BENG/CSE/BIMM 181 COURSE (WINTER 2024)
Molecular Sequence Analysis
https://canvas.ucsd.edu/courses/51979

This is an unusual flipped class with complicated logistics - the syllabus alone is 19 pages long! But do not panic - we are providing a Table of Contents to help you navigate this syllabus. It is important that you read the **entire** syllabus to familiarize yourself with the course logistics.

INSTRUCTOR AND TAs

Instructor: Pavel Pevzner

- email: ppevzner@ucsd.edu
- web site: https://bioalgorithms.ucsd.edu/
- office: EBU3b 4236

Teaching Assistants/Tutors:

- Marcus Fedarko (email: mfedarko@ucsd.edu)
- Vikram Sirupurapu (email: vsirupur@eng.ucsd.edu)

Course Details:

	Days	Time (PT)	Location	Zoom link
Class	Monday and Wednesday	5:00 – 6:20 PM	HSS 1330 or online	https://ucsd.zoom.us/j/98473410249
Discussion Session	Friday	12:00 – 12:50 PM	CENTR 216 or online	https://ucsd.zoom.us/j/98826741593

By default, class and discussion meetings are **in-person (with no zoom option)** unless specified otherwise. Please see the detailed weekly schedule at the end of this syllabus.

Office hours:

	Days	Time (PT)	Location	Zoom link
Pavel Pevzner	Tuesday	1:00 – 3:00 PM	EBU3b 4236 or online	https://ucsd.zoom.us/j/92148499219
Vikram Sirupurapu	Monday	2:00 – 4:00 PM	EBU3b B215 or online	https://ucsd.zoom.us/j/93557514941
Marcus Fedarko	Thursday	12:00 – 2:00 PM	EBU3b B275 or online	https://ucsd.zoom.us/j/91786756212

By default, office hours are **in-person (with no zoom option)** unless specified otherwise. TAs can also hold office hours by appointment under special circumstances.

THIS IS A FLIPPED CLASS!

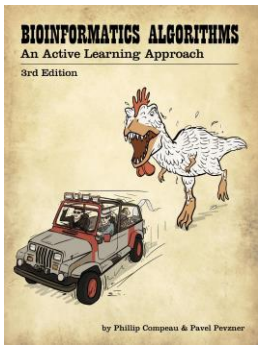
Course Philosophy. The "Two Sigma Problem" is a key paradigm of educational psychology introduced by Benjamin Bloom in 1984. It states that students achieve a two standard deviation improvement in learning performance when they are tutored individually compared to a traditional lecture in a large classroom. The reasoning behind the "Two Sigma Problem" is that students learn different material in different ways and at different paces, but the traditional lecture is completely blind to this conundrum.

How, then, should a course be run? The instructor of any large class unfortunately does not have time to tutor every student individually and thus cannot address the "Two Sigma Problem." CSE 181 differs because it is powered by an interactive version of the "*Bioinformatics Algorithms: An Active Learning Approach"* project co-developed by Phillip Compeau (Carnegie Mellon University) and Pavel Pevzner (UCSD). This class thus operates on a "flipped" model, in which students read and complete assessments in the interactive text (as well as watch lectures) *before* the class starts. This allows for class time to be used not for transmissive lectures but rather for guided discussions with students covering both the interactive text and recent advancements in bioinformatics.

What is a flipped class? The UC Online Innovative Learning Technology Initiative (ILTI) encourages professors to transform their classes into online offerings available across various UC campuses. Dr. Pevzner has been funded by ILTI and NIH to develop new online approaches to bioinformatics education at UCSD. This class was the first fully flipped class at UCSD - since 2014 all learning materials in this class are available online (rather than presented in the classroom) so that students can start learning *before* the class starts. For reference, watch the TedX video "The Era of Online Learning" where CSE professor Niema Moshiri (who was a student in this class in 2015) discusses the advantages of flipped courses over the traditional classroom format.

Prerequisites. The course assumes some prior background in biology, some algorithmic culture (the CSE 101 course is a strictly enforced prerequisite), and some programming skills.

Textbook. This class closely follows the textbook *Bioinformatics Algorithms: an Active Learning Approach* by Phillip Compeau and Pavel Pevzner that has been adopted by 200+ instructors from 45+ countries and that represents the MOOCBook of the *Bioinformatics Specialization* on Coursera with hundreds of thousands of enrolled students. The 3rd edition of this textbook (with a chicken-dinosaur on the cover) was published by Active Learning Publishers in 2018.



You DO NOT NEED to buy the hard copy of this textbook (unless you want to). After the pandemic started, Active Learning Publishers partnered with Cogniterra (previously known as Stepik) to make the book available as an interactive online MOOCBook. The MOOCBook includes various additional materials (such as videos, programming challenges, automated homework testing environment, etc.) and is available to UCSD students at the discounted price of $69.95. Note that the online textbook on Cogniterra differs from the hard copy version as it represents an intermediate 3^+ version between the 3rd and 4th edition (we are still working on the 4th edition).

