Introduction and Input/Output

Competitive Programming I (Fall 2019)

Ninghui Li (Purdue University)

CP1 (1 credit)

Pre-requisite: 180

A new version is taught this semester

- Unit 1: Using arrays (3 weeks + 1 competition)
  - Prefix sums, two pointers, sliding windows, difference array, usage of Set, Map, PriorityQueue, Stack.
- Unit 2: Recursion and search techniques (3 weeks + 1 competition)
  - DFS, floodfill, BFS, backtracking
- Unit 3: Other topics (2 weeks + 1 competition)
  - Pointers and linked lists, bit operations

Examples will be given in Java or C++; program submission can be done in Java, C/C++, Python.

CP2 (will become a 2-credit course in Spring 2020)

New version will be first taught in Spring 2020. Format similar to CP1 this semester, with twice the workload.

In the future, may be offered just once per year, and taught by faculty

Topics:

- Graph algorithms, Dynamic Programming, String processing, Computational Geometry, etc.
- Can be viewed programming complement to 381.

CP3: Preparation for ACM ICPC (1 credit)

Will be offered every semester, and can be taken repeatedly.

In the future: Prepare for ICPC through extensive practice and discussions.

ACM ICPC: International Collegiate Programming Contest

- Yearly coding competition.
  - Region qualification contest
  - World Finals

- Teams of three from colleges around the world compete to solve the most coding problems in a limited amount of time.

Why Taking CP Courses

1. Become a better programmer / problem solver.
    a. Lots of practice of applying data structures and algorithmic ideas.
    b. Working on interesting and carefully selected problems, many have been extensively tested.
    c. Read others' code, solution ideas. Many problems have many submissions and discussions on line.
2. Help prepare for job interviews.
3. Prepare to compete in ICPC (The International Collegiate Programming Contest). CP1 is only the first step.
4. Have fun.

Region ICPC 2018

Course Pages

Official Course Page

https://www.cs.purdue.edu/homes/cs390cp/

Unofficial Page from my homepage, the schedule page may be more up to date here.

https://www.cs.purdue.edu/homes/ninghui/courses/390_Fall19/

Grading Policy for CP1

- Out of total of 115: 90+ for A- or higher, 80+ for B- or higher
- Attendance: 10 out of 115:
- Problem sets: 40 out of 115: For each of the 8 lecturing weeks, seriously attempt at least 5 (example or additional) problems during the week after the lecture. (You will be asked to put down how many problems you've attempted, and how many you have solved in next week's signup sheet.)
- Three competitions: 65 out of 115. Competition held during lecture time in classroom/lab. Each lasts 110 minutes, but remains open for six days. Students can solve problems in competitions after 110 minutes, for reduced grades. (Competition should be done individually with no external help.)

HackerRank

- Our competitions will be on hackerrank.com.
  - Take the appropriate Hackerrank Username Registration survey on course webpage
- Ask for help on the course piazza page.
  - This is the primary channel of communication.
  - Mark your post as private if you are sharing your code segments with TAs.
- We cannot reveal the test files, even after the course is over.
  - If you think there is an error with a specific test case, please post on Piazza. This is rare.
- For competition, cheating is defined as obtaining help from any other person, posting problems anywhere, or sharing solutions in any way. Code will be checked for similarities.

Easy Multiplication: Problem 1 of Introductory Competition

Given a stream of integers, whenever seeing 0, output the remainder modulo 1e9+7 of the product of all the numbers inputted so far (0's not included). Also output product at the end.

Input Format:

Each integer (absolute value less than 1e9) is entered in a new line. End the program when you see a blank line.

Output Format: Output on a new line each product modulo 1e9 + 7.

Three things to know for this problem: How to parse inputs. Choose appropriate primitive type. Know the caveat of %.

Create a Template Code for Reading Data

Possible goals of template code:

- Code for reading input is intuitive and requires not too much typing.
- Sufficiently efficient
- Can accommodate both standard in and file input
- Try throwing exceptions when reading logic doesn't match data

Java input choices

- Scanner (uses regular expression for parsing, slow)
- StreamTokenizer (may parse things differently from what we need)
- StringTokenizer(BufferedReader.readline()) (okay)
- BufferedReader.readline().split() (okay)
- Using Reader class, and does paring oneself (fast and long, low-level code)

Resources for Java I/O

See Fast I/O in Java in Competitive Programming at Geeksforgeeks for code for different options

Personally, I use a template modified from method 3 of the above article, using a user-defined class that uses BufferedReader + StringTokenizer

Speedtest Results: Reading and Parsing

	Scanner	Stream-Tokenzier	String-Tokenzier	String split	Stream-Reader
100K numbers 600K file size	161 ms	54 ms	67 ms	74 ms	11 ms
1M numbers 6.7M file size	755 ms	210 ms	162 ms	160 ms	27 ms
10M numbers 79M file size		1910 ms	1042 ms	899 ms	206 ms

