CS 31100-CP2 Competitive Programming II (Fall 2023)

Welcome (back) to Competitive Programming!
Classes are held on Thursdays from 6 PM to 7:50 PM in LWSN B151.

1 Contact Information

1.1 Instructor

- Name: Zhongtang Luo
- E-mail: luo401@purdue.edu
- Office Hours: Wednesday 2:00 - 3:00 PM at HAAS 271 or by appointment

1.2 Teaching Assistants

- Name: Otavio Sartorelli de Toledo Piza
- E-mail: osartore@purdue.edu
- Office Hours: Tuesday 10:30 AM - 12:00 PM, Thursday 10:30 AM - 12:00 PM at HAAS G072
- Name: Egor Gagushin
- E-mail: egagushi@purdue.edu
- Office Hours: Monday 11:00 AM - 12:30 PM, Wednesday 2:00 - 3:30 PM at HAAS G072

1.3 Supervising Instructor

- Name: Ninghui Li
- E-mail: ninghui@purdue.edu

2 Course Description

CP2 teaches experienced programmers additional techniques to solve interview and competitive programming problems and builds on material learned in CP1. This includes specific algorithmic techniques, advanced algorithms, advanced problem types, and more. (See course schedule.) It can be viewed as a programming complement to CS 381, with some overlap in content.

The class will consist of lectures where we will review a topic and then cover problems within that topic. During the problem review part, students are expected to interact with each other and then with the rest of the class in a think-pair-share format. This facilitates group discussion and strongly reinforces the methodologies we use for problem solving.

3 Course Design

This course was designed based on engineering education principles surrounding Content, Assessment, and Pedagogy. It has been through several teaching iterations and professional revisions in order to get to this point. If you have

questions about why we do things a certain way, please reach out and ask! We are happy to discuss any of these things and more with you.

4 Enduring outcomes

- Knowledge of how to solve all the major advanced categories of competitive programming problems quickly;
- Knowledge of how to decipher a problem to know what type(s) of problem it is;
- Knowledge of how to decompose a problem into its composite parts;
- How to analyze a potential algorithm in space and time using standard computer science notation.

5 Important To Know Outcomes

- How to solve certain specific problems that are canonical examples of the major advanced categories;
- Ability to implement advanced new/novel algorithms from pseudocode.

6 Learning Objectives

At the end of this course, students will be able to:

- Differentiate between advanced categories of problems in computer science including backtracking, simulation, advanced dynamic programming, advanced graphs (including trees and DAGs), mathematics, string processing, and range queries, as well as subcategories in each of these categories;
- Implement well-known solutions (as discussed in class) to advanced categories of problems in computer science (listed in LO1). Identify when a problem spans multiple categories of problems (as listed in LO1);
- Design algorithms to solve problems that span multiple categories of problems;
- Use new/novel algorithms to solve advanced problems;
- Recognize key characteristics of certain problem types listed in LO1;
- Classify an advanced problem by those key characteristics;
- Deconstruct an advanced problem into subproblems that can be solved individually;
- Determine runtime and space usage of a potential solution using big-O notation to judge if the potential solution will work;
- Create an efficient solution to a problem based on analysis of the problem type, the deconstructed problem parts, and the time and space constraints of the problem;
- Reflect on how one came up with a solution to a problem in order to better recognize patterns in problem types.

These learning objectives were generated with careful thought and consideration of Big Ideas, Guiding Concepts, Enduring Outcomes, Important-to-Know Outcomes, Difficult Concepts and Misconceptions and was designed with reference to the Revised Bloom Taxonomy for Educational Objectives.

6.1 Method of Evaluation

Weekly problem sets with 3-4 problems each. To get credit for a problem, one must submit code that passes all test cases, meaning outputting the correct answers under the specified time and memory limitation. This ensures that students must use sufficiently efficient algorithms and implementations and think through corner cases. When an implementation is partially correct, one must spend a substantial amount of time debugging with known input and output. Note that this "singular" (but very wide) method of evaluation is very common in competitive programming courses around the world. We have found that the most effective way of ensuring the students have a strong understanding of the material is to assign multiple short coding problems a week over certain topics, such that they accumulate a significant amount of experience by the end of the semester. This repeated experience reinforces the material to a much higher degree than tests or quizzes or conceptual-based assignments can.

