Lecture 1 Introduction

CS101 Algorithms and Data Structures

Instructor：Dengji Zhao; Yuyao Zhang; Zhice Yang

2020-09-08

Outline

- What will we learn from this course?

- How we manage this course?

Definition of Data Structure

- A data structure is a scheme for organizing data in the memory of a computer.

- The way in which the data is organized affects the performance of an algorithm for different tasks.

- 数据结构（data structure）是计算机中存储、组织数据的方式。通常情况下，精心选择的数据结构可以带来最优效率的算法（algorithm）。

Ex1 How to arrange books on the bookshelf?

Ex1 How to arrange books on the bookshelf?

- The following two operations are essential for efficiently arranging your books:

    - **Operation 1:** how to insert new books?

    - **Operation 2:** how to find/access an existing book?

Ex1 How to arrange books on the bookshelf?

- **Method 1:** randomly insert new books.

  - **Operation 1:** how to insert new books?

  Insert the book wherever there is an available space.
  Nice and easy!

  - **Operation 2:** how to find/access an existing book?

  It depends …

Ex1 How to arrange books on the bookshelf?

- **Method 2:** insert new books according to the alphabets order of the first letter.

  - **Operation 1:** how to insert new books?

  EX: we bought a new book "Algorithm".

  - **Operation 2:** how to find/access an existing book?

  EX: Binary search!

Ex1 How to arrange books on the bookshelf?

- **Discussion 1:** is **Method 2** absolutely better/more efficient than **Method 1**?

Method 1: randomly insert new books.

Method 2: insert new books according to the alphabets order of the first letter.

- **Discussion 2:** how can we further improve **Method 2**?

Ex1 How to arrange books on the bookshelf?

- **Method 3: cluster books according to different topics** (computer science, economics, agriculture, politics…), then insert new books according to the alphabets order of the first letter.

  - **Operation 1:** how to insert new books?

    EX: we bought a new book "Algorithm".

  - **Operation 2:** how to find/access an existing book?

    EX: Binary search for topic first, then binary search for book title.

- **Discussion 3:** how much space we should preserve for each topic? How many topics is an optimism option?

The efficiency of a method/algorithm highly depends on the organization&amount of the data.

Ex2 How to implement a function PrintN?

- Implement a function named PrintN, when input a positive integer N, print all the positive integer from 1 to N.

```c void PrintN (int N) { int i; for (i=1; i<=N; i++) { printf("%d\n" , i); } return; } ```	```c void PrintN (int N) { if (N); PrintN(N-1); printf("%d\n" , N); } return; } ```
Loop implementation	Recursive implementation

- Let N = 100, 1000, 10000, 100000, … …

Ex2 How to implement a function PrintN?

- Implement a function named PrintN, when input a positive integer N, print all the positive integer from 1 to N.

```c
# include <stdio.h>
void PrintN ( int N );
int main ()
{ int N;
  scanf ("%d", &N);
  PrintN(N);
  return 0;
}
```

```
10
1
2
3
4
5
6
7
8
9
10
Press any key to continue_
```

```
99977
99978
99979
99980
99981
99982
99983
99984
99985
99986
99987
99988
99989
99990
99991
99992
99993
99994
99995
99996
99997
99998
99999
100000
Press any key to continue_
```

Loop implementation

The efficiency of a method/algorithm depends on the occupation of RAM.

Ex3 compute the summation for a polynomial at a fixed value x.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

```
double fpoly1 ( int n, double a[ ], double x )
{ int i;
  double p = a[0];
  for (i = 1; i <=n; i++)
      p += (a[i] * pow( x, i) );
  return p;
}
```

$$f(x) = a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + x(a_n)) \cdots))$$

```
double fpoly2 ( int n, double a[ ], double x )
{ int i;
  double p = a[n];
  for (i = n; i > 0; i-- )
      p = a[i-1]  + x* p;
  return p;
}
```

- clock(): capture consumed time for running a function. The unit of the captured time is ***clock tick***, which depends on the CPU.
- CLK_TCK is a constant that presents the number of ***clock ticks*** per second.

