哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

数据结构
Data Structures

Chapter 1 Introduction to Data Structures

Prof. Yitian Shao
School of Computer Science and Technology

Instructor Information

Prof. Dr. **Yitian Shao** (Chinese: **邵奕天**)

- School of Computer Science and Technology
- Research interests: **haptic interfaces**, **robotic tactile sensing**, and **wearable technologies**

Academic Background

- Junior professor (W1) of Electrical and Computer Engineering, TU Dresden, Germany (2022-2023)
- Postdoctoral Researcher, Max Planck Institute for Intelligent Systems, Stuttgart, Germany (2021-2022)
- Ph.D., Electrical and Computer Engineering, UC Santa Barbara, U.S. (2015-2020)

Email: shaoyitian@hit.edu.cn (I also teach High-level Language Programming)

Teaching Assistant

Lin Xu
School of Computer Science and Technology

Grading **Quiz** and **Lab** submissions

Email: sjjg2025@gmail.com

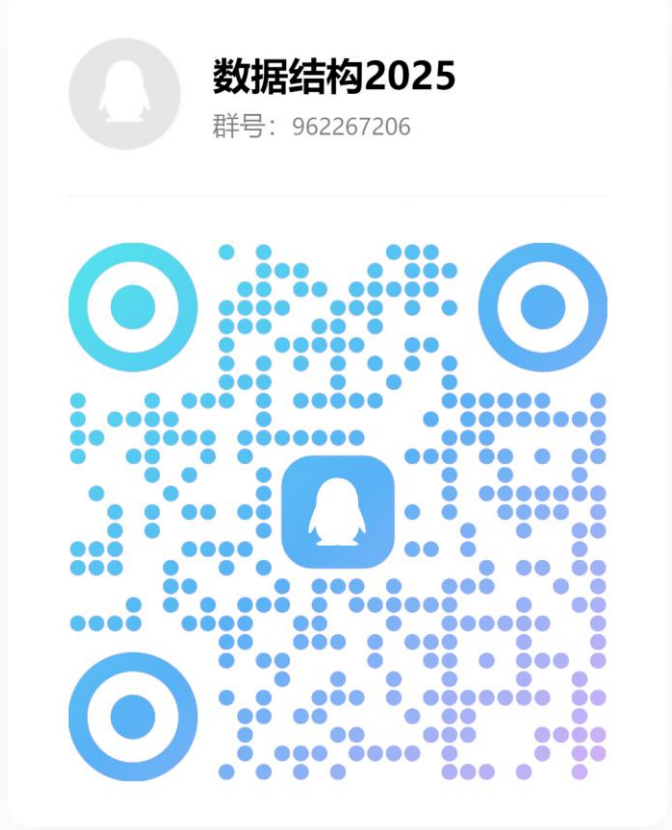
Let's Get Connected!

- Course related discussion: **QQ**
  - Posting homework
  - Sharing course materials
  - Course related Q&A

- **Step 1**: Download and install QQ on your mobile/PC: im.qq.com/index

iOS

Android



- **Step 2**: Inside your QQ, scan the QR code above

Overview

- In ***Data Structures***, we learn:
  - Basic **abstract data types** and their **logical structures, storage structures** and **operations**
  - Corresponding **algorithms** and **typical applications**
  - The storage and application of some **advanced data structures**
  - Common **searching** and **sorting** algorithms
- Course objectives:
  - Understand the relationship between the data structures and algorithms
  - Properly use data structures and algorithms
  - Analyze code in time complexity (and space complexity)
  - Design effective algorithms and data structures to solve practical problems

Expectations

- **Attend lectures on time**
- **Complete all assignments on your own** and **submit on time**
- Engage in class discussion
- Communicate with instructor if need help

- All lectures are given in **English**
- All assignments and exams must be completed in **English**

Academic Integrity

Quiz **Lab** **Final Exam**

- **Do not** look at other people's solution code
- **Do not** give your solution code to others, or post it elsewhere
- **Report** any inappropriate activity you see performed by others

All assignments are checked for similarity! Copied solutions suffer significant grade penalty!

