CS Town Hall F21 Survey Responses

Aggregated and Anonymized Responses to survey responses

Curriculum

Courses that students enjoyed

Which course(s) did you really enjoy?	What made these courses enjoyable?
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 111 - Operating Systems Principles, CS 131 - Programming Languages	Good, engaging lecturers and thick, meaningful projects that really force you to learn something
CS 1 - Freshman Computer Science Seminar, CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 118 - Computer Network Fundamentals	Detailed instructions, reasonable expectations
CS 131 - Programming Languages, CS 132 - Compiler Construction, CS 231	I liked the topics covered in those classes. For 132 in particular, I liked how it was project focused and there was significant programming involved, which is not really true for any other CS class I have taken at UCLA.
CS 33 - Introduction to Computer Architecture, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles, CS M151B - Computer Systems Architecture, CS M152A - Introductory Digital Design Laboratory	Professor's well organized slide and workload, passionate TA and professor
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture	
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS 118 -	interesting subject matter

Computer Network Fundamentals, CS 131 - Programming Languages, CS 132 - Compiler Construction, CS 180 - Introduction to Algorithms and Complexity	
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS M51A - Logic Design of Digital Systems, CM124 - Machine Learning Applications in Genetics, CS M146 - Introduction to Machine Learning, CS 180 - Introduction to Algorithms and Complexity, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	The professors and content
CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	Good content and teaching was thorough
CS 32 - Introduction to Computer Science II, CS 131 - Programming Languages, CS 134 - Distributed Systems, CS 181 - Introduction to Formal Languages and Automata Theory	Interesting topics with very visible applications in industry, fun problems
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS 131 - Programming Languages, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	Intersting course material, cool projects (like the bomb lab in 33, A* implementation in 32, ocaml lab in 131), professors had a deep understanding and were interested in topic
CS 131 - Programming Languages, CS 161 - Fundamentals of Artificial Intelligence, CS 181 - Introduction to Formal Languages and Automata Theory, CS 183 - Introduction to Cryptography	emphasis on theory and rigor
CS 111 - Operating Systems Principles, CS 174A - Introduction to Computer Graphics, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	These courses have fun and relevant applications.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS M51A - Logic Design of Digital Systems	31 and 32 were thorough introductions to computer science topics with interesting lectures and meaningful homework assignments. m51a covers interesting ideas and builds up to a overarching goal throughout the quarter which

	makes the class meaningful
CS 32 - Introduction to Computer Science II	I liked doing the projects.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II	professors were engaging and the homework/projects were doable. content was easy to learn
CS 32 - Introduction to Computer Science II, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	Great professors. Nachenberg for 32, Sarrafzadeh for 180, and Sahai for 181.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II	They were heavily focused on just learning coding and doing projects, which were enjoyable and rewarding to complete
CM124 - Machine Learning Applications in Genetics, CS 143 - Database Systems, CS M184 - Introduction to Computational and Systems Biology	Reasonable workload, interesting and relevant content that I could readily use outside of class, good organization (everything built on top on each other nicely)
CS 30 - Principles & Practices of Computing, CS 31 - Introduction to Computer Science I, CS 118 - Computer Network Fundamentals, CS 143 - Database Systems, CS M151B - Computer Systems Architecture, CS 161 - Fundamentals of Artificial Intelligence	Great professors and interesting, engaging curriculum.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles	The professors! They were great lecturers, and gave us fewer great homework assignments/projects as opposed to a barrage of pointless memorization exercises each week
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II	The consistency of the class work, e.g. a project every 1-2 weeks, as well as the pacing which I found to be good.
CS 31 - Introduction to Computer Science I, CS M51A - Logic Design of Digital Systems, CS M148 - Introduction to Data Science, CS 181 - Introduction to Formal Languages and Automata Theory	Lectures, homework/projects, and examinations were all connected (i.e., the course didn't feel disconnected). The professors who taught them were also very clear in their lectures.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 -	For all of them, good prof, interesting topic and very well designed projects.

Introduction to Computer Architecture, CS 143 - Database Systems, CS 161 - Fundamentals of Artificial Intelligence, CS 174A - Introduction to Computer Graphics	
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS 143 - Database Systems, CS 174A - Introduction to Computer Graphics	The hands-on approaches that they had, especially the revamped 35L/97 and 111
CS 30 - Principles & Practices of Computing, CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CM121 - Introduction to Bioinformatics, CM122 - Algorithms in Bioinformatics, CM124 - Machine Learning Applications in Genetics, CS M146 - Introduction to Machine Learning, CS 180 - Introduction to Algorithms and Complexity	
CS 31 - Introduction to Computer Science I, CS 174A - Introduction to Computer Graphics, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	Projects were directly related to the material learned in class. They had concrete, visual results.
CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 131 - Programming Languages, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 111 - Operating Systems Principles, CS 180 - Introduction to Algorithms and Complexity	33 - enjoyed reinman and the content itself was very interesting and labs were fun 32 - enjoyed nachenberg and content was good to learn and helped with interviews 111 - enjoyed eyolfson and content is in same vein as 33 so enjoyable for similar reasons 180 - I enjoy professor hsieh and content is very high level and proof based which I enjoy
CS 143 - Database Systems, CS 145 - Introduction to Data Mining, CS M146 - Introduction to	I like the new development of using big data to making informed decisions! I also like learning

Machine Learning, CS 161 - Fundamentals of Artificial Intelligence, CS 174A - Introduction to Computer Graphics	how to emulate graphics using code.
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS 136 - Introduction to Computer Security, CS M146 - Introduction to Machine Learning, CS 168 - Computational Methods for Medical Imaging, CS 181 - Introduction to Formal Languages and Automata Theory, CS 183 - Introduction to Cryptography	I generally enjoyed the professors I took them with as well as the content of the courses; I prefer lower level and conceptual classes and these courses aligned well with my interests. They are also among the most practical courses I have taken (especially 111) since I am now working in industry with low level code.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS 130 - Software Engineering, CS 188 - Special Courses in Computer Science: Secure Software Design and Development	Passionate lecturers as well as projects that were longer-term but had plenty of resources for help if one were to get stuck.
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS M51A - Logic Design of Digital Systems, CS 143 - Database Systems, CS 144 - Web Applications, CS M151B - Computer Systems Architecture, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	Professor John Cho is such a gem, he is a really great communicator and is the only professor who seems to understand student questions the first time they're asked. He make 143 and 144 really enjoyable and they are my favorite CS courses at UCLA. The topics taught in CS 33 and M151B are pretty interesting, learning how programs are converted into silicon and I found Reinman's flipped classroom format to work fairly well. Korf's M51A offering adapted very well to the virtual format, and I appreciated him writing and drawing everything by hand. This made the content super easy to follow and really easy to take notes. Definitely the best professor for M51A.

	I also really enjoyed the way the computer vision 188 was structured. Though the lectures were not the most clear at times, the projects were all very enjoyable and the content taught was always linked to real world applications
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS M51A - Logic Design of Digital Systems, CS 174A - Introduction to Computer Graphics, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	For each of these courses, I found the material interesting and the professors engaging. I could not say the same for any of the other computer science courses I took.
CS 1 - Freshman Computer Science Seminar, CS 31 - Introduction to Computer Science I, CS 131 - Programming Languages, CS 145 - Introduction to Data Mining, CS M146 - Introduction to Machine Learning, CS 161 - Fundamentals of Artificial Intelligence, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision, CS M192A - Introduction to Collaborative Learning Theory and Practice	proofs, pedagogy
CS 111 - Operating Systems Principles, CS 143 - Database Systems	
CS 35L/97 - Software Construction Laboratory, CS M146 - Introduction to Machine Learning	Good professors, material, and TAs
CS 30 - Principles & Practices of Computing, CS 31 - Introduction to Computer Science I	The professors and the fact that the workload was not overbearing. I felt like doing the projects and the homework was actually building on what I was learning and they werent too long or hard to where I was constantly stressing out.
CS 32 - Introduction to Computer Science II, CS 144 - Web Applications, CS 188 - Special Courses in Computer Science: Scalable Internet Services	In both CS 32 and CS 144, I got to build chonkier projects. I like that feeling. I actually did not take 188 Scalable Internet

	Services but I heard glowing reviews about it from last year and I'm extremely sad that it isn't being offered this year.
CS 174A - Introduction to Computer Graphics	The topic interested me.
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory	Enjoyable content presented in a mostly lecture format. Midterms allowed me to make up project grades.
CS 143 - Database Systems	
CS 131 - Programming Languages, CS 132 - Compiler Construction, CS M146 - Introduction to Machine Learning, CS M148 - Introduction to Data Science, CS 180 - Introduction to Algorithms and Complexity, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	CS188/132/146/148 - elective classes with specifications I am very interested in CS131 - I am interested in programming language, and the discussions of concurrency, parsing, and functional programming have been very helpful to me
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture	
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 131 - Programming Languages, CS 161 - Fundamentals of Artificial Intelligence	
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems	CS 32: Carey CS 33: The labs were fun and I liked learning about the lower-level parts of computers CS 35L: Filled in some knowledge gaps about Linux CS M51A: Same deal as 33; I like learning about the lower-level parts of computers
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles	CS 31 - Knowing the content ahead of time, and Smallberg, exams were fair and fun CS 32 - Smallberg and Carey, Knowing the content ahead of time, I love data structures, exams were fair and fun, Project 3 unironically was the most fun I've had doing a project

	CS 35L - Knowing the first half of content already, TA Daniel who gave the hints slides made the experience 100 times better CS M51A - Only because of Korf and no exams
CS 32 - Introduction to Computer Science II, CS 111 - Operating Systems Principles, CS 143 - Database Systems, CS 188 - Special Courses in Computer Science: Scalable Internet Services	Real world applicable
CS 32 - Introduction to Computer Science II, CS 132 - Compiler Construction, CS 134 - Distributed Systems, CS M151B - Computer Systems Architecture, CS 161 - Fundamentals of Artificial Intelligence, CS 168 - Computational Methods for Medical Imaging, CS 188 - Special Courses in Computer Science: Scalable Internet Services, CS 188 - Special Courses in Computer Science: Turn Your Idea into Company	the profs, the material
CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 130 - Software Engineering, CS 131 - Programming Languages, CS 132 - Compiler Construction, CS 161 - Fundamentals of Artificial Intelligence, CS 239 - Quantum Computing	CS 33: I really liked the hands-on nature of the labs. They were fun but also clearly connected to the course material, and I still remember the content to this day (i.e. I feel like I really "learned" it). CS 130: I took this with the Google engineers, and it was a very hands-on/practical course with great advice from industry. I would love to have more classes with outside perspective! CS 131: I love programming languages. I liked learning about logic programming, which I had no idea even existed beforehand! CS 132: Compilers is a great culmination class for CS, since it's a combination of many different parts of the curriculum (and I feel like I'm using everything I learned). In addition, Palsberg's approach to grading is extremely fair: having the grading framework published with instantaneous feedback makes me feel like I know what I'm being graded on. The LL(1) Academy (a set of online practice modules that exactly mimics the midterm) made it easy for me to prepare for the

	exams. CS 161: Darwiche is a very engaging lecturer, and I also enjoyed quite a few of the projects (esp the A* one). It was great having an AI class that wasn't just ML! CS 239: Quantum programming is just so cool! Would love to see more opportunities to learn cutting-edge technology; the partnership with Google/IBM was a once-in-a-lifetime experience.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles, CS 145 - Introduction to Data Mining, CS 180 - Introduction to Algorithms and Complexity	
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 143 - Database Systems	
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture	I thought they were very well-structured, solid introductions to Computer Science and Computer Organization. In addition, my professors Smallberg, Nachenberg, and Reinman were all very great lecturers and were great at answering any questions students had about the subject material.
CS 1 - Freshman Computer Science Seminar, CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS 131 - Programming Languages, CS 132 - Compiler Construction, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	I think huge credit has to go to the professors and teaching staff for making the course content interesting. The courses were "paced" well in the sense that the prerequisites made sure we were prepared for the courses. The lectures, content and assignments were especially interesting.
CS 32 - Introduction to Computer Science II, CS M146 - Introduction to Machine Learning, CS 161 -	

Fundamentals of Artificial Intelligence, CS 180 - Introduction to Algorithms and Complexity, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	
CS 32 - Introduction to Computer Science II, CS 111 - Operating Systems Principles	Jon Eyolfson made CS 111 super enjoyable, with all the projects being submitted via a git push and test cases built into the vm, and the flexible late days. CS 32 was fun because of the content and Carey Nachenberg.
CS 30 - Principles & Practices of Computing, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 111 - Operating Systems Principles, CS 145 - Introduction to Data Mining, CS 180 - Introduction to Algorithms and Complexity	The Professor
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory	
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS M51A - Logic Design of Digital Systems, CS M151B - Computer Systems Architecture, CS M152A - Introductory Digital Design Laboratory	Good lecturers, interesting content, and projects/homework assignments that were challenging but doable. Many of the assignments helped me reach a deeper understanding of the material, as I applied concepts learned in class.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS 143 - Database Systems, CS 145 - Introduction to Data Mining, CS M146 - Introduction to Machine Learning, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	
CS 32 - Introduction to Computer Science II, CS	

M146 - Introduction to Machine Learning, CS 180 - Introduction to Algorithms and Complexity	
CS 1 - Freshman Computer Science Seminar, CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS M51A - Logic Design of Digital Systems	Interesting assignments, clarity in teaching.
CS M51A - Logic Design of Digital Systems, CS 143 - Database Systems, CS M151B - Computer Systems Architecture	For 143: professor Cho's teaching style made materials easier to comprehend by examples. For M51A (prof Abari): less exam pressure. The class pace didn't make me feel overwhelmed.
CS 32 - Introduction to Computer Science II	Usefulness, enthusiasm of the professor
CS 31 - Introduction to Computer Science I	Professor Smallberg paces the course well and teaches in an easy to understand manner.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 143 - Database Systems, CS M148 - Introduction to Data Science, CS 181 - Introduction to Formal Languages and Automata Theory	Professor
CS 32 - Introduction to Computer Science II	interesting or useful content without an overwhelming amount of work
CS 32 - Introduction to Computer Science II, CS M51A - Logic Design of Digital Systems, CS 143 - Database Systems, CS 161 - Fundamentals of Artificial Intelligence, CS 174A - Introduction to Computer Graphics, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	Blend of interesting subject matter and challenging but enjoyable difficulty level
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS 143 - Database Systems, CS M146 - Introduction to Machine Learning, CS M148 - Introduction to Data	

Science, CS 180 - Introduction to Algorithms and Complexity, CS 181 - Introduction to Formal Languages and Automata Theory	
CS 32 - Introduction to Computer Science II, CS 131 - Programming Languages, CS 161 - Fundamentals of Artificial Intelligence, CS 181 - Introduction to Formal Languages and Automata Theory, CS M184 - Introduction to Computational and Systems Biology, CS 188 - Special Courses in Computer Science: Introduction to Computer Vision	Interesting topics, clearly presented. I think the course topics are the dominant factor in my enjoyment of these subjects.
CS 31 - Introduction to Computer Science I, CS 32 - Introduction to Computer Science II	Great professors!
CS 32 - Introduction to Computer Science II, CS M51A - Logic Design of Digital Systems	enthusiastic professors

