Week 1.1, Monday, August 19
Questions
- I welcome clarification questions during lectures

Remote Lecture?
- I am attending CRYPTO in Santa Barbara
- Prof. Atallah will teach class on Wednesday
- I will be back on campus on Friday (Skype won’t be the norm!)
- Kevin Xia (GTA) can relay questions today
Maximum Enrollment*: 150

Wait List: 17 (as of 8/18/2019)

I will speak to the CS registrar to see if it is possible to open up additional slots after the first week

- Seniors who still need to take the course would receive priority
- I cannot make definitive promises
Course Personnel

Instructor
- Professor J. Blocki

Graduate TAs
- Kevin Xia (head TA)
- Tunazzina Islam
- Hai Nguyen
- Ahammed Ullah

Undergraduate TAs
- Michael Cinkoske
- Noah Franks
- Hiten Arun Rathod
- Abhishek Sharma (Head UTA)
- Himanshi Mehta
- Utkarsh Jain

Office Hours will be posted on Piazza (Office Hours Begin 2nd Week)

Instructor Office Hours (Week 1): Fri, 2:30-3:30 PM
Course Resources

Course website

- [www.cs.purdue.edu/homes/jblocki/courses/381_Fall19/](www.cs.purdue.edu/homes/jblocki/courses/381_Fall19/)
- Course policies, syllabus, course work, links to related material

Piazza

- [https://piazza.com/class/jyk8rg9el1s5m6](https://piazza.com/class/jyk8rg9el1s5m6)
- For discussion on course material, assignment questions
- Slides, assignments and solution sketches are posted
- Sign up!
Course Resources

Gradescope
- submitting typed assignments
- View graded exams
- Submit re-grade requests*
  (re-grade policy discussed later)

Blackboard
- register your clicker before Friday!
- Grades
- (Optional) CS580 Lectures
Assignments: 20%
- 7-8 written assignments
- Must be typed, submitted on Gradescope

Clickers: 5%
- You can communicate with your neighbors
- No makeups (lowest score dropped)

Exams
- closed book & notes, cheat sheet allowed
- Midterm 1: 20% (Sep 25th. 8PM)
- Midterm 2: 20% (Oct 30th. 8PM)
- Final exam: 35%
Prerequisites for 381

- **CS 182**
  - Applications of discrete math in CS
  - Proof techniques: direct, indirect, induction
  - Abstractions, recursion, counting

- **CS 251**
  - Very good understanding data structures (their use, implementations, limitations, tradeoffs)
  - Ability to judge and think through an implementation without coding
  - No programming assignments in 381
Data structures

- Stacks, queues, search trees (binary, balanced), priority queues, hashing, trees, graphs
- Operations on data structures under different implementations
More on prerequisites

- Algorithms
  - Searching and sorting
  - Graph and tree traversals (BFS, DFS, pre, post, inorder)
  - Computing graph properties
  - Examples of greedy algorithms: Shortest path, min spanning tree
  - Analyze asymptotic performance of given code
Course Highlights

- Introduction to the Analysis of Algorithms
  - Central to all of Computer Science!
- Techniques to Design Efficient Algorithms
  - Greedy, Divide and Conquer, Dynamic Programming, Network Flow, Reductions etc…
  - Desiderata: Efficient, Concise and Correct

- Analysis
  - Analyze required resources to execute (space/time)
  - Prove that the algorithm is correct
LIFE-CRITICAL SYSTEM VERIFICATION

"If it fails, people die."

Theoretical computer scientists harness the power of logic and mathematics to provide a provable guarantee of safety.
Course Highlights

- Problem Solving!
  - Abstracting the essential features of a problem
  - Developing creative, efficient, and non-obvious solutions to problems.
  - Analyzing an algorithm’s performance and resource usage in a machine and language independent way.

- Understanding computational limitations
  - Are there problems that don’t permit efficient algorithmic solutions
  - P vs. NP (Million Dollar Problem)
What you submit needs to be understandable (and typed)

We want human readable pseudo code

- Target audience is a human TA not a compiler!
Homework Solutions

- Graded for correctness, clarity, conciseness, rigor, and efficiency
- Type using any software supporting math notations (good opportunity to learn LaTeX)
- Read homework guidelines on course webpage
Cannot Solve Problem?

- Partial Credit
  - Clearly identify a reasonable approach to solve the problem
  - Maximize partial credit by identifying gaps in your attempted solution

- Better to acknowledge that you don’t know than to pretend you solved it.
  - 15% credit for simply admitting “I could not solve the problem”
  - Can receive 0% credit for bad/obfuscated `solutions`
Exams

- Closed Book (No Phones, Laptops, Calculators, Smart watches etc...)
  - Allowed one sided page of handwritten notes
  - Communicate only with course staff during exam

- Disability Requiring Special Accommodation:
  - Contact instructor promptly (first 3 weeks of class)
  - Need official letter from Disability Resource Center

- Final Exam (TBD)
  - Do not book travel before last possible date (Dec 14th)
Regrade Requests

- Graders are human, please be patient!

- Submit on Gradescope
  - Within 14 days of return of assignment (exception for final exam)
  - Should clearly explain what you think the grader missed
  - Not an opportunity to expand your answer!
  - Double Check Carefully
    - Your grade can go up or down!

- Appeal Rejected Regrade?
  - 2 points bonus I agree you are correct (for your trouble)
  - 2 points deduction if I agree with TA (most common outcome)
Collaborations is allowed, but you must acknowledge collaborators on your homework.

Don’t cheat yourself out of learning how to solve problems!
- Suggestion: spend at least 15 minutes thinking about each problem yourself before collaborating.

