Wrapping Up

Final

- On gradescope
- Can download pdf of exam, will have a link to overleaf
- 24 hours from start to finish (Gradescope will warn you that you are starting that timer)
- Turn it in on gradescope, just like assignments
- Please be sure not to start the exam until you’re ready to spend 24 hours on it
- Multiple submissions OK; turning in after deadline not possible unless I manually intervene
- Good idea to turn in an early version

Final: Honor code

- I’ve put in some failsafes
- There are plenty of ways to avoid them
- Please don’t :)
- High stakes

Admin

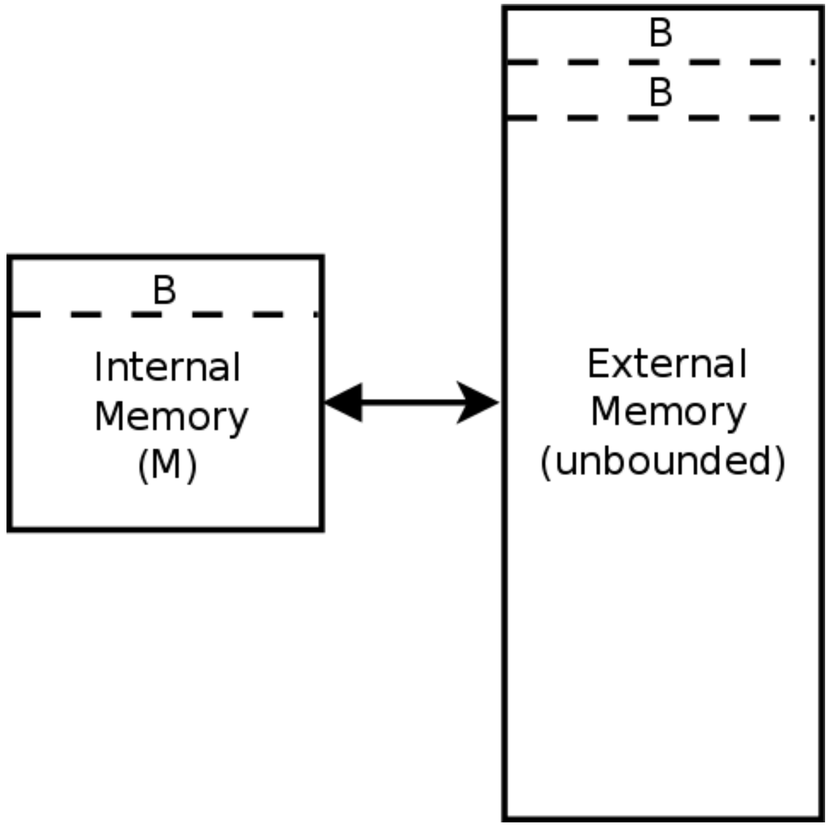
- Office hours/Assignment 10 discussion:
    - Friday. 4-6pm
    - (Note that this was moved back an hour)
- Assignment 9 back this evening
- New sample student answers out later today (I'll announce on Glow)
- Questions?

Other Models of Computation

Aka: Applying what we've learned in practice

External Memory Model

- Algorithmic model to measure asymptotic cache performance
- Doesn't capture everything
- Ignores computation cost! Only cost of moving information
- But can help indicate how cache-efficient an algorithm is



Cache-efficient algorithms for:

- Sorting (run generation/timsort, multi-way merge sort)
- Dictionaries (B trees)
- Matrix multiplication
- Dynamic programming

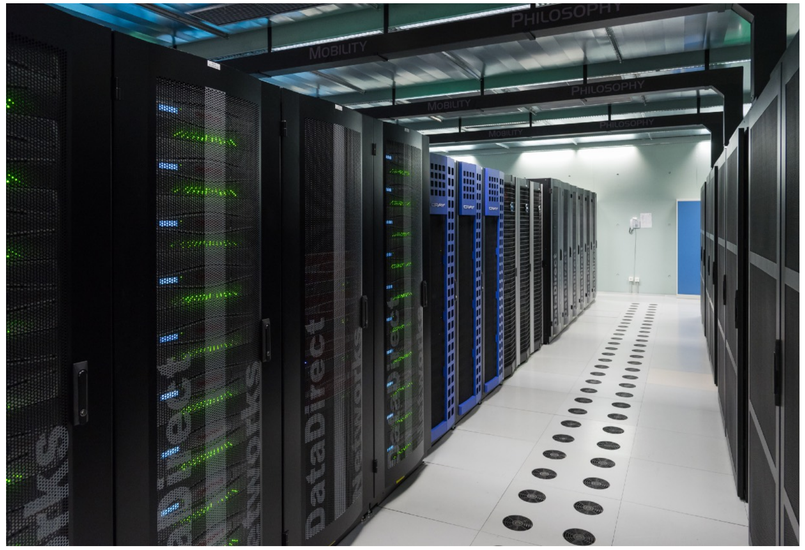
Operations Aren’t the Same

- Addition and subtraction are fast, multiplication is fast
- Division and modulo are slow
- Integers are faster than floats (?)
- Can we model this in theory?
- Not really. Asymptotics ignore this intentionally
  - Going from $O(2^n)$ to $O(n)$, or even $O(n^2)$ to $O(n \log n)$, is more important

