Nifty Assignments

Nick Parlante, Julie Zelenski
(moderators)
Stanford University
nick.parlante@cs.stanford.edu
zelenski@cs.stanford.edu

Peter-Michael Osera
University of Pennsylvania
posera@cis.upenn.edu

Marty Stepp
Stanford University
stepp@cs.stanford.edu

Suzanne J. Matthews
United States Military Academy
suzanne.matthews@usma.edu

Allison Obourn
University of Washington
aeobourn@cs.washington.edu

Mark Sherriff
Luther Tychonievich
Ryan Layer
University of Virginia
(mark) sherriff@virginia.edu
(luther) lat7h@virginia.edu
(ryan) rl6sf@virginia.edu

David R Raymond
United States Military Academy
david.raymond@usma.edu

Josh Hug
University of California, Berkeley
hug@cs.berkeley.edu

Stuart Reges
University of Washington
reges@uw.edu

Categories and Subject Descriptors
K.3.0 [**Computers and Education**]: General

General Terms
Algorithms, Design, Languages

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

Abstract
A great CS assignment is a delight to all, but the path to one can be most roundabout. Many CS students have had their characters built up on assignments that worked better as an idea than as an actual assignment. Assignments are hard to come up with, yet they are the key to student learning. The Nifty Assignments special session is all 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 each assignment. A key part of Nifty Assignments is the mundane but vital role of distributing the materials – handouts, data files, starter code – that make each assignment ready to adopt. The Nifty Assignments home page, **http://nifty.stanford.edu**, gathers all the assignments and makes them and their support materials freely available.

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. See the nifty.stanford.edu home page for more information.

SIGCSE '15, March 4-7, 2015, Kansas City, MO, USA.
ACM 978-1-4503-2966-8/15/03.
http://dx.doi.org/10.1145/2676723.2677327

Counting Squares (CS0/CS1) - Mark Sherriff, Luther Tychonievich, and Ryan Layer

This intro assignment is used to help introductory students think about the process of creating a basic algorithm, with a particular emphasis on generality and efficiency. Students try to think of different ways for a robot to count all the squares on a grid using a very basic set of commands. Code is provided in both Python and Java to allow them to test out their solutions; we have also had success using it "unplugged" on paper. We have found that this assignment is a great introduction to computer science and programming in general. It's a relatively easy problem to start with if you consider a square grid, but the complexity can quickly escalate as the students compete for the fastest solution (efficiency) while considering grids with missing squares or irregular shapes (generality). This assignment can be used as a quick glimpse into AI for early classes as well. Counting Squares has been used with great success with both our introductory course as a lab session and as a group activity in lecture. We have also used it with middle school students in a camp setting.

Speed Reader (CS1) - Peter-Michael Osera

Many people, students especially, wish they could read faster. In addition to the various speed reading courses out there today, we can employ software to help us learn to read faster. Most of these programs employ Rapid Serial Visual Presentation (RSVP) to help you focus your eyes and eliminate sub-vocalization, allowing you to read at more than twice the rate of the average person. However, are these programs truly effective? Are we able to read at this rate and still comprehend what we are reading?

Luckily, speed readers are simple programs that we can have our students implement in a CS1 course and run experiments over to see for themselves! In this assignment, students build a prototype speed reader that reads in plain text files and displays them in the RSVP style. They are then charged to run small usability studies on their friends and families to determine if RSVP is effective.

The speed reader program has highly flexible requirements. At its core, students practice the basics of file IO, string tokenization, and animation. Depending on the instructor's preference, students can implement a curses-style console-based speed reader or a fancier speed reader in a GUI. The assignment is also extensible in many ways to give it as much depth as the instructor desires.

Geo Location (CS1) - Stuart Reges

This assignment involves writing a simple class that is a variation of another class. The CS education community has struggled to find simple class examples that don't seem contrived or confusing. The most successful examples involve some form of scaffolding where the instructor provides a lot of supporting code. This assignment also involves scaffolding, but the interface is very simple. The code calls a method that accesses the Google Maps API to get information about a location given its coordinates (latitude and longitude).

Students are shown the implementation of a GeoLocation class that keeps track of information about a location. Students intuitively understand that a location involves two pieces of information: a latitude and a longitude. It also has the nice property that these values are often expressed in two different ways: as three integers (hours, minutes, seconds) and as a single real number, which means that it makes sense to have two different constructors. It also has a nontrivial bit of behavior that you would not want to make others implement. Instead of something like the boring but classic Fahrenheit to Centigrade conversion, this class includes a method that finds the distance between two locations on Earth, which is not a simple calculation.

