BIONF II/CSE 282/BENG 202 COURSE (WINTER 2023)
Molecular Sequence Analysis

https://canvas.ucsd.edu/courses/42026/

This is an unusual flipped class with complicated logistics - this syllabus alone is 20 pages long! But do not panic - we are providing a Table of Contents to help you navigate it. It is important that you read the entire syllabus to familiarize yourself with the course logistics.

INSTRUCTOR AND TAs

Instructor: Pavel Pevzner (ppevzner@ucsd.edu)

- web site: https://bioalgorithms.ucsd.edu/
- office: EBU3b 4236
- phone: (858) 822-4365

Teaching Assistant: Marcus Fedarko (mfedarko@ucsd.edu)

Course Details. By default, all class meetings are in-person only (with no Zoom option) unless specified otherwise.

	Days	Time (PT)	Location	Zoom link
Class	Monday and Wednesday	6:30 – 7:50 PM	CSB 005 or online	https://ucsd.zoom.us/j/92885436720

Please see the detailed weekly schedule at the end of this syllabus.

Office Hours:

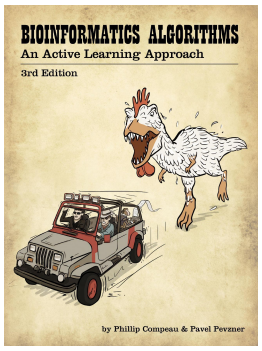
	Day	Time (PT)	Location	Zoom link
Pavel Pevzner	Tuesday	3:00 – 5:00 PM	EBU3b 4236 or online	https://ucsd.zoom.us/j/99511232171
Marcus Fedarko	Friday	2:30 – 4:30 PM	EBU3b B215 or online	https://ucsd.zoom.us/j/98690930138

The TA can also hold office hours by appointment under special circumstances.

THIS IS A FLIPPED CLASS!

Prerequisites. The course assumes some prior background in biology, some algorithmic culture (e.g., an equivalent of the CSE 101 course on algorithms at UCSD), 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 book (with a chicken-dinosaur on the cover) was published by Active Learning Publishers in 2018.



Although it was a required textbook for this class until the start of the pandemic, you DO NOT NEED to buy the hard copy of this textbook anymore. After the pandemic started, Active Learning Publishers partnered with Stepik to make the book (along with all additional materials that include videos, programming challenges, etc.) available as an online MOOCBook at the discounted price of $69.95. Note that the online textbook on Stepik slightly differs from the hard copy version as it represents an intermediate version between the 3rd and 4th edition that is now being completed. It includes the new *feedback autograder* (still in beta version) that may greatly help you to debug your code.

Please go to "Bioinformatics Algorithms @ UCSD Winter 2023 (CSE 282)" at Stepik to enroll in the course. **It is important that your name on Stepik EXACTLY matches your name on Canvas – please, do not use alternate names, e.g., "Mike" on Stepik instead of "Michael" on Canvas.**

Piazza.
1. You should be automatically added to Piazza ("BENG 202 / CSE 282 [WI23]"). 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).

What is a flipped class? The UC *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 the ILTI and by the 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.

Automated homework testing on the Stepik 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 Stepik 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). You can attempt a HW problem on Stepik by clicking on the green "Click to start solving" button.

- After clicking on this button, you will be given a novel dataset on which you are expected to run your code locally on your machine; afterward you can upload the answer (the output of your code) to Stepik.
- You will be notified if your output matches the correct output for this dataset (some problems may have multiple accepted solutions).
- Each HW problem has a five-minute time limit.
- You can re-attempt a HW problem arbitrarily many times.
- You can use any programming language to solve the HWs.

New feedback autograder. Starting this year, Stepik includes an optional *feedback autograder* (for most HW problems) accessible using the link "Visit (optional) autograded code challenge"). It does not impact your grade but helps you to debug your code by providing a useful feedback.

