DS 320: Algorithms for Data Science — Spring 2024

Instructor: Prof. Kira Goldner
Email: goldner@bu.edu
Office Hours: Tuesday 3:30-4:30PM and by appointment
Office Location: CCDS 1339, 665 Commonwealth Ave

Lectures: Tuesday and Thursday 2:00-3:15PM, EPC 209
Discussion Section A2: Wednesday 3:35-4:25PM, FLR 121
Discussion Section A3: Wednesday 4:40-5:30PM, FLR 121

Teaching Fellows: Nathaniel Quisel, Ketan Suhaas Saichandran
Email: njquisel@bu.edu, ketanss@bu.edu
Office Hours: TBD
OH Location: TBD

Course Description: This course covers the fundamental principles underlying the design and analysis of algorithms. We will walk through classical design methods, such as greedy algorithms, design and conquer, and dynamic programming, focusing on applications in data science. We will also study algorithmic methods more specific to data science and machine learning. The course places a particular emphasis on algorithmic efficiency, crucial with large and/or streaming data sets, for which multiple scans of data are infeasible, including the use of approximation and randomized algorithms.

Prerequisites: Required prerequisites are DS-110 and DS-121 or equivalent. Equivalents include for DS-110: CS-111, and for DS-121: CS-131 or MA-293. This is a **theoretical problem-solving** and **proof-writing** course. Additional useful background may include Discrete Math, Combinatorics, Linear Algebra, Data Structures and related algorithms, and Probability (none required, but the more the better).

Course website: `https://www.kiragoldner.com/teaching/DS320/`. There will also be a Piazza website for the course: `http://piazza.com/bu/spring2024/ds320/home`.

Learning Objectives: The objectives of this course are to:

- get comfortable understanding and writing formal definitions and statements.
- practice creative problem solving and thinking algorithmically.
- write clear and convincing arguments.
- domain-specific skills: identify algorithmic problems within applications; determine when to apply which technique; analyze runtime.

BU Hub: This course satisfies **Quantitative Reasoning II** and **Toolkit Critical Thinking**.

Quantitative Reasoning II: Throughout this course, we will build a toolkit of quantitative tool for solving algorithmic problems and proving correctness. Students will face complex problems from a breadth of applications and determine when each tool is appropriate and how to use it to design and analyze algorithms. They will learn to communicate clear and logically correct arguments in proofs. This will build on prior math and proof skills from discrete math and combinatorics (DS-120, DS-121, and DS-122 or equivalent).

Toolkit Critical Thinking: Students will learn algorithmic reasoning—intuition for how to consider any possible input instance and the requisite steps needed to transform the instance into the desired output. This form of thinking is not only essential in computing and data sciences, but transformative in understanding a world shape by algorithmic technology. Students will also hone their proof skills, learning rigorous mathematical reasoning to express why things are true, as well as the skills to evaluate their own and others' arguments.

Feedback on Learning Outcomes: Students will complete weekly problem sets that require them to solve algorithmic problems and write proofs analyzing their solutions. They will be encourage to work collaboratively in small teams to learn from one another. They will receive prompt grading with feedback on their solutions, and thus on their quantitative reasoning and critical thinking. During lectures, students will learn and practice both algorithmic and analysis techniques. In-class exercises will include opportunities to observe ideas from other students as well as to hear feedback from other students and from the instructor.

Books and Other Course Materials: There are no required books. There will be suggested readings from various textbooks and lecture notes suggested along with the course. If you *prefer* to have a textbook to follow along with, the book we will follow most closely is "Algorithm Design" by Kleinberg and Tardos, ISBN-10: 9780321295354.

Assignments and Grading:

- Weekly problem sets: 45% in total.
- Two midterms: 15% each.
- Final exam: 20%.
- Class participation—in class and via piazza: 5%.

Homework (45%):

- Expect to spend at least 10 hours per week on homework.
- **Late policy:** You may use up to 4 late days throughout the semester, but not more than 2 days on a given assignment. For each instance, you may only use an integer number of late days. Outside of this policy, no late submissions will be accepted.
- Your lowest homework score will be **down-weighted to 30%** at the end of the semester.

- Your written assignments must be prepared with LaTeX, not handwritten.
- You must hand in your homework via Gradescope (`https://www.gradescope.com/courses/346235`), which will be due at 11:59pm on the day assigned.
- **Regrade requests:** Regrade requests must be submitted within 7 days of receiving the graded assignment and only via Gradescope. You must also submit an explanation detailing which problems were graded incorrectly and an argument that the submitted solution is indeed correct. Regrades may only be requested if it is believed that a correct answer was marked as incorrect, not because insufficient partial credit was given to an incorrect or partially correct solution. If you request a regrade, you accept that the entire assignment/exam will be regraded, not just the problem(s) believed to be graded incorrectly.

Homework Collaboration Policy:

- For many people, algorithmic problem-solving is a collaborative endeavor. As such, you may work with up to **three** other classmates on the weekly homeworks for the course. However, the assignments you hand in must be written up by yourself and represent your own thoughts and work. In particular, you may discuss ideas with your classmates in person, but as a rough rule, nobody should leave the room with anything written down. If you really understand the discussion, you should be able to reconstruct it on your own. **As a hard rule, you must write up your arguments and problem sets individually. You may not use the internet or other references other than the course materials, unless told otherwise.**
- You must write your collaborators' names on the top of your assignment. Crediting one's peers is an important habit. If you do not work with collaborators, list "Collaborators: None." Separate rules apply to your exams, see below.
- Finally, make sure you adhere to BU's academic conduct policy, which I take very seriously: `https://www.bu.edu/academics/policies/academic-conduct-code/`.

Two Midterms (15% each): There will be two in-class midterms. The problem format will be similar to homeworks, except cumulative on material and designed to be solved in 1h15 time period. The midterms will take place roughly from February 22 and March 28 (exact dates subject to change).

Final Exam (20%): The final exam will be a closed-book in-class exam during finals period.

Participation (5%): In-class participation will be graded via participation cards every two weeks. Each student will receive a participation card with their name on it, and when they participate in good faith during class, the instructor will collect the card (counting their participation for this period) and return it at the end of the two-week period. Class attendance will not otherwise be taken. Piazza statistics will be used to supplement in-class participation.

Reasonable Accommodations: If you are a student with a disability or believe you might have a disability that requires accommodation, please contact the Office for Disability Services (ODS) at (617) 353-3658 or access@bu.edu to coordinate any reasonable accommodation requests. ODS is located at 25 Buick Street on the 3rd floor. If you have an accommodation, please report it to me in the first two weeks of the semester.

List of Topics:

- Runtime and Asymptotics
- Induction and Sorting
- Abstract Data Types
- Depth-First Search and Topological Sort
- Breadth-First Search and Testing Bipartiteness
- Greedy Algorithms:
  - Dijkstra
  - Caching
  - Scheduling
  - Huffman Codes
- Divide & Conquer:
  - Mergesort
  - Closest Pair of Points
  - Integer and Matrix Multiplication
- Dynamic Programming:
  - Weighted Interval Scheduling
  - Segmented Least Squares
  - Knapsack
  - Bellman-Ford
- NP Completeness
- Linear Programming & Duality
- Zero Sum Games and the Minimax Theorem
- Multiplicative Weight Update
- Stable Matching
- Approximation Algorithms: Randomized and Online