Syllabus

CS 41100, CP3 Competitive Programming III (Spring 2025) **Zhongtang Luo**
Purdue University March 30, 2025

1 Course Information

- Course Number and Title: CS 41100, Competitive Programming III
- CRN: 69765-CP3
- Meeting Time & Location: Tuesday and Thursday 5:30 pm–6:45 pm, LWSN B134
- Course Credit Hours: 2
- Course Web Page: `https://cs.purdue.edu/homes/luo401/teaching/cs411-spring2025`
- Prerequisites: Undergraduate Level CS 31100 Minimum Grade of C

2 Contact Information

2.1 Instructor

- Name: Zhongtang Luo
  - E-mail: luo401@purdue.edu
  - Office Hours: Monday 2:00 pm–3:00 pm at DSAI B022 or by appointment

2.2 Teaching Assistant

- Name: Jimmy Dinh
  - E-mail: dinh16@purdue.edu
  - Office Hours: Monday 3:30 pm–5:00 pm at DSAI B047, Friday 11:30 am–1:20 pm at LWSN Commons
- Name: Leo Lee
  - E-mail: lee4247@purdue.edu
  - Office Hours: Monday 3:30 pm–5:00 pm at DSAI B047, Friday 11:30 am–1:20 pm at LWSN Commons
- Name: Egor Gagushin
  - E-mail: egagushi@purdue.edu
  - Office Hours: Thursday 3:30 pm–5:30 pm at DSAI B022

2.3 Supervising Instructor

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

3 Course Description

CP3 teaches experienced programmers additional techniques to solve competitive programming problems and builds on material learned in CP1 and CP2. This includes algorithmic techniques. Primarily, CP3 prepares students to compete in programming contests, which means most class time is focused on simulating contest environments and teaching teamwork and communication alongside problem practice.

The course revolves around three aspects that are essential in problem-solving. Together they form the enduring outcomes of the course:

EO-1. **Observation skills** reduce a new algorithmic problem to a known problem that can be solved.
EO-2. **Techniques** solve known algorithmic problems efficiently.
EO-3. **Implementation** by coding and debugging builds a solution.

A comparison of CP1, CP2 and CP3 is available below:

Course	Focus
CS 21100 - CP1	Basic observation skills. Some techniques on graph problems.
CS 31100 - CP2	Summarization and expansion of CP1 observation skills. Basic techniques on various topics.
CS 41100 - CP3	Review of CP1 observation skills and CP2 techniques. Advanced techniques. Implementation practices.

3.1 Learning Outcomes & Assessment

Students will be able to …

LO-1. Select and use appropriate observation skills covered in CP1 (*Search, Greedy, Dynamic Programming, BSTA*), CP2 (*Pruning, Perspective, Sweep Line, Monotonic Queue, Dynamic Programming on Tree, DFS Order on Tree, Bitmask*) and CP3 (*Monotonicity, Offline*) to understand and reduce problems to known algorithmic problems. (EO-1)

LO-2. Select and apply appropriate techniques covered in CP1 (*DFS/BFS, Shortest Path, Floodfill, Topological Sort, Tarjan, Union Find Set, Minimum Spanning Tree*), CP2 (*Binary Exponentiation, Linear Sieve, ExGCD, Combinatorics, Inclusion-Exclusion, Sparse Table, Fenwick Tree, Basic Computational Geometry, Convex Hull, Rolling Hash, Trie*) and CP3 (*Half-Plane Intersection, Adaptive Simpson, Linear Recurrence, Segment Tree, LCA, HLD, Network Flow, KMP, AC Automata*) to solve known algorithmic problems. (EO-2)

LO-3. Implement and debug a solution to algorithmic problems efficiently. (EO-3)

LO-4. Combine LO-1 – LO-3 to solve algorithmic problems, performing at Candidate Master (rating 1900+) level in Codeforces contests. (EO-1 – EO-3)