6.2 How to Succeed in this Course

In order to be successful, students should:

- Be familiar with:
  - Programming in Java, C++ or Python;
  - Basic data structures and algorithms encountered in CP1 in topics like the 4 major programming paradigms (complete search, greedy, divide and conquer, dynamic programming), other algorithmic topics (binary search the answer/bisection, meet-in-the-middle, prefix sum and difference arrays, two pointers, sliding window), and basic graph algorithms covering strongly/connected components, floodfill, topological sort, and shortest paths;
  - We will briefly review important concepts as the need arises.
- Actively participate in class discussions. Practice is by far the best way to become better at CP and the topics therein.
- Think about how you came up with a solution after you do so. The more you can identify these indicators, the faster you will get a good grasp on CP material.
- Practice, practice, practice! That is the tried and true way to get better at CP as testified by many competitive programmers!

Prerequisites: CP290-CP1 (strongly recommended) OR instructor approval.

7 Course Policies

Students may leave early/arrive late if they are quiet upon entry and exit. If you miss a large portion of class, contact the professor to inquire about missed materials. The only penalty for missing a class is that you will not have access to the course materials (slides, recording) for that session.

Students may use whatever technology they wish during class if it is not disruptive to others. However, it is recommended to refrain from using your devices as group discussion is performed frequently.

There is no preparation required for class.

If technology issues arise with assignments, contact the professor immediately.

If you are going to talk in class, please WHISPER.

8 Learning Resources, Technology, and Texts

8.1 Course Text (OPTIONAL)

Competitive Programming 4 by Halim, Halim, and Effendy. Content in CP1 is drawn mostly from the first half (book 1) and content in CP2 is drawn mostly from the second half (book 2). CP3 utilizes special topics in book 2.

8.2 Discussion Forum

Signup for Piazza. (See Brightspace for a link.)

Piazza is one official source of announcements in this class (besides Brightspace).

You are expected to be courteous, respectful, and professional when posting at the discussion forum.

- **When making a private Piazza post with questions about failing a test case, if you are debugging you must provide a description of both what your approach is as well as what you have done to try to solve the issue. We expect you to have tested your code with your own test cases before you ask us to review your code.**
  - If you do not provide these two things, we will ask you to provide them before we help you.
  - Consider the following types of edge cases to write (if the situation warrants): cases that cause integer overflow, very large input cases, very small test cases, test cases that represent the maximum input constraints, cases that are made to test the edges of the problem, just to name a few. It could be beneficial to use a programming language such as python to systematically generate test cases.

- Piazza is also intended for clarification questions of general interest. If you have a general clarification question about a homework assignment or lecture this is the right place to ask.
- If you are not sure whether a posting is appropriate, make sure it is private, i.e., if you are asking for instructor feedback about your source code/current solution the question must be private.
- Do not use Piazza to post answers to any of the assigned problems!
- Piazza is not the forum for complaints about an assignment or the class. (Please bring any concerns to the attention of the instructor.)

8.3 Problem Sets

Problem sets will be given on Kattis. (See Brightspace for a link.)

9 Assessment

The only assignments in this course are the weekly homework problems (usually 3). Each week, students will have an opportunity to earn bonus credit (the details are described below). Additional bonus point opportunities may be offered throughout the semester.

9.1 Grade Calculations

There are 12 problem sets (1 per week) on purdue.kattis.com. The grading scale is as follows:

- A+ for solving 33 or more problems, A for 29, A- for 27;
- B+ for 25, B for 23, B- for 21;
- C+ for 19, C for 17, C- for 15 (last passing grade);
- D+ for 13, D for 11, D- for 9;
- F for below 9.

Additionally, to get a grade of C- or above, a student must complete at least one problem from 11 different problem sets (i.e., 15 points distributed across at least 11 topics). This is to ensure a comprehensive knowledge of all topics.

The course can be taken based on pass/fail. A student must achieve a C- for a Pass, including the requirement of one problem from 11 different problem sets. **THE PASS/FAIL DEADLINE FOR A 12 WEEK COURSE COMES BEFORE THE NORMAL PASS/FAIL DEADLINE!**

9.2 Weekly Bonus Point

Every week, you are given the opportunity to earn 0.66 points on your final grade as defined above (the equivalent of solving 2 additional problems every 3 weeks). To do this, you will need to submit well-formed, commented solutions for each problem (note that you must have successfully completed the problem). Each problem solution you submit will be worth 0.22 points for a total of 0.66 points per week. Whether or not your submission meets the guidelines is ultimately up to the discretion of the grader, but they have the following guidelines.