```c
#include <stdio.h>
#include <time.h>

clock_t  start, stop;
/* Clock_t is the variable returned by function clock(). */
double duration;
/* Record the running time for a function. Time unit is second. */
int main ()
{
  start = clock ();  /* Start timing. */
  Myfunction();
  stop = clock ();   /* Stop timing. */
  duration = ((double) (stop - start))/CLK_TCK;

  return 0;
}
```

Ex3 compute the summation for a polynomial $f(x) = \sum_{i=0}^{9} i \cdot x^i$ at a fixed value $x = 1.1$, $f(1.1)$.

```c
double fpoly1 ( int n, double a[ ], double x )
{ int i;
  double p = a[0];
  for (i = 1; i <=0; i++)
      p += (a[i] * pow( x, i) );
  return p;
}
```

```c
double fpoly2 ( int n, double a[ ], double x )
{ int i;
  double p = a[n];
  for (i = n; i > 0; i-- )
      p = a[i-1]  + x* p;
  return p;
}
```

$$f(x) = \sum_{i=0}^{9} i \cdot x^i$$

```c
#include <stdio.h> #include <time.h> #include <math.h>
clock_t start, stop;
double duration;
#define MAXN 10  /*maximum order of the polynomial */
double fpoly1 ( int n, double a[ ], double x )
double fpoly2 ( int n, double a[ ], double x )
int main ()
{  int i;
   double a[MAXN]; /*save the coefficient of the
polynomial*/
   for ( i=0; i<MAXN; i++) a[i] = (double) i;

   start = clock ();
   fpoly1(MAXN-1, a, 1.1);
   stop = clock ();
   duration = ((double) (stop - start))/CLK_TCK;
   prinft ("ticks1 = %f\n",(double) (stop - start));
   prinft ("duration1 = %6.2e\n", duration));

   start = clock ();
   fpoly2(MAXN-1, a, 1.1);
   stop = clock ();
   duration = ((double) (stop - start))/CLK_TCK;
   prinft ("ticks1 = %f",(double) (stop - start));
   prinft ("duration1 = %6.2e\n", duration));
 return 0;
}
```

```
ticks1 = 0.000000
duration1 = 0.00e+000
ticks2 = 0.000000
duration2 = 0.00e+000
Press any key to continue
```

```c
#include <stdio.h>
#include <time.h>
#include <math.h>

......
#define MAXK 1e7
/*maximum repeat time of the test function */
......
int main ()
{ ......

  start = clock ();

  for ( i=0; i<MAXK; i++)
       fpoly1(MAXN-1, a, 1.1);  /* repeat the test function to get enough clock ticks*/

  stop = clock ();
  duration = ((double) (stop - start))/CLK_TCK/MAXK;
  /* compute running time for single function duration */
  prinft ("ticks1 = %f\n",(double) (stop - start));
  prinft ("duration1 = %6.2e\n", duration));

   ......

