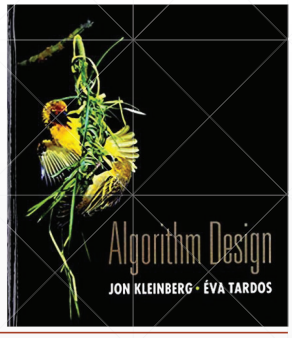
Algorithms

CS 336: Design and Analysis of Algorithms

Introduction

- Instructors: Konstantin Makarychev & Donald Stull
- TAs: Charles Cui, Sanchit Kalhan, and Liren Shan
- PMs: Sanath Angalakudati, Peizhi Liu, Liam O'Carroll, Sisilia Sinaga, Brando Socarras, Andrew Su, and Nathan White
- Textbook:
  - "Algorithms Design" by Kleinberg & Tardos



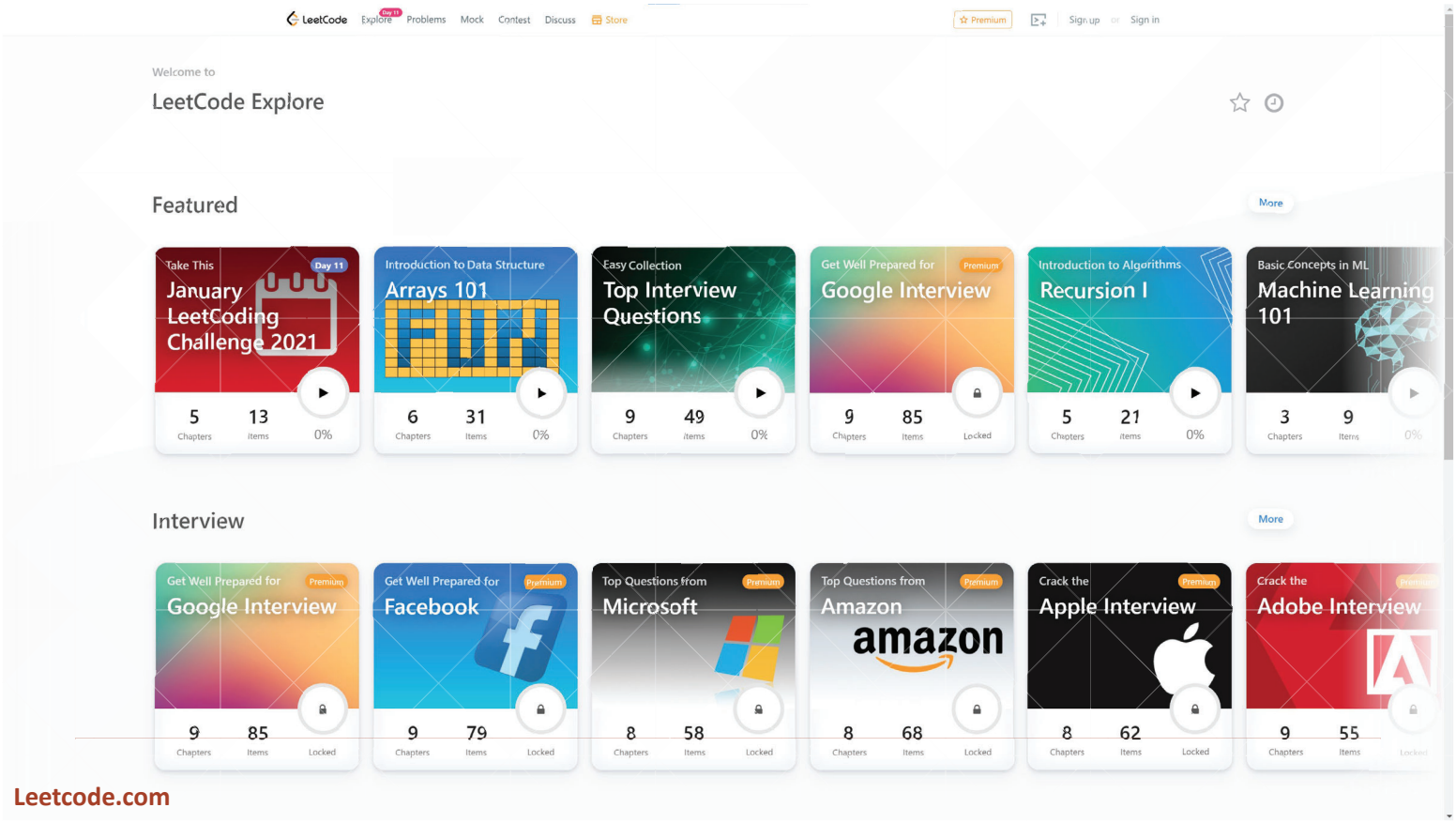
What is this course about?