"Well formed" is defined as the following:

- Not messy competition code (example in the slides). Cleaned up competition code is ok, but it must be easy to follow.
- Overall cleaned up and looks good (as you've been taught in other classes).
  - i.e. don't submit code with variable names like "mom" or "thisClassBoresMe".
  - Remove debugging statements and commented out code (unless it provides some sort of insight into your solution).
  - The code must actually produce the correct answer (meaning that if you don't solve the problem you won't be able to get the extra credit on that problem).

"Commented" is defined as the following:

- We are not going to be extremely strict on comments. Write comments that briefly (1-2 lines max) explain your thought process/code organization throughout your code and in particular at complicated parts.

- There is absolutely no need to submit something as formal as JavaDocs or function headers (unless you feel it is warranted).
- **You must include a comment at the top that looks like the following:**
  `Author: My Name`
  `It is [is not] ok to share my code anonymously for educational purposes.`
  (See details below for this.)

The bottom line is to abide by this principle: **write such that a random CS person (from freshman to professor) in Lawson could immediately walk through your code and understand what it is supposed to do.**

Note that this means that you can solve roughly 2.5 problems a week and as long as you submit the bonus points opportunity for every problem you solve, you will get an A+ ($2.5 \times 12 + 2.5 \times 12 \times 0.22 = 36.6 > 33 =$ A+).

9.3 Purpose

There are an abundance of reasons why you should write clean code, but here are our primary focuses:

- In interviews, if you write messy, unorganized and uncommented code you will not do as well, so it's good to get in the practice.
- If you complete this assignment regularly, it will help you learn to write code cleanly as you go instead of all at once at the end.
- Writing comments as you go helps both you come back and others read your code for the first time and easily understand what it does and what its purpose is.

9.4 Logistics

On Brightspace, there will be 1 optional assignment/week where you will submit code for any of the three problems you wish to get extra credit for. Each problem is worth either 0 or 0.22 points, for a total of 0.66 homework points per week In every file, you must Put an "author" line near the top of the file. Indicate whether or not it is ok to show your (anonymous) solution to others. This is to promote understanding of different ways to solve the problem and ways to program. They will be due at 12 PM noon the Friday after the problem set is due. This is to let you focus on solving the problems before their deadline (after your problem set deadline) as well as to give the TAs time to grade them before weekend (noon Friday and not midnight). Aka 12 hours after the problem set is due. We will not take late submissions without good reason (as determined by the syllabus – see below).

9.5 Important Note

The upload of these anonymized submissions are for the purposes outlined above – mainly to help you understand different approaches or styles or coding languages.

These solutions are for ACADEMIC PURPOSES ONLY and MAY NOT be used for ANY OTHER REASON. Additionally, you may not share these solutions with students who have not taken this course without direct permission from the current Professor(s) and/or GTA of this course.

9.6 Late Days

- Three total late days over the semester (in day increments)
- You MUST submit the bonus credit for the problem (bonus credit submission time counts in late days)
- Submit the bonus credit to the late submission assignment on Brightspace to indicate that it is done (filename must match problem name).
- The day recorded on Brightspace for your bonus credit submission will be the day used to calculate remaining late days.
- Submission method will be announced in class.

10 Academic Integrity

Academic Integrity is a critical foundation for any form of higher education, and Purdue University takes this concept seriously. All submitted assignments will automatically be checked for plagiarism. Any student found guilty of plagiarism and/or other forms of academic dishonesty will automatically fail this course, and face any additional consequences that the University deems necessary. To know and understand what is academic integrity, what is expected from you, and what you should NOT do, read carefully this document: Academic Integrity. **Posting homework questions (or any other part of an assignment) online is NOT ALLOWED.**

Examples of academic dishonesty include (but are not limited to):

- Searching for solutions online and copying (whole or portions) of the code in your submitted solution.
- Copying (whole or portions of) other (current or past) student's solutions.
- Share your solution code with another student.
- Submitting code that is based on reverse-engineering of the expected answer without solving the problem. Such code would fail on almost all test data similar to the one used in the system.
- Submitting a solution which you do not understand (e.g., would not be able to explain to the TA/professor)
- **Submitting a solution open.kattis.com for a problem assigned during its assignment window.** Contact the instructor if you need help submitting to purdue.kattis.com.

The Kattis platform checks submissions against other users' submissions and warns the instructor when matches are found.

We encourage you to interact among yourselves, so long as the activities do not fall under the above categories.