- **Solving a question using the feedback autograder does not award you any credit**: please solve problems using the standard autograder and use the feedback autograder when you experience difficulties in debugging your code.
- In difference from the default autograder (that simply tells you whether you solved the problem or not), the feedback autograder provides you with an example of the input data that your code does not handle correctly (and the correct answer for this input).
- In difference from the standard autograder, the feedback autograder allows you to provide your code directly to Stepik, which will then test your code on a (hidden) compendium of input datasets.
- You will pass the feedback autograder if your code works correctly on all these datasets. It also includes a “function stub” that represents a helpful starting point for writing your code.
- While the standard autograder is language-agnostic, the feedback autograder currently supports only Python 3.10 and C++20.

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

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

- A link to most lessons is available from the Bioinformatics online specialization web page at Coursera.

- Private YouTube channel with lecture videos
- FAQs
- Problem Register for the textbook (see section "*HW Problem Register*" below)

GRADING

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

- **HWs** (50% of the score). HWs will be issued each week and will include from 5 to 8 coding challenges. HWs should the result of individual work. 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.
    - **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*").
- **Midterm exam** (15% of the score).
- **Assessments based on invited lectures** (5% of the score).
- **Class Project** (30% of the score).
- **Final exam** (Pass or Fail). The final will be waived for most students based on assessing whether a student followed the policy outlined in the section *"Do not submit the code before you understand how the algorithm works!"* 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, 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, 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

How do I submit the solutions of programming challenges? You will have to submit all your solutions of the HW programming challenges at the Stepik content delivery platform. Stepik currently accepts just the solutions of the HW programming challenges, not the code files used to compute these solutions: the code files that resulted in successful solutions on Stepik should be uploaded to the corresponding assignments in Canvas ***prior to the HW deadline for follow-up plagiarism checking* (*no exceptions*).** Do not upload the code if your

solution was not successful on Stepik.

The textbook and extensive FAQs are designed in such a way that they contain all information (including explicit or implicit hints) needed for you to succeed in HWs. Thus, no hints on how to solve HWs will be provided before the HW deadline.

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. This book unifies the descriptions and test cases for all HW problems.

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 automatic grader 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. For information on how we enforce this policy, please see section "*You may not get credit for a HW problem you have solved… an unusual two-stage HW grading approach*".

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 learning 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 (including quizzes, the midterm, 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 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! This unusual credit is well-deserved since asking a good question about a problem is sometimes as difficult as 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 that will require a specially scheduled one-on-one meeting with the instructor/TAs where you will be asked questions related to various HWs. If the instructor concludes that you did not follow the "*Do not submit the code before you understand how the algorithm works!*" approach for a specific weekly HW, the results of this *entire* HW will be nullified.

Do not use external packages to solve HWs! To be consistent with other CSE classes with automated HW checking and strict anti-plagiarism enforcement, 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 prevented students from solving some HW problems.

Addressing learning breakdowns. Each student who experienced a learning breakdown 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.

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.

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 in the section "DETAILED SCHEDULE". 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.

Reviewing online materials. Students can review the online materials (Stepik, FAQs, 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. 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).

MIDTERM, FINAL, NON-GRADED QUIZZES, AND INVITED LECTURES

Midterm and Final. The midterm and final exams will consist of newly designed programming challenges that 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) and to mark all new lines where you made changes as compared to the code for *A*. 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.

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 invited lectures. We will not have quizzes during some (busy) weeks.

Please bring the fully-charged laptop (some quizzes will require 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 (even more convenient) on your smartphone.

Avoid anti-correlation between HW and midterm scores/quiz results! The quizzes and midterm exam 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, final, and quiz 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 section "*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.

Invited lectures. Since the 200+ instructors who have 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.

You will be given quizzes that test your understanding of the topics discussed in these invited lectures. We encourage you to ask questions during these lectures to help improve your understanding of the underlying concepts.