Please go to "Bioinformatics Algorithms CSE 181 @ UCSD Winter 2024" at Cogniterra to enroll in the course.

Important: Please use this exact link to enroll. The interactive book is used by many instructors and thus has many clones. In the past, some UCSD students mistakenly enrolled in a different (non-UCSD) clone of the book and later realized that they are getting zero credit at UCSD :-)

Important: Your name on Cogniterra should EXACTLY match your name on Canvas – please, do not use alternate names, e.g., "Mike" on Cogniterra instead of "Michael" on Canvas.

Piazza.
1. You should be automatically added to Piazza (BENG / BIMM / CSE 181). If you are not added, please contact the TAs via email.
2. Piazza is only used for logistics, announcements, and general course-related doubts. Piazza shall not be used to discuss weekly homework problems (which have to be completed individually).

Automated homework testing on the Cogniterra platform. Computer science legend Donald Knuth once said "*I don't understand things unless I try to program them.*" We share his "*Learning through programming*" approach - all HWs in the class represent coding challenges that are described in the textbook and can be submitted at the Cogniterra platform. You can use any programming language to solve the HWs.

This class provides an automated homework testing environment aimed at learning bioinformatics through programming. Like in real life, there will be no partial credit for programming assignments - you either solve the problem by the deadline (full credit) or not

(zero credit).

The interactive MOOCBook has a new *feedback autograder* that helps with debugging your code and significantly improves on the previously used *language-agnostic autograder*. The feedback autograder is now the default autograder in this class. Since it currently supports only four languages (Python, C++, Java, and Go), students who are not familiar with these languages may want to learn one of them to take advantage of the debugging capabilities of the feedback autograder. **If you are not familiar with one of these four languages, please email the TAs by January 9, 2023.**

Since we are still developing the feedback autograder, some HW problems that you will encounter in the last three weeks of this class have only a subset of the four languages (Python, C++, Java, and Go) available for their solution. We are working on improving the feedback autograder and plan to implement it for all four languages and all HW problems in this class in the next few weeks.

Feedback autograder. The feedback autograder helps you to debug your code by providing useful feedback:

- In contrast to the language-agnostic autograder (that simply tells you whether you solved the problem or not), the feedback autograder provides you with some examples of the input data that your code does not handle correctly (and the correct answer for these inputs).
- In contrast to the language-agnostic autograder, you use the feedback autograder by providing your code directly to Cogniterra; Cogniterra will then test your code on a compendium of input datasets. You will pass the feedback autograder if your code works correctly on all these datasets.
- In contrast to the language-agnostic autograder, the feedback autograder includes a "function stub" that represents a helpful starting point for writing your code.
- The feedback autograder currently supports Python 3.10, C++20, Java 17, and Go for all HW problems issued in the first seven weeks of this class. We are now working on extending it to all problems.

We encourage you to provide your comments on the feedback autograder and suggestions (the link to a survey where you can provide comments is given on Cogniterra). We sincerely appreciate your comments as we work to improve this course for future cohorts of students.

Online resources. The Cogniterra platform provides all online resources for this course. You can also find some additional resources online at the following locations:

- Private YouTube channel with lecture videos: http://www.youtube.com/user/bioinfalgorithms/
- FAQs: http://bioinformaticsalgorithms.com/faqs.htm
- Problem Register for the textbook (see section "HW Problem Register" below): Google Drive link

GRADING

Computing your total score. The total score will be composed of the following components:

- **HWs** (60% of the score).
    - **Scoring HWs.** Every HW problem is 1 point. If you solved *N* out of *M* problems in a specific HW, your score for this HW is computed as (*N/M*)*100%. Your total HW score is the average of the scores of all individual HWs (after dropping your lowest HW score; see section "How do missed classes/HWs affect my score?").
    - **Communication skills** in bioinformatics. If you haven't solved all HW problems in a specific HW, you can add at most 1 point to this HW by asking a well-formulated question (see section "Addressing learning breakdowns").
- **Two midterm exams** (30% of the score).
- **Assessment based on "Recent Advances in Bioinformatics" lectures** (10% of the score).
- **Final exam** (Pass or Fail). IMPORTANT: Failing the Final exam implies failing the class independently of your other scores.

How do missed classes/HWs affect my score? We understand that you may miss some sessions of the class due to unforeseen circumstances, illness, graduate school interviews, etc. To help you deal with these circumstances, your overall HW score will be computed from your top *n*-1 individual scores, where *n* is the total number of HWs in this class. Thus, you can miss one HW in this class, no questions asked, to account for medical or family-related absences, job and graduate school interviews, etc. (no need to inform the instructors if you have to miss a single class). However, you will have to provide an official justification for each missed session if you want to obtain credit for more HWs than specified above.

