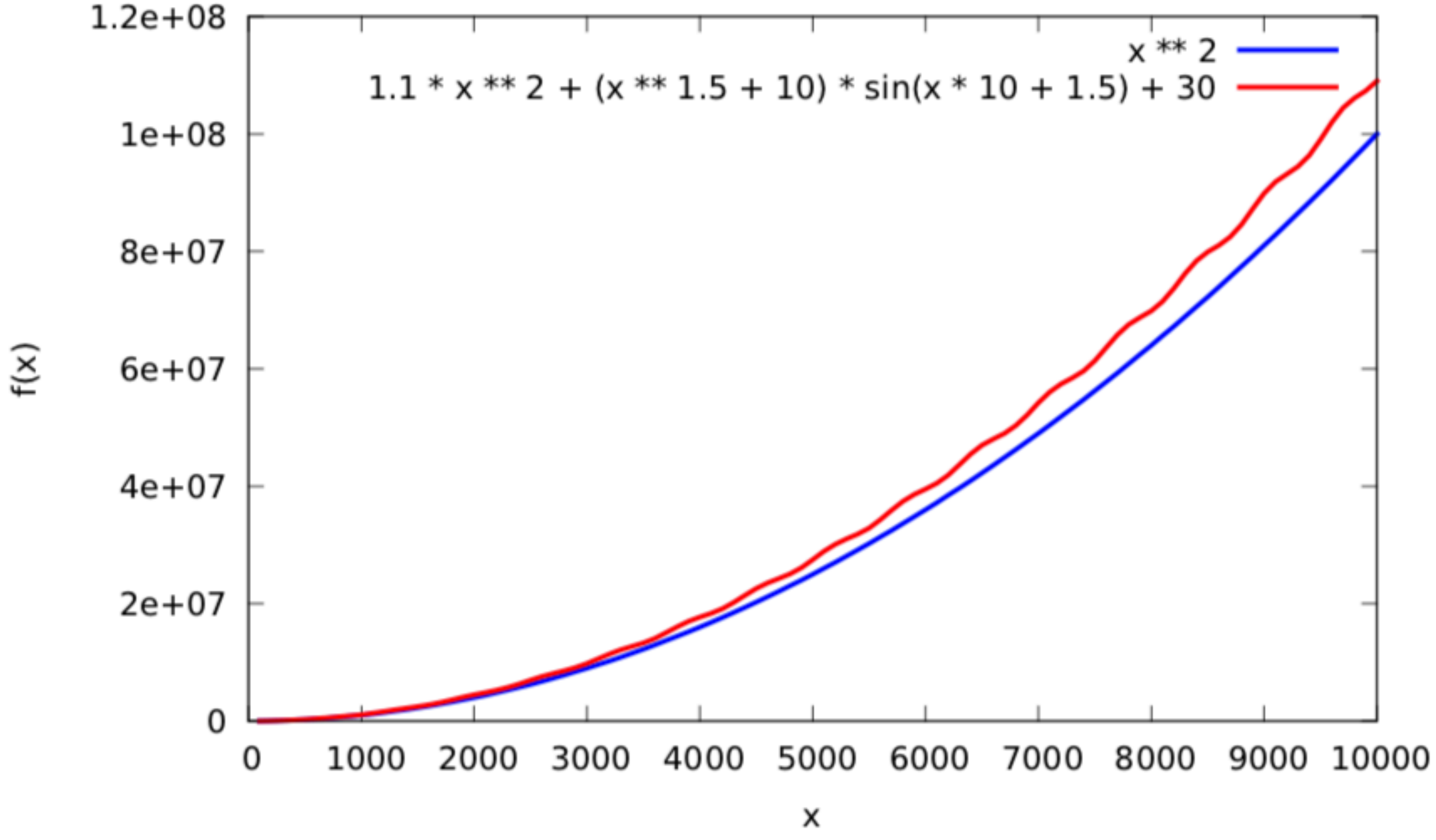
$\boxed{g \geq f}$

Big O-notation: Theta

$$f_1(x) = x^2$$

$$f_2(x) = 1.1x^2 + (x^{1.9} + 10)\sin(10x + 1.5) + 30$$





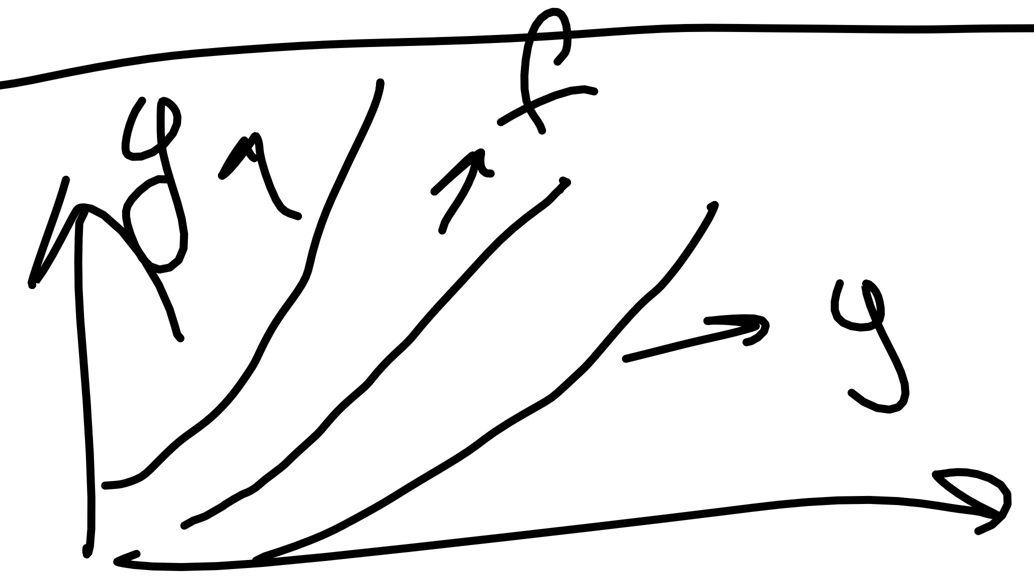
Big O-notation: Theta

Def. Let $f, g: \mathbb{N} \to \mathbb{N}$, then

$f = \Theta(g)$, if $\exists \underbrace{c_1, c_2} \in \mathbb{N}$ $\exists N \in \mathbb{N}$:

such that: $\forall n \geq N$:

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$



Big O-notation: Omega → lower bound

<u>Def:</u> $f, g : \mathbb{N} \to \mathbb{N}$, then

$f = \Omega(g)$, if $\exists C \in \mathbb{N}, N \in \mathbb{N}$, such that $\forall n \geq N$: $f(n) \geq C \cdot g(n)$

<u>N.B</u> $f = \Omega(g) \iff g = O(f)$

Big O-notation: Omega

lower bound

$O, \Omega, \Theta(g)$

upper bound

Binary Search Idea

Problem $A[n] = [1, \ldots, n]$

x in A →

$A = [1, 2, 4, 6, [8], 10, 11]$