We will learn how to

1. Design fast and efficient algorithms.
2. Analyze the running time & prove correctness of algorithms.

Why does this course matter?

1. It will help you to get a good job!
   - Most interview questions are on algorithms.
   - Many problems in this course are taken from actual job interviews!



Leetcode.com

Why does this course matter?

1. It will help you to get a good job!
   - Most interview questions are on algorithms.
   - Many problems in this course are taken from actual job interviews!

Why does this course matter?

1. It will help you to get a good job!
2. It will teach you how to write fast and efficient code!
   - If you want your programs to be fast, use good algorithms.

Why does this course matter?

1. It will help you to get a good job!
2. It will teach you how to write fast and efficient code!
3. It will show you how to analyze algorithms and introduce you to theoretical computer science.

Schedule

- **Track #1:** Tuesday & Thursday, 11am — 12:20pm
- **Track #2:** Monday & Wednesday, 5pm — 6:20pm

- **Q&A and problem-solving sessions:**
  - Monday, 6:00pm - 6:50pm
  - Tuesday, 5:00pm-5:50pm
  - Tuesday, 6:00pm - 6:50pm
- **Office Hours:** see Canvas

Grading

- Midterm: 25%
- Final: 25%
- Homework: 50%
- + points for active class participation.

Grading

- **A** and **A-** at least **35%-40%** of class
- **C+** or less at most **15%-20%** of class

If your score $\geq 90\%$, you will get an **A** or **A-**.

Homework Assignments

- Weekly homework assignments.
- Most assignment have two or three parts:
  - Quiz Questions;
  - Algorithmic Problems; and
  - Programming Assignment.

Quizzes

Questions that test your understanding of the material.

- **Question:** $n = O(n \log_2 n)$
- **Answer:** *True* or *False*

Quizzes

Questions that test your understanding of the material.

- **Question:** $n = O(n \log_2 n)$
- **Answer:** *True* or *False*
- **Defn:** $f(n) = O(g(n))$ if there exists a positive C, integer n_0 such that for every $n \geq n_0$

$$f(n) \leq C\ g(n)$$

- Submit your answers using Canvas Quizzes.

Algorithmic Problems

Problem 1. Alice started a company that sends people to the Moon. She contacted potential customers, and each customer gave her the maximum price he or she is willing to pay for a round trip ticket. Now, Alice needs to set a price for tickets. This price must be the same for all customers. If Alice sets the price to be x, all customers who are willing to pay x or more will buy tickets and pay x; all customers not willing to pay x will not buy tickets. The cost of sending a customer to the Moon is denoted by *cost*. Your goal is to design a fast algorithm that finds the price that maximizes Alice's profit. The algorithm receives:

- *max_prices* – array containing maximum prices customers are willing to pay;
- *cost* – the actual cost of sending a person to the Moon.

Algorithmic Problems

- Your task is to
  1. Design an algorithm that solves the problem.
  2. Prove that your algorithm is correct.
  3. Analyze its running time.

➢ Write a clear comprehensive solution.

➢ Typeset your solutions in LaTeX, Microsoft Word, etc.

➢ Submit your solutions in the PDF format using Canvas.