  return 0;
}
```

```
ticks1 = 10093.000000
duration1 = 1.01e-006
ticks2 = 1375.000000
duration2 = 1.38e-007
Press any key to continue
```

The efficiency of a method/algorithm depends on the design of the algorithm.

Definition of Data Structure

- **Data structure**, way in which data are stored for efficient search and retrieval.

- Different data structures are suited for different **operations**.

- **Algorithm** is a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation.

Abstract Data Type (ADT 抽象数据类型)

- Abstract: The method that we describe the data type, does not depend on the implementations.

  - Not related to the computer that stores the data.

  - Not related to the physical structure that stores the data.

  - Not related to the algorithm and language that implements the operation.

- We only care about "*how to design*" the objective data sets and related operations, not how to "*implement*" a data structure.

EX4 Abstract data type of a $matrix$

- Data type: $Matrix$
- Objects: a $M \times N$ matrix $A_{M \times N} = (a_{ij})$ $(i = 1, \cdots, M; j = 1, \cdots, N)$ is composed by a number of $M \times N$ array of $<a, i, j>$, where a present the value of the matrix element, i present the no. of row, and j present the no. of column.
- Operations: for an arbitrary matrix $A, B, C \in Matrix$, and integers i, j, M, N
- $Matrix \ create \ (int \ M, int \ N)$: return an empty matrix of $M \times N$;
- $int \ GetMaxRow(\ Matrix \ A \ )$: return the number of rows;
- $int \ GetMaxCol(\ Matrix \ A \ )$: return the number of columns;
- $ElementType \ GetEntry(\ Matrix \ A, int \ i, int \ j)$: return the element of matrix A in row i, column j;
- $Matrix \ Add \ (Matrix \ A, \ Matrix \ B)$: if the dimension of matrix A and B are the same, return matrix $C = A + B$, otherwise error;
- $Matrix \ multiply \ (Matrix \ A, \ Matrix \ B)$: if the number of columns of matrix A is equals to the number of rows of matrix B, return matrix $C = AB$, otherwise return error;
- ……

Outline

- What will we learn from this course?

- How we manage this course?

Course info

- **Instructors**

  ❑ Yuyao Zhang zhangyy8@shanghaitech.edu.cn SIST #2-302F

  ❑ Dengji Zhao zhaodj@shanghaitech.edu.cn SIST 1A304E

  ❑ Zhice Yang yangzhc@shanghaitech.edu.cn SIST 1A404F

- **TAs**

  ❑ 11 Tas (3 Masters, 6 Year-4, 2 Year-3)

    - Master: Yao Zhang, Yonghang Guan, Weikai Xu

    - Year4: Youjia Wang, Teng Xu, Yaoxin Ge, Haoxin Liu, Ze Song, Zhuo Chen

    - Year3: Yining She, Longwen Zhang

Course info

- **Classes**

  ❑ Tue 10:15-11:55; Thu 8:15-9:55

  ❑ 教学中心 203&302

- **Review/Quizzes/Discussions**

  ❑ Format: 5 groups, 40 students each (Week 2-16)

  ❑ Length: 60mins per week per group

  ❑ Location: TBA

  ❑ Time: Tuesday | Thursday

  ❑ Instructors: all TAs

 ❑ Contents: quizzes, and homework solutions

Course info

- Piazza Course Forum

  ☐ http://piazza.com/shanghaitech.edu.cn/fall2020/cs101/home

  ☐ You are encouraged to ask questions and participate in discussions

  ☐ Course schedule, slides, office hour etc. are published on the forum

  ☐ Invitation has be sent to your ShanghaiTech email (haven't received it?)

- Office Hours

  ☐ Location and Time: see course forum

- Homework

  ☐ Submit to gradescope https://www.gradescope.com/courses/180750

Course info

- Reference Book

  ❑ Introduction to Algorithms (3rd ed.). Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. MIT Press. ISBN，9780262033848.

  ❑ Algorithm design. Jon Kleinberg, Éva Tardos. Pearson. ISBN, 978-0321295354.
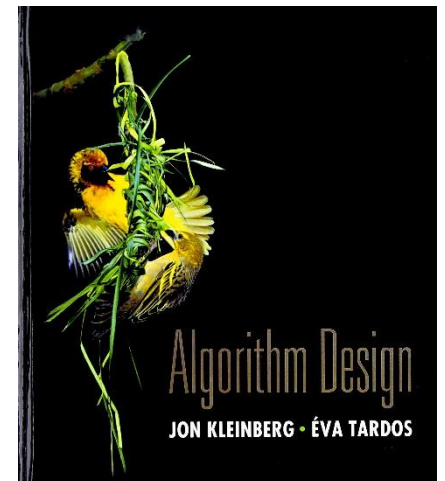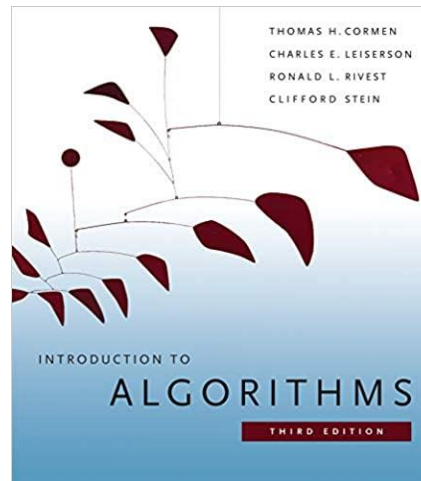