HOMEWORKS

HWs should be the result of individual work. HWs will be issued each week and will include from 5 to 7 coding challenges.

- You are NOT allowed to search for solutions of HWs on any online resources.
- HW submissions will be subjected to automatic plagiarism checking.
- You can discuss HWs with your classmates *after* the HW deadline when you are preparing the survey question.
- No hints on how to solve HWs will be provided before the HW deadline. The textbook and FAQs are designed in such a way that they contain all information (including explicit or implicit hints) needed for you to succeed in HWs.

How do I submit the solutions of programming challenges? Please submit all your solutions to the HW programming challenges at the Cogniterra platform using the feedback autograder. Cogniterra automatically stores your code for further plagiarism checking.

HW Problem Register. To help with solving HW problems, you may consult the online book *Problem Register for "Bioinformatics Algorithms: An Active Learning Approach"* written by Parker Côté and Ryan Eveloff. Parker (now a bioinformatician at Dana Farber Cancer Institute)

and Ryan (now a bioinformatician at Exact Sciences) took this class in Winter 2021 and have done a great job to unify the descriptions and test cases for all HW problems and help future students during the time they work on HWs. Nikitha Kalahasti (bioinformatics student at UCSD) and Mustafa Guler, Jackie Vo, and Jiayi Li (computational biology students at Carnegie Mellon) further helped to improve the Register in 2023.

Do not submit the code before you understand how the algorithm works! It is important that you understand the ideas behind each algorithm that you implement in this course. We do not want you to blindly code a "line-by-line" implementation of pseudocode to pass the autograder without understanding how the algorithm behind this pseudocode works. That is why we ask you **not to submit** the code unless you can write down an explanation of how your solution works in your own words without opening the textbook. In other words, do not submit your solution if you cannot outline the key idea of your algorithm (and write its pseudocode) without copying pseudocode from the textbook. See section "You may not get credit for a HW problem you have solved… an unusual two-stage HW grading approach" that explains how we enforce the approach "Do not submit the code before you understand how the algorithm works!"

Trying to implement a HW problem is often a good way to address your learning breakdowns. That is why sometimes it makes sense to start coding even before you fully understand the algorithm in an attempt to address your learning breakdown. However, if your breakdown has not been cleared even after coding, it is better not to submit your program and instead prepare a well-formulated question for the survey that may add a point to your grade.

We will waive the Final Exam for students who follow this approach. To test whether you follow this approach, most assessments in this class (e.g., the midterms and the final) will be designed to test how well you understand your OWN previously submitted HW programs (see below). The decision on whether to waive the Final Exam will also take into account how well students answer questions raised by the professor during class meetings.

You may get credit for a HW problem even if you did not solve it! If you fail to solve one of the HW problems, you will have an opportunity to submit a question that describes a learning breakdown that you experienced and explains why this breakdown makes it difficult to solve a specific problem. If this description presents a well-formulated summary of your learning breakdown (see below), you will get a point even if you failed to solve a specific HW problem!

At the first glance, it may appear that getting a point for a question about a HW problem is easier than solving this problem. This class will teach you that this unusual credit is well-deserved since asking a good well-formulated question about a problem is sometimes even more difficult than solving it! You can submit a description of a learning breakdown for at most one unsolved problem per each week's HW.

You may not get credit for a HW problem you have solved… an unusual two-stage HW grading approach. To enforce "*Do not submit the code before you understand how the algorithm works!*" (and to disincentivize cheating) we will use the following two-stage approach to HW grading. The automatically generated grade for a HW becomes final only if

there is no anti-correlation between HW and midterm scores/quiz results (see below) during the entire quarter. High anti-correlation triggers a case-by-case audit of all submitted HWs and quizzes. If the instructor concludes that you did not follow the "*Do not submit…"* approach for a specific HW, the results of the entire weekly HW will be nullified. This analysis of your HWs/quizzes may require a specially scheduled one-on-one meeting with the instructor/TAs where you will be asked questions related to various HWs and quizzes.

Do not use external packages to solve HWs! To be consistent with other CSE classes with automated HW checking and strict anti-plagiarism enforcement (like CSE100R), we insist on using the policy: "You can use *anything* that is built into the language and its standard libraries, but you *cannot* use *any* external libraries". Using the native implementations of basic data structures (e.g. hashmap/dictionary, array/list, queue, stack, heap, etc.) is fine, but using things like full-fledged graph libraries is not allowed. That being said, you are free to implement your own data structures, e.g., you can implement your own Node/Edge/Graph classes as you see fit. Make sure that, outside of things you implement yourself, you only use native basic data structures to solve the problems. Here are some examples:

- **C++:** You can use vector, string, etc. even though you need to use "#include <vector>", "#include <string>", etc. because they are built into the C++ Standard Template Library, but we do not allow use of external libraries like Boost.
- **Python:** You can use Counter, defaultdict, etc. even though you need to use "import collections" because the Python "collections" module is part of the Python Standard Library (https://docs.python.org/3/library/collections.html), but we do not allow use of external libraries like NumPy, SciPy, pandas, NetworkX, iGraph, etc.