Programming Assignments

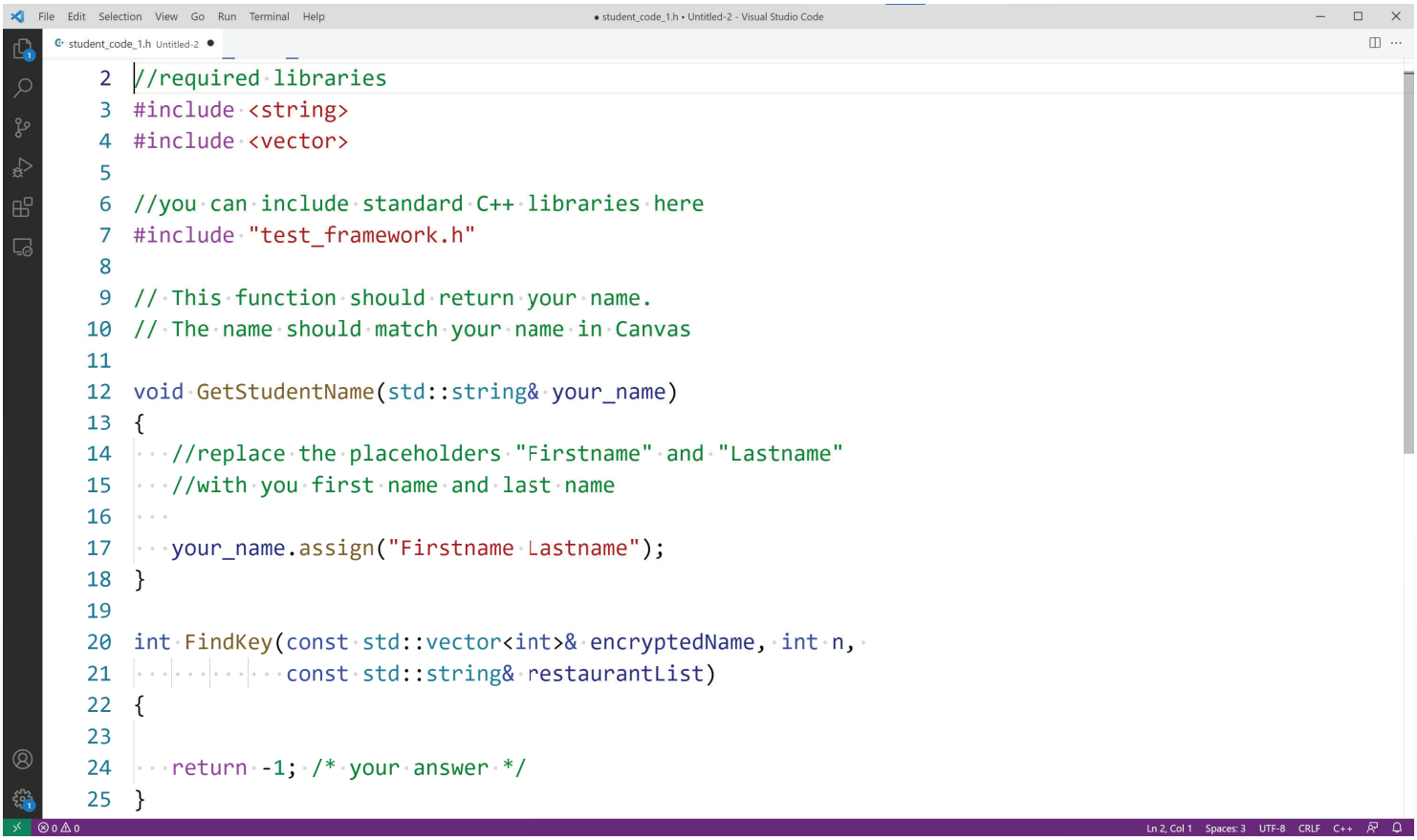
- Implement all your algorithms in C++.
  - Your code should be compatible with C++17.
  - You can use the C++ Standard Library (std::).
  - … but don't use other libraries (e.g., Boost).

