# 549T, Fall 2014
# Parallel and Sequential Algorithms

## Recitation 1: Overview

Kunal Agrawal and I-Ting Angelina Lee

*August 25, 2014*

# Requirements for CSE 549

- One in-class presentation on an assigned paper
- Read and discuss (on Piazza and in class) all assigned papers
- Final project (in groups of 2-4): Project proposal, a short presentation, and a written report.

# Logistics for Assigned Readings

- We'll assign a paper to read for most Fridays.
- Each week, 2-3 students will present the assigned paper(s).
- On the Tuesday before you present in class, you should practice your presentation to us and incorporate our comments into your presentation.
- All students should read all assigned papers and participate in discussions.

# List of Assigned Readings

- R2 (09/05): Work-Stealing Analysis
- R3 (09/12): Introduction to Cache Analysis
- R4 (09/19): Cache-Oblivious Algorithms
- R5 (09/26): Cache-Aware and Cache-Oblivious Sorting
- R6 (10/03): Cache Analysis of Work-Stealing
- R7 (10/10): Introduction to Concurrent Programming
- R8 (10/24): Concurrent Queue
- R9 (10/31): Reducers
- R10 (11/07): Parallel Breath-First Search

# Final Project Timeline

- Send us a (private or public) Piazza note about your project ideas by Oct. 07.
- Schedule appointments on / before Oct. 07 if you want to chat in person about project ideas.
- A 2-4 page project proposal is due on Oct. 17. It should contain some preliminary literature review and a precise problem definition.
- Project proposal should contain a "Plan B."
- A 20-min final project presentation in class on Nov. 14 and Nov. 21.
- Final project report due on Dec. 05.

# Possible Final Project Ideas

- Comparing concurrent data structures (implementation, experimental evaluation, and possible improvement)
- Cache-aware vs. cache-oblivious algorithms
- Parallelization of a real-world application
- Augment the Cilk Plus or Cilk runtime system to collect performance data, such as cache-misses.
- Design and analyze a new scheduler, for example, scheduling on processors with different speed that allow preemption at certain points.
- Improve upon the implementation of reducer hyperobjects