- Occasionally something like: $O(n \log n)$ additions, $O(n)$ divisions

Parallelism

- Modern computers almost always have multiple compute cores
- If we have p identical “processors” can we speed up our algorithms?
  - Maybe by a factor of p?
- Many models for algorithm analysis
  - PRAM is as above, classical model
  - MapReduce model: massive number of cores, want to minimize communication rounds
  - Many others



GPUs

- Really popular way to improve performance
- Key for things like ML, Bitcoin mining
  - And games/graphics of course
- Can do millions of operations in parallel—if they are all the same
- Theory model for how we can use GPUs?
- No, not yet
- One reason: algorithms is about cross-platform performance
- GPU implementations need to be extremely carefully tailored to the exact parameters of the GPU
- Less so as time goes on

Using Algorithms

- Asymptotically efficient method in terms of time
- Eye towards:
  - Other costs (like cache efficiency)
  - Abilities of modern hardware (parallelism, GPUs)

Wrapping Up CS 256

We Did It!

Algorithmic Design Paradigms

- Graph Traversals
  - Can do a lot with just BFS and DFS!
- Greedy Algorithms
  - Simple solution/ hard to show why they work
- Divide & Conquer/ Recurrences/Recursion Trees
  - Powerful design technique
- Dynamic Programming
  - Exponential-seeming problems with polynomial time recursive solutions
- Network Flows
  - Many complicated looking optimization problems in disguise
- Randomized Algorithms
  - Simple, elegant algorithms that work really well!
- Approximation Algorithms
    - Worst-case guarantees for intractable problems

Problem Solving Strategies

- How to make an abstract problem concrete
- How different parts of the problem commute/compose together
- **Problem reduction**: recognizing one problem in another
  - Reduce an instance of Problem **A** to Problem **B**
  - Use algorithm for Problem **B** to solve it
  - Why it works (necessary and sufficient conditions)

Biggest Takeaways:

- Learning to think algorithmically
- How to formalize your intuition

Flip Side of Design: Intractability

- How do we draw boundaries between what is computable and what is impossible
- No matter how much we'd want it, drawing these boundaries isn't easy
- Brings up the famous P versus NP question
- When we can’t prove something is unconditionally hard, we prove it conditioned on some conjecture we all believe is not true
    - SAT/ 3SAT, Independent Set, Vertex cover
    - Set cover, Hamiltonian Cycle/Path
    - Subset Sum, Graph Coloring, Clique
- Shows the power of mathematical characterization
- ***If you are CS major, you will see these again in CS 361***

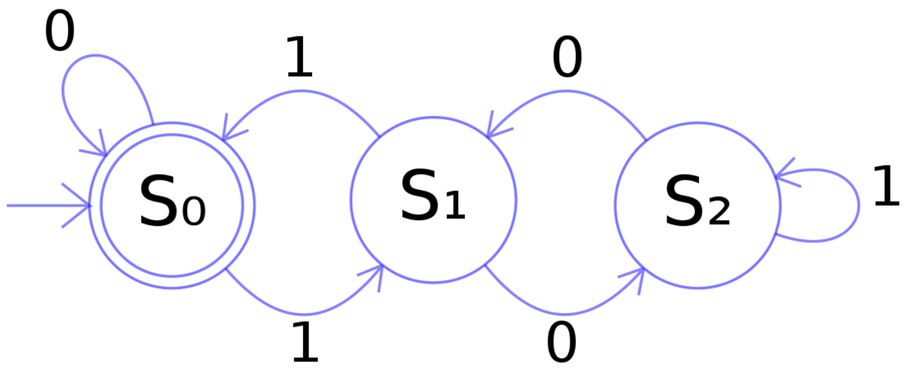
Other Courses at Williams

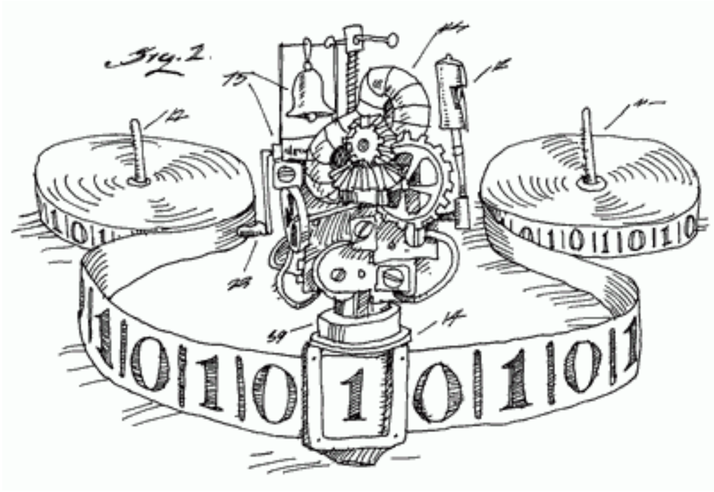
Beyond CS 256:
CSCI 361: Theory of Computation