Recommended compilers:

- **Clang (clang++),** GNU C++ (g++), Microsoft Visual C++ (cl).

Programming Assignments

- Let's go over Homework Assignment #0.
- Download:
  - *problem_solver_1.cpp* – main file in the project
  - *student_code_1.h* – this file should contain your solution.
  - *test_framework.h* – a library responsible for input & output
  - *problem_set_1.in* – sample problems.
- Place all files in a new folder/directory.
- Open *student_code_1.h* and implement two functions:
  - *GetStudentName* should return your name;
  - *FindKey should solve the problem.*



Programming Assignments

- Compile your code:
  - `clang++ -std=c++17 problem_solver_1.cpp -O3 -o problem_solver`
  - `g++ -std=c++17 problem_solver_1.cpp -O3 -o problem_solver`
  - `cl /EHsc problem_solver_1.cpp`
- Run your program:

```
Problem set #1. Your algorithm solved all test problems correctly. Don't
forget to submit your source code and file 'solution_1.dat' via Canvas.
```

- Submit *student_code_1.h on* Canvas.

Academic Integrity

- Don't cheat on homework assignments!
- No collaboration unless it is *explicitly permitted*.
  - When it is permitted, you still must acknowledge all students you worked with in your solution.
  - You must write down your solution completely on your own.
- Cheating is wrong.
- The consequence of cheating may be very serious.

Running Time

GTA Online update fixes loading time, after player discovers issue

No more long wait times just to log on.

Oscar Gonzalez March 16, 2021 12:57 p.m. PT

LISTEN · 01:12

Getting online to commit some crime is about to get faster.
Rockstar Games

GTA Online suffers from ridiculously long loading times, as any player can confirm, but a user identified the cause. After more than seven years, developer Rockstar Games

Replace spaces with dashes

```
void ReplaceSpaces(char* str)
{
    for (int i = 0; i < std::strlen(str); i++)
    {
        if (std::isspace(str[i]))
        {
            str[i] = '-';
        }
    }
}
```


How to measure performance?

- Measure the running time.
- The running time is the amount of time it takes to run an algorithm.
- It is convenient to measure time in elementary CPU operations.
- The exact number of operations depends on the CPU architecture, so we use big-O notation.

Examples

- Sorting: How much time do we need to sort n numbers?

Examples

- Sorting: How much time do we need to sort n numbers?
  - Bubble Sort – ?
  - Quick Sort – ?

Examples

- Sorting: How much time do we need to sort n numbers?
  - Bubble Sort – $O(n^2)$ operations.
  - Quick Sort – $O(n \log n)$ operations.

Examples

- How to calculate the sum of all numbers from 1 to n?

```
int ComputeSum (int n)
{
    int S = 0;
    for (int i = 1; i <= n; i++)
    {
        S = S + i;
    }
    return S;
}
```

Running time?

```
int ComputeSum (int n)
{
    int S = 0;
    for (int i = 1; i <= n; i++)
    {
        S = S + i;
    }
    return S;
}
```

Running time? O(n)

Better Solution

- We can use formula:

$$S_n = \frac{n(n+1)}{2}$$

The running time of computing S_n is $O(1)$.

Why is this formula correct?

Prove by Induction

- Let $S_n = 1 + \cdots + n$.
- We prove by induction on n that $S_n = \frac{n(n+1)}{2}$.
- **Base case:** The claim holds for $n = 1$, since for $n = 1$, we have $S_n = 1$ and $\frac{n(n+1)}{2} = 1$.
- **Inductive step:** Suppose that the statement holds for n (inductive hypothesis). We prove it for $n + 1$.