Some students may want to use external libraries like NumPy, JGraphT, etc. for small utilities (e.g. the array data structures available in NumPy). Even though we would personally be okay with *just* the NumPy array, we are concerned that as soon as we allow one function in NumPy, students will ask about other functions in NumPy, other Python packages, etc. Thus, to be consistent with other flipped classes at CSE, we insist on the above policy. If you have any questions about whether or not something is allowed in the HWs, please feel free to ask on Piazza or contact the TAs.

COMMUNICATION SESSIONS

Communication skills in bioinformatics. Communication skills are important in every discipline but they are even more crucial in interdisciplinary fields like bioinformatics. That is why you will be given credit even for unsolved HW problems if you master the art of asking well-formulated questions that describe your learning breakdowns that prevent you from solving these problems.

An important goal of this class is to teach students how to diagnose their INDIVIDUAL *learning breakdowns* and to resolve them by asking well-formulated questions. A learning breakdown refers to a concept that students have struggled with even after spending significant time trying to address this breakdown (e.g., thinking deeply about this concept, checking FAQs and other learning materials, etc.). In this class, we will focus on learning breakdowns that

prevent students from solving some HW problems.

Addressing learning breakdowns. Each student who experienced a learning breakdown related to an *unsolved* HW problem may file a *single* well-formulated question related to their breakdowns by the Survey deadlines specified in the schedule below. We understand that you may have multiple breakdowns; there will be time to address all your breakdowns during our class meetings and office hours. However, the partial HW credit (1 point) will be given only for a description of a single breakdown that you have submitted. Do not submit the survey questions if you solved all HW problems.

It is important that you invest time in formulating the question so that the instructor can address it based *only on your formulation*, without additional clarification. The self-contained description of your breakdown should explain how it prevented you from solving a *specific* HW problem. You will have to explain how you started solving this problem and which specific concept(s) prevented you from solving it.

There will be nine Survey deadlines in this class. Please file your questions at the class Canvas website, under Assignments>Surveys. A "well-formulated question" means that your peers (and the instructor!) are able to understand the specific difficulty you are having and to help you to overcome the learning breakdown. For example, "I don't understand how this algorithm works, can you please explain it again?" is not a well-formulated question and will not be given credit because it does not describe your *specific* learning breakdown, does not allow an instructor to diagnose what caused it, and does not describe why this specific breakdown prevented you from solving a specific HW problem.

Some students may prefer not to file any learning breakdowns (e.g., students who did not experience any learning breakdowns on a given week's HW) and this is perfectly fine. Note that all students will be answering questions of the instructor (and questions from other students) during class meetings and answers to these questions are taken into account when deciding on whether to waive the Final.

Learning breakdowns versus curiosity questions. Learning breakdowns reflect challenges that make it difficult for a student to solve a specific HW problem - you have to specify which HW problem and explain how your learning breakdown prevents you from solving this problem. "Curiosity questions" like:

- Are there any alternative ways of estimating the location of the replication origin?
- How do we select the size of the window in the Clump Finding Problem?
- How do we select the constant K for the partial suffix array?
- Why does this example assume that K=5 and not 10?

are not classified as learning breakdowns because they do not affect the understanding of the follow-up materials. If you only face curiosity questions while going through the chapter, this means you have not really had a breakdown. Also, you cannot file questions that simply repeat "Exercise Breaks," "STOP and Think" boxes, FAQs, or indirectly ask to provide hints for homework problems.

All books have small errors that should not be filed as learning breakdowns unless an error prevents you from solving a HW problem.

Filing a Survey report. As discussed above, you can optionally file a Survey report by one of the Survey deadlines specified below. This report specifies a learning breakdown that a student is struggling with. The following information is required for each Survey report:

- Information about the HW problem you failed to solve (specify the name of the problem) because of the specific learning breakdown you intend to describe.
- A detailed description of the learning breakdown (pointing to a specific page/paragraph).
- A well-formulated question that will explain to the instructor how to help you to address this specific learning breakdown.

Additionally, we ask that your Survey report follows these guidelines:

- Your description should explain how this specific breakdown prevented you from solving the specific HW problem. The description of the breakdown should describe the algorithmic rather than programming challenges – e.g., questions like "How do I implement the Burrows-Wheeler Transform in Java" will not be given credit.
- Your breakdown report should be self-contained, i.e., the instructor should be able to understand the cause of the breakdown without additional verbal clarifications from you. There will be no credit if you file a breakdown/question that has been already addressed in the available resources.
- It is a student's responsibility to check the FAQs and Charging Stations (not to mention the text of the entire chapter) to ensure that this question has not been addressed yet (otherwise, there will be no credit for the communication session).
- Your questions should refer to the textbook rather than the videos (or power points) since videos represent incomplete and error-prone versions of the learning materials. It is important that the question relates to the specific learning breakdown (and a specific paragraph in the textbook) and specific HW problem rather than being an open-ended question. For example, we appreciate questions like "What is the future of nanopore-based sequencing technology?" or "Can I apply Hidden Markov Models to gene prediction?" and they will be answered in the follow-up communication session. However, no credit will be given for such general open-ended questions.