In Winter 2023, we will cover additional materials contributed by the following instructors:

- Phillip Compeau, Department of Computational Biology, **Carnegie Mellon University**
- Alexey Gurevich, Computer Science Department and Institute for Pharmaceutical Research at **Saarland University**, Saarbrücken, Germany
- Ben Raphael, Computer Science Department, **Princeton University**

Here is information about the research-oriented lectures by collaborating professors from other universities that use the same textbook:

- Phillip Compeau (CMU): ***How do we measure gene expression: transcript assembly and quantification.***

  This lecture is related to the topics of sequence comparison (chapter 5), clustering (chapter 8), and read mapping (chapter 9).
- Alexey Gurevich (Saarland): ***How do we compare LONG genomic sequences?***

  This lecture is related to the topics of genome assembly (chapter 3) and sequence comparison (chapter 5).
- Pavel Pevzner (UCSD): ***The long-read revolution in genome sequencing.***

  This lecture is related to the topic of genome assembly (chapter 3).
- Ben Raphael (Princeton): ***Cancer evolution.***

  This lecture is related to the topics of genome assembly (chapter 3), genome rearrangements (chapter 6), and evolutionary tree construction (chapter 7).

In addition, we will have a COVID-19-related lecture that discusses how bioinformatics contributes to emerging problems in personalized immunogenomics and antibody discovery:

- Pavel Pevzner (UCSD): ***Personalized immunogenomics.***

Last but not least, we will have a lecture that describes an open-ended biological problem and raises an important question on how to transform it into a well-defined computational problem:

- Andrei Osterman (Sanford-Burnham-Prebys Medical Discovery Institute): ***Predictive nutriomics approach to balancing human gut microbiome.***

You are encouraged to use the problems outlined in this lecture for selecting your class project.

CLASS PROJECT

Forming a group for the class project. The students are encouraged to work in small groups of 3 students. We recommend that each group includes students with both biological and computational backgrounds. Every group should work on the project without communicating with other groups. Piazza (see link above) has a "Search for Teammates" page; we encourage using this to find other group members for the class project.

The art of problem formulations. Every bioinformatician faces three challenges:

- Select a biological problem to work on
- Transform a biological problem into a computational one
- Solve the resulting computational problem and benchmark the solution

The class project encompasses the development of a computational problem formulation and algorithmic / software solutions for a bioinformatics problem relevant to one of the topics covered in the class. Transforming a (typically imprecise) biological problem into a well-defined computational problem is a major and often under-appreciated challenge in bioinformatics (for details on this, see section "*What is a Well-Defined Computational Problem?*" below). Some biologists somehow assume that as soon as a biological problem is formulated, its transformation into a computational problem will be automatically taken care of by bioinformaticians. Moreover, some biologists are not trained to distinguish a "well-defined" (e.g., precisely describing the Input, the Output, and the Objective Function) problems from an "ill-defined" problem. On the other side of things, computational scientists may not realize that identifying a new bioinformatics problem - and transforming the corresponding biological problem into a computational challenge - are both difficult tasks.

The projects in this class typically require significant efforts to transform a biological problem into a computational one. You are encouraged to read the relevant biological literature to better understand the subject area. Students will be guided through various stages of bioinformatics research: formulating the problem, designing the research plan, responding to the criticism of the reviewers, preparing the presentation, writing the paper, etc. It is important that you start working on the project as soon as possible and file the progress reports reflecting your work on the project by the specified deadlines (see section "CLASS PROJECT SCHEDULE"). It is important to complete the project on time in order to allow time for end-of-quarter events like presentations; deviations from this schedule will negatively affect your grade.

What is a Well-Defined Computational Problem? At a high-level, a well-formulated problem requires precise Input and Output (it should be clear how to determine whether the Output is correct given the Input). If a problem is well-formulated, there would be no ambiguity if a mathematician or computer scientist without any biological background attempt to solve it.

Some students may find it difficult to distinguish between a well-defined and an ill-defined problem. A few tips for developing a well-defined problem (and/or verifying that a problem is well-defined):

- Whenever possible, define terms using precise mathematical notation and avoid using biological terms. For example, instead of referring to a *genome*, you may refer to a *string of symbols*. If you insist on using biological terms in your input/output description you should first define them as purely mathematical terms. The language commonly used for biology research neither requires nor encourages the precision this project demands.
- Be specific and precise when describing the problem's objectives, constraints, input, and output.
- After you have defined all needed terms, the following problem formulation describing the input and output should be short and elegant - it should not take up more than a paragraph. If it is longer, you may have not defined all the necessary terms before the problem formulation starts.
- Problem formulation should be independent of implementation; even problems that are intractable can still be well-formulated.