3.1.1 Performance Task

To assess LO-1 – LO-4, the student is provided with three competitive programming problems every assessment session (contest). The online judge website grades every problem on a pass/fail basis. The student gets instant feedback and can revise within the time frame. The student is either assessed individually or in teams of three. The student is tasked to solve the problems with the following rules, consistent with ICPC-style contests:

1. You have 90 minutes to solve 3 problems in class. Problems solved in class net 2 points.
2. However, you are not required or expected to solve every problem in class! You may bring unsolved problems back home and upsolve them as homework before the due date. Every problem nets 1 point as homework.
3. As a rule of thumb, you need 4 points every week for an A, and 2 points every week for a P over 12 sessions. 2 lowest performances are dropped when calculating the grade.
4. You may bring your laptop or use the lab computer. However, cellphones and the Internet are **not** allowed other than submission.
5. Instead, you may bring a printed-out reference to help you solve the problem. You must type the solution yourself.
6. For educational purposes, we may ask you to present your code in class.
7. To track your progress in the course, you are required to fill in the solve information and your reflections in a Google Sheets document. We will provide the relevant template in class.
8. **For team contest:** One teammate is allowed to code **no more than** one problem in class. You should discuss and decide who codes what after reading the problems. A solve in the team counts as 2 points for every team member. (For teams of two, you may attempt the third problem with anyone as long as the first two solves are done by different people. To put it more elegantly and generally, the difference of solves between team members in contest should not be greater than one.)
9. **For team contest:** Homeworks (upsolving) are still individual and no plagiarism is allowed. Only the coder can use the partial solution code he/she writes in class if there is one.

Problem Selection For the three problems every session:

1. Problem 1 reviews and examines the student's knowledge from CP1 and CP2. The problem's solution involves skills and techniques from these two prior classes, with a focus on the observation skill (EO-1).
2. Problem 2 assesses the student's understanding of the current topic in CP3. The problem's solution involves techniques from the latest topic (EO-2).
3. Problem 3 assesses the student's implementation abilities. The problem involves extensive coding and debugging and requires a mastery of these abilities from the student (EO-3).

The order of the problems may be randomized.

Upsolving Trying to solve these problems during the contest session is not all there is to this course! In fact, what you successfully do in the contest is no more than a practice and review of topics you already understand. On the other hand, what you failed to solve in the contest is a golden opportunity to compare yourself with other students, identify your own weaknesses and do some catch-up. **For this**

reason, I ask you to write a very short reflection (around 10 words) on every problem in the Google Sheets document, together with the solve information. The following checklist may be helpful for you to reflect on your progress on every problem after every contest.

Level 1. **I solved the problem within the allotted time.**
Congratulations! You have demonstrated your capabilities in the contest. If you feel that there is still room to improve your efficiency, you are more than welcome to utilize the advice below. Otherwise, feel free to move on.

Level 2. **I solved the problem after the contest.**
A perfect student should be able to get 2 problems every session. If you feel you are ever stuck for a significant period in the contest for some reason, feel free to check below on the reason you get stuck for some advice. You can also drop in during my office hours and we can walk through your contest process to identify the ways to improve your efficiency.

Level 3. **I know how to solve the problem, but my code runs into some bugs.**
Debugging is often a painful process for many people, including myself. You may want to try a variety of debugging methods (i.e. static, dynamic, stress test, etc.). You may find it helpful to write down the exact bug (integer overflow, typo, logic issues, etc.) in the reflection, as knowing what bug happens most frequently in your code helps you to speed up your debugging process significantly.
You can also drop in during any office hours if you cannot find the bug in your code.

Level 4. **I know how to solve the problem, but I don't know how to implement it.**
Unfortunately, as we enter CP3, the complexity in the code's logic steadily increases, and you will start to face what we call implementation-heavy problems. These problems resemble what you will face in your future software engineering jobs: not so algorithmically complicated, but rather a bunch of logic that you have to implement correctly. I often find it helpful to take out a pen, think about and write down the structure of the code. What is the most concise way to implement the logic? How many functions do I need to make? What is the logic flow in each function? As I think through these aspects, I gradually begin to grasp what I need to code and become more comfortable to code it.