Courses that students think need revamping

Which course(s) do you think needs revamping?	What exact changes would you make to the courses you selected above, if any?
N/A: I haven't taken any of these courses	
CS 131 - Programming Languages, CS M152A - Introductory Digital Design Laboratory, CS 174A - Introduction to Computer Graphics	
CS 32 - Introduction to Computer Science II, CS 111 - Operating Systems Principles, CS 131 - Programming Languages	CS 32: Cover modern C++ (e.g. smart pointers) and the implementation of data structures like self-balancing trees. CS 111: Have students implement an operating system themselves instead of just learning about how it works. CS 131: Have students implement a programming language themselves (e.g. an interpreter) instead of just learning about how it works.
CS 35L/97 - Software Construction Laboratory, CS M146 - Introduction to Machine Learning, CS 174A - Introduction to Computer Graphics	CS 35L could have split into two courses
None	
CS 111 - Operating Systems Principles, CS M152A - Introductory Digital Design Laboratory	
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS 131 - Programming Languages	Standardize eggerts exams so they’re not as chaotic
CS 145 - Introduction to Data Mining	145: Focus less on specific methods and algorithms and more on different types of data and mining on those
CS M51A - Logic Design of Digital Systems, CS 145 - Introduction to Data Mining, CS M146 - Introduction to Machine Learning, CS M148 - Introduction to Data Science, CS M151B - Computer Systems Architecture, CS M152A - Introductory Digital Design Laboratory	CS 145, CS M146, CS M148 are 80% the same class
CS 1 - Freshman Computer Science Seminar, CS 31 - Introduction to Computer Science I	I wish CS31 was more interesting. I think that the lectures could be made more interesting.
CS 111 - Operating Systems Principles, CS M152A	CS M152A: unfair grading

- Introductory Digital Design Laboratory	
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles	The classes are too time-consuming w.r.t. their number of units.
CS 1 - Freshman Computer Science Seminar, CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory	The first portion of CS 33 data/bit manipulation felt very overwhelming because it was covered very briefly despite it being a fundamental to the rest of the course. I think more focus on that portion in the first few weeks would be beneficial to the learning experience for the rest of the course. I do not think that there needs to be multiple lectures on OpenMP, but data manipulation probably does warrant more focus. CS m51a also goes more in depth on bits and bytes so recommending students to take m51a alongside 33 might be a good compromise. I have mixed feelings on 35L. I really appreciate that we have a course that covers miscellaneous software tooling topics and skills but the way it is presented (wide variety of topics crammed in a very short period of time) can be very overwhelming which compromises the learning experience. Assignments can still maintain the "learn on your own" attitude because I agree that that is a very effective way to learn quickly but they should all have a "tutorial" or "hold your hand" section to ease students into the assignment. For example, Assignment 3 has us follow an official React tutorial before asking us to expand our knowledge by then solving a more difficult problem. This process made me feel a lot more comfortable and confident in React and ultimately strengthened my learning experience.
CS 1 - Freshman Computer Science Seminar, CS 35L/97 - Software Construction Laboratory	CS 1 just didn't feel very effective as a course since some speakers talked to us as if we had prerequisite knowledge of what they were talking about so I just didn't know what was happening some days, but also I don't know if it'd be interesting if everything got "dumbed down". The

	homework was beyond the scope of the class sometimes. CS 35L doesn't let you go in depth in any of the topics covered since you go through so many :(
CS 33 - Introduction to Computer Architecture	the content in cs 33 is completely new to the vast majority of students, and therefore it takes a bit longer for information to sink in. personally, i felt that while the content in cs 33 was not very challenging, it took a long time to actually understand & learn, and I felt that the pace of the course was too fast.
none of the above	
CS 31 - Introduction to Computer Science I	CS31 was not lenient on partial credit and it made me, a non-CS major, loose enjoyment in CS. Because I misread one sentence in the specs, I got a 30% even though I did everything else correctly.
CS 35L/97 - Software Construction Laboratory	CS 35L has too much content everywhere, in my opinion. I think that it should focus more on a few topics, rather than spread out so much over so many. Maybe remove python + lisp and focus more on html, javascript, node/react, shell, and git since these things are what people will primarily use for their group project.
CS 111 - Operating Systems Principles, CS 131 - Programming Languages	111: Integrate some of its early content into CS 111 or 35L, Remove the beaglebone based assignments, 131: Reorder course content so that students understand that main principles behind developing a programming language early on and THEN discuss each language on a case by case basis (explaining the choices made in that languages design and how it works), make programming assignments simpler - it's not reasonable to expect us to write a parser with 2 weeks of OCaml knowledge (maybe put that at the end of the quarter, might be helpful with prepping for the final)

CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles	CS 35L: Slow down the pace of the curriculum so students can actually learn these topics instead of just skimming the surface. CS 111: Focus on core ideas for a few languages rather than trying to squeeze in a bunch of languages that are difficult to learn in 10 weeks.
CS 33 - Introduction to Computer Architecture, CS M151B - Computer Systems Architecture	CS 33: This class is so ridiculously boring and difficult that it sucks any possible enjoyment out of it. I'd rather cover lesser material and actually enjoy it than span so much material and learn nothing. CS M151B: Why do I have to take this class and why is the material so niche and obtuse? I understand the need to acquaint oneself with the hardware but I really think this should be an elective, because I have no interest whatsoever in pursuing anything remotely related to the material covered in this class, and I was able to decipher that after taking CS 33 and CS M51A.
CS 35L/97 - Software Construction Laboratory	I would remove Emacs from the course and possibly cover it in a later or upper division course. Currently, I think there's too much information being covered in the course of 10 weeks and while Emacs takes 2-3 weeks to cover, I find that NodeJS, Python, Bash, React, Git, and other technologies are a far more valuable use of that time.
CS 111 - Operating Systems Principles	CS 111: Very disconnected material. I felt like I was taking two 4-unit classes within one class. This was because the lecture material (and all the readings) seemed tangentially related to the very difficult projects.
CS 35L/97 - Software Construction Laboratory	
CS 130 - Software Engineering	Scrum process is not utilized to its full potential in the tiny projects done over a quarter of cs130.
CS M51A - Logic Design of Digital Systems	CS M51A: Was a good class, but the workload is high to the point where much of the homework feels like busywork and not testing my

	understanding of the course.
CS 130 - Software Engineering	I think the project section is rather inflexible in terms of the type of project students can choose to work on. Due to the requirements and format of the project, it's much easier to do some sort of web or mobile app than some sort of standalone project like machine learning or video game. This is worsened by the disorganized team formation, since I at least personally found it difficult to find a team since almost every team planned on making some sort of web or mobile app and therefore and I had no marketable skills to convince others to add me to their team. I don't know what improvements would actually fix these issues, but I would like to see them addressed.
CS 131 - Programming Languages, CS M152A - Introductory Digital Design Laboratory	CS M152A: remove this class from the curriculum, if possible. Otherwise, do as recommended and remove Verilog while adding Arduino C programming. CS 131: teach the programming language of choice before the project of said language is due. Clarify specifications of homeworks.
CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS M146 - Introduction to Machine Learning	111: Would've liked to see an implement scheduler, implement malloc, or implement mini OS project like at top CS schools such as Berkeley. Some info was out of date such as optimizing for hard drives (kudos to Reiher for skipping some of that stuff though.) 118: Would've liked to see a project relating to link/network layer (something with BGP or WiFi would've been really cool) in addition to TCP. The Wireshark project was very surface level. I did not feel prepared to work on network-related stuff at a company after taking this class. M146: Too basic. Would like to see a class for engineers interested in applications and not just proving bounds, but also not struggling with the mathematical concepts.

CS 174A - Introduction to Computer Graphics	The extensions of 174A should be offered more often.
CS 131 - Programming Languages, CS 174A - Introduction to Computer Graphics	CS 131: I feel that the amount of content covered in this class is fundamentally not possible to learn within the span of a quarter. The expectations that are given to students by the professors of this course are significantly higher than those of any other course, and while the content can be interesting, often lectures lag behind on the required information needed to complete projects. On top of this, there is a major disconnect between the content covered in lecture and the skills required to effectively complete any of the assignments. CS 174A: The content of this course feels outdated given how rapidly modern computer graphics have developed. While I understand this is meant as an introductory course, using old libraries to render images feels clunky at best and is not a great way to learn where one can actually go with computer graphics.
CS 111 - Operating Systems Principles, CS 161 - Fundamentals of Artificial Intelligence	CS 111: offer more up to date projects that weave better with the readings and lectures CS 161: standardize the information taught and dabble more into less outdated information
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS M152A - Introductory Digital Design Laboratory	I'm not sure what should be cut from 35L and 111, but both courses seemed impossibly full of information at the time I took them; so much so that there was no way I could retain any or even most of it. For M152A, removing Verilog and adding in Arduino sounds like a great idea.
CS 35L/97 - Software Construction Laboratory, CS 130 - Software Engineering, CS 131 - Programming Languages	introduce more formal methods to COM SCI 131
CS 111 - Operating Systems Principles, CS 131 - Programming Languages, CS M152A - Introductory Digital Design Laboratory	

CS 35L/97 - Software Construction Laboratory	CS35L/97: Make this class 5 units or allow more time for discussions.
CS 32 - Introduction to Computer Science II, CS 35L/97 - Software Construction Laboratory	
CS 111 - Operating Systems Principles, CS 131 - Programming Languages, CS M152A - Introductory Digital Design Laboratory	I took 111 with Harry Xu and I do not know if it is better now with Eyolfson, but I felt that the projects and lecture were not very relevant to each other. I appreciate it much more when the projects actually help me think that I'm applying some lecture principle. (Especially if the project is not the focus of the course, unlike John Cho's 144.) With 131 assignments, I also thought that the lecture was unrelated, although maybe less so. I think the projects could be less difficult especially since they are implemented in new paradigms of programming languages. The second OCaml project (about language rules??) was that hard and for what???? And with CS M152A, please replace Verilog with something else.
CS 31 - Introduction to Computer Science I, CS M51A - Logic Design of Digital Systems	
CS 35L/97 - Software Construction Laboratory, CS M152A - Introductory Digital Design Laboratory	
CS 1 - Freshman Computer Science Seminar	CS 1 - Make a weekly seminar speaker series over Fall quarter with open attendance so all years can come if they are interested, and we are not required to go to seminars which we are not interested in
CS 180 - Introduction to Algorithms and Complexity	
CS 1 - Freshman Computer Science Seminar	CS 1 - There needs to be more effort put into this seminar. Professors coming once a week, and TAs giving homework that scales from easy to needing to perfectly understand the material

	(which the professors said was okay not to understand) and quizzes that were just copy pasted from the slides. Why have the quizzes in that case? And since it's a seminar, why not have more effort in discussion to delve on the topic? What is the point of grading that class? It's supposed to be an exploration into possible topics. Maybe do Homework during discussions as exploration. I would've learned more that way than having to self teach the entirety of the content in order to do a single homework assignment
CS 33 - Introduction to Computer Architecture, CS 35L/97 - Software Construction Laboratory, CS 131 - Programming Languages, CS 161 - Fundamentals of Artificial Intelligence	
CS 131 - Programming Languages	CS 131: I wish the lecture content could line up with the projects; the projects were pretty difficult and took so long to do so I had to start early, but I didn't have any context from lecture since we didn't cover it yet. So doing the homework vs going to lecture felt a bit disjoint; also the homeworks were so long I didn't feel like I got to enjoy the strengths, weaknesses, differences between all the languages. Except Prolog, that homework was a good length and helped me understand how Prolog was useful.
CS 1 - Freshman Computer Science Seminar, CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles, CS 131 - Programming Languages, CS 145 - Introduction to Data Mining	CS 1: this class has so much potential, but in its current state I feel that it's not very helpful. I would like to see lectures that are slightly more interactive or directly focused on what students can do in a field; the discussions should be actual discussions. CS 35L: I still think the workload / unit count for this class needs to be reevaluated. I appreciate the changes made with 97, but I think more still needs to be done. CS M51A: I think this class would benefit significantly from some sort of interactive

	example (ex being able to "play" with state machines or explore the material with real hardware). CS 111: In taking this with Reiher, I think there's quite a bit of disjointness between the lectures and the labs. I've heard that Eyolfson's class is great, and we should take the approach of more hands-on labs that are directly tied to the course material; similar to CS 33 labs! CS 131: while I really enjoyed this class, I really think that the homework is extremely overbearing. Homeworks should be assigned *after* the language is taught in class (instead of being the mechanism to learn the language), and I would appreciate it if large projects were broken up into smaller disjoint pieces. CS 145: this class has too much overlap with CSM146. I would like to see a larger focus on non-ML methods, and/or a deeper exploration of the course content.
CS 1 - Freshman Computer Science Seminar, CS 131 - Programming Languages	
CS 35L/97 - Software Construction Laboratory, CS M51A - Logic Design of Digital Systems, CS 111 - Operating Systems Principles, CS M146 - Introduction to Machine Learning, CS 180 - Introduction to Algorithms and Complexity	
CS M152A - Introductory Digital Design Laboratory	Good suggestion in the question itself! C programming would be a welcome change from Verilog although my complaint with CS M152A is not so much with the programming language but in the manner the course is administered. We need lectures from professors! Our TA's are mostly great and helpful but it sucks to just be working on labs without gaining a better understanding of what's going on underneath. Everyone tends to be very clueless when taking this course; neither the TA nor the students seem to know what's going on.