Prove by Induction

- We prove by induction on n that $S_n = \frac{n(n+1)}{2}$.
- **Base case:** The claim holds for $n = 1$, since for $n = 1$, we have $S_n = 1$ and $\frac{n(n+1)}{2} = 1$.
- **Inductive step:** Suppose that the statement holds for n (inductive hypothesis). We prove it for $n + 1$. Write:

$$S_{n+1} = 1 + \cdots + n + (n + 1)$$

- We prove by induction on n that $S_n = \frac{n(n+1)}{2}$.
- **Base case:** The claim holds for $n = 1$, since for $n = 1$, we have $S_n = 1$ and $n(n + 1)/2 = 1$.
- **Inductive step:** Suppose that the statement holds for n (the inductive hypothesis). We prove it for $n + 1$. Write:

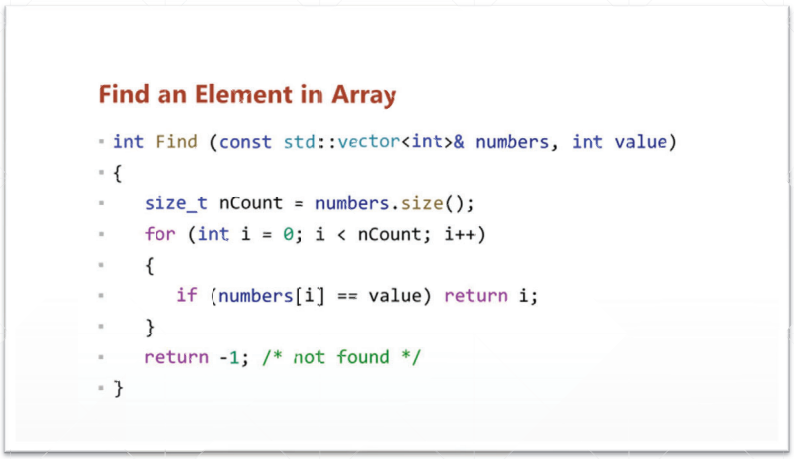
$$S_{n+1} = \underbrace{1 + \cdots + n}_{S_n} + (n + 1) = S_n + (n + 1)$$

By the ***inductive hypothesis***, $S_n = n(n + 1)/2$. Thus

$$S_{n+1} = \frac{n(n + 1)}{2} + (n + 1) = \frac{n^2 + 3n + 2}{2} = \frac{(n + 1)(n + 2)}{2}$$

Find an Element in Array

```
int Find (const std::vector<int>& numbers, int value)
{
   size_t nCount = numbers.size();
   for (int i = 0; i < nCount; i++)
   {
      if (numbers[i] == value) return i;
   }
   return -1; /* not found */
}
```



- Running time is $O(n)$, where n is the size of the array.
- Use data structures (e.g., ***hash tables*** or ***balanced trees***) instead of linear search. For sorted arrays, use ***binary search***.