For C++

If you code in C++ and use C++ I/O exclusively, you can use

```
std::ios::sync_with_stdio(false);
```

This disables the synchronization between the C and C++ standard streams. By default, all standard streams are synchronized, which allows one to mix C and C++ style I/O and get sensible and expected results. If you disable the synchronization, don’t mix C++ and C I/O.

```
cin.tie(nullptr);
```

This unties `cin` from `cout`. Tied streams ensure that one stream is flushed automatically before each I/O operation on the other stream. Tied streams are needed for interactive programs, and are unnecessary for CP.

Ranges of Java Primitive Types

Type	Sizes (bytes)	Range
byte	1	-128 to 127
short	2	-32,768 to 32767
int	4	App. -2.15e9 to 2.15e9
long	8	App. -9e18 to 9e18
float	4	App. -3.4e38f to 3.4e38f
double	8	App. -1.8e308 to 1.8e308
char	2	0 to 65,536 (unsigned)

A Little Bit of Number Theory

Definition. Given an integer X and a positive integer K, the **remainder** R **of X modulo K** is the remainder when dividing X by K.

- More precisely, R is the unique integer in [0, K) such that there exists an integer Q where X = Q*K + R.
-

How to compute remainder of X modulo K?

In Java: 12%7 == 5, but -2%7 == -2?

However, -2 modulo 7 is 5 because -2 = (-1)×7 + 5.

When X is negative, you need to use something like

((X % K) + K) % K to compute X modulo K

A Little Bit More about Number Theory

Fact: (X-Y) is a multiple of K if and only if the remainder of X modulo K is the same as the remainder of Y modulo K.

Example. (12- (-2)=14) is multiple of 7, and their remainders modulo 7 are both 5.

Fact. $(X \times Y \bmod K) = ((X \bmod K) \times (Y \bmod K)) \bmod K$

Example 1. $12 \times 4 \bmod 7 = 48 \bmod 7 = 6$

$(12 \bmod 7) \times (4 \bmod 7) = 5 \times 4 = 20$ $\quad 20 \bmod 7 = 6$

Example 2. $(-2) \times 3 \bmod 7 = -6 \bmod 7 = 1$

$(-2 \bmod 7) \times (3 \bmod 7) = 5 \times 3 = 15$ $\quad 15 \bmod 7 = 1$

Sources of Problems Used in the Course

LeetCode

- Some problems have solution explanation.
- The discuss page for each problem has many solutions posted by users.
- The page for each of your submission contains figures showing time and space distribution, clicking shows sample solutions. (Can check what do the fastest solutions do.)

Kattis

- Problems used in this course taken from the Methods to Solve webpage, which is associated with the Competitive Programming book.
- That page has a one-line hint for the problems
- There is a Kattis Hint Giver for chrome browser (haven't tried)

Sources of Problems Used in the Course

USACO

- Three levels (Bronze, Silver, Gold) before the 2015-2016 season
- Four levels (Bronze, Silver, Gold, Platinum) since December 2015
- This course mostly use the old Bronze and new Silver problems.
- After creating an account (anyone can do this) and logging in, one can submit solutions (except for the 2011-2012 season)
- The contest page includes test data and solution
    - If one fails a test case, one can download and examine the data.
    - One can read solution discussion and code.

The Big-O Notation

- We consider the running time (or space consumption) of a piece of code (or an algorithm) as a function of its input size, typically denoted by N.
- The exact number would depends on the units one is using, and factors such as computer speed, compilers, …
- To abstract away from that, we use O(f(N)), which f(N) is a function of N.
- Typical complexity include O(N), O(N^2), O(N logN), O(N^2 logN), O(2^N), O(N!), etc.
- Time complexity O(N) means that for whatever units of measuring time, and run-time environment, there exists a constant C such that the running time in input of size N is no more than C*N.

Format of Test Files

- Typical format:
  - Single Case
  - Multiple cases with case number
  - Multiple cases end with sentinel:
    - New line
    - 0 / # / -1 etc.
    - EOF

- This week’s contest is for learning about different formats of input test files.
  - Develop a template that you like.

Debugging

- Accepted! (AC) - Yay!!!!!

- Compiler Error (CE)
- Runtime Error (RE)
- Presentation Error (PE)

- Time Limit Exceeded (TLE)

- Wrong Answer (WA)

CPU Club

- The competitive programming club is for anyone who is interested in competitive programming
- The primary focus of CPU Club this Fall is to participate in ICPC.
- It is more accelerated and will focus on working in a team to solve programming problems in the ICPC format.