You will be given a credit (1 point towards the corresponding week's HW score) for good questions that comply with the above description. Yes, getting a point for a question about your learning breakdowns is not easy but we believe that such questions help you to develop your communication skills in bioinformatics!

Group activities. We will have group activities during some communication sessions and lectures in the "Recent Advances in Bioinformatics" series. During short group activity sessions (typically 5-10 minutes), the instructor will pose questions and students will be

working in small groups (typically 3-4 people at the adjacent seats) to answer these questions and present their solutions in class.

Reviewing online materials. Students can review the online materials (Cogniterra, FAQs, Problem Register, etc.) at their convenience (all materials are available on the first day of classes) but should be prepared to answer questions about all the materials by the Communication Session dates specified below. Even if a HW for a specific chapter does not cover all topics in this chapter, you will need to read the entire chapter to prepare for questions during the communication session. The Study periods below are merely suggestions to help you get organized for this class - you can work on whatever schedule you find convenient (for example, you could solve all HWs during the first week of classes if desired).

MIDTERMS, FINAL, NON-GRADED QUIZZES, AND "RECENT ADVANCES IN BIOINFORMATICS" LECTURES

Midterms and Final. The midterms and final exams will consist of newly designed programming challenges that typically represent modifications of problems that have been previously given as HWs in the class. Therefore, to solve a novel problem A* that originated from a HW problem A, you merely need to slightly modify the original code for A (that you have already submitted). If you successfully solved problem A on its corresponding HW, you **are required** to modify your submitted code for A (instead of implementing A* from scratch). Thus, you will need to have the code from all your previous HWs available when you start the midterm/final. If you did not solve problem A on its corresponding HW, you can solve problem A* from scratch during the midterm/final.

To provide a more comfortable coding environment, the midterms will be conducted in a different auditorium (TBA) rather than our HSS 1330 classroom.

Non-graded quizzes. Quizzes will either (1) be given in the format described above, (2) will ask you to describe one of your previous HW solutions in your own words, or (3) will ask you to solve simple problems that test your knowledge of some basic concepts. These unannounced quizzes will not be graded but instead used for deciding which students have high anti-correlation and thus are not eligible for waiving the final exam. Most quizzes will be given on Mondays (if time allows) but some quizzes may be issued on Wednesdays, depending on when we have time left after the communication sessions or lectures in the "Recent Advances in Bioinformatics" series. We will not have quizzes during some (busy) weeks.

Please bring a fully-charged laptop (some quizzes will require a bit of programming or answering questions online) and color pens (some quizzes, particularly for the "Genome Rearrangements" chapter, will require red, blue, and black pens or pencils) to each class meeting. We will also be solving online algorithmic puzzles that you can solve either on your laptop or on your smartphone.

Avoid anti-correlation between HW and midterms scores/quiz results! The quizzes and midterms should represent a simple task for students who followed our "*Do not submit the code before you understand how the algorithm works!*" policy. However, they will be difficult

tasks for students who submitted HWs without deep understanding of the algorithms behind them or (worse!) for students whose HWs do not represent an independent effort. All midterm and final solutions will be checked for plagiarism.

If it turns out that your high HW score "anti-correlates" with your low midterm score and the results of the quizzes, it likely means that you have not taken the subsection "*Do not submit the code before you understand how the algorithm works!*" seriously - or, worse, that the HWs you submitted do not represent independent work. The midterm and final will be designed in such a way that you *will have to* modify the code in one of your previously submitted HWs to solve each of the problems. Therefore, the best way to prepare for the midterm/final and to pass this class is to make sure that you submit your *own independent HWs* each week and understand how each algorithm implemented in this HW works.

Lectures on recent advances in bioinformatics. Since 200+ instructors (who adopted our textbook) cover similar materials, there is an opportunity to share various educational materials that extend the topics covered in the textbook and enhance the educational experience of students across various universities. Each lecture in this series will be given either by the instructor (in class) or by one of our invited speakers (either in person or on zoom).

You will be given a graded quiz that tests your understanding of the topics discussed in these lectures. We encourage you to ask questions during these lectures to help improve your understanding of the underlying concepts.

In Winter 2024, we will cover additional materials contributed by the following bioinformaticians:

- Phillip Compeau, Professor at Department of Computational Biology, **Carnegie Mellon Universit**y
- Alexey Gurevich, Professor at Computer Science Department at **Saarland University**, Saarbrucken, Germany
- Ben Raphael, Professor at Computer Science Department, **Princeton University**
- Stefano Bonissone, Chief Scientific Officer at **Abterra Biosciences**

ACADEMIC INTEGRITY

To detect instances of academic integrity violations in programming assignments we will use third-party plagiarism detection software.