Your project should have an interesting biological application and be, at the same time, algorithmically interesting and non-trivial. However, this has no bearing on whether the problem is well-formulated. Even if the explanation of the biological impact were to be removed, the description of what is expected from the program should be precise and should form a non-trivial algorithmic problem.

Grading criteria for the class project:

- transformation of a biological problem into a computational one
- formulating a well-defined computational problem
- reviewing the previous research in the area
- writing a self-contained and concise report
- proposing efficient algorithmic solutions
- sensible benchmarking design
- clear description of results in the report
- insightful discussion of further directions
- complete bibliographic review

ACADEMIC INTEGRITY

To detect instances of academic integrity violations in programming assignments we will use third-party plagiarism detection software. You may find the tutorial "Plagiarizing source code" at the link:

https://libraries.ucsd.edu/assets/elearning/cse/cseplagiarismexternal/story.html

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. 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 another student uses your solution in a HW. Please note that, if you solved a HW before the start of the class (e.g., in the Fall 2022 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.

RESOURCES FOR STUDENTS

Diversity and Inclusion. 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 Disability (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 their 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.

DETAILED SCHEDULE

(subject to change)

A detailed schedule is given below. Most classes will be in-person. Some classes (TBD) may be given on Zoom.

Homework deadlines are at 11:59 pm (PT) on the specified dates. Please note that HWs do not necessarily include ALL PROBLEMS from a given chapter. Read information below for the list of excluded problems for each chapter - the HWs contain only 59 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 Stepik 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 Stepik; for a list of which exact problems are required in the course, please see this spreadsheet (just the problems labelled with "Kept" in their "Used in class?" column are required).

As discussed above in the section "*Automated homework testing on the Stepik platform*": **please submit solutions generated by your code to Stepik, and please submit the code files used for these problems to Canvas.**

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

- Monday, January 9. Introductory lecture.
- Wed, January 11. Pavel Pevzner. ***From a Real-World Problem to an Algorithmic Challenge: the Art of Problem Formulations.***

Martin Luther King Jr. Day. Monday, January 16 (no class)

- Wed Jan 18. Andrei Osterman (Sanford-Burnham-Prebys Medical Discovery Institute) ***Predictive nutriomics approach to balancing human gut microbiome.***

Replication Origin (Chapter 1)
- HW deadline: Tue, Jan 17
- Survey deadline: Wed, Jan 18, 3 pm
- Communication Session, Mon, Jan 23
- Study: Mon, Jan 9 - Tue, Jan 17
- Excluded HW problems: 1C (Find reverse complement), 1D (Find all occurences of a pattern in a string), 1G (Hamming distance), 1J (Find frequent words with mismatches and reverse complements), 1L (PatternToNumber), and 1M (NumberToPattern).
- Total: 8 problems.

Regulatory Motifs (Chapter 2)
- Communication Session, Wed, Jan 25
- HW deadline: Thu, Jan 26
- Survey deadline: Fri, Jan 27, 3 pm
- Study: Tue, Jan 17 – Tue, Jan 24
- Excluded HW problems: 2D (Greedy motif search), 2E (Greedy motif search with

pseudocounts)
- Total: 6 problems.

Assembly (Chapter 3)
- Communication Session, Mon, Jan 30
- HW deadline: Tue, Jan 31
- Survey deadline: Wed, Feb 1, 3 pm
- Study: Tue, Jan 24 – Tue, Jan 31
- Excluded HW problems: 3A (Generate the k-mer composition of a string), 3B (Reconstruct a string from its genome path), 3K (Generate contigs), 3L (Generate a string spelled by a gapped genome path), and 3M (Generate all maximal non-branching paths in a graph).
- Total: 8 problems.
- Wed, Feb 1. Pavel Pevzner (UCSD).
  The long-read revolution in genome sequencing.