	So more professor involvement with even one lecture a week would be beneficial imo.
CS M152A - Introductory Digital Design Laboratory	CS M152A if kept at all should do a better job of actually teaching Verilog and skills needed for the labs.
CS 111 - Operating Systems Principles, CS 118 - Computer Network Fundamentals, CS 161 - Fundamentals of Artificial Intelligence	161: revamping of the projects
CS 1 - Freshman Computer Science Seminar, CS M51A - Logic Design of Digital Systems	CS 1: It feels like the speakers and the professor are never on the same page. This process needs to be more unified and coherent. CS M51A: I know people dislike verilog, but I would appreciate some form of digital circuit design software to submit through rather than going through things by hand.
CS 111 - Operating Systems Principles	CS 111: Make labs relevant to course material, or at least touch on some of the concepts in class.
CS 1 - Freshman Computer Science Seminar, CS 131 - Programming Languages, CS 161 - Fundamentals of Artificial Intelligence	CS 161: More python, less LISP. CS 1: More variety in guest lecturers. CS 131: lower workload
CS 35L/97 - Software Construction Laboratory, CS M152A - Introductory Digital Design Laboratory, CS 181 - Introduction to Formal Languages and Automata Theory	Verilog is actually an okay language, but the instructions are vague and the time to self-study is too much compared to its units. Also replacing it with a new, more practical hardware language is not a bad idea. 35L/97: The problem of this class is it's too brutal for students who just getting started with CS (especially for transfers). The workload is too much and the exams made students felt like they learn nothing from it. This class should be divided to 2 small lab classes so students can have more time to absorb the materials.
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles	CS 35L/97: Relate the lectures to the homework/projects. CS 111: Remove the need for a Beaglebone and revamp the projects accordingly.
CS 1 - Freshman Computer Science Seminar	CS 1: Not that I don't like AI, deep learning and

	machine learning, but at least five or six of the lectures we had were on some form of the these topics. I also think the general structure of the discussion sections, in which the TA essentially repeats what was said during the lecture and we take a five question open note quiz is more or less just a waste of time, not really offering anything substantial.
CS 35L/97 - Software Construction Laboratory	I don't understand why CS35L includes a group project on top of a normal amount of assignments and tests. I feel like the tests and assignments significantly detract from the amount of effort I wanted to put into the group project.
CS 131 - Programming Languages, CS 145 - Introduction to Data Mining	145/M146: Make 145 more different from M146 131: Go slower and/or make the projects less difficult/more incremental
CS M51A - Logic Design of Digital Systems, CS 118 - Computer Network Fundamentals	CS M51A: More opportunities to design and analyze circuits to aid understanding. I feel like my understanding is kind of surface level without this practice. CS 118: More opportunities to interact with networking systems and tinker with them to gain a more intuitive understanding of how the systems work, rather than just reading the slides / textbook.
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles, CS 180 - Introduction to Algorithms and Complexity	CS 180: Add assignments that give practical meaning to the algorithms that we learn (more actual code rather than pseudocode only)
CS 33 - Introduction to Computer Architecture	CS 33, make the assignments not take 20 hours every week.
CS 35L/97 - Software Construction Laboratory, CS 111 - Operating Systems Principles	reduce workload by pruning material or spreading out across multiple classes. alternatively increase unit counts

New Courses that students want to see added

Are there any CS classes or topics you'd like to see added to the curriculum?
Generally more modern and practical courses - web design, python, UI/UX
Yes. Functional programming: this is briefly covered in 131 but does not include much about the type system or more advanced topics such as modules or monads. Data structures and algorithms: the current topics covered in 32 and 180 are quite basic and do not include things like the implementation of self-balancing trees for instance. Operating systems: 111 only covers the concepts of OS and I would like a more advanced class where students implement an OS from scratch themselves (CS 235 seems to involve this but it doesn't seem to have been offered in more than 10 years). Programming language theory: 231 is an introductory PLT class but there are no further classes in this area. Compilers: 132 is an introductory class and there are no further classes in this area. Constructive logic and category theory: this might belong to the math department but these topics are useful in programming language theory. Programming language design: CS C137A/B have not been offered in many years.
app development
search engines, distributed systems
Optimization
Probabilistic Programming & Relational Learning
Classes related to fintech, a discrete math class that is dedicated to CS (has more topics related to CS, or can easily see CS application). Also I would be interested in further theoretical computer science classes (after 181)
Computer Animation
Courses about UI/UX design, a more beginner/accessible web development course - CS 144 has a lot of prereqs to learn basic skills like HTML/CSS (maybe w/ just CS 31/32 prereqs instead of a whole chain of upper div requirements). I noticed that CS 144 isn't even offered this year :(
Mobile App Development
Blockchain technology, API design
Distributed Systems, Cloud; Web and App Development
I know there's a 188 class for this, but it'd be cool if Natural Language Processing was offered more often!
Can we get some more classes on video game related technology?
Game Engines or Game Physics Simulations

Game Development
A second class on algorithms and complexity following 180, a more rigorous version of 231, blockchain class, more ml, performance engineering
no
Game Development Classes (either creating or working with existing engines) through an engineering focus (linear algebra, physics simulations)
Game Development
Additional classes in computer graphics and its applications especially in video games. I think that's an area of computer science that hasn't been explored in the current CS curriculum at UCLA since game development contains challenges that require interdisciplinary knowledge from physics and mathematics.
Computer Vision
We need a course on video game development. As a previous student leader for UCLA ACM Game Studio, I've seen firsthand the enthusiasm that a portion of the CS community has for game dev. Right now, ACM is the only resource aspiring game developers have at UCLA, and I would love to see that change.
representation learning
green computing
EC ENGR: C147 should be added to CS, not EE department
More offerings for the 174 track, game development
I would love to see more game development related offered! The lack of game dev courses at UCLA had made me almost attend USC over UCLA.
I'd love to see some undergrad level classes on quantum computing. Even just a seminar would help
Greater focus on software engineering and less on theoretical concepts
Game-related classes, more proof based algorithm classes, actual Software Engineering focused classes
Distributed Systems
CS 134 !!!!!!!!!!! more software engineering stuff, I liked the 188 scalable internet services; more 188 classes taught by software engineers or people who founded startups. honestly any of the old 188s all sounded super cool, but I was too busy taking my required classes.. I don’t think I’m into CV or NLP
Functional programming, game development, in-depth web applications, quantum computing (undergrad), computational mathematics

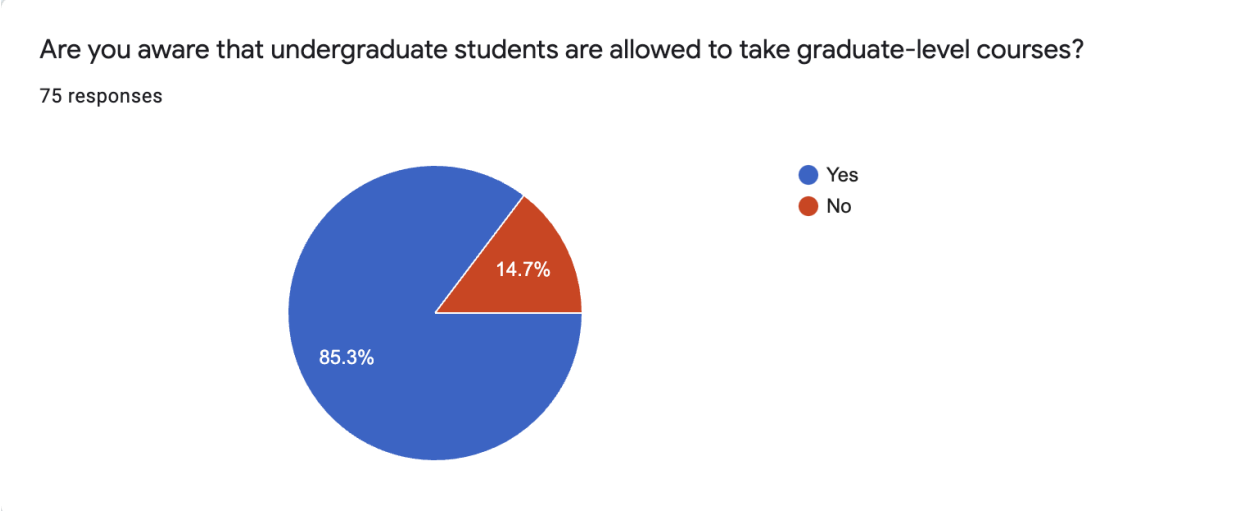
Computer Vision, Web/Mobile/Full-stack development
Operating Systems Projects/Implementation
Reinforcement Learning, Deep Learning (should really have a CS offering instead of EE147), Robotics
Computer vision
Computer Music, Physical Modeling, Debugging,
Game development, web development
More classes on computer vision, natural language processing (would be great if they were offered more frequently). The topic of virtual reality would be interesting to be added to the curriculum.
Computer gaming,
web development;
Computer vision, web applications,
Distributed Systems
I think the content of ECE C147 (deep learning) should be added to the CS curriculum as well. The content of CS 238 Quantum Programming would be nice in the undergrad curriculum. Also, robotic manipulation and deep generative models.
HCI, UI/UX, Deep Learning, VR/AR

Courses that students want to see removed

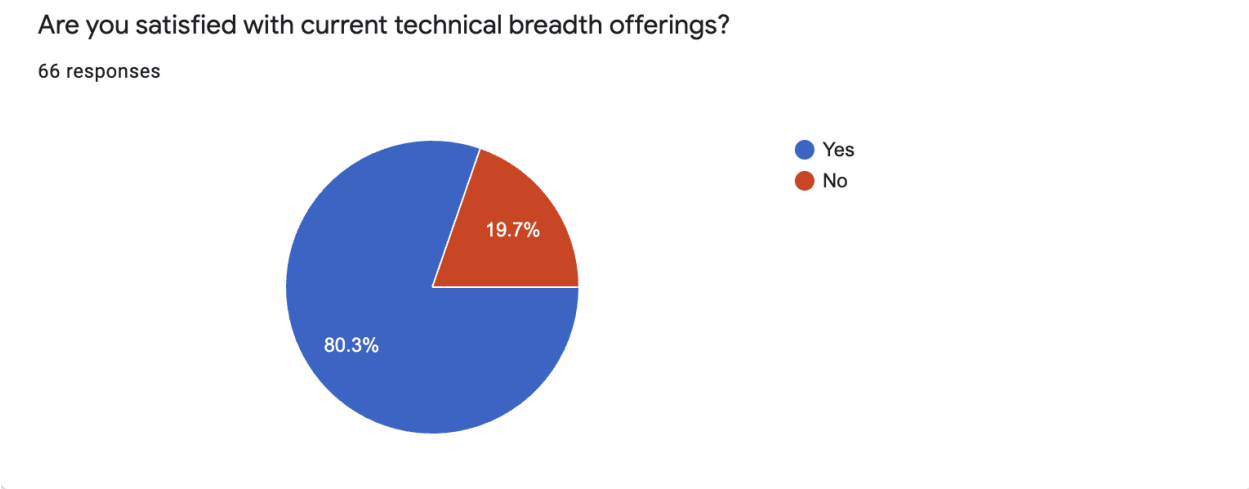
No, but in case anyone else says 131 I want to say that I think 131 is highly relevant and should NOT be dropped
Physics series, math 33b
Yes. CS M51A and CS M152A should not be required for CS majors as they are irrelevant for software work (CS M151B is enough). People who want to focus in that area can still take them if they want.
Ethics
CS97 :(
CS M152A
CS M152A
CS 1
cs m51a and onwards
M51A, M151B, M152A
CS M151B
CS35L
Engr 183EW, at least its current contents have nothing to do with its title.
181 might be good for students interested in theory, but it is too niche for all C's students to have to take, in my opinion.
CS M152A
M151B - Happy the class exists, don't think it should be a requirement
no
I feel that some of the physics requirements could be loosened, particularly the physics lab requirement.
engineering ethics
M152A M152A M152A M152A M152A M152A
Not necessarily irrelevant but I hope they can work more javascript into lower divs
M152A
CS 181 - seems completely not useful for industry/real world, simply a history of computation class, should be an elective if students are interested in computation CS M152A - should be an elective, too close to hardware/low-level to be a requirement for all students

CS 1 - Make a weekly seminar speaker series over Fall quarter with open attendance so all years can come if they are interested, and we are not required to go to seminars which we are not interested in
CS M152A (CS M51A is enough knowledge for pure CS, I don't know many CS alums who could readily do M152A things, but all remember M51A level material)
152a
I don't think 152A needs to be dropped, but I do think it needs to be significantly reworked.
M51A. Logic Design of Digital Systems, M152A. Introductory Digital Design Laboratory
None that I think are irrelevant, but I think it would be good to break up CS 35L into a couple more dedicated classes towards the subject material covered in that class
CS M152A: Major EE vibes
CS M152A, CSM151B
M152A
Digital Design
verilog
Physics 4AL!!!!!!!!! CS M152A; CS M151B
CS M152A

Are you aware that undergraduate students are allowed to take graduate-level courses?



Are you satisfied with current technical breadth offerings?

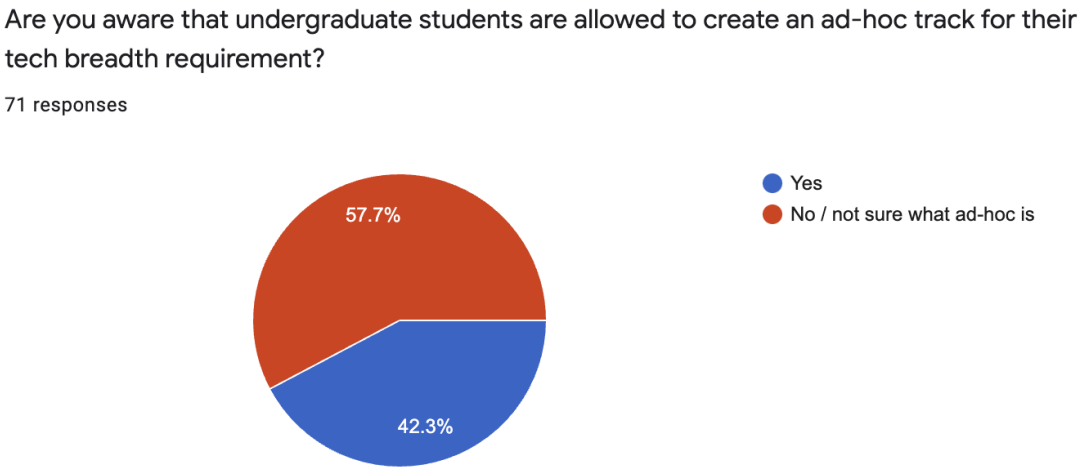


If not [satisfied with current offerings], which additional technical breadth offerings would you like to see?

Political Science
I'd rather we just excluded sci-tech courses altogether, and let CS students graduate earlier or at the very least deal with a slightly lighter courseload.

Game development classes!
DESMA used to be considered for sci-tech. For students interested in game development, removing this option has been very upsetting because the only game courses currently taught at UCLA are in DESMA.
English classes should be allowed for sci-tech
for sci-tech: English, Gender Studies, LGBTQ+ Studies
I would love if the DESMA track was brought back.
I understand DESMA was removed because of the bandwidth of that department, but it would be great if that were still possible as a tech breadth.
philosophy
Philosophy, English, Tech Ethics, more humanities in general
Statistics
Tech breadth: Computational biology (specifically the dynamic modeling track courses). Also, a tech breadth or sci-tech in quantum information could be interesting.
It's not fair and makes no sense that I can't take additional tech management classes for my sci-tech if that's my tech breadth. I'm super interested in entrepreneurship and it's strange the department is dissuading me from being able to include this learning for my degree.
Design Media Arts

Are you aware that undergraduate students are allowed to create an ad-hoc track for their tech breadth requirement?



Are you satisfied with how we teach ethics in the computer science program? Why or why not?