Talk to the instructors/TA if you need help with the assignment (but **do not wait until the quiz or exam day**)

Grading

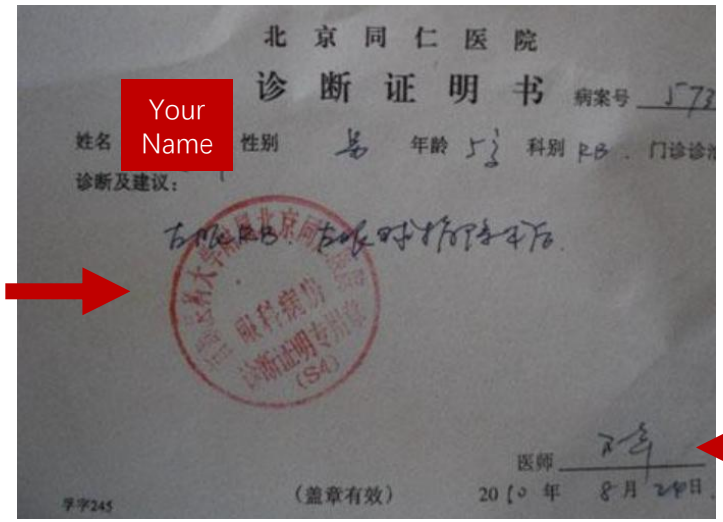
- 24 lectures from Week 1 to Week 13
  - **Tuesday 2pm-3:45pm** and **Thursday 10:30am-12:15pm**, **T5-507**

- Course assignments and exams
  - **Exercises will not be graded! (Complete them if you want to prepare yourself for the quiz and final exam)**
  - **In-class quiz** x 4 **= 40% points (Complete and Submit in class)**
  - **Lab report** x 4 **= 20% points (Complete and Submit in lab)**
  - **Final exam = 40% points**

Late submission not accepted, no excuse!

Special cases

- **Require a valid proof** for late submission / Absenteeism
- **Must inform the instructor** before taking any actions
- **Submit a valid proof** no later than a week



Valid Stamp

Valid Signature

Class Attendance

- **Class attendance is enforced** and recorded every single class
- Bring a laptop to the class if possible
- **If you miss more than 8 lectures, you will be disqualified to attend the final exam**

Class Attendance

- **Class attendance is enforced** by our university
- Technical issues – **Get help from H719 (Main building)**
- Install **WeChat** or the **Attendance Check APP**

安卓 Android

苹果 IOS

Class Attendance Check!

- Technical issues – **Get help from H719 (Main building)**
- Open **WeChat** or the **Attendance Check APP**
- Now, **scan the QR code** using WeChat or Attendance Check APP, ask for help if you encounter any problem.

Schedule

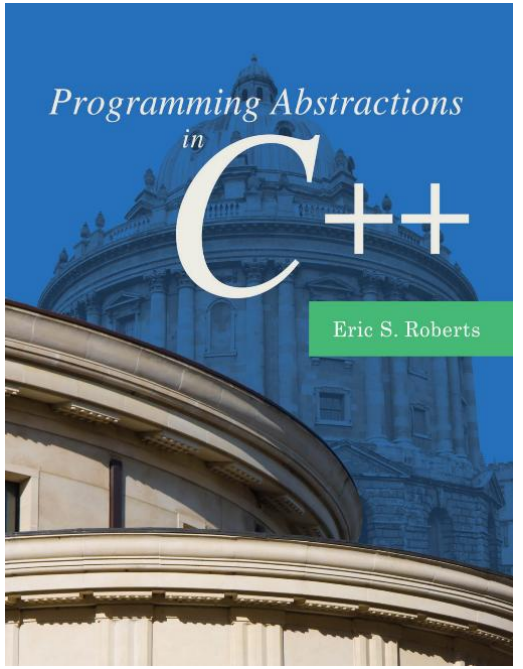
	Tue	Wed	Thu
Week 1: Introduction (2/25)			Review of C++ Programming (2/27)
Week 2: Array and Matrix (3/4)			Basics of Algorithm (3/6)
Week 3: Linked Lists I (3/11)			Linked Lists II (**3/13 Quiz Day!**)
Week 4: Stacks and Queue I (3/18)		Lab 1	Stacks and Queue II (3/20)
Week 5: Strings I (3/25)			Strings II (**3/27 Quiz Day!**)
Week 6: Trees I (4/1)			Trees II (4/3)
Week 7: Trees III (4/8)		Lab 2	Trees IV (4/10)
Week 8: Trees V (**4/15 Quiz Day!**)			Graph I (4/17)
Week 9: Graph II (4/22)		Lab 3	Graph III (4/24)
Week 10: Searching Algorithms I (4/29)			*(holiday)*
Week 11: Searching Algorithms II (5/6)			Searching Algorithms III (**5/8 Quiz Day!**))
Week 12: Sorting Algorithms I (5/13)		Lab 4	Sorting Algorithms I (5/15)
Week 13: Sorting Algorithms I (5/20)			**Final Exam** (Date to be determined)