Alignment, Part 1 (Chapter 5, ending before (not including) "Penalizing Insertions and Deletions")
- Communication Session, Mon, Feb 6
- HW deadline: Tue, Feb 7
- Survey deadline: Wed, Feb 8, 3 pm
- Study: Tue, Jan 31 – Tue, Feb 7
- Excluded HW problems: 5A (Find the minimum number of coins).
- Total: 8 problems (5B-5I). You may want to solve 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.
- Wed, Feb 8. Seminar- Invited talk: Alexey Gurevich (Saarland University).
  How do we compare LONG genomic sequences?

Alignment, Part 2 (Chapter 5, starting from (including) "Penalizing Insertions and Deletions")
- Communication Session, Mon, Feb 13
- HW deadline: Tue, Feb 14
- Survey deadline: Wed, Feb 15, 3 pm
- Study: Tue, Feb 7 – Tue, Feb 14
- Total: 5 problems (5J-5N).

***Midterm exam*: Wednesday, Feb 15**

Presidents' Day. Monday, February 20 (no class)

Rearrangements (Chapter 6)
- HW deadline: Tue, Feb 21
- Survey deadline: Wed, Feb 22, 3 pm
- Communication Session, Wed, Feb 22

- Study: Tue, Feb 14 – Tue, Feb 21
- Excluded HW problems: 6I (GraphToGenome), 6J (2-BreakOnGenomeGraph), and 6K (2-BreakOnGenome).
- Total: 8 problems.

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

- Mon, February 27. Ben Raphael (Princeton).
  Cancer Evolution.
- Communication Session, Wed, March 1
- HW deadline: Thu, March 2
- Survey deadline: Fri, March 3, 3 pm
- Study: Tue, Feb 21 – Tue, Feb 28
- Excluded HW problems: 9F (Finding shortest non-shared substring), 9H, (Pattern Matching with the Suffix Array)
- Total: 7 problems (9A-9E and 9G, 9I).

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

- Communication Session, Mon, March 6
- HW deadline: Tue, March 7
- Survey deadline: Wed, March 8, 3 pm
- Study: Tue, Feb 28 – Tue, March 7
- Excluded HW problems: 9P (TreeColoring), 9Q (Partial Suffix Array of a String), and 9R (Suffix Tree from a Suffix Array)
- Total: 6 problems (9J-9O).
- Wed, March 8. Phillip Compeau (Carnegie Mellon University).
  How do we measure gene expression: transcript assembly and quantification.

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

- Communication Session, Mon, March 13
- HW deadline: Tue, March 14
- Survey deadline: Wed, March 15, 3 pm
- Study: Tue, March 7 – Tue, March 14
- Excluded HW Problems. Implement all problems from Chapter 8 except for 8D, (Implement the Soft k-Means Clustering Algorithm) and 8E (Hierarchical Clustering). **Do not** implement any problems from Chapter 10 except for the only problem, 10C (Viterbi algorithm).
- Total: 4 problems (8A-8C and 10C).
- Wed, March 15. Project presentations.

Final exam: **Wednesday, March 22, 7 pm – 9:59 pm**

SCHEDULE AT A GLANCE

(subject to change)

- Mon, Jan 9, Introductory lecture
- Wed, Jan 11, Pavel Pevzner. ***From a Real-World Problem to an Algorithmic Challenge: the Art of Problem Formulations.***

- Mon, Jan 16, Martin Luther King Day (no class)
- Wed, Jan 18, Andrei Osterman (Sanford-Burnham-Prebys Medical Discovery Institute) ***Predictive nutriomics approach to balancing human gut microbiome.***

- Mon, Jan 23, Communication Session (DNA Replication)
- Wed, Jan 25, Communication Session (Motifs)

- Mon, Jan 30, Communication Session (Assembly)
- Wed, Feb 1, Pavel Pevzner. ***The long-read revolution in genome sequencing.***

- Mon, Feb 6, Communication Session (*Alignment, Part 1*)
- Wed, Feb 8, Alexey Gurevich (Saarland University). ***How do we compare LONG genomic sequences?***

- Mon, Feb 13, Communication Session (*Alignment, Part 2*)
- **Wed, Feb 15, *Midterm***

- Mon, Feb 20, President's Day (no class)
- Wed, Feb 22, Communication Session (*Rearrangements*)

- Mon, Feb 27, Ben Raphael (Princeton). ***Cancer Evolution.***
- Wed, Mar 1, Communication Session (*Detecting Mutations, Part 1*)

- Mon, Mar 6, Communication Session, (*Detecting Mutations, Part 2*)
- Wed, Mar 8, Phillip Compeau (Carnegie Mellon University). ***How do we measure gene expression: transcript assembly and quantification.***

- Mon, Mar 13, Communication Session (*Clustering* and *Hidden Markov Models*)
- Wed, Mar 15, Project presentations.