$x = [9]$

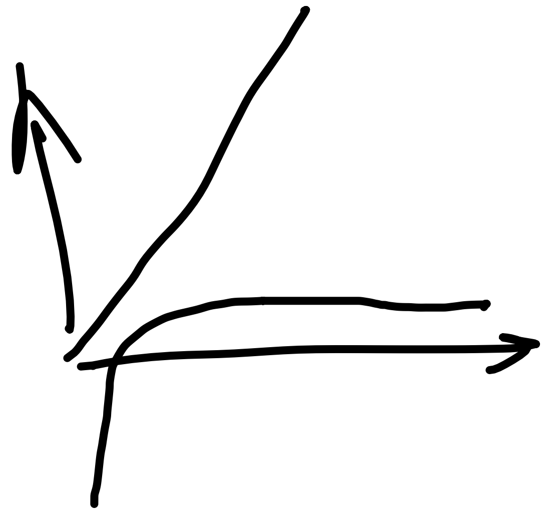
x > pivot

x < pivot

/2

[1 1]

Binary Search Pseudocode



$[\quad 5 \quad]$

```
function binary_search(A, n, T) is
    L := 0
    R := n − 1
    while L ≤ R do
        m := floor((L + R) / 2)
        if A[m] < T then
            L := m + 1
        else if A[m] > T then
            R := m − 1
        else:
            return m
    return unsuccessful
```

$O(n) \leftrightarrow O(\log n)$

$$T(n) = O(1) + T\left(\frac{n}{2}\right)$$

$$T(n) = \underbrace{O(1) + O(1) + O(1) + \dots}_{\log n \cdot O(1)} = O(\log n)$$

Your questions!

```
function binary_search(A, n, T) is
    L := 0
    R := n − 1
    while L ≤ R do
        m := floor((L + R) / 2)
        if A[m] < T then
            L := m + 1
        else if A[m] > T then
            R := m − 1
        else:
            return m
    return unsuccessful
```