Textbook and Online Resources

Eric Roberts, ***Programming Abstractions in C++***, Pearson. 2013. (Electronic version acceptable)

Online Course: Stanford CS106B

(The content of this course is developed based on it)



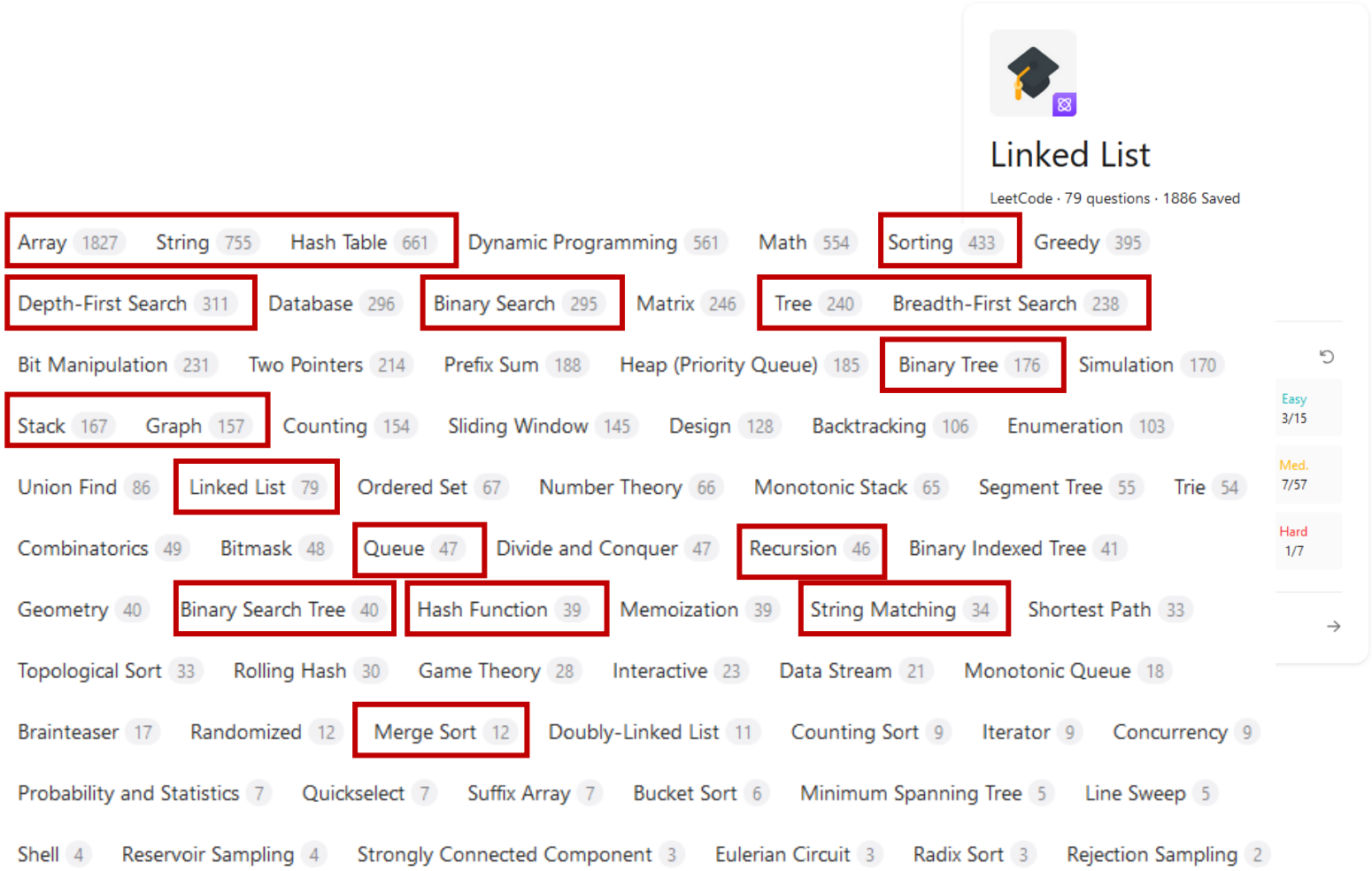
Other online references and materials

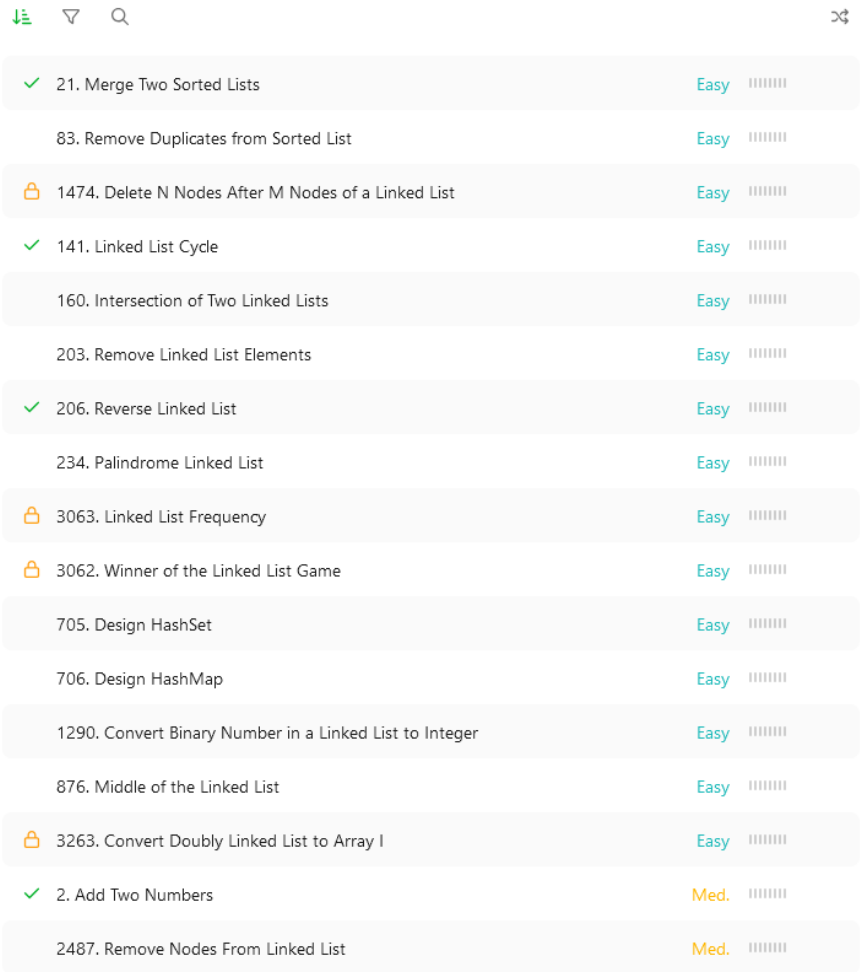
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms (3rd Edition), The MITPress. 2009.
- Robert Sedgewick, Kevin Wayne, Algorithms (4th Edition), Addison-Wesley Professional. 2011.
- leetcode.com

LeetCode for Homework and Self Practice

leetcode.com/problemset/algorithms/

- 3431 Problems – can cover all topics in this course (start from the easy ones!)