Level 5. **I know the related technique, but I don't know how to solve this specific problem.**
As we encounter more and more techniques in CP2 and CP3, a common difficulty is to link the technique to the problems we are solving. In general, I do not find it very helpful to get stuck thinking about any problem for more than 1 hour if you have absolutely no idea how to solve it. Instead, after we go over the solution in the lecture, try to ask yourself: *How could I have thought the solution myself? What is the observation I am missing here?* In other words, you need to identify a possible way for you to come up with the solution yourself so that you do not get stuck in future contests on a similar problem.
If you find it difficult to conjure a possible way for you to come up with the solution yourself, feel free to talk to me after class or during office hours.

Level 6. **I don't know the related technique.**
In a sense this is the easiest thing to fix: you already understand what you don't know! Fortunately, there are a lot of online resources for every technique we covered in CP1, CP2 and CP3. I generally found Google to be reasonably helpful. You may need to glance over a couple of different blogs before you have a more comprehensive understanding, but that is totally fine. You can also talk to me after class or during office hours and I can either explain or help you find relevant resources.

3.1.2 Problem Setting (Extra Credit)

Students may earn up to 6 extra points by proposing and setting a competitive programming problem for future uses.

- Students may work in teams of at most 3. Everyone in the team is expected to have a similar workload. Notably, **everyone is expected to code a solution** to ensure accuracy.
- Students should use Polygon (`https://polygon.codeforces.com`) to develop the problem. The problem is expected to be complete, with a statement, a checker, a validator, tests, solutions (2+ AC and 1+ TLE, if applicable), and a tutorial. Students should create a problem and add the instructor (zhtluo) to review the problem.
- Students should submit the draft of a statement by **Topic 3**, the draft of a complete problem by **Topic 6**, and a complete problem by **Topic 9**. The instructor will review the drafts and provide feedback. For this reason, late submissions may not be accepted.
- The problem might be used in the course or school contests. For this reason we ask that you **do not discuss the problem with other people**.
- The problem will be graded by the rubric below.

Grade	2 pts	1 pts	0 pts
Difficulty	The problem is of appropriate or higher difficulty for CP3.	The problem is of appropriate difficulty for CP1 or CP2.	The problem is trivial.
Completeness	The problem is complete.	The problem lacks enough solutions to ensure accuracy or lacks a tutorial.	The problem is not complete.
Clarity	The statement and tutorial is clear and easy to follow.	The statement or tutorial needs clarification.	The statement or tutorial cannot be understood.

3.2 Grade

The following scale is used to assign a grade over 10 assessment sessions, **after dropping the 2 lowest performances from a total of 12 sessions**:

Grade	A+	A	A-	B+	B	B-	C+	C	C- (P)	D+ (NP)	D	D-	F
Point	45	40	38	35	30	28	25	20	18	15	10	8	≤ 7

Late Policy Every student is given three free late days in total for upsolves to account for unexpected events. For extraordinary circumstances, please contact the instructor. In addition, 2 lowest performance sessions are dropped when calculating the grade.

Attendance Students are expected to be present for every meeting of the classes in which they are enrolled. Only the instructor can excuse a student from a course requirement or responsibility. When conflicts or absences can be anticipated, such as for many university-sponsored activities and religious observations, the student should inform the instructor of the situation as far in advance as possible. For unanticipated or emergency absences when advance notification to an instructor is not possible, the student should contact the instructor as soon as possible by email. When the student is unable to make direct contact with the instructor and is unable to leave word with the instructor's department because of circumstances beyond the student's control, and in cases of bereavement, the student or the student's representative should contact the Office of the Dean of Students via email or phone at 765-494-1747.

Since for contests, an unannounced absence creates an unfair burden on the rest of the team and unnecessary stress on the logistics of the course, students absent from contest sessions without being excused by the instructor will be given a **0** for that contest with **no** opportunities for upsolve. For this course, **arriving more than 10 minutes late** or **leaving more than 10 minutes early** without being excused counts as missing the class. In the event of an anticipated absence, inform the instructor of the situation as far in advance as possible.