All the work in the course should be your own. **Since plagiarism was detected in previous sessions of this class (with serious consequences for the students involved), we invest significant effort in checking your code and comparing it with a database of existing solutions over the last decade.** Using various web resources (that provide solutions to coding challenges) for solving HWs is considered a violation of the academic integrity policy.

Please do not post your solutions on the Internet and do not share your solutions with classmates since this may trigger a violation of the academic integrity policy, for example in

the case when your schoolmate uses your solution in a homework. Please note that, if you solved a HW before the start of the class (e.g., in the Fall 2023 quarter) and used web resources for solving it, this may also trigger a violation of the academic integrity policy. If this is the case, you have to redo the program from scratch since otherwise it may be marked as a violation by our plagiarism checking tool. Below please find answers to some Academic Integrity FAQs:

Can I reuse code from previous problems solved in this class (e.g. using HW1 code in HW2)?

- Yes

Can I consult material from previous classes like CSE 101?

- Yes

Does the formatting of input/output matter for code submission on Cogniterra?

- No. But the logic driving code has to be the same.

I finished these HW problems earlier, before the class started. Can I reuse code from there?

- Yes, as long as you have not copied from online resources to write the original code, shared the code with anyone else OR made the code public on GitHub, Rosalind, etc.

Can I collaborate on "Stop and Think" and "Exercise Breaks"?

- Yes.

Can I collaborate on optional problems?

- No. This is because many optional problems relate directly to non-optional HW problems.

Limitations of the autograder. The Cogniterra autograders are calibrated in such a way that they only award points for time-efficient and memory-efficient solutions. But since they are not bullet-proof, some students tried to dupe the autograders by submitting inefficient solutions that however passed the autograder. For example, even though some problems in the course specifically require memory-efficient solutions, it is difficult to design an autograder that passes all memory-efficient solutions and does not pass all memory-inefficient solutions.

Please follow the specified HW requirements and do not try to dupe the autograders. To enforce these requirements, HW solutions that violate these requirements will be nullified.

RESOURCES FOR STUDENTS

Diversity of thoughts, perspectives and experiences. We are committed to fostering a learning environment for this course that supports a diversity of thoughts, perspectives and experiences, and respects your identities (including race, ethnicity, heritage, gender, sex, class, sexuality, religion, etc.). Our goal is to create a diverse and inclusive learning environment where all students feel comfortable and can thrive.

Our instructional staff will make a concerted effort to be welcoming and inclusive to all students in this course. If there is a way we can make you feel more included please let us know, either in person, via email/discussion board, or even in a note under the door. We welcome your perspectives and input.

If you experience any sort of harassment or discrimination, please contact the instructor or the Office of Prevention of Harassment and Discrimination: https://ophd.ucsd.edu/.

Students with Disabilities. We aim to create an environment in which all students can succeed in this course. If you have a disability, please contact the Office for Students with Disabilities (OSD) to discuss appropriate accommodations. We will work to provide you with the accommodations you need, but you must first provide a current Authorization for Accommodation (AFA) letter issued by the OSD. You are required to present your AFA letters to the instructor and to the OSD Liaison in the department in advance so that accommodations may be arranged.

Basic Needs. If you are experiencing any basic needs insecurities (food, housing, financial resources) please visit http://thehub.ucsd.edu/ for information about available resources to help you.

SCHEDULE

Detailed schedule (subject to change) is given below. Most classes will be in-person. Some classes (TBD) may be given on Zoom.

Homework deadlines are at 11:59 pm on the specified dates. Please note that HWs do not include ALL PROBLEMS from a given chapter. Read information below for the list of included/excluded problems for each chapter - the HWs contain only 50 problems while the relevant chapters contain over 100 problems.

The "excluded problems" here are listed based on the 3rd edition of the book; however, there are a few differences between this edition and the Cogniterra course (which is based on an in-progress 4th edition of the book). Please always check the NAME of the excluded problem (rather than its number) to see which problems are excluded. In the event of discrepancies, please go with what is on Cogniterra; for a list of which exact problems are required in the course, please see this spreadsheet (just the problems labeled with "Kept" in their "Used in class?" column are required).

Survey deadlines are at 3 pm (PT) on the specified dates.