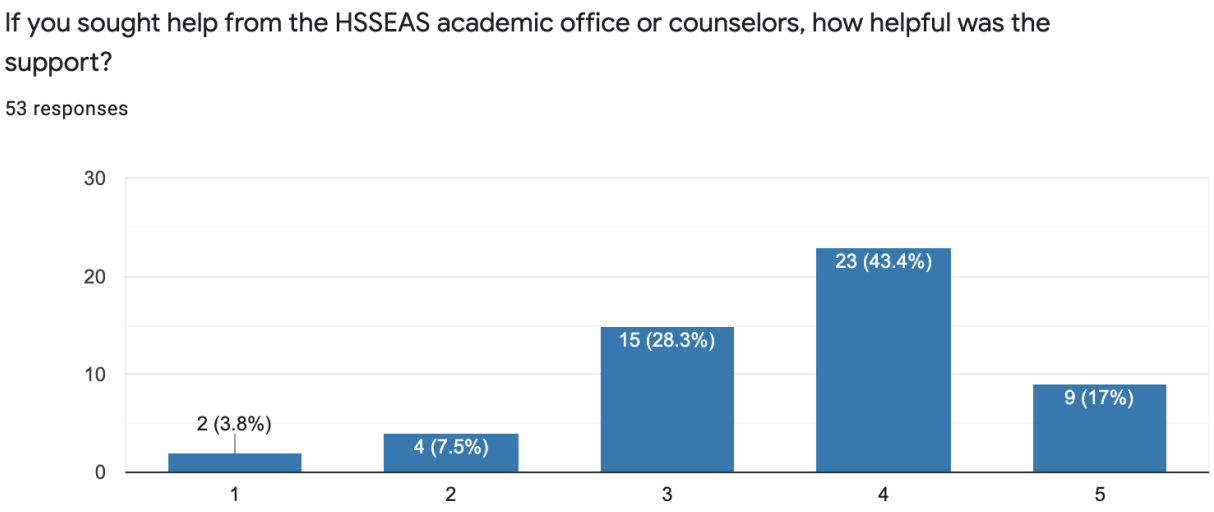
Replace physics series with more classes related to ethics
Yes, one course of ethics is enough and manageable
No, I would prefer not having the ethics requirement and working it in small portions into other classes instead
Less about writing, more about actual case studies and collaboration
I think it would be more useful if ethics was sprinkled into our classes where it can be applied so that it is more clear where ethical boundaries exist. For example in the AI class or operating systems we could talk about where ethics is important
No, no practical knowledge. No one is interested.
I wish that ethics were more integrated into all of our computer science courses instead of being separated into its own course. The separation makes it seem as if ethics is a side topic that should only be considered in parallel to software development when in reality ethical and accessibility considerations play into every component of the final application and by extension the development process. Having a writing course is good, I just wish that the discussion of ethics was integrated more thoroughly in the CS curriculum.
No, though I haven't taken the course, I haven't heard good things about it.
Ethics should not be a class that students take only once. It should be integrated into the curriculum of every CS class, where applicable. For example, Professor Sarrafzadeh talked about how data science might be used to mislead people or cause negative outcomes in the first few lectures of CS M148.
No. I think that the ethics class should be revamped and also less centered around writing. My ethics class felt like two different classes. One where we read and write and one where we look at case studies. I think the focus should definitely be solely around ethics than the writing portion.
Nope. I'm in my 3rd year of college and I haven't even had a chance to take the ethics course because it fills up so quickly. What good is an ethics course after I've already taken so many CS classes and started working? I should be wrangling with those ethical questions while I work not after the fact.
N/A - Have not taken ethics
did not take ethics yet
No. Engr 183 at least is boring and scares people away from the topics it covers.
No, I don't like how so much of the course is spent on teaching writing practices, and how early in the morning the class is.

No. There seems to be little practical application in using these ethical frameworks when everyone I know forgets they exist after taking the class. They are treated as abstract ways of thought that are not to be used in your career. The overall message of the class seems to be "be ethical" but doesn't provide relevant cases to most of our careers. Obviously, the failure of a large company to do some task is bad but it doesn't relate to our careers, where it is more likely we will see someone claiming credit for someone else's code or embezzling.
No. I took 185EW. The main thing I learned from that class was how to divide up team project work. Some stuff on avoiding killing people by reporting faulty products. Nothing on how to think about negative societal consequences of my work. Perhaps better suited to Aero/Mech/Civil than CS.
N/A -- have never taken engineering ethics
I took the new ethics course that focused on computer science topics and felt that it was a very good use of my time. While I can't speak to the other ethics courses myself, from what friends have said my class was a lot more streamlined and relevant to topics that would actually affect me as opposed to, e,g, a computer science student discussing the morals of a skyscraper.
No. It's a huge waste of time in its current iteration
I took the experimental ethics course that was geared toward Computer Science specifically, and I think that is a great improvement over the generalized engineering ethics course which has little relevance for CS students. However, I also think bundling the engineering writing requirement into the same course as ethics is a bad idea because the course tries to teach you BOTH writing techniques and ethics which are not really related. It was like two courses in one.
No, it's not engaging or relevant. We should integrate ethics into every class.
No, because the ethics we teach are strictly neoliberal (predictive policing doesn't need more diverse datasets, it needs to not exist in the first place) and Western/Capitalism-focused. Also stop putting the reform of engineering ethics on unpaid students who have enough to deal with.
No. There are too many cheaters.
No. For context, I took 183EW with Donald Browne and Gershon Weltman. The course definitely demonstrates more-so the failures of managers/executives to respond to when engineers found issues with the product, rather than failures of the engineers themselves. Also, learning ethics is essentially equivalent to learning when to tell something is "good" or "bad," which I believe at this point in life we should already be able to tell that ourselves. Either 1) stop masquerading an Ethics course as an Engineering Ethics course OR 2) Teach us ways to organize in order to stand up for what we believe is right. How should we discuss our actions with other engineer co-workers? How should we consolidate power to override our executives? How do we navigate the line between preserving ourselves within the system vs. wanting to oppose a part of the system? Maybe we could instead study engineers who were whistleblowers. What were the consequences? Did they find allies or were they shunned? What laws protect whistleblowers and how effective are they?

In conclusion, the older engineering ethics could definitely be revamped because there are so many more interesting questions we could be exploring about an engineer's social responsibility.
Yes
N/A
no, there should be a dedicated class for this.
No; although I haven't taken the class (and won't because of petitions), everyone hates the class
There should be more emphasis and explicit mentions during classes.
Yes
Yes I took 182 with Villaseñor and I thought it was interesting+eye opening.
No. I think the generic EW classes are too disjoint from computer science majors and the problems they will face in academia and in industry. The CS-specific offering of the class (which I took) still didn't reach the level of depth I was looking for; I felt like it focused too much on "obvious" ethical judgements (ex: don't lie about building a stable bridge), and less on trickier ethical situations (ex: predictive policing, data privacy). I also felt like the discussions were very poorly used, and the inclusion of Writing II is not done very productively (there is not as much focus on developing specific writing skills, and it's completely removed from the course material).
I think it would be nice if ethics were introduced in every class rather than just having one ethics-oriented class. For example, CS m146 is a great class to incorporate ethics into, but we don't cover that.
I am taking ENGR 183EW currently and I think the class has been good so far. I guess instead of doing a lot of writing, we could probably work more on problems as teams. Instead of doing ethics worksheets, I think in person team scenarios where teams of students are asked to solve problems while applying ethical principles would be much better . Basically, a more "hands-on" approach. A fun class which allows us to explore ethics and actually understand by applying them without the burden of solving worksheets and writing papers.
No, the ethics class is not very informative and does no require students to think much about impacts of the specific technologies they learn about or will be building.
Have not taken it, but have not heard good things.
No, nobody takes the requirement seriously and it's all crammed into one quarter
Not sure.
Haven't taken the ethics course yet, but have heard from others that it is uninspiring
I feel that the discussion section in ethics is much too long, and the class is more of a writing + ethics class than an ethics class.
Yes

If it were more embedded into the topics we learn rather than a separate class that'd be better. Showing the potential bad effects of technology while learning it would be more effective.
no, engr 185ew was one of the worst classes I've taken here.

If you sought help from the HSSEAS academic office or counselors, how helpful was the support?



UCLA CS compared to peer institutions

How does UCLA Computer Science compare with the curriculum offered at peer institutions?	**In particular, what is your opinion on our coverage of data science and machine learning classes in undergraduate courses?**	**How can the UCLA Computer Science department further improve the curriculum?**
	I think the coverage could be better; it seems like other universities offer more DS/ML courses as undergraduate options, while UCLA CS has 2 or maybe 3 at best (with a lot of overlap). Working with the ECE department to cross-list some of their courses like C143A and	

	C147 would be a good step, as would developing seminar courses that can be taught concurrently with grad-level courses.	
I think UCLA CS is much less rigorous than the CS programs at many other institutions. For instance, I took the equivalent of CS 31+32 at Carnegie Mellon University during one summer in high school before coming to UCLA. There we covered formally proving properties about programs, the implementation of AVL trees and union-find, the mathematical definition of big-O and proving complexity of algorithms, memory layout, and did much more programming in projects and labs compared to in CS 31+32 at UCLA (which I was required to take anyways). Also, UCLA CS lacks advanced classes in many areas of CS (taking into account graduate courses as well), including functional programming, programming language theory, operating systems, compilers, and logic.	Sufficient.	- Allow students who have enough programming and computer science experience to skip CS 31 and 32 (perhaps by offering some sort of qualifying test, even if they do not have credit for a formal course equivalent to those). I did not learn anything in those two classes. - Increase the amount of programming in assignments for most CS classes. I think there is way too little programming for most classes currently for students to adequately understand the material; the only class that is somewhat programming heavy is CS 132. Too many topics are only covered conceptually in class and not implemented. - Allow programming projects to be submitted to a grading server to be run on test cases and receive a score, as many times as students want before the due date (again, the only class I have taken that does this is CS 132). This is more reflective of real programming. This could cause scores to increase so it could be balanced out by making the programming projects harder as mentioned above.

		- Add an honors program and honors versions of courses, similar to what the EE department does, for students who want more challenging material. - Add more CS courses so that students can better specialize in certain areas of CS.
	can have machine learning in other aspect except genetic	recording lectures, sharing excellent projects in class
	Could be more	
Good	Too many of the same material being retaught	Recording lectures
I think that UC Berkeley teaches discrete math (CS70) as a cs class with content that we do no cover. I think that this content is important for algorithms and data structures.		Available grading scripts for projects, recording lectures, livestreamed lectures
	good	recording lectures, more theory classes like cryptography
UCLA CS courses tend to be more theoretical than courses taught at other institutions		
	There seems to be a lot of different data science related courses, but I am still unsure as to what the difference between all of them is ie why should I take Intro to Data Science instead of Intro to Data Mining or vice versa?	
	i feel that there aren't any courses that go beyond the level of machine learning that m146 covers, which is not a ton	recording lectures, more structured lecture

	I think we should not strive to cover more data science and machine learning in CS undergraduate courses. After all, there exist graduate classes or undergraduate classes in other departments (math, stats, ECE) that cover a wide range of topics in this fields already. More effort should be devoted to making students aware of these classes rather than just copying them over to the CS catalog. Concretely, it would be good if the CS department can more closely work with ACM AI since this committee of ACM already does so much to interface with undergraduates interested in machine learning and data science.	
		CS33 and CS M51A both teach integers and floating-point representations in the machine, but I think only one class really needs to.
Personally, I think UCLA CS is slightly better in curriculum offering because schools like Berkeley and Stanford start off with teaching Python and how to apply Python to make "cool things". Berkeley also has made several classes like OS(111) optional which is quite problematic (understanding concurrency and scheduling especially is really important)	Quite good - By the time you have taken CS 131, you will be able to learn any new programming language really easily and that includes R. We've also learned a good amount of Python by then as well and know how to use Python libraries. I also think our lower level math requirements are completely reasonable in this context as they give you sufficient knowledge to understand the theory behind ML. From	Don't reuse projects so much - we work really hard on our class projects and it's unfair that we cannot show them to potential employers to showcase our skills without the dept calling it "cheating". It would be also nice if courses provided lecture notes or make "READMEs" with somewhat of a transcript of the lecture - for some of us, reading through can help more than listening or "coding along"

	personal experience, after taking these classes and no outside training/tutorials, I started working in a lab where my duties largely entail using data science/ml approaches to solve problems. The learning curve was completely reasonable and I was able to grasp whatever I needed with 2 months time.	
	I think these needs to be more coverage of data science in the CS curriculum. Whether that means it is weaved into current classes's curriculums or there are new classes offered.	Providing grading scripts for projects would be extremely useful.
I'm unimpressed. Stanford has an iOS development course that is incredibly popular, and we have no equivalent to that. USC has game design courses, and again we have no equivalent to that (or at least not one that specifically entails to that genre).	Two years in, haven't had a chance to take a single one. Why aren't these classes mandatory instead of some of the more pointless low level ones, especially given the direction software and the world are moving in.	Recording lectures is incredibly helpful, I hope that continues. Recording one discussion section too is very helpful. Sticking to timed, open-book tests that were implemented during covid, as they were much better tests of knowledge and required just as much effort on the professor's part as our own. Prof. Eggert actually did a great job of that in CS 97 in the winter.
I would say that UCLA computer science spends too much time on theory rather than hands-on practice. The Berkeley curriculum for CS tends to be a lot more hands-on and even for their first-year classes, because they spend so much more time on hands-on application and what exists in the real world, they are able to cover much	N/A - Have not taken any data science or ML courses yet	Use Git rather than zip files for uploading code! Use GitHub organizations to track who pushed code and for students to see what commits were made. Use Gradescope for grading assignments and to be able to give feedback on written assignments/hoemwork.