How to Succeed in this Course Below are a few tips that might help you.

- **Prepare for the stress.** Solving problems in a strictly-timed environment is a stressful event for everyone. Unfortunately, this is a common occurrence in competitive programming and interview exams. In general, having more contest experience helps, so this course employs multiple low-stake contest sessions with free drops available to minimize stress. You may also find participating in online contests on Codeforces and AtCoder helps to build a rich competition experience, as they provide some incentive (rating) but are also generally low-stakes.
- **Be an active thinker in the contest.** As you approach CP3 and beyond, you will start to find that implementation starts to become easier but coming up with ideas for the problems remains difficult. To practice solving the problems, you need to be efficient not only in implementing a solution but also in coming up with ideas, and the best way to practice is to think about the contest problems in class actively instead of relying on your teammates.
- **Study and survey a wide range of sources for one topic.** For most topics in this class (and in competitive programming in general), there are many different resources online. The ability to search and study these resources is essential to your success in competitive programming and problem-solving, even outside of this class. In general, do not be afraid to glance over 10 articles online and find the one that helps you understand the topic more thoroughly. You will need this skill once you graduate from the course and study yourself.

3.3 Logistics

The following resources and platforms are used in this course.

3.3.1 Helpful Resources

None of the following resources are required for the course. Nevertheless, they may help you understand competitive programming and topics in the class.

- *Competitive Programming 4* by Halim, Halim, and Effendy
- *Algorithms for Competitive Programming*: `https://cp-algorithms.com/index.html`
- *Codeforces Catalog*: `https://codeforces.com/catalog`

3.3.2 Brightspace

Announcements and grades are published on Brightspace. Please check regularly for announcements that you may have missed.

3.3.3 Codeforces

Assessment sessions will be done on Codeforces (`https://codeforces.com/`).

- You need to create an account on the platform if you do not have one already and provide us with the account name. You also need to join the Codeforces group for this class. Sign-up details will be announced in class.
- For team-based contests, you need to create a team (`https://codeforces.com/teams`) with your teammates added to the team before you register for the contest.

3.3.4 Google Sheets

We use Google Sheets to help you track your progress throughout the course. You are required to fill in your progress and your team's progress on Google Sheets. Details will be announced in class.

3.4 Tentative Course Schedule

Topic (Tuesday)	Contest (Thursday)
Topic 0: Introduction, Implementation	Individual Contest
Topic 1: Geometry: Review, Half-plane Intersection, Adaptive Simpson	Team Contest
Topic 2: Combinatorics: Review, Linear Recurrence	Individual Contest
Topic 3: Range Query: Review, Segment Tree Hard	Team Contest
Topic 4: Monotonicity: DP Review, Optimization	Individual Contest
Topic 5: Tree: Review, LCA, HLD	Team Contest
Topic 6: Game Theory: SG Function, Search	Individual Contest
Topic 7: Network Flow: Min Cut, Min Cost	Team Contest
Topic 8: Fast Fourier Transform	Individual Contest
Topic 9: String: KMP, AC Automata	Team Contest
Topic 10: Offline: CDQ, Mo's Algorithm	Individual Contest
Topic 11: Final Contest	Team Contest

4 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 be checked for plagiarism. Any student found guilty of plagiarism and/or other forms of academic dishonesty may 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 this document: `https://spaf.cerias.purdue.edu/integrity.html`. **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 that you do not understand (e.g., would not be able to explain to the TA/professor)

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 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.

5 Course Evaluation

During the last two weeks of the course, you will be provided with an opportunity to evaluate 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 two weeks to complete this evaluation. Your participation is an integral part of this course, and your feedback is vital to improving education at Purdue University. I strongly urge you to participate in the evaluation system.

6 Disclaimer

Additional information regarding university policies is available in Brightspace.

This syllabus is subject to change. Changes will be announced in Brightspace and/or by e-mail.