How to Pass Arrays to Functions in C++

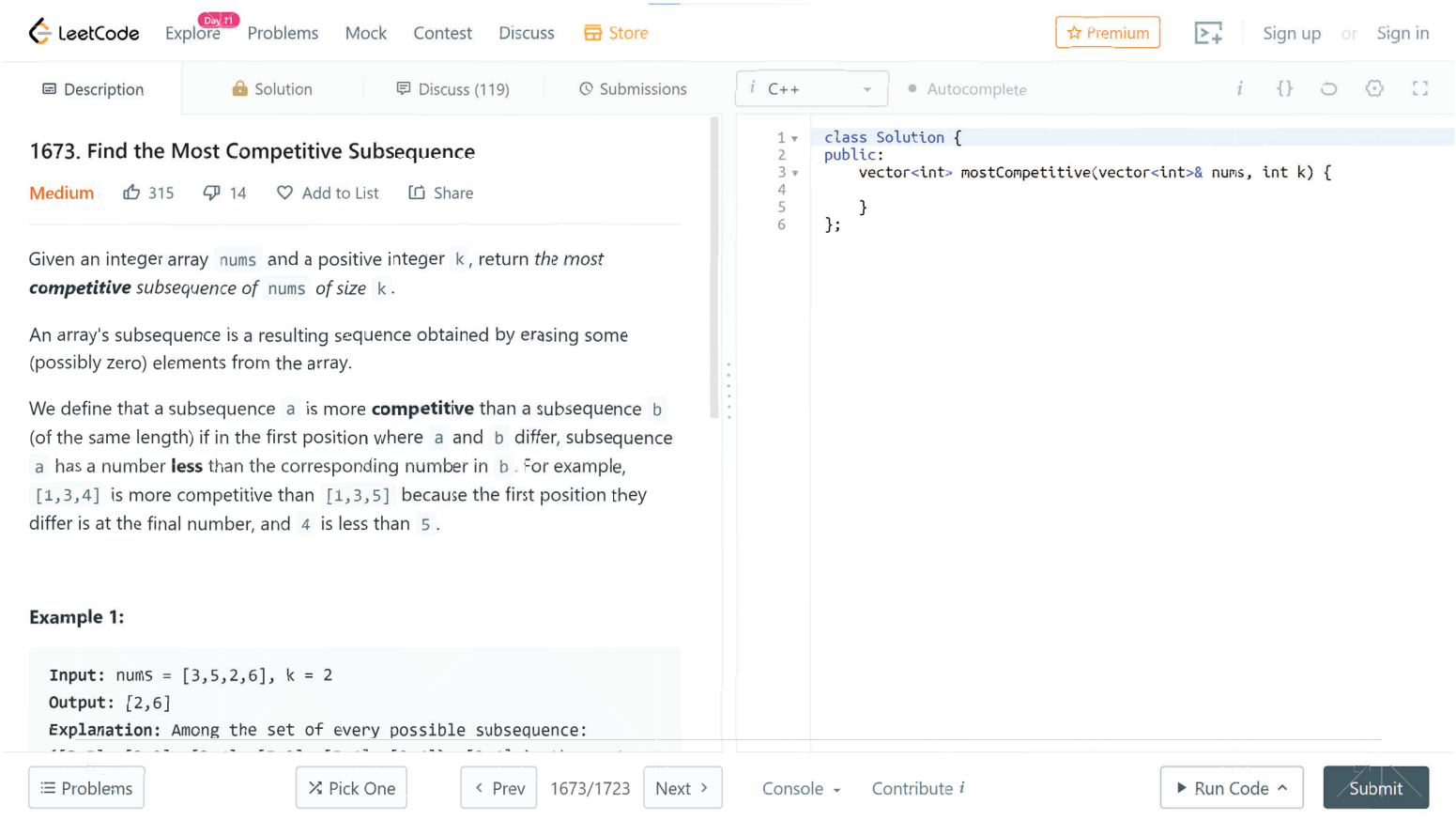
- `int Find(const std::vector<int>& numbers, int value)`

- Pass arrays by *reference* or use *iterators* (std::span since C++20)
- Use *const* modifier if you don't intend to modify the array.
- Feel free to use functions from the standard library. For instance, you can use std::find for linear search.

Optimization Problems

Find the optimal solution

- We are given a set of possible – *feasible solutions*
- Our goal is to find the best (e.g., most valuable or cheapest) solution among all feasible solutions.
- The number of *feasible solutions* is exponentially large, so we cannot go over all of them.



Description | Solution | Discuss (76) | Submissions | C++ | Autocomplete

630. Course Schedule III

Hard 933 36 Add to List Share

There are `n` different online courses numbered from `1` to `n`. Each course has some duration(course length) `t` and closed on d_{th} day. A course should be taken **continuously** for `t` days and must be finished before or on the d_{th} day. You will start at the 1_{st} day.

Given `n` online courses represented by pairs `(t,d)`, your task is to find the maximal number of courses that can be taken.

Example:

```
Input: [[100, 200], [200, 1300], [1000, 1250], [2000, 3200]]
Output: 3
Explanation:
There're totally 4 courses, but you can take 3 courses at most:
First, take the 1st course, it costs 100 days so you will finish
it on the 100th day, and ready to take the next course on the
101st day.
Second, take the 3rd course, it costs 1000 days so you will
finish it on the 1100th day, and ready to take the next course
```

```
1  class Solution {
2  public:
3      int scheduleCourse(vector<vector<int>>& courses) {
4
5      }
6  };
```