- Monday, January 8. Introductory lecture
- Wed, January 10. *Recent Advances in Bioinformatics* series

Martin Luther King Jr. Day. Monday, January 15 (no class)

Replication Origin (Chapter 1)

- HW deadline: Tue, Jan 16
- Survey deadline: Wed, Jan 17, 3 pm
- Communication Session, Wed, Jan 17
- Study: Mon, Jan 8 - Tue, Jan 16
- 6 HW problems (excluded problems are shown in gray):

(1A) Compute the Number of Times a Pattern Appears in a Text
(1B) Find the Most Frequent Words in a String
(1C) Find the Reverse Complement of a DNA String
(1D) Find All Occurrences of a Pattern in a String
(1E) Find Patterns Forming Clumps in a String
(1F) Find a Position in a Genome Minimizing the Skew
(1G) Compute the Hamming Distance Between Two Strings
(1H) Find All Approximate Occurrences of a Pattern in a String
(1X) Implement ApproximatePatternCount (does not appear in the hard copy of the book)
(1I) Find the Most Frequent Words with Mismatches in a String
(1J) Find Frequent Words with Mismatches and Reverse Complements
(1K) Generate the Frequency Array of a String (does not appear on Cogniterra)
(1L) Implement PATTERNTONUMBER
(1M) Implement NUMBERTOPATTERN
(1N) Generate the d-Neighborhood of a String

Regulatory Motifs (Chapter 2)

- Communication Session, Wed, Jan 24
- HW deadline: Thu, Jan 25
- Survey deadline: Fri, Jan 26, 3 pm
- Study: Tue, Jan 16 – Tue, Jan 23
- Mon. January 22. *Recent Advances in Bioinformatics* series
- 5 HW problems:

(2A) Implement MOTIFENUMERATION
(2B) Find a Median String
(2C) Find a Profile-most Probable *k*-mer in a String
(2D) Implement GREEDYMOTIFSEARCH
(2E) Implement GREEDYMOTIFSEARCH with Pseudocounts
(2F) Implement RANDOMIZEDMOTIFSEARCH
(2G) Implement GIBBSSAMPLER
(2H) Implement DISTANCEBETWEENPATTERNANDSTRINGS

Assembly (Chapter 3)

- Communication Session, Mon, Jan 29
- HW deadline: Tue, Jan 30
- Survey deadline: Wed, Jan 31, 3 pm
- Study: Tue, Jan 23 – Tue, Jan 30
- Wed, Jan 31. *Recent Advances in Bioinformatics* series
- 6 HW problems:

(3A) Generate the k-mer Composition of a String
(3B) Reconstruct a String from its Genome Path

(3C) Construct the Overlap Graph of a Collection of k-mers
(3D) Construct the de Bruijn Graph of a String
(3E) Construct the de Bruijn Graph of a Collection of k-mers
(3F) Find an Eulerian Cycle in a Graph
(3G) Find an Eulerian Path in a Graph
(3H) Reconstruct a String from its k-mer Composition
(3I) Find a k-Universal Circular String
(3J) Reconstruct a String from its Paired Composition
(3K) Generate the Contigs from a Collection of Reads
(3L) Construct a String Spelled by a Gapped Genome Path
(3M) Generate All Maximal Non-Branching Paths in a Graph

Alignment, Part 1 (Chapter 5, ending before (not including) "Penalizing Insertions and Deletions")

- Communication Session, Mon, Feb 5
- HW deadline: Tue, Feb 6
- Survey deadline: Wed, Feb 7, 3 pm
- Study: Tue, Jan 30 – Tue, Feb 6
- 7 HW problems. You may want to solve the problem 5N from the next HW (Finding topological ordering of a graph) before embarking on this HW since it may help you to solve this HW. You have covered problem 5N in your Algorithms course.

(5A) Find the Minimum Number of Coins Needed to Make Change
(5B) Find the Length of a Longest Path in a Manhattan-like Grid
(5C) Find a Longest Common Subsequence of Two Strings
(5D) Find the Longest Path in a DAG
(5E) Find a Highest-Scoring Alignment of Two Strings
(5F) Find a Highest-Scoring Local Alignment of Two Strings
(5G) Compute the Edit Distance Between Two Strings
(5H) Find a Highest-Scoring Fitting Alignment of Two Strings
(5I) Find a Highest-Scoring Overlap Alignment of Two Strings

***First Midterm Exam*: Wednesday, Feb 7 (covering all topics discussed by this date)**

Alignment, Part 2 (Chapter 5, starting from (including) "Penalizing Insertions and Deletions")

- Communication Session, Mon, Feb 12
- HW deadline: Tue, Feb 13
- Survey deadline: Wed, Feb 14, 3 pm
- Wed, Feb 14. *Recent Advances in Bioinformatics* series
- Study: Tue, Feb 6 – Tue, Feb 13
- 5 HW problems:

(5J) Align Two Strings Using Affine Gap Penalties
(5K) Find a Middle Edge in an Alignment Graph in Linear Space
(5L) Align Two Strings Using Linear Space

(5M) Construct a Highest-Scoring Multiple Alignment
(5N) Find a Topological Ordering of a DAG

President's Day. Monday, February 19 (no class)

Rearrangements (Chapter 6)

- HW deadline: Tue, Feb 20
- Survey deadline: Wed, Feb 21, 3 pm
- Communication Session, Wed, Feb 21
- Study: Tue, Feb 13– Tue, Feb 20
- 5 HW problems.