- **Wed, Mar 22 (7 pm – 9:59 pm), *Final***

CLASS PROJECT SCHEDULE
(subject to change)

All deadlines are at 11:59pm PT on the selected dates, with the exception of the final presentation slides (which are due at 6:30pm PT). The first deadline asks you to submit the document by emailing the TA; we'll then create student groups on Canvas. Starting with the second deadline, all project deliverables will be submitted through Canvas.

- **Friday, January 20.** Deadline for **selecting the project you plan to work on**. By this deadline (or earlier) you will have to specify what project you selected (1 page) and email this information, along with a list of all other student(s) in your group, to the TA. The description should be concise but sufficient to communicate the importance of a biological problem and sketch potential algorithmic challenges.
    - Since this class focuses on algorithms rather than statistics or machine learning, only problem formulations presenting algorithmic challenges are accepted.
    - The project will be assigned on a first-come, first-served basis. If your project has been already selected by multiple groups, you will have to choose another project. Try to select the project well before the deadline to make sure that you have a variety of projects to choose from. (One exception: multiple groups are allowed to choose a project based on the talk by Prof. Osterman on January 18, although these groups should still work independently.)
    - Please specify how you learned about this problem (e.g., a journal paper, a professor, a conference, the in-class presentation by Prof. Osterman on January 18, etc.).
    - Do not select problems that you have encountered and solved during your previous rotations.
    - Do not select well-studied problems for which a computational problem formulation has been already published in a journal.

  Read recent research papers relevant to the project and make sure that you understand all aspects of the project. Start working on transforming it into well-defined computational problem(s).

- **Friday, January 27.** Submit (on Canvas) an at most 1-page long **computational problem formulation and work plan for your project**. Your project will be assigned to another group who will write an initial review and will decide whether the problem is well-formulated; similarly, you will be assigned another group's problem formulation and work plan to review.

  In a typical class, most initial problem formulations have various issues. Moreover, since some selected projects do not lend themselves to viable problem formulations, some groups will have to completely change their initial project. That is why you should focus on your problem formulation (rather than the work plan and algorithms for your project) until yours is accepted by the instructor.

  Meet with the instructor and TA shortly afterward to discuss the problem formulation you proposed.

- **Friday, February 3.** Submit **feedback on the other group's document**. This should be a short report (at most half a page) describing the potential pitfalls of the other group's proposed problem formulation and work plan. Note that these reviews will count towards your project grade.

- **Friday, February 10.** Submit an **updated problem formulation and work plan**, based on the feedback you've received thus far. *If your problem formulation has not yet been accepted by the instructor, your main focus here should be fixing it.*

- **Friday, February 17.** Submit a **corrected and extended description of the project specifying a detailed plan of your work** in the next ~3 weeks. The submission should specify your problem formulation, research plan, algorithmic challenges, and software implementation efforts. Prepare a list of a few milestones, with deadlines for achieving each of them. *If your problem formulation has not yet been accepted by the instructor, your main focus here should be fixing it.*

- **Friday, February 24**. Submit a **summary of your progress and preliminary results** (including your problem formulation). Even if your problem formulation has not yet been accepted by the instructor, you can move on to working on the rest of the project starting with this submission.

- **Friday, March 10.** Deadline for a **5-page long paper**. You are expected to meet the instructor and TA to make a short powerpoint presentation (based on the 5-page paper) describing your project, the remaining challenges you are facing, and the results. The projects will be presented in the class in the last day of the quarter.

- **Wednesday, March 15**. **In-class presentations** for the projects. (Your slides should be submitted by 6:30pm PT this day - just before the start of class.)