Warning regarding Github Copilot and other automatic code-generation software:

If your submission is flagged as similar or identical to other submissions on Kattis and it is because you used Github Copilot or other automatic code generation software, this will count as a violation of the academic integrity policy. By using automatic code generation, you are switching from a creative process to a review-based process. It becomes much harder to spot errors and prevents you from learning the material to the standards required.

11 Tentative Course Schedule

- Topic 0: Introduction, Simulation and Ad Hoc
- Topic 1: Scan Line: Discretization, Monotonic Stack/Queue
- Topic 2: Range Query: RMQ and Fenwick Tree
- Topic 3: Range Query: Segment Tree and Lazy Update
- Topic 4: Geometry: Basics, Convex Hull, Convex Optimization
- Topic 5: Number Theory: Prime Numbers, GCD, exGCD
- Topic 6: Combinatorics: Counting, Inclusion-Exclusion, Linear Recurrence
- Topic 7: Tree: LCA, DP, DFS Order
- Topic 8: Graph: Matching, Network Flow
- Topic 9: String: Hash, KMP, AC Automata
- Topic 10: String: DP, Bitmasking, Plugging
- Topic 11: Random Problem Solving

12 Nondiscrimination Statement, Attendance Policy, Emergency Preparedness

Links to these and other university policies are provided on Brightspace under "University Policies."

13 Students With Disabilities

Purdue University strives to make learning experiences as accessible as possible. If you anticipate or experience physical or academic barriers based on disability, you are welcome to let me know so that we can discuss options. You are also encouraged to contact the Disability Resource Center at: drc@purdue.edu or by phone: 765-494-1247.

We cannot arrange special accommodations without confirmation from the Disability Resource Center. It is the student's responsibility to notify the Disability Resource Center (http://www.purdue.edu/drc) of an impairment/condition that may require accommodations and/or classroom modifications.

14 Mental Health Statement

If you find yourself beginning to feel some stress, anxiety and/or feeling slightly overwhelmed, try WellTrack. Sign in and find information and tools at your fingertips, available to you at any time.

If you need support and information about options and resources, please see the Office of the Dean of Students for drop-in hours (M-F, 8 am-5 pm).

If you're struggling and need mental health services: Purdue University is committed to advancing the mental health and well-being of its students. If you or someone you know is feeling overwhelmed, depressed, and/or in need of mental health support, services are available. For help, such individuals should contact Counseling and Psychological Services (CAPS) at 765-494-6995 during and after hours, on weekends and holidays, or by going to the CAPS office of the second floor of the Purdue University Student Health Center (PUSH) during business hours.

More information regarding mental health and basic needs security can be found under the Student Support and Resources tab on Brightspace.

15 Diversity & Inclusion Statement

In our discussions, structured and unstructured, we will explore a variety of challenging issues, which can help us enhance our understanding of different experiences and perspectives. This can be challenging, but in overcoming these challenges we find the greatest rewards. While we will design guidelines as a group, everyone should remember the following points:

We are all in the process of learning about others and their experiences. Please speak with me, anonymously if needed, if something has made you uncomfortable. Intention and impact are not always aligned, and we should respect the impact something may have on someone even if it was not the speaker's intention. We all come to the class with a variety of experiences and a range of expertise, we should respect these in others while critically examining them in ourselves.

16 Course Evaluation

During the last two weeks of the semester, you will be provided with an opportunity to give feedback on this course and your instructor. Purdue uses an online course evaluation system.

You will receive an official email from evaluation administrators with a link to the online evaluation site. You will have up to 10 days to complete this evaluation. Your participation is an integral part of this course, and your feedback is vital to improving education at Purdue University. We strongly urge you to participate in the evaluation system.

17 Copyright

Online educational environments, like all learning environments, should provide opportunities for students to reflect, explore new ideas, post opinions openly, and have the freedom to change those opinions over time. Students enrolled in and instructors working in online courses are the authors of the works they create in the learning environment. As authors, they own the copyright in their works subject only to the university's right to use those works for educational

purposes (visit Purdue University Copyright Office). Students may not copy, reproduce or post to any other outlet (e.g., YouTube, Facebook, or other open media sources or websites) any work in which they are not the sole or joint author or have not obtained the permission of the author(s).

18 Modifications

This syllabus may be modified at any time. Any modifications will be announced via e-mail and Brightspace announcements.

19 Credits

The current course content, design, and syllabus were originally created by Ethan Dickey during Spring and Fall of 2022. He has experience in designing content for and teaching competitive programming, experience and passion in designing courses and is currently working on completing the Teaching and Learning in Engineering Graduate Certificate offered by Purdue University.