deeper, much more practical topics in computer science before UCLA students cover them. I would say that UCLA CS curriculum is also somewhat outdated: for example, Stanford teaches JavaScript for their intro to CS courses and I believe that teaching JavaScript or Python in CS31 and CS32 would be a more practical and pragmatic exercise than teaching C++ which is no longer as mainstream or widely used as JavaScript/Python/Java in building general purpose software.		
	I took both M146 and M148. I think these two classes went very well together, since m146 gave me the mathematical foundation needed to really do well in m148, where the implementation was the main focus. I think before m148 was added as an official class, I thought the coverage was lackluster.	
		Available grading scripts for projects
IDK		Please add something related to graphics, physics simulation, or game engines in general. These are complicated enough to be taught in class and useful enough to either push technological development or help students land a job in the industry.
USC has several 2d/3d interactive application courses,		

such as ITP-380 and ITP-485. Of the ~900 CS and CS-Games majors at USC, 65 are taking ITP-380 this semester. Because of the limited graphics-centered courses offered at UCLA, courses similar to these would contribute greatly to UCLA's CS electives, and it would help broaden the scope of the CS program here.		
I think the UCLA computer science curriculum is lacking courses covering subjects related to game development. It would be great if the department could offer a course similar to CS 113 Computer Game Development at UC Irvine, or any of the 17 game-related courses in the computer science department at USC.	I think the machine learning side has really good coverage, but based on my internship experience I think more emphasis on the data science part could be helpful for students looking to work in industry. In particular, the data science courses could focus more on data engineering, extracting useful information from "messy" data, and augmenting data with additional data sets. This would also help differentiate the various data science and machine learning courses a bit more since there would be a bit less overlap in topics covered.	
UCSD offers much more computer vision and graphics classes to their undergraduates, while our computer vision class is still experimental. There are two graphics classes now, better than last year but a far cry from UCSD's many courses.	We need more computer vision courses to be competitive in the job market.	Create basic testing scripts for students so they can check if they are on the right track.
	I'm not very happy with M146 and I've heard 145 and M148 are	

	very similar	
UCI and USC offer much stronger game development programs through their CS departments. I believe that, with video games rising as an industry and many game dev. giants like Riot Games close to campus, UCLA should also be developing a competitive program to feed into this growing industry.	Concepts are too similar between 145, 146, 148.	Add game development classes.
One of my largest gripes with the UCLA Computer Science curriculum was its lack of focus on content geared towards computer graphics and game development that would be relevant for a job in such a field. The game development experience I was able to gain was entirely through student organizations, which while engaging and enjoyable, were not a suitable replacement for a rigorously designed coursework. I am aware that other institutions, such as USC, have entire departments dedicated to game development, and having been part of ACM Studio for 3 three years, I can say that there is an equally large community at UCLA that would love to take advantage of such resources.		
Significantly audit all classes to make sure at least some up to date material is covered	I really like what Professor Majid and Mirzasoleman are doing with M148. It's taught in a very accessible fashion and is a great entry way into the world of ML	More transparent grading. I know Cho and Palsberg publish their autograder

	and data science.	
USC offers game development courses and an entire game development minor. Students like me who are exclusively interested in entering the games industry are at a huge disadvantage at UCLA where we have no access to game dev courses in the curriculum.	I do not have strong opinions about it.	I think Covid showed a lot of us that recorded lectures are a convenience that should be the norm. I also would like to see UCLA implement a few more self-guided projects where students can make something of their own choosing, outside of just CS 130. A big strength of CS 174A in my opinion was a project where students had to make the graphics for a game or video of their own creation. At Berkeley, students work on apps of their own design even in introductory classes, and at UCLA students have to turn to ACM and other extracurriculars for their more practical self-guided projects.
I like how theoretical it is.	I think the ML classes could be more theoretical and proof-heavy.	make grading scripts available for projects
i saw a school (i forgot which one L) that offered data science in the context of afrofeminism. I would probably be dead before UCLA even considered offering that class in the engineering school (although it would definitely be offered in L&S).		
	We need more machine learning classes.	Reducing overlap between classes.
	I wish to learn more about machine learning I think we need more.	Reduce overlap and probably different pre reqs for 35L if it is not changed, also allow for curves and make it more accessible for those who did not previously have a background in

		CS and want to start learning
	I took the Intro to Machine Learning class but I don't think I learned anything. It would be more interesting if there was a project-based class that helped us apply the concepts to a project of our own design?	- grading scripts for projects would be nice - recording lectures is great; when it has auto-cc it's even better - more project-based classes that teach industry skills?? - 2-unit technical interview prep crash course?????
USC has a premier game development and game design tech breadth program that I think I would have benefited from majorly. Despite being able to work in the game industry, at times I still feel vastly under-equipped in terms of hard skills, like proprietary engines and what core topics to study, and all of the progress I have made is by myself or with student-led organizations.		
There aren't as many game development classes at all		Continuing to record lectures would be super helpful
I've heard that Berkeley has much more CV/ML course offerings than we have.	We should have more breadth of coverage on ML topics instead of just 1 algorithms course and 1 data science course that have large amounts of overlap in contents.	For specifically recorded lectures - Implement Dr. Palsberg's method of having a TA/student dedicated to reading the Zoom chat for questions. Many times in other classes the professor is not able to both read chat and cover material, and will end up ignoring/disabling chat, which is detrimental to students who ask questions through there.
	I am a Ling and CS major and we have not touched anything relating to machine learning.	

	not enough data science required for ling/cs	
Although it's a weeder class at Berkeley, their EECS16 series provides lower-div exposure to building hardware. It's required for both EECS and CS majors. Our closest equivalent I'd say is M51A, but the course is based on pure design rather than implementation. I've heard that we do more hardware stuff in our 151 sequence, but that's an upper-div which most people don't take until late sophomore or junior year, while my friends at Berkeley are taking EECS16 during their first semester of sophomore year.	Have not taken them	As a very vague statement, introduce topics in a way that promotes the mindset that they're good in themselves rather than promoting the mindset of how useful it is with respect to SWE and industry
USC provides Game Making classes. Game companies would value this over a pure CS degree with no exposure.	I'm not interested in these topics but I know people would be more interested in having more, especially data science.	Available grading scripts for projects (not just a basic tester, the FULL tests should be available to students. This is supposed to encourage learning and understanding) Make hardware classes more high level. ALL LECTURES SHOULD BE RECORDED GOING FORWARD. A lot of professors should learn from Smallberg's online exams and Eyolfson's exams for testing.
I think it’s lacking compared to institutions like Berkeley, CMU, MIT and Stanford. When planning courses, I noticed a much smaller emphasis on distributed systems and infrastructure related topics	Sufficient	Have more undergraduate distributed systems classes

compared to theirs.		
We don’t have distributed systems class this year	I think there is a lot of coverage; we have Intro to Data Science and Intro to ML and then data mining	Recording all lectures!!! Recorded lectures have helped so much with my time management+being able to follow along in lecture, slow down when I need to take my time to digest the material, speed up when the material is review so I don’t lose focus. I also really like take home exams because it’s a lot less anxiety inducing; I don’t have to worry about external factors like getting a bad middle seat in Dodd where the desk is slanted forward so my exam is falling off the desk, the student next to me is left handed so they’re basically taking their exam in my lap, and it’s an open note exam so I have 45 pages of notes stuffed under my elbow, feeling really claustrophobic... Caltech has all take home exams, and I feel like it would allow us to actually do our best on the exams and use them as a learning experience rather than an anxiety inducing one.
I would love to see more student-led classes like UC Berkeley's DeCal program. UISE is not sufficient (since it doesn't allow enforced prereqs). Other colleges (ex Stanford, MIT, UC Berkeley) are more aggressive with offering experimental classes (their	I think there should be a clearer path of what classes to take when. CS 145 and M146 have too much overlap in their content (about 6 weeks), and my peers have complained that Math 33A does not adequately prepare them for the class. We should be cross-listing more advanced classes in the EE department,	Available grading scripts for projects is a great idea. Palsberg and Cho doing it in their classes makes the class much more enjoyable and grading more fair. Course materials and websites should be more accessible to those with visual impairments. This includes screenreader

version of CS 188), often bringing in lecturing professors. I'd like to see more of that on topics that are popular but not necessarily research interests of UCLA profs (ex game development, differential privacy, computational modelling).	and working more closely with the math department to make our offerings consistent. I think CS M148 is a great step in the right direction!	accessibility, higher contrast for colorblind students, and when possible, recording lectures. I would like to see more collaboration with other departments; joint classes between CS and the Info Studies, Math, Public Policy, etc. departments is a much better use of UCLA as an academic institution!
		Why are most CS class lectures 2 hours? They are so dense that it is hard to retain everything, I would do much better with more frequent 50 minutes or 1 hours lectures. I also feel that all lectures should be recorded--it makes things more accessible. Why do CS majors have to take so many physics classes? It makes no sense at all. There are also a lot of low-level computer engineering-like classes which don't seem relevant for CS majors, but more so for CE majors. Also, the CS curriculum is very theoretical at UCLA--I would love to have more practical classes which would better prepare me for industry.
	I have yet to take the courses, but from what I heard, CS M146 is very mathematical in nature. I feel like project-based courses for Machine Learning and Data Science would be great, but I am happy that UCLA does have these classes offered, even if	Recording all lectures would definitely be great! In addition, reducing overlap would be good as well, particularly with discussion sections.

	maybe there were things to improve upon.	
We can have our Discrete Math and Probability requirements offered as CS courses taught by computer scientists from the CS department instead of other departments. I am comparing this with UC Berkeley who have CS 70 to teach discrete math and probability and Stanford has CS 103 and CS 109 which cover discrete math and probability respectively. I guess it would help create a more tightly knit CS community and allow us to tailor the courses to make them most relevant for CS topics. CMU has 15-410 which is Operating System Design and Implementation, which gives crucial project experience in systems programming which is hard to learn on our own compared to stuff like game development or machine learning for which there are clubs. Would love to see a CS 111 "part 2" where all we do for a quarter is a project implementing an OS or parts of it. CS 132 does this pretty well for compilers by the way. Also, please bring back CS 134: Distributed Systems! Very interesting topic and course, which seems to not be offered anymore sadly :(Also, can we revise the Math requirements to not require		Create a separate School of Computer Science which runs on a semester system. Now that would help us go next level.

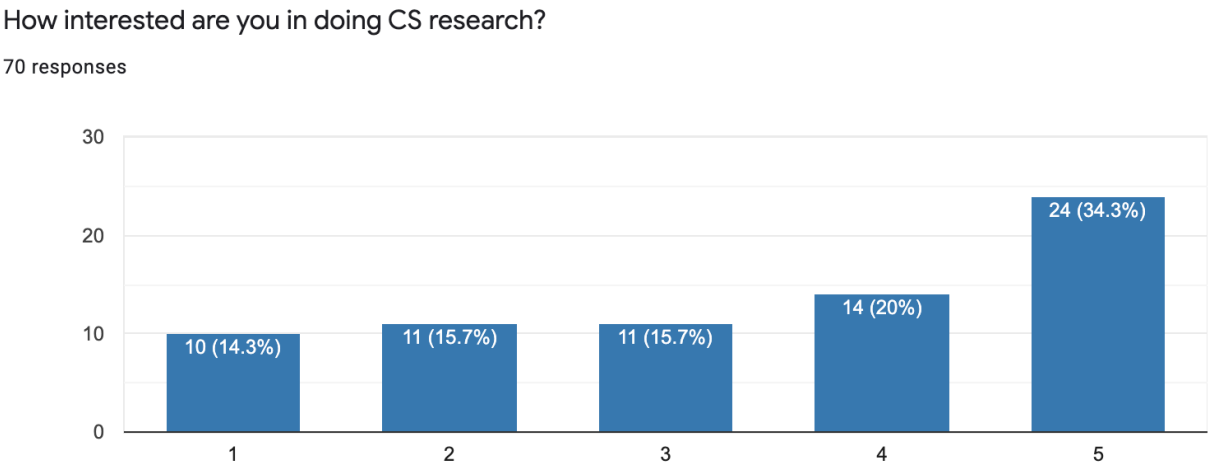
Math 32B and 33B of every CS major? UCSD has done away with those requirements for CS majors, and they can be taken by those CS majors who intend to use those topics in their upper divs. We could better spend that time learning logic, more discrete math or more from the CS 33 topics or even just another elective.		
Stanford, Berkeley, and MIT all have much richer choices of electives.	Current coverage is poor. We are missing permanent NLP, Reinforcement Learning, Unsupervised Learning, Roboitcs, and more courses.	
	Needs to be a little more unified	
	We should cover it more early on. We in general, I feel like our projects for lower division classes can be revamped to be more sampler-esque.	
		One thing I really enjoy about CS 161 with Professor Van den Broeck is how he uploaded all of his Zoom lectures from a previous quarter. This gives a lot more flexibility in my schedule, as I can go to class or decide to watch the appropriate recorded lecture on my own time.
	I would love more classes dedicated to ML	Recording lectures + discussions, available grading scripts for projects
UCLA CS curriculum is overall more theoretical. A lot of other schools will have more application and programming		Recording lectures and reducing overlap with other classes will be very useful. More applications for courses such as

involved. For example, UC Berkeley has a "Full Stack Deep Learning" course with a group final project at the end, which is useful since it provides a comprehensive introduction to integrating the stack with artificial intelligence while providing practice and freedom for student's to explore other topics when generating their final project idea.		CS180 or ones more theoretically based.
	I took Data Science class. It was really helpful understanding the myth behind the ML/DS field. (and to realize I wasn't that interested in them)	cs143 auto grader is really convenient. Other classes should have that as well so students won't be confused about edge cases. More practical projects, less theoretical exams. I think most engineers and CS learn by examples and mistakes while doing but not just hearing from lectures.
		Grading scripts for projects, recording lectures always.
	Pretty good!	alter physics requirements
		Having recorded lectures has been very helpful
UIUC integrates more recent technologies and applications into their introductory computer science classes. One example that stood out to me was that one of the projects in their CS32 equivalent was based on simple handwriting symbol detection. I feel that classes like M146, M148, and the 188 series really allow students to get	Having audited a few classes from both, I really enjoy M146 and M148 but wish there was a way to incorporate those concepts more into the core curriculum instead of having them be an elective focus	

excited about real-world applications like those mentioned above, but it would make the curriculum so much richer overall if projects in core classes (e.g. 32, 111) also touched on more recent interesting applications.		
	I feel our coverage is quite good and broad. I've taken 145, M146, and M148 and received A's in all of them. I also know we now have NLP and Computer Vision, which means we have 5 data science/machine learning courses, which is more than ample enough .	Available grading scripts/test cases for projects Recording lectures
I believe Stanford and UC Berkeley offer a more comprehensive set of deep learning / machine learning courses.	I believe they are insufficient for what I wanted to learn, and I frequently am going outside of class to get information in machine learning / deep learning. Data science is ok I think.	I think I would internalize class concepts better if there were more opportunities to generate or interact with the systems that we are studying. For example, making more circuits in M51A, analyzing simple networks on the command line in 118, and writing algorithms in 180. There are sometimes big projects in these courses, but it would be nice if the homeworks incorporated more practice of these skills.
		Limit hours of homework every week. Professors seem to have the mentality of "well I've been through it so they should have to as well," this greatly increases the anxiety and depression levels among students who have to undertake the work.

	I think the addition of 148 is a big improvement. I don't know how I feel about 145 and 146, they seem to overlap in a lot of material. I'd like to see a non graduate level deep learning class offered by the CS department itself, as well as more classes on cutting edge ML research like graph neural networks, vision transformers etc.	standardized requirements for project specs

CS Research



If you are already involved in CS Research, how did you find this opportunity? If you are not yet involved but are interested, what is the primary roadblock?
I'm not sure where to start. It's intimidating to cold-email a professor or graduate student at a lab when I feel like my knowledge as an ungrad just doesn't measure up, but I don't see any listings for positions specifically for undergrads, so I don't know how else to get involved.
Research portal lists few opportunities, not hearing back from profs
I emailed a professor.