Reference all sources used.
- Failure to acknowledge a collaborator or key source is cheating!

All submitted solutions must be written entirely in your own words!

You must understand your solution completely.
- If you could not explain your solution to course staff that is considered cheating!

Reference:
https://www.cs.purdue.edu/homes/jblocki/courses/381_Fall19/syllabus/syllabus.pdf
Late Homework

- Not Late
  - No Penalty
- Less than 24 hours late
  - 10 point penalty (out of 100 possible points)
  - Subtracted from final score
- 24 to 48 hours late
  - 25 point penalty (out of 100 possible points)
  - Subtracted from final score
- More than 2 days late
  - No Credit 😞
Course materials, including slides, tests, and other course materials, are protected by copyright.

- I am the owner of the copyright in the materials I create.
- CLRS/KT/W/Susanne etc… are the owners of the material I use from the slides they provide.

You may make copies of course materials for your own use.

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly without written consent.

Similarly, you own copyright in your answers to assignments.

- If I am interested in posting your answers on the course web site, I will ask for your written permission.
Piazza Rules of Conduct

- Piazza is intended for clarification of questions of general interest.
- Piazza cannot be used to post answers to assignments, detailed descriptions of solutions, or hints.
- Piazza is not the forum for complaints about an assignment, exam, or the class.
  - Any concerns should be brought to the attention of the instructor.
- Be courteous and professional when posting/emailing and use appropriate language.
- If you are not sure whether a posting is appropriate, make sure it is made private or e-mail us.
Your tasks

- Register your clicker
- Sign up on Piazza
- Suggested Reading
  - CLRS: Sections 10 and 12 (data structures)
  - CLRS: Appendices A and B (discrete math)
- Assignment 1 will be posted by Friday
  - Review of material needed for algorithm analysis
- Attend PSOs to review proof techniques
A Remark about Hashing

- Hashing (for insert, delete, search) is important and effective
- Algorithms in CS381 do not use hashing

Collisions can lead to poor worst case performance

- To get $O(1)$ expected performance per operation, one needs the right table size, a good hash function and good collision resolution
**Warm Up: Sorting**

*Input:* \( n \) numbers \( a_1, \ldots, a_n \) stored in an array \( A \)

*Output:* sorted sequence of size \( n \)

(increasing = non-decreasing)

**Algorithms you should know**
- Bubble sort, Insertion Sort, Selection Sort
- Merge Sort, Quicksort, Heapsort

**Review**
- asymptotic performance, stable sorting, in place sorting
- create the sorted sequence in an incremental way
- start with a sorted sequence of length 1 and insert one more element in each iteration

**INSERTION-SORT** \( (A, n) \)

For \( j \leftarrow 2 \) to \( n \) do

1. \( key \leftarrow A[j] \)
2. \( i \leftarrow j - 1 \)
3. While \( i > 0 \) and \( A[i] > key \) do
   - \( A[i+1] \leftarrow A[i] \)
   - \( i \leftarrow i - 1 \)
4. \( A[i+1] = key \)

1 \hspace{1cm} i \hspace{1cm} j \hspace{1cm} n

- sorted
- key
What is the running time of Insertion Sort?

Number of times the while-loop is executed depends on the input

- increasingly sorted input is fast; decreasing is slow.
- Worst case? $\sum_{j=2}^{n} j < n^2$
- Average case?

What all do we count/have to count when analyzing time?

- In (internal) sorting algorithm we generally count the number of comparison

Pseudo code has two nested loops
- while loop moves left from \( j \) to 1
- total time won’t be more than quadratic.

Note: A doubly nested loop does not necessarily result in quadratic time

Worst case: \( T(n) = O(n^2) \)
- Work is bounded by summing the first \( n-1 \) integers which is equal to \( n(n-1)/2 \)
- Time is proportional to \( n^2 \)
- Also, \( T(n) = \Theta (n^2) \)
INSERTION-SORT \((A, n)\)

for \(j \leftarrow 2\) to \(n\) do


\(\text{key} \leftarrow A[j]\)

\(i \leftarrow j - 1\)

while \(i > 0\) and \(A[i] > \text{key}\) do

\(A[i+1] \leftarrow A[i]\)

\(i \leftarrow i - 1\)

\(A[i+1] = \text{key}\)

INSERTION-SORT \((A, n)\)

for \(j \leftarrow 2\) to \(n\) do

**Pre-Condition:** \(A[1] \leq A[2] \ldots \leq A[j - 1]\)

\(key \leftarrow A[j]\)

\(i \leftarrow j - 1\)

while \(i > 0\) and \(A[i] > key\) do

\(A[i+1] \leftarrow A[i]\)

\(i \leftarrow i - 1\)

\(A[i+1] = key\)


Post-Condition when \(j=n\) \(\Rightarrow\) entire array \(A\) is sorted.

\[
\begin{align*}
\text{key} & \leftarrow A[j] \\
n & \leftarrow j - 1
\end{align*}
\]

Define $A_{\text{orig}}[1,\ldots,j-1] := A[1,\ldots,j-1]$

while $i > 0$ and $A[i] > key$ do

\[
\begin{align*}
A[i+1] & \leftarrow A[i] \\
n & \leftarrow i - 1
\end{align*}
\]

Invariant: $A[1,\ldots,i-1] = A_{\text{orig}}[1,\ldots,i-1]$ (untouched)

$A[i+2,\ldots,j] = A_{\text{orig}}[i+1,\ldots,j-1]$ (shift once*)

key $< A[i+2]$

$A[i+1] = key$