(6A) Implement GREEDYSORTING to Sort a Permutation by Reversals
(6B) Compute the Number of Breakpoints in a Permutation
(6C) Compute the 2-Break Distance Between a Pair of Genomes
(6D) Find a Shortest Transformation of a Genome into Another by 2-Breaks
(6E) Find All Shared k-mers of a Pair of Strings
(6F) Implement CHROMOSOMETOCYCLE
(6G) Implement CYCLETOCHROMOSOME
(6H) Implement COLOREDEDGES
(6I) Implement GRAPHTOGENOME
(6J) Implement 2-BREAKONGENOMEGRAPH
(6K) Implement 2-BREAKONGENOME

Detecting Mutations, Part 1 (Chapter 9 before (not including) "Inverting Burrows-Wheeler Transform")

- Communication Session, Mon, Feb 26
- HW deadline: Tue, Feb 27
- Survey deadline: Wed, Feb 28, 3 pm
- Study: Tue, Feb 21 – Tue, Feb 28
- 7 HW problems:

(9A) Construct a Trie from a Collection of Patterns
(9B) Implement TRIEMATCHING
(9C) Construct the Suffix Tree of a String
(9D) Find the Longest Repeat in a String
(9E) Find the Longest Substring Shared by Two Strings
(9F) Find the Shortest Non-Shared Substring of Two Strings
(9G) Construct the Suffix Array of a String
(9H) Implement PATTERNMATCHINGWITHSUFFIXARRAY (does not appear on Cogniterra)
(9I) Construct the Burrows-Wheeler Transform of a String

***Second Midterm Exam*: Wednesday, Feb 28 (covering all topics discussed by this date)**

Detecting Mutations, Part 2 (Chapter 9, starting from (including) “Inverting Burrows-Wheeler Transform")

- Communication Session, Mon, March 4
- HW deadline: Tue, March 5
- Survey deadline: Wed, March 6, 3 pm
- Study: Tue, Feb 27 – Tue, March 5
- Wed, March 6. *Recent Advances in Bioinformatics* series
- 4 HW problems:

(9J) Reconstruct a String from its Burrows-Wheeler Transform
(9K) Generate the Last-to-First Mapping of a String (not present on Cogniterra)
(9L) Implement BWMATCHING
(9M) Implement BETTERBWMATCHING
(9N) Find All Occurrences of a Collection of Patterns in a String
(9O) Find All Approximate Occurrences of a Collection of Patterns in a String
(9P) Implement TREECOLORING
(9Q) Construct the Partial Suffix Array of a String
(9R) Construct a Suffix Tree from a Suffix Array

Clustering (entire Chapter 8) and *Hidden Markov Models* (Chapter 10) before (not including) “Classifying proteins with profile HMMs.”

- Communication Session, Mon, March 11
- HW deadline: Tue, March 12
- Survey deadline: Wed, March 13, 3 pm
- Study: Tue, March 5 – Tue, March 12
- Wed, March 13. Short lecture and graded quiz in the *Recent Advances in Bioinformatics* series
- 5 HW problems:

(8A) Implement FARTHESTFIRSTTRAVERSAL
(8B) Compute the Squared Error Distortion
(8C) Implement the Lloyd Algorithm for k-Means Clustering
(8D) Implement the Soft k-Means Clustering Algorithm
(8E) Implement HIERARCHICALCLUSTERING

(10A) Compute the Probability of a Hidden Path
(10B) Compute the Probability of an Outcome Given a Hidden Path
(10C) Implement the Viterbi Algorithm

Final: Mon, March 18, 7 pm – 9:59 pm

SCHEDULE AT A GLANCE

- Monday, January 8. Introductory lecture
- Wed, January 10. *Recent Advances in Bioinformatics* series

- Mon, January 15 *Martin Luther King Jr. Day* (no class)
- Communication Session, Wed, Jan 17

- Mon. January 22. *Recent Advances in Bioinformatics* series
- Wed, Jan 24 Communication Session,

- Mon, Jan 29. Communication Session,
- Wed, Jan 31. *Recent Advances in Bioinformatics* series

- Communication Session, Mon, Feb 5
- **Wed, Feb 7 *First Midterm Exam***

- Mon, Feb 12 Communication Session
- Wed, Feb 14. *Recent Advances in Bioinformatics* series

- Monday, Feb 19 *Presidents' Day* (no class)
- Communication Session, Wed, Feb 21

- Communication Session, Mon, Feb 26
- **Wed, Feb 28 *Second Midterm Exam***

- Communication Session, Mon, March 4
- Wed, March 6. *Recent Advances in Bioinformatics* series

- Communication Session, Mon, March 11
- Wed, March 13. Short lecture and graded quiz in the *Recent Advances in Bioinformatics* series

- **Final:** Mon, March 18, 7 pm – 9:59 pm