LeetCode: How to Use

Run Submit

Description | Editorial | Solutions | Submissions

21. Merge Two Sorted Lists

Solved

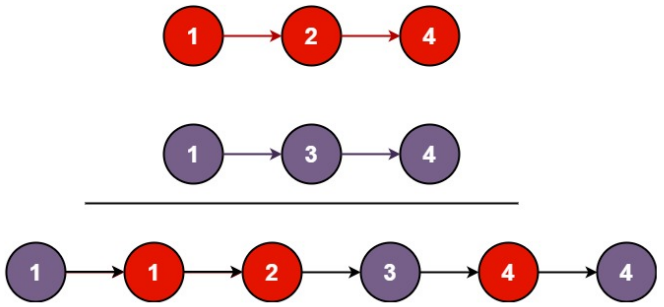
Easy | Topics | Companies

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:




```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []
Output: []
```

Example 3:

```
Input: list1 = [], list2 = [0]
Output: [0]
```

Constraints:

- The number of nodes in both lists is in the range `[0, 50]`.
- `-100 <= Node.val <= 100`
- Both `list1` and `list2` are sorted in **non-decreasing** order.

Seen this question in a real interview before? 1/5

Yes No

Code

C++ Auto

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        ListNode* dummy = new ListNode(0);
        ListNode* op = dummy;