In the assignment, students first practice being a client of the GeoLocation class and then they write a variation of the class that keeps track of additional information about the location (it's name, address, and associated tags). Our observation is that students make many classic mistakes as they learn about objects in this assignment. For example, instead of storing their own GeoLocation object that computes the distance between two locations, they often copy the code from the GeoLocation class. That kind of confusion is useful to clear up early so that we can point out that one object often delgates some task to another object to complete.

The nifty part is the ability to access the Google Maps API. This makes the assignment more compelling. It also allows for many extensions to the basic assignment. For example, we wanted to have a large amount of data of places of interest near the University of Washington. This wasn't easy to get directly from Google, so we created our own through crowdsourcing. Each student submitted entries for a massive data file that we compiled.

Packet Sniffing in Python (CS1) – Suzanne J. Matthews and David R. Raymond

This lab highlights the real-world dangers of packet-sniffing by enabling students to analyze a packet capture (PCAP) file using the Scapy package in Python. After a quick overview on how information is transferred on wired and wireless networks, we discuss the ethics and legality of packet sniffing unsecured wireless networks (like those at coffee shops, airports, and hotels). During the main portion of the lab, students write Python code to extract and inspect packets contained in a PCAP file purportedly collected from four individuals surfing the web using a coffee shop's unsecured wireless network. The PCAP file is in fact artificial; it was created on our sandboxed networks. The end goal is for students to collect as much information as possible (e.g. names, e-mail addresses, passwords, occupations, e-mail contents) about the individuals at the coffee shop. Most students are shocked that they can read other people's e-mails and learn what websites they visited. The exercise quickly underscores the need for packet encryption, and promotes discussion on how students can best protect themselves. The lab can easily be tailored as a homework assignment or project. We assume students have covered the basics of sequence, selection and iteration in the Python language, and are knowledgeable about Python lists and dictionaries.

Melody Player (CS1/CS2) - Allison Obourn and Marty Stepp

This assignment involves writing a class to play music. Students are given files containing the notes of songs in a simple text format. Part of what makes the data interesting is that most songs have repeated sections commonly called a refrain or chorus. Students implement a class to store notes and rests in a song. Repeated sections must be represented without duplication. They write methods to perform various operations such as playing the song, changing its octave/pitch or tempo, reversing it, etc. Students are given a library for playing individual tones. They can be provided a UI or can write it themselves.

We have given this assignment using arrays/lists and using stacks/queues, so it can fit in mid/late CS1 or early CS2. The assignment is extensible and makes it simple to adjust the list of operations that the students will implement; for example, advanced students may implement playback from a given time index.

The data is interesting without being "big" data, so it is simple to visualize and debug. Students understand the data and understand what they are expected to do, while getting good practice of linear data structures. Also, students often like multimedia content, but many of such assignments focus on graphics. It's fun to hear the songs play and to perform manipulations on familiar melodies. Students also enjoy writing their own custom songs.

Seam Carving: A Content Aware Image Resizing Algorithm (CS2) - Josh Hug

Given an image, it is sometimes desirable to scale that image in only one dimension, for example when moving from a 16:9 aspect ratio to a 4:3 aspect ratio. The easiest approach is to simply crop the image, or with a bit more work, to rescale the image. However these approaches either remove image content or make everything in the image look skinnier or fatter than it should.

Seam carving is a simple and elegant image resizing technique published in 2007 by Shai Avidan and Ariel Shamir that avoids these flaws by rescaling in a content-aware manner. Specifically, seam carving reduces the size of an image by one pixel of height (or with) at a time by removing 'seams', which are contiguous paths of pixels with minimum total energy.

Despite being a relatively new published result, the algorithm is simple to explain and implement, and the quality of the results can be quite impressive for the right kinds of images. Furthermore, when applied to the wrong sorts of images (like the student's own faces), the results can be both horrifying and hilarious.

The assignment can be cast as a difficult challenge involving nested for loops over a two dimensional array, as an example of dynamic programming, or as a graph traversal problem on a directed acyclic graph. There are also opportunities for exploration, including finding ways to expand the size of images, remove specific regions of images, and more.