Nifty Assignments

Nick Parlante
Julie Zelenski
Stanford University
nick.parlante@cs.stanford.edu
zelenski@cs.stanford.edu

Evan M. Peck
Bucknell University
evan.peck@bucknell.edu

Kevin Wayne
Princeton University
wayne@cs.princeton.edu

John DeNero
Christopher Allsman
Tiffany Perumpail
Rahul Arya
Kavi Gupta
Catherine Cang
Paul Bitutsky
Ryan Moughan
UC Berkeley
{denero,callsman,tperumpail,rahularya,
kavi,catherinecang,pbitutsky,rmoughan}
@berkeley.edu

David J. Malan
Brian Yu
Harvard University
malan@harvard.edu
brian@cs.harvard.edu

Carl Albing
Naval Postgraduate School
carl.albing@nps.edu

Keith Schwarz
Stanford University
htiek@cs.stanford.edu

ABSTRACT

The Nifty Assignments special session is about promoting and sharing the ideas and ready-to-use materials of successful assignments.

Each presenter will introduce their assignment, give a quick demo, and describe its niche in the curriculum and its strengths and weaknesses. The presentations (and the descriptions below) merely introduce the assignment. A key part of Nifty Assignments is the mundane but vital role of distributing the materials – handouts, data files, starter code, rubrics, autograders – that make each assignment ready to adopt. Each assignment presented has complete materials freely available on the Nifty Assignments home page nifty.stanford.edu.

If you have an assignment that works well and would be of interest to the CSE community, please consider applying to present at Nifty Assignments.

CCS CONCEPTS

• Applied computing → Digital libraries and archives

KEYWORDS

Education; assignments; homeworks; examples; repository; library; nifty; pedagogy

Housing Algorithms: Developers as Decision Makers (CS0/CS1) — Evan M. Peck

How can we train students to carefully reflect on their social responsibility as programmers? In this CS1 assignment, we pair sociotechnical issues with the technical content. While practicing conditional statements, students design an algorithm to assign housing priority on campus. Their program asks a series of questions, assigns points for answers which sum to a total priority at the end of the questionnaire.

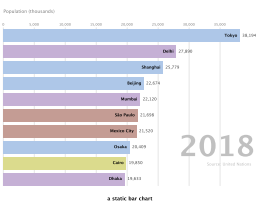
Each student group uses a human-centered design process to interview their classmates, and then decides on the important questions that should drive housing priority. Students determine test cases and write code that reinforces their understanding of conditional statements. The output is simple and the project can be completed in just a few hours, but the challenges model those faced by real-world algorithms.

Students observe a programming context in which there are no *right* answers – only complex tradeoffs that rarely have neutral outcomes. They reflect on which people their algorithm benefits and which are likely to be disadvantaged. The assignment touches on algorithmic transparency and fairness early in their CS career. Most importantly, we believe it signals to students that hard problems in CS aren't always hard for purely technical reasons, but for the contexts in which they are embedded.

Bar Chart Racer (CS1) — Kevin Wayne

In Bar Chart Racer, student write a program to produce *animated bar charts*. Animated bar charts are a surprisingly simple, yet powerful, way to tell a story about categorical data over time.

For example, to visualize the 10 most populous cities in the world from 1500 to 2019, students successively draw 520 individual bar charts (one per year of data), with a short pause between drawings. Each bar chart contains bars for the 10 most populous cities in that year, sorted in order of population, and colored according to world region.



We provide a simple library (in either Java or Python) to draw static

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03.
https://doi.org/10.1145/3328778.3372574

bar charts. We also supply compelling data files, ranging from geography and business to sports and entertainment.

The assignment combines graphics and real-world data to create captivating visualizations. It is easy to explain, authentic, and appealing to a diverse range of students. It gives students practice with several core CS topics including sorting an array/list, defining a total order for a user-defined type, reading a text file one line at a time, and computer animation.

Typing speed test (CS1) — John DeNero, Christopher Allsman, Tiffany Perumpail, Rahul Arya, Kavi Gupta, Catherine Cang, Paul Bitutsky, and Ryan Moughan

Students create a web app similar to TypeRacer.com where players compete to type a quotation as fast as possible. They build a simple Python web server, starting with computing typing speed and accuracy measurements, then adding nifty features like autocorrect and multiplayer support.

This assignment covers a number of CS1 topics including using collections (lists, strings, dictionaries), list comprehensions, higher-order functions, and iteration. Computing string edit distance for the autocorrect feature is a motivating application of recursion. While the project is designed for a Python-based course, the open-source React-based web GUI provided to the students is easy to read and extend. This assignment lends itself nicely to extensions involving faster or more accurate autocomplete, as well as post-game analysis of a typing race.

This project was developed for week 5 of a fast-paced CS1 course in which most students have prior programming experience. An integrated autograder and full test suite guides students through the process and allows them to verify the correctness of their implementation as they work.

DNA (CS1) — David J. Malan and Brian Yu

This assignment is a whodunit, a CSI-style problem for which students write a program to search a DNA database to find a match to a "suspect". This application of computer science to biology asks students to explore DNA profiling and how forensic investigators can identify to whom a given sequence of DNA belongs. Students learn about Short Tandem Repeats (STRs), short sequences of DNA that repeat consecutively at specific locations within a person's DNA. The number of times each STR repeats can be used to identify someone based on their DNA.

Students compute the STRs from a target DNA sequence and compare to the database of STR counts to identify the person to whom the DNA (most likely) belongs. We use this assignment to introduce students to Python and to give students practice with loops, string manipulation, and file I/O.

Color My World (CS1/CS2) — Carl Albing

Students solve the mystery of the unknown image while learning about pseudo-coloring and color maps to convert raw data into beautiful images.

Computing can produce lots of numbers fast; pictures can help us understand those numbers. Often those pictures don't match an actual physical visibility; (e.g., what color are microwaves from space?) so we can be creative, choosing and assigning colors to bring out detail in those images based on our data.

In this assignment, students are given a data file of raw numbers and convert those numbers into a compelling visual image. Application of a color map produces the pixel values of their image. Variations of the assignment make it suitable for more advanced students. While early programming courses can focus on looping through the pixels, more advanced classes in scientific programming, high performance computing, computer graphics, or data science, can be given less starter code and require more analysis of the raw data.

This assignment challenges students with an open-ended creative aspect as well as its tangible result— a pile of numbers turned into an amazing image.

Recursion to the Rescue! (CS2) — Keith Schwarz

This assignment is a trio of recursion exercises that get students solving problems with real-world stakes. Using recursive backtracking and memoization, students construct hospital schedules, determine where to stockpile supplies for disasters, and probe the extremes of the Electoral College. In doing so, students explore classic NP-hard problems and see how the techniques they're learning in the classroom translate into practice.

The three pieces can used individually or mixed and matched into a single assignment.

- *Doctors Without Orders*: Students write a function to schedule patient appointments while respecting restrictions on the amount of time each doctor has available.
- *Disaster Planning*: Students determine where to stockpile emergency supplies so that each city either has supplies or is adjacent to one that does. The sample data files model the road and rail networks of various countries and regions.
- *Winning the Presidency*: Students write a function that determines the minimum number of popular votes that would have been needed to be elected President of the United States in each year since 1828. The results are surprising!

Many students find these problems rewarding because of their inherent interest in the topics explored. For students who go onward to take courses in algorithms or complexity, this assignment has the additional benefit of introducing and motivating three classic NP-hard problems: bin packing, minimum dominating set, and the knapsack problem.