        while (list1 && list2) {
            if (list1->val <= list2->val) {
                op->next = list1;
                list1 = list1->next;
            } else {
                op->next = list2;
                list2 = list2->next;
            }
            op = op->next;
        }
        op->next = list1 ? list1 : list2;
        return dummy->next;
    }
};
```

Saved Ln 10, Col 4

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

list1 =
[1,2,4]

list2 =
[1,3,4]

Output

[1,1,2,3,4,4]

Expected

[1,1,2,3,4,4]

Abstract Data Types (ADT)

What is Abstract Data Types (ADT)?

- **Data structures** can be assembled to form hierarchies. The **atomic data types**—such as int, char, double, and enumerated types—occupy the lowest level in the hierarchy.
- To represent **more complex information**, you combine the atomic types to form larger structures. These larger structures can then be assembled into even larger ones …
- Collectively, these assemblages of information into more complex types are called **data structures**

What is Abstract Data Types (ADT)?

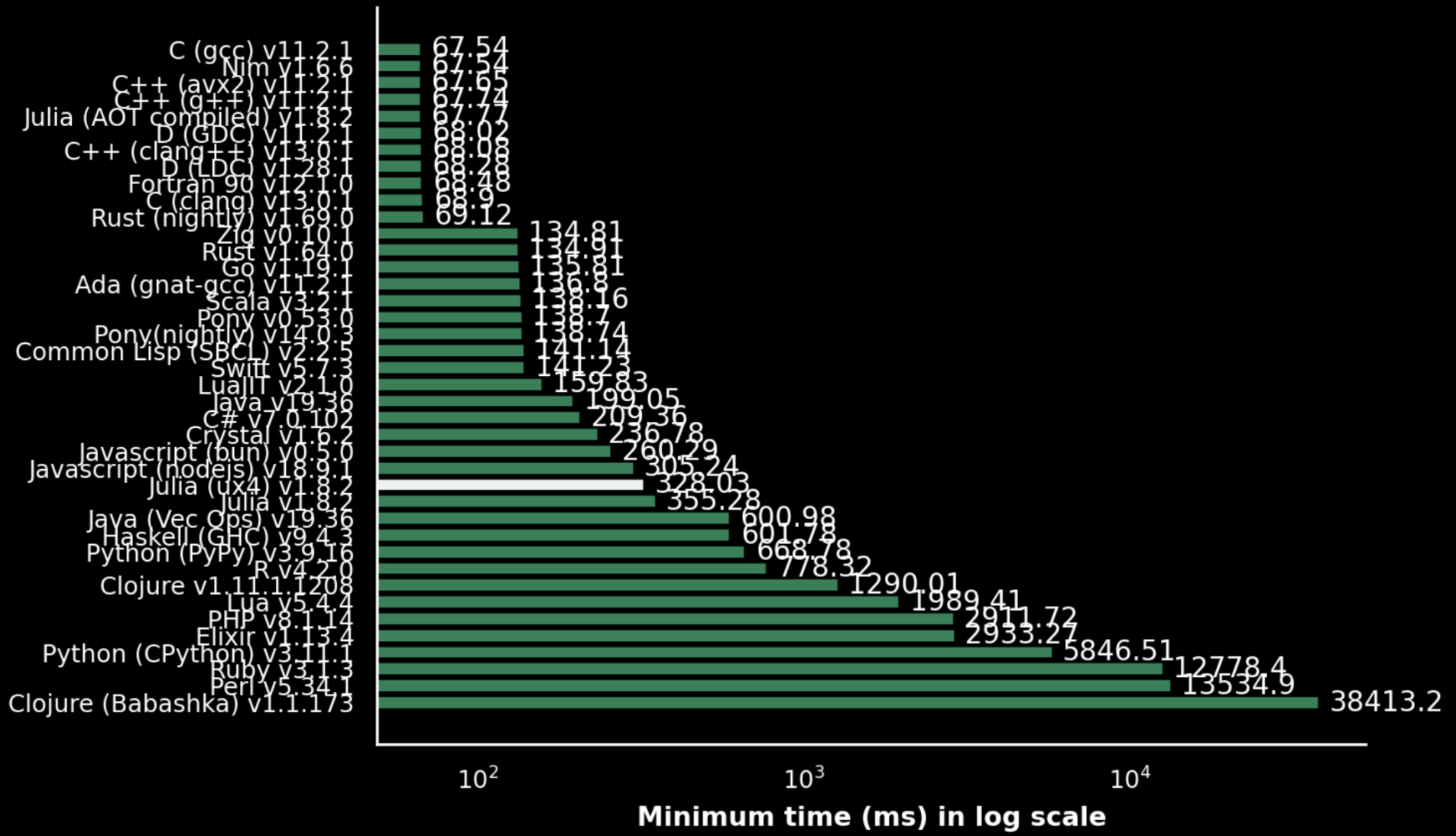
- **A type defined in terms of its behavior rather than its representation** is called an **abstract data type**, which is often abbreviated to **ADT**
- ADT are central to the **object-oriented style of programming**, which encourages **thinking about data structures in a holistic way**

Programming language for this course

- C++ (High-level Language Programming)
- Brief review of C++ syntax will be included in this course

Speed comparison of various programming languages

Method: calculating π through the Leibniz formula 100000000 times



Generated: 2023-02-05 18:52

https://github.com/niklas-heer/speed-comparison

Programming language: C++

- C++ is a programming language developed in 1983 by Bjarne Stroustrup
  - one of the world's **most widely used** languages today
  - built for systems programming with **high speed/efficiency**
  - built on older C language by adding **object-oriented programming**
  - continues to be **improved over time** (latest version: C++ 26)

Basic C++ programming skills you need

- Syntax
- Compile and execute
- Variable, expression, string, **function**, **struct**, **class**

Exercise 1.1

- Homework exercises assigned after each lecture will not be graded, however, you should complete them to **get yourself prepared for quizzes and final exam**!
- Learn how to use LeetCode
- Mark your first step on LeetCode, complete [LeetCode 58]

58. Length of Last Word

Easy | Topics | Companies

Given a string `s` consisting of words and spaces, return *the length of the **last** word in the string.*

A **word** is a maximal substring consisting of non-space characters only.