Course Schedule

Week	Date	Content
1	9/8 Tue	Introduction
	9/10 Thu	Elementary Data Structures: Array and Lists
2	9/15 Tue	Stack and Queue
	9/17 Thu	Big O/Theta/Omega
3	9/22 Tue	Hash Table
	9/24 Thu	Sorting: Insertion, Bubble
4	9/29 Tue	Sorting: Merge
	Public Holiday	
5	Public Holiday	
	10/8 Thu	Sorting: Quick
6	10/13 Tue	Divide and Conquer
	10/15 Thu	Trees: Introduction, DFS, BFS
7	10/20 Tue	Binary Trees
	10/22 Thu	Heap and Heap Sort
8	10/27 Tue	Binary Search Trees
	10/29 Thu	Balanced Binary Search Trees: AVL

Course Schedule

Week	Date	Content
9	11/3 Tue	Disjoint sets
	11/5 Thu	Middle Term Exam
10	11/10 Tue	Graphs: Introduction, Traversal
	11/12 Thu	Minimum Spanning Trees
11	11/17 Tue	Greedy 1
	11/19 Thu	Greedy 2
12	11/24 Tue	Topological Sorts
	11/26 Thu	Shortest Path Algorithm: Dijkstra
13	12/1 Tue	A*
	12/3 Thu	Floyd-Warshall Algorithm
14	12/8 Tue	Dynamic Programming
	12/10 Thu	Knapsack Problem
15	12/15 Tue	P+NP
	12/17 Thu	NPC
16	12/22 Tue	Median of Medians
	12/24 Thu	Review

Course Policy

- Plagiarism

  ❑ All assignments must be done individually

  - You cannot copy directly from any other source

  - You cannot share solutions with any other students

  - Plagiarism detection software will be used on all the assignments

  ❑ Ways of collaboration

  - You may discuss together or help another student such as debuging his or her code; however, you cannot dictate or give the exact solutions.

Course Policy

- Plagiarism

  ❑ Punishment

  - When one student copied from another student, both students are responsible.

  - Zero point on the assignment or exam in question.

  - Disqualified from receiving any awards recommended by the school and from any competitive studying opportunities (e.g., international exchange).

  - Repeated violation will result in a F grade for this course as well as further punishment at the school/university level.

Plagiarism: Example 1

- Alex and Bob were roommates.

- Bob let Alex use his laptop to complete an assignment.

- Alex copied Bob's solution for the assignment.

Plagiarism: Example 2

- Leslie asked if Morgan could send her his code so that she could look at it (promising, of course, not to copy it).

- Morgan sent the code.

- Leslie copied it and handed it in.

Plagiarism: Example 3

- Garry and Harry worked together on a single source file initially and then worked separately to finish off the details.

- The result was still noticeably similar with finger-print-like characteristics which left no doubt that some of the code had a common source.

Plagiarism: Example 4

- Jordan uploaded the projects to GITHUB.com without setting appropriate permissions. Kasey found this site, downloaded the projects and submitted them. Both are guilty.

  ❑ This applies to any public forum, news group, etc., not just gitub.com…

Real Plagiarism Examples

- Copied a piece of the codes from others or online repositories.

- Copied someone's solution from his/her USB drive.

- Copied a piece of others' codes and change all the variable/function names.

- Unusual solutions appeared in different submissions.

Course Policy

- **Grading**

  ❑ Exams (45%): middle term: 25%; final: 20%

  ❑ Weekly Homework (20%): non-programming questions

  ❑ Programming Tasks (20%): 4 programming tasks (each lasts 3 weeks)

  ❑ In-Class Quizzes (15%): in lectures and discussions

Take Home Message

- Algorithms and Data Structure is ~~one of~~ the most important course during your undergraduate.

- You will sometimes feel it very tuff, but please always keep on going, we are here to help.