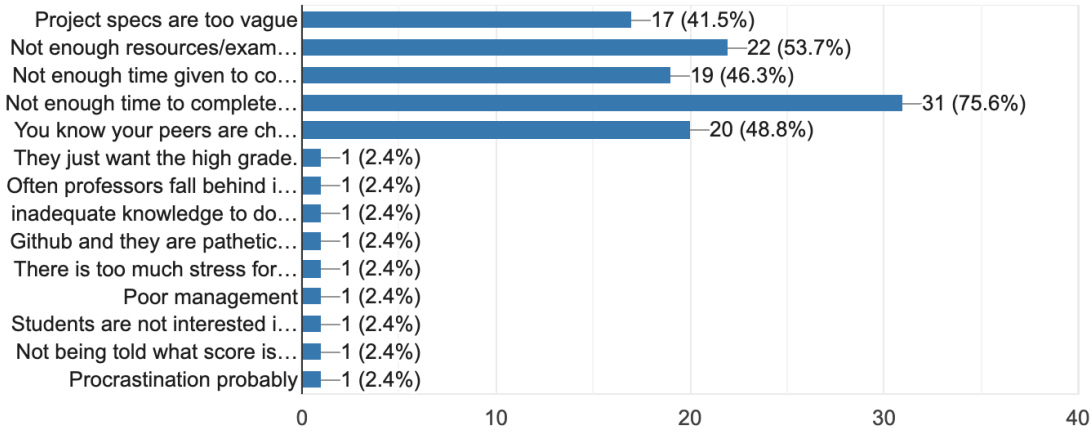
Finding avenues in to meet profs
Professors don’t want me
Reached out directly to professor
Lack of undergraduate openings.
I am developing technical solutions for UCLA Health but am not part of a formal research lab. I want to try joining a lab at some point, but am unsure as to what all the labs are, which one take undergraduates, what they do, etc.
i knew an upperclassman who introduced me to the lab she was working in
I emailed people in the CS department to ask for research opportunities.
I think the primary roadblock is just experience, especially early on in the major.
I have no idea how to get into it, and the corporate world of internships just seemed more appealing (and paying) so I haven't worked to rectify that.
I found it by cold emailing professsors until I found one that was intereste din taking me into the lab.
I don't know where to get started. Also, because most of my peers were focused on going into industry, I hadn't even considered the research path until now (I am a third year). Feeling "late" to research makes it harder to get started.
Not enough time
seems I should take some more relevant classes before getting into it
n/a
through CS 1
internet research initiative - howevver i want to get involved in research that combines CS and other disciplines but barely anything in the CS dept offers this
I joined a biology lab that uses computer science for research.
´The main roadblock with everything in my life: I don't know where to start and the activation energy for the chemical reaction that leads me to start is extremely high
Time, approachability of professors I've never met
Undergraduate research portal. However, I wanted to find distributed systems research, but can’t find any even with outside googling around.
too busy :/ and it's hard to find what research opportunities are available, lab websites, research portal, etc. are all outdated!
I found the opportunity through a lot of google searches.
A lack of guidance on how to apply to CS research labs.
Emailing professors

The roadblock is being able to talk to some of these professors since they teach upper division courses not all undergraduates can enroll in early on.
I don't think I'm qualified
I don't have a lot of experience right now, but am interested in finding new opportunities to learn more
Although I am very interested in research, most of the areas I'm interested in happen to be in ECE - so not necessarily a problem with CS research per se

Academic Honesty

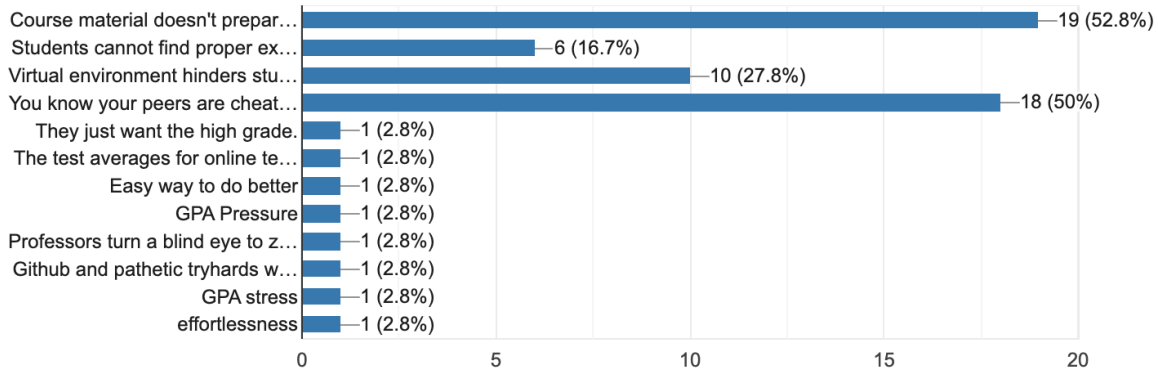
What do you think prompts students to cheat on assignments/projects?

41 responses



What do you think prompts students to cheat?

36 responses



Suggestions on how the CS department can mitigate cheating

If you have any other thoughts about cheating at UCLA CS or suggestions on how the CS department can mitigate cheating, please leave them here.
Online proctoring solutions quite frankly suck, and don't disincentivize cheating. Online non-proctored (or Zoom-proctored) tests are fine if they're open note, but proctored tests should be in-person.

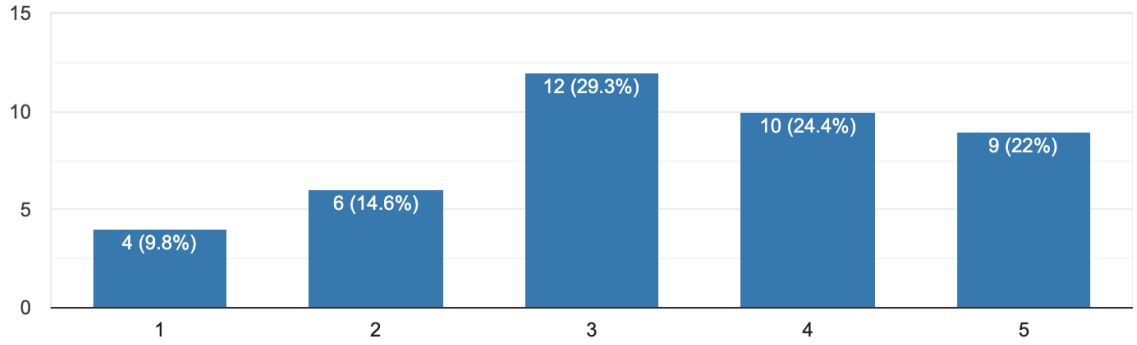
Don't reuse the same assignments every quarter, and make assignments and exams harder
The culture of cheating sucks and is part of what makes UCLA a second rate CS school (below Berkeley/Stanford/MIT/CMU) in my opinion
while this requires effort in the part of the professor, it would be great to have fresh exam/assignment questions every time. i understand it might take a lot of effort to do this, but it eliminates the possibility of students cheating. also, when it comes to exams, it then gets rid of any unfair advantage some students might have from access to old exams.
I personally dont think cheating can be eradicated having experienced online school
Prof. Cho's projects/specs are a good example of a reasonable assignment that can be done without 90% of the students using GitHub. They include clear directions, specific links to tutorials, reasonable expected work, and good connection to course content
Knowing my peers are cheating on exams is a big factor in curved classes. Putting honest students at a direct disadvantage seems very unfair. Sometimes, the professor isn't clear about the curving of the class, which prompts more people to cheat as well.
I don't hear about people cheating on exams, just the projects because of the heavy workload.
Cheating is rampant in the UCLA CS community, and far too often people discuss it as if the people who are cheating are morally corrupt or hindering their own learning by doing so. The truth is, the CS curriculum is so demanding that cheating becomes a necessity for many if not most students. Each time I cheated, I would look at an online posting of a previous student's solution in order to help better understand a given homework assignment and one of its possible solutions. Cheating like this could save me anywhere from one hour to an entire day, and despite saving time like that I still never felt I had enough time at any point throughout college. The amount of time that professors demand from their students in completing coding assignments is frankly unacceptable in my eyes, and that will need to change before we see any improvement with CS academic dishonesty.
People who copy code from github for projects should be publicly humiliated in front of their parents, and then expelled.
Please give students time and respect their mental health. Also giving assignments so students can practice what they actually learn in class could help instead of giving hard assignments where we have never learned or talked about how to do it.
Regarding "How much do you think your professors have considered your mental well-being," I think having a hard policy on not allowing any late assignments is a sign of not considering mental well-being. I appreciate Eggert's late policy. I appreciate when profs say "Hey you can turn stuff in late, just notify us and we can work something out. You don't need to explain" because sometimes you're just In The Thick Of It and it's painful to explain. Obviously on the opposite side is the TAs/graders/profs who have to deal with late assignments. (how about we abolish the idea of grades whatsoever)
Placing such a large emphasis on projects massively encourages cheating. Perhaps each project

could be broken down into modules which are then tested and graded as smaller things(eg: The game we make in CS 32 could be broken down into components such as rendering, physics, etc.)? You could still collect the project at the same time, but I think that students really cheat because they don't want to get a failing grade on a large assignment.
dont give assignments on random things we did not learn @eggert
Make project requirements more reasonable and based off of what was covered in class.
I've never even thought about cheating in classes where the projects are interesting to me, or if I can at least see the end-goal. When projects become outdated, tedious, and confusing, the temptation to cheat grows
My personal reasons for cheating (only ever on assignments) have been lack of time and/or how to start the project was never discussed in class. I believe the biggest culprits of this are 111 and the old 35L, where I understood the theory behind the project, but had no idea how to start or translate the abstract ideas I learned in class into code. For 131, I believe that the class goes much too fast-paced (learning a new language every ~2 weeks is quite ridiculous to me), especially since the projects are not simple and would be fairly difficult in other languages. For 131, I believe the class should be slower, project difficulty lowered, and/or having more of a difficulty buildup for learning each new language. On a personal level, I'm morally fine with cheating on homework, but am not and have never cheated on a test. To me, homework and projects are to help aid and reinforce the student's understanding of the subject, and tests are meant to test them. Thus, I'd much rather cheat on a homework and understand the material (thus achieving the goal of the homework) rather than be completely lost, fail the homework, and not understand the material (opposite goal of homework). To mitigate this cheating from my ethical standpoint, professors should: lower the weight of homework, make homework only based on completion, and/or release the solutions for the homework after the due date (or even before! if a student uses the solutions and thus understands the homework, didn't the homework achieve its goal?).

Diversity and Inclusion

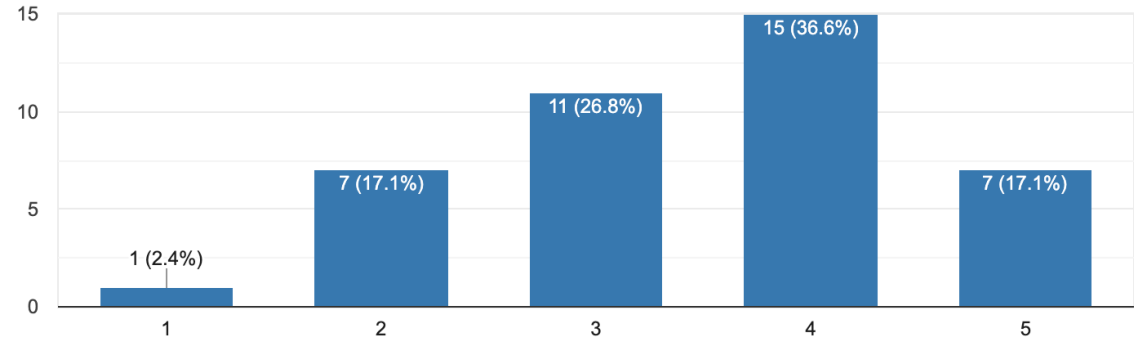
Do you feel like you belong in the CS major?

41 responses



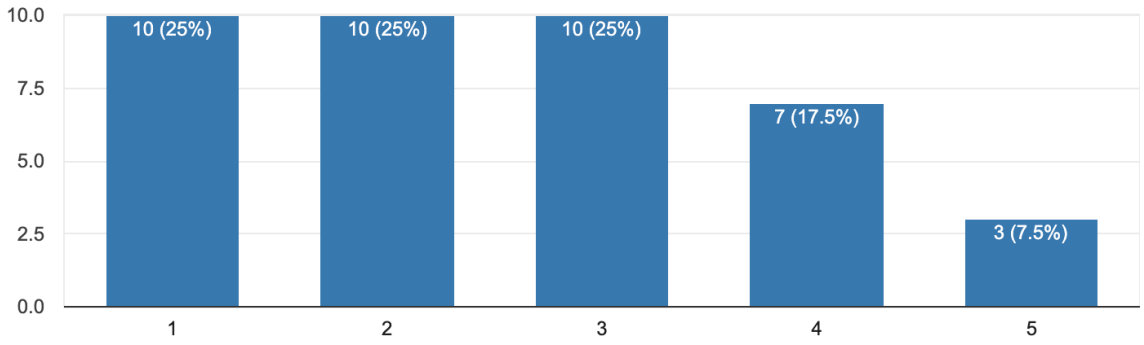
Do you feel like your professors, TAs, and classes are inclusive?

41 responses



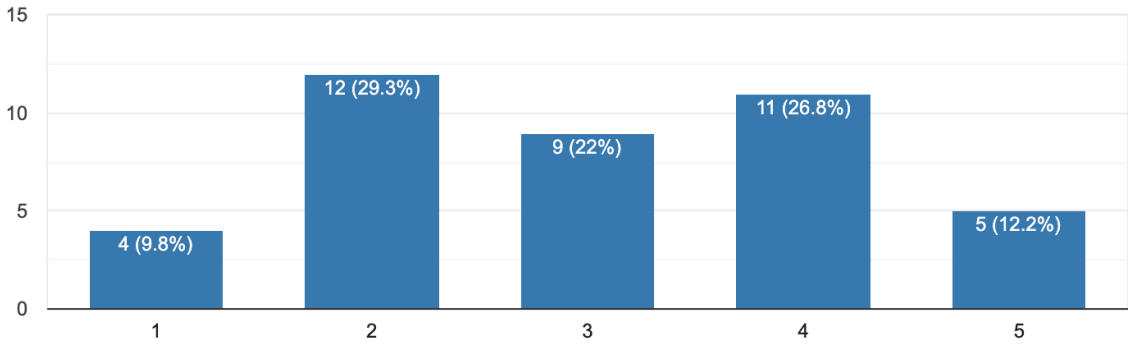
Is the CS major is welcoming to minorities?

40 responses



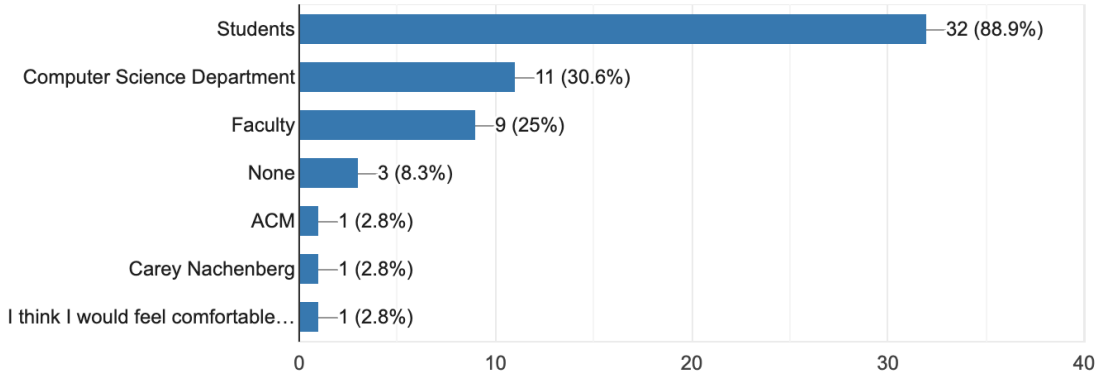
Do you feel supported in UCLA CS?

41 responses



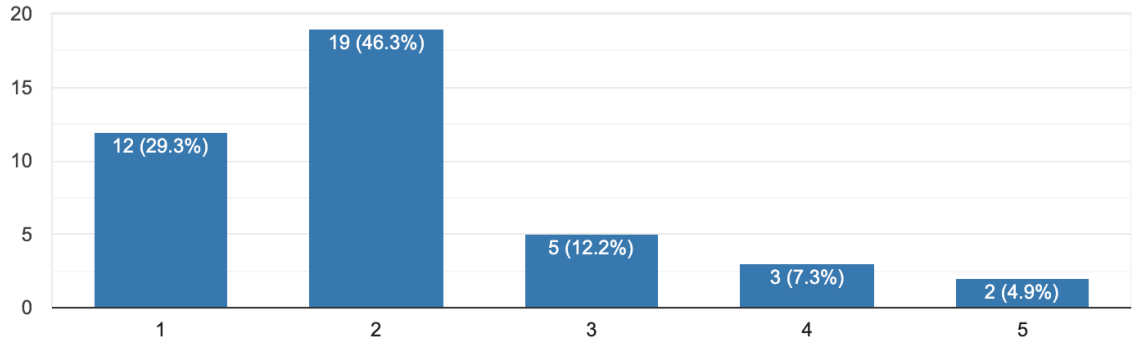
Who do you feel comfortable raising concerns to about diversity and inclusion?

36 responses



Do you feel that the image of the UCLA CS community is representative of all students, regardless of background, experience, etc?

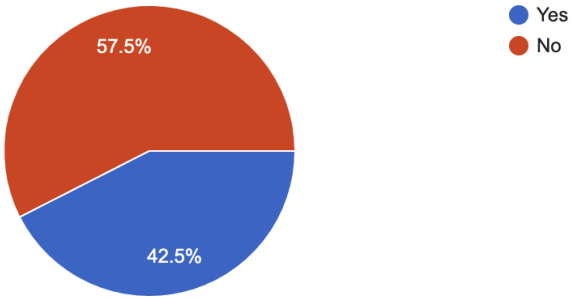
41 responses



Implicit bias in the CS community

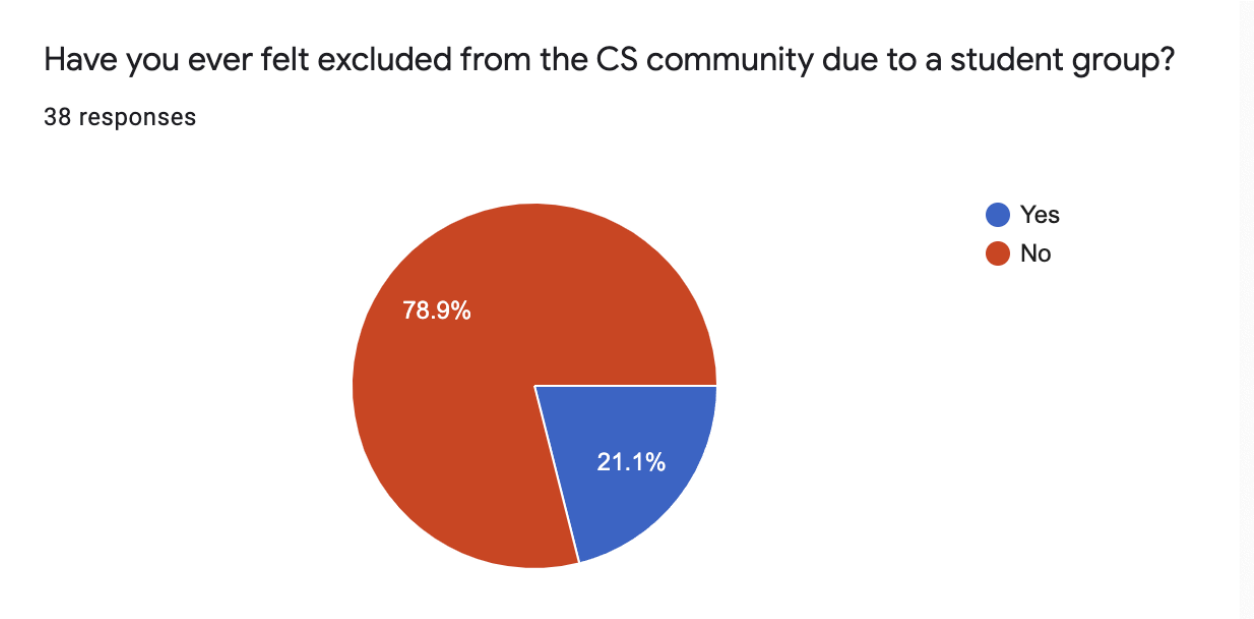
Have you ever experienced implicit bias from the UCLA CS community?

40 responses



If you answered yes, please share your experience with implicit bias, if you are comfortable with doing so
I have experienced implicit bias on multiple occasions from professors and TAs in my CS courses at UCLA. These incidents have been based on my gender. I have been talked down to and treated differently than my male counterparts in multiple instances.
As a female student, I get treated very differently from my male peers, both from professors and other students. They assume a lower level of competence from me. In several group projects, I have been consistently talked over or my ideas ignored. In general, professors tend to refer to any generic student as male. Sometimes, they remember to add a "or she" to a usage of "he" with a statement on how it is possible for anyone to be a generic student. It's not great to see a female name pop up once on a test when the rest of the example programmer names are male. Female students are not seen as a default possibility.
being talked over / ignored in a group setting
stable matching in 180
Male students and professors are less respectful to female and non-binary students.
Some male students have expressed that they believe women in CS are naturally inferior. A male CS student, unaware that I am in CS, told me they thought women were handed internships at big tech companies simply because they are women. This discounts the work I put in to earn my internship offers and makes me believe that I did not earn the internship purely off of my own merits.

Implicit bias in the clubs

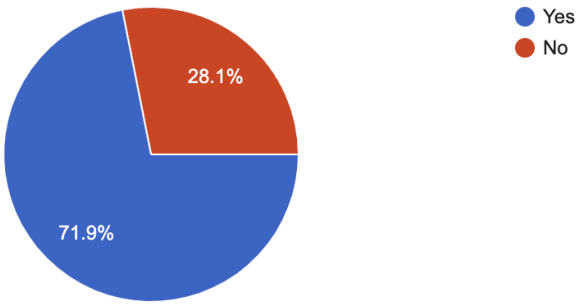


If you answered yes above, please share your experience with feelings of exclusion, if you feel comfortable doing so
some committees in acm are pretty intimidating. other cs clubs like devx make me wonder what im doing with my life
ACM did not make me feel welcome as a low income student
IEEE
There are cults, aka "fraternities/sororities" that waste people's time and energy.
Some groups take pride in rejecting many club members/being an "elite" community. I think this is an uninclusive way of running a club. Prime example is DevX!

Feeling welcomed because of clubs

Have you ever felt particularly welcomed into the CS community due to a student group?

32 responses



If answered yes above, please share your experience of feeling welcomed, if you feel comfortable doing so
In ACM they made a point to give all minorities in CS an interview, regardless of previous experience, which I appreciated. It allowed us to show our current skills, whereas I feel a lot of one's success in CS is usually contingent on things that you studied, worked on, or participated in in before college, either directly or indirectly.
despite some committees in acm being intimidating, i LOVE acm and i am so glad i came to ucla because of it
ACM
Society of Women Engineers
ACM is very welcoming and friendly
ACM's gay events <3
Learning Assistant Program
Bruinwalk was really inviting to me as a freshman
ACM is welcoming of all backgrounds.
ACM - it has been a great way to meet other people in the CS community, and learn about different fields that I hadn't heard about before! Creative Labs - I like how they take interdisciplinary approaches to tech seriously, and it's good to meet people outside the CS major!
ACM is very welcoming.
ACM
Mainly extra curricular clubs that are very welcoming. The no judgement towards people with less

experience, the help provided, and the kind attitude.

Specific moments of feeling excluded

Are there any other specific moments where you felt excluded? If you feel comfortable sharing, tell us about what happened
Nothing specific can be done to address this, but a clear gender imbalance/being only 1-2 females in a room can feel really really really overwhelming and prevent an individual from enjoying an event to the fullest
Many cs/tech student orgs on campus are very industry-focused, which makes sense, but as someone who wanted to pursue grad school/research, this continuously created a sense of self-doubt from the lack of relatability.
There should be a crackdown on cults.
I think the specific moment is when I look around my 200+ lectures and there is no one else that looks like me or when I look at the professors for my classes and TAs and I dont see any that look like me. The field as a whole needs revamping though.
I feel unsafe with Professor Gafni, a racist and misogynistic person (this isn't for debate, there are more than enough evidence points to back this claim up), leading the department. It's bad for the image of the CS department. I don't feel like I belong as a result.
Gafni's comments made me feel excluded as a Chinese-American.
There's a lot of elitism around how smart you are. People have looked down on me when I've asked for help or expressed confusion around me. It wasn't just that they said no to helping, but they acted like I was some weirdo for needing help.

Shoutouts to people that included you

On a more wholesome note: is there a specific person(s) (a professor, TA/LA, student leader, peer) who has made you feel particularly included? What did they do?
Matt Wang! (President of ACM.) He always makes it clear how important diversity and inclusion is to him, both in words and policy, so I trust that he is working on systemic changes.
Yes! George Varghese, John Cho, & Pradeep Dogga. They were extremely inclusive and supportive while taking their classes. They really value mental health and inclusion and it shows!
My 181 TA, Hadley Black, really helped me in trying to pursue grad school. He was very approachable and understanding of my situation, and I felt comfortable asking for help.
John Cho made me feel very included in CS 143, he took student feedback very seriously
Carey Nachenberg has made me feel more at home as a queer student in computer science. I attended QWER Hacks a few years ago and even though he had to call in sick for his keynote speech, we still got to see his slides and it felt good to see someone who I already looked up to owning his identity as an LGBTQ+ computer scientist. Also, shout out to the students who organized QWER Hacks for giving us a space like that.
Kai-Wei Chang has a very inclusive and supportive lab!
Shoutout Reinman for being the only professor I saw to say Black Lives Matter!
Dr. Eggert and Dr. Yang have been phenomenal.
I just love Carey Nachenberg for making CS 32 fun, which is probably a way to be inclusive
Tim Hunter, professor of Ling185A
Carey brought lots of people from varying backgrounds in during his lectures, which gave a nice tone to the class
Smallberg and Carey are quite possibly the greatest people in the CS department. I want to shout out Jon Eyolfson as well, although he isn't part of the CS department officially. He's such an amazing instructor as well and makes me feel included in the class.
Miryung Kim has been so welcoming to students, especially in her advising sections and for inviting international students to thanksgiving with her! Plus, she's just so nice, and it's very refreshing (especially since other professors can be ... distant). I really look up to Arjun, Sharvani, and Yvonne in ACM. They all did a great job of making diversity and inclusion a focus for the club.
Carey Nachenberg, John Cho
A lot of people especially the officers of UPE and leaders of Bruinwalk have made me feel extremely included. Their sincere attitude to help students, and encouragement is very appreciated coming from a student who did not have a strong background in CS before college.

Carey Nachenberg does a wonderful job of making sure women's voices are heard during his lectures! He makes an effort to use varied pronouns when using examples instead of "defaulting" to just using he/him pronouns.

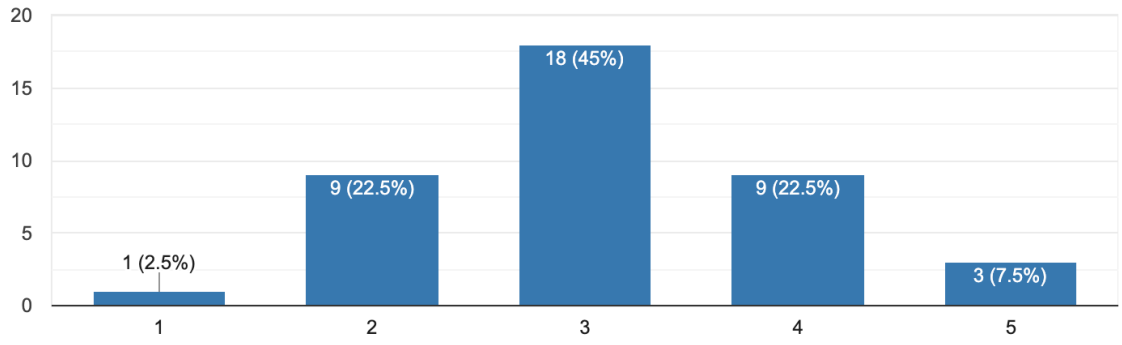
Specific moments of feeling included

Are there any other specific moments where you felt included? Tell us what happened!
When professors (I remember examples from Carey, Smallberg, and Sarrafzadeh, my STATS 100 prof) sometimes describe an example person/computer scientist during lecture and use "she" or "them" instead of "he," I am always slightly surprised because male is usually the default. Waiting for the day when we get an example person who uses neopronouns~

Imposter Syndrome

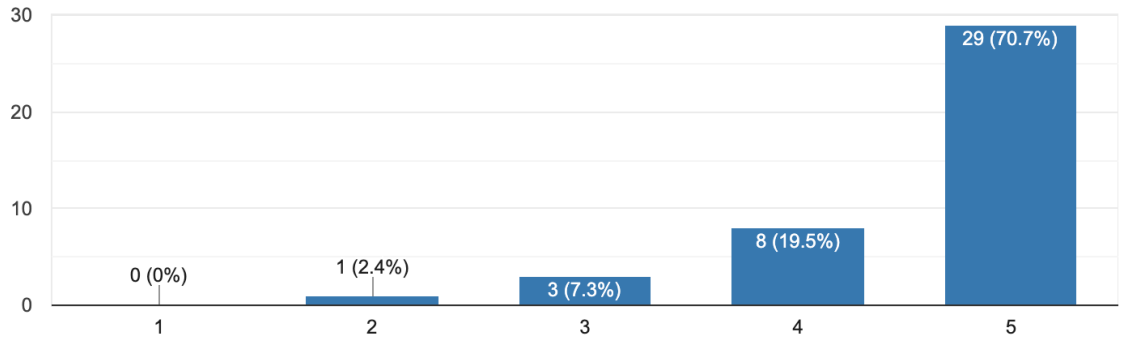
There is too much pressure in the CS major regarding GPA