Required course if you are CS major

- Formal framework for investigating both the computability and complexity of problems
- Several models of computation including finite automata, regular languages, context-free grammars, and Turing machines
- These models provide a mathematical basis for the study of computability theory

Prerequisite: CS 256

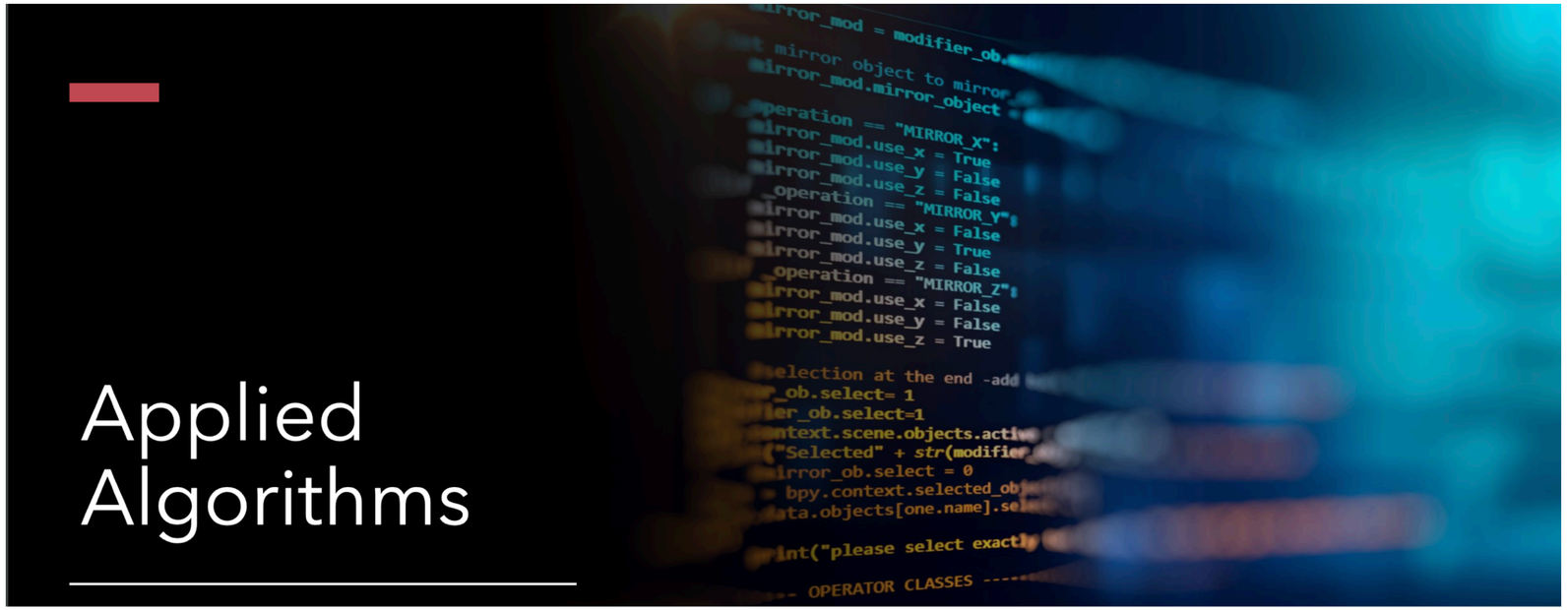




Beyond CS 256:
CSCI 358: Applied Algorithms

- Bridging the gap between theoretical running time and writing fast code in practice
- How to implement algorithms in an efficient way
- Learn new algorithmic techniques not covered in 256

Prerequisites:
CS 256 and CS237

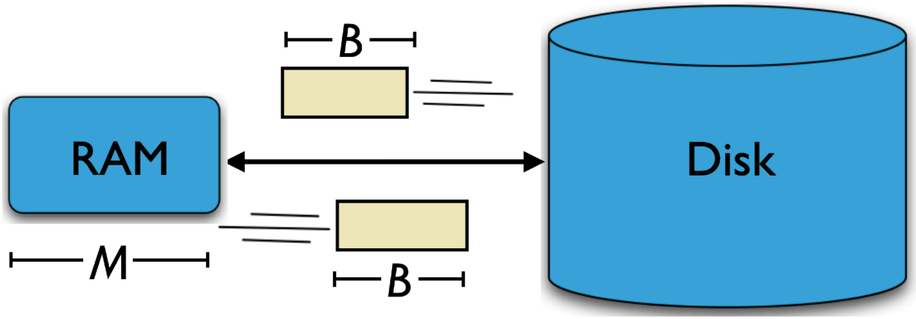