40 responses



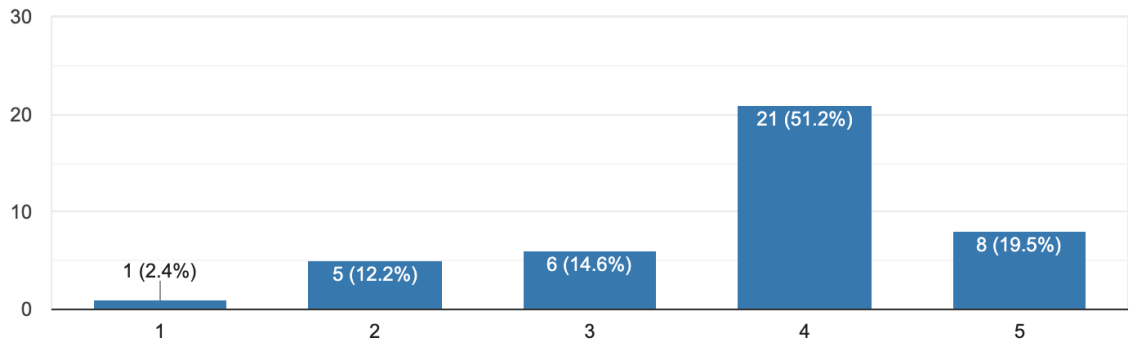
There is too much pressure in the CS major regarding internships

41 responses



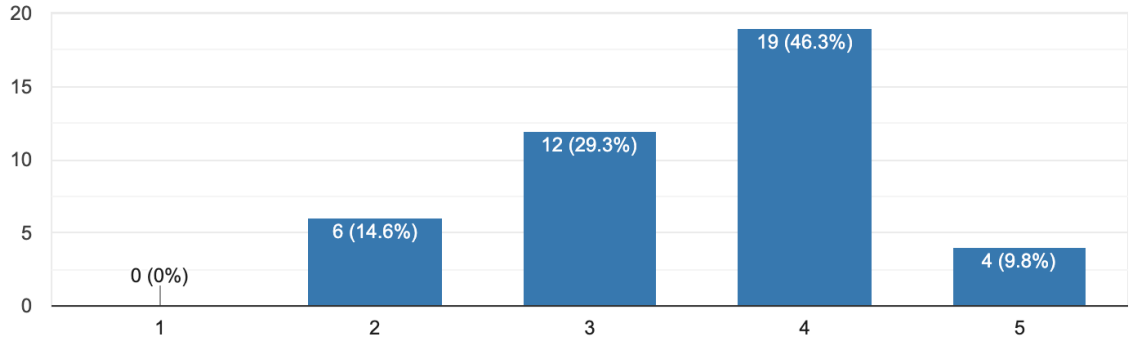
I feel comfortable asking my peers for help

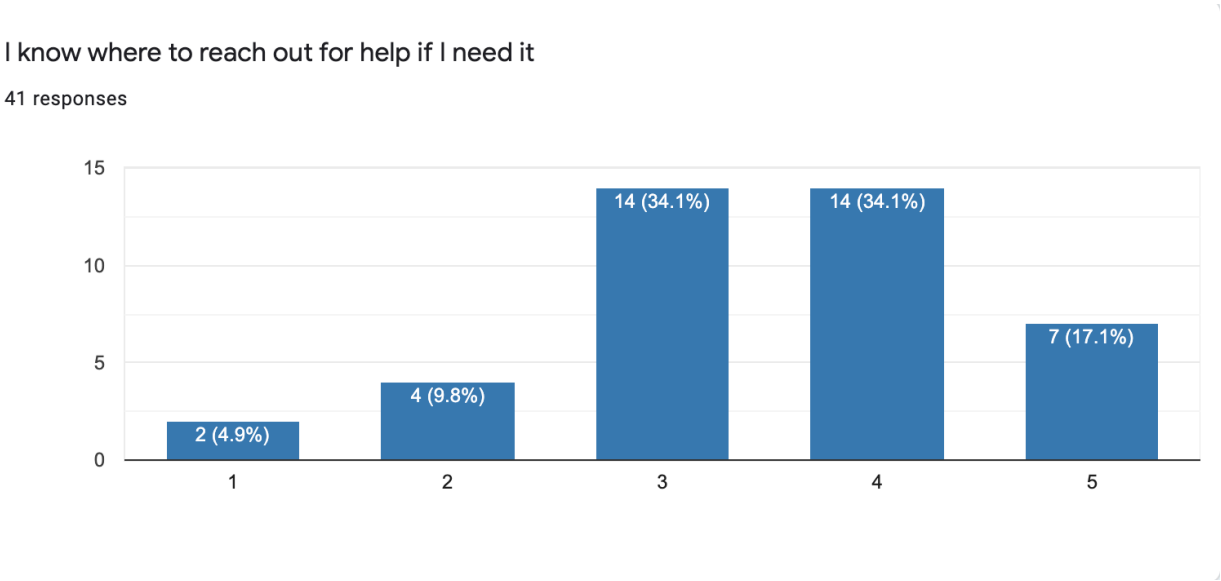
41 responses



I feel comfortable asking my professors and TAs for help

41 responses





Feelings of inadequacy

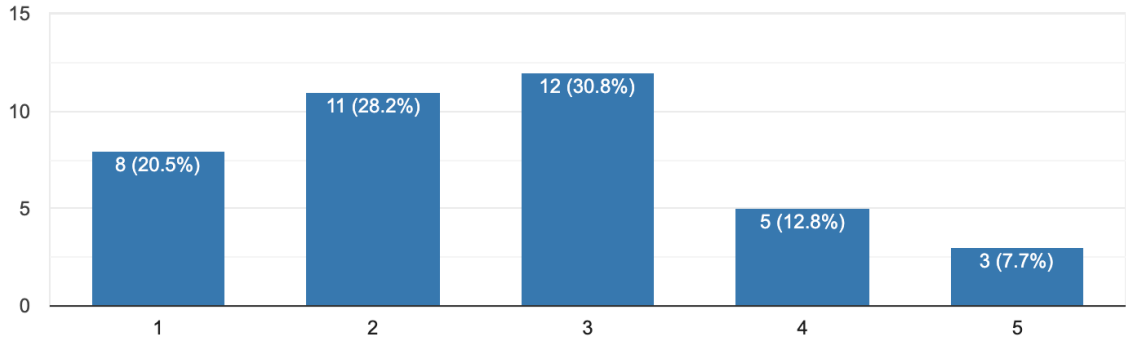
Have you ever experienced feelings of inadequacy while at UCLA? Please share your experience if you are comfortable with doing so
Yeah, constantly and deeply! There isn't really a specific experience I can point to... it is probably more of a personal issue than anything else.
Yes
internships. i hate the internship application process in general and am lowkey considering putting all my effort into research just so i dont have to deal with that bs
I constantly feel inadequate. I will be graduating soon and I feel like I have no job prospects ahead of me and everything is downhill from here.
Yes I feel like people think my achievements were handed to me more/easier because I am a woman and have had people say things to me about how I got my internship/research
The whole attitude towards internships in general! It's really common for people simultaneously to be endlessly thinking about/applying for/leetcoding and then just casually bringing up that they got into an internship as if its the easiest thing in the world (spoiler alert: it's not!) There are factors other than technical abilities that play a role and here, if you feel like you're not willing/able to go through the internship grind, there's an underlying attitude where you are looked down on by your peers.
This mainly came from the overwhelming pressure for getting internships / going into industry.
Not having an internship during my 2nd year summer
I absolutely have felt extremely inadequate in UCLA computer science. I have been surrounded by

high-achieving students who are going to work at FAANG companies, and this environment causes me to feel like I'm not good enough. I made the decision early on that I didn't want to take a heavy tech job or one at a huge corporate company, and even with my confidence in that decision I constantly felt a pressure from my peers that a Facebook or Amazon tech job should be what I'm striving for and achieving.
Have felt like my research output was insufficient to get into grad school
yeah literally all the time i was doing really bad during recruiting season and everyone else seemed to have their sh** together
I made a typo and almost quit a class on the first day because I could not find the typo.
I am a ling and cs major who wants to transfer into the CS major and I feel as though there is way too much pressure on grades, and it stresses me out to much to the point where I have too much anxiety to even start my hw. I always feel inadequate compared to my peers because I feel like everyone is so far ahead of me.
Ayo those internships tho!!!! 4th year and I haven't done a single one :)))))) It's mostly my fault but I would probably feel more supported if our classes taught us industry skills and there was a 2 or 3-unit class that just covers interview prep
Always
I feel uncomfortable when people ask me what companies I've interned at as I have not done an internship yet.
I have often felt inadequacy, especially when comparing my experiences with the experiences of my peers. This mainly includes the topic of internships (many have had prior internship experience) and general knowledge.
In pretty much every class I feel like an imposter, I feel this is largely due to the fact that the professors make almost every class I've had in the CS department super hard and time consuming. They expect you to pick things up fast and don't give much leeway.

UCLA CS and accessibility

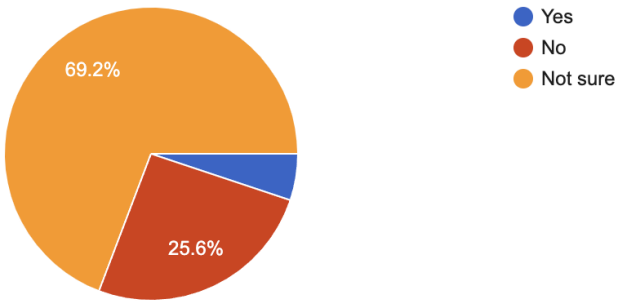
The CS Department is actively listening to student input about diversity and inclusion

39 responses



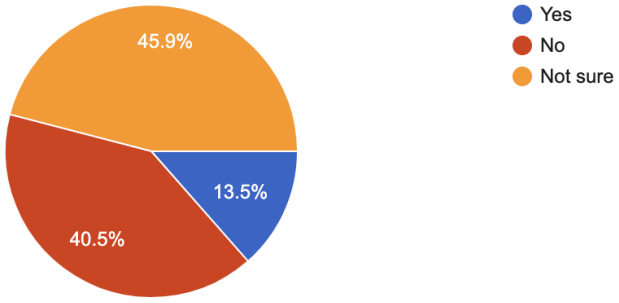
The CS Department provides ample support for people with disabilities

39 responses



The curriculum and course structure of UCLA CS are friendly to everyone, regardless of access to resources or disability

37 responses



Do you have any comments on UCLA CS and accessibility?
Most content (ie for classes) is not visually accessible at all. Also there seems to be a clear split between professors who are nice and open to answering questions and those who would make you feel like you know nothing if you try to answer a question
The strict deadlines set by many of the professors in the department are not very accommodating to students with health conditions. For example, I have a health condition that ensures I effectively cannot think for around 4 days a month. If one of those days happens to be before a CS project deadline, I have lost valuable time and need to crunch projects in time, as there aren't flexible deadlines. The same goes for any other student who is ill for any reason, or dealing with emotional or mental health issues. Eggert's usage of a late deadline has been extremely helpful to me and other students in that regard, allowing us more time to understand our progress but recognize the delay with the percentages cut off from our grades.
UCLA CS should record all lectures and make slides available before class.
We should have all CS classes be remote whenever possible. Our major is to be stuck behind a screen.
Certain classes require so much time and effort that it doesn't feel like the class is accommodating to individuals who require working a part time or have certain disabilities.
Course websites and powerpoints should be visually accessible! Also, having all classes be recorded is super helpful for disabled students, especially if physically getting to class is very hard.

How can the CS Department further improve their focus on diversity and inclusion?

How can the CS Department further improve their focus on diversity and inclusion?
Please replace Professor Eli Gafni as chair of the department.
I don't know how realistic this is, but I would love to see more diversity among professors.
not have someone who clearly does not care about diversity and inclusion as the department chair : D
Reassess the way they present themselves to students (both in person and online)
I have not taken a single class with a female professor. I feel like having a female professor in one of my earlier classes would have helped me feel like I belonged more and decreased my sense of imposter syndrome.
It's really hard for me to believe that the CS Department can have a focus on diversity and inclusion when its current chair has used terms like "Wuhan Virus" then justifies this behavior by bringing up that "he has a Chinese wife" and "loves Chinese" with a wink face emoji. This language alone makes me feel unsafe in interacting with CS faculty and department leadership. His apology was also inadequate. Student's aren't stupid, we can tell that half of Gafni's apology wasn't even written by him.
By actually publicizing and supporting D&I work done by students while it's happening instead of inhibiting it and taking credit for it afterwards. Also stop relying on unpaid student labor.
They do not need to.
more diverse faculty, people with different experiences
As a freshman I joined organizations because of my minority identity, but I had to put in the work to seek those out. I think it would be helpful if upon being admitted to UCLA CS, every student gets mailed and emailed a brochure of CS/engineering orgs they could join. Then, orgs for minorities could be highlighted. However, this also costs money so we know it's not gonna happen :)))
Being receptive to initiatives brought about by student orgs
I would like to see professors and administrators put in a better effort to understand student problems. I have never (in my 4 years here) been asked by a professor or administrator on how they can do better, and to me that's quite upsetting. In addition, the department needs to do a better job of showing actionable steps taken from feedback!

How can professors further improve their focus on diversity and inclusion?

How can professors further improve their focus on diversity and inclusion?
I suppose, assume as little prior knowledge as possible?
Use gender-neutral words when referring to made-up people. Do not treat female students patronizingly and treat them like they are any different from the rest of the CS student body.
Treat student feedback more seriously, tell us what changes have been made from such feedback (like end of quarter student course evaluations)
Treating everyone "equally" is not enough :/ make sure to acknowledge the ways you and the school are trying to support Black, Latinx, and Indigenous students, LGBTQIA+ students, and neurodivergent people. Make your service diversity and inclusion-oriented. Stop placing the burden of mentoring minoritized students on minoritized professors. Talk about how your research impacts diversity and inclusion. Recruit minoritized students to your lab.
They do not need to.
Something I really like is when professors offer more opportunities for feedback throughout the quarter (I've heard that John Cho and George Varghese both do this). Profs should do this more!

If you have any other thoughts about diversity at UCLA CS, please leave them here

If you have any other thoughts about diversity at UCLA CS, please leave them here
Diversity is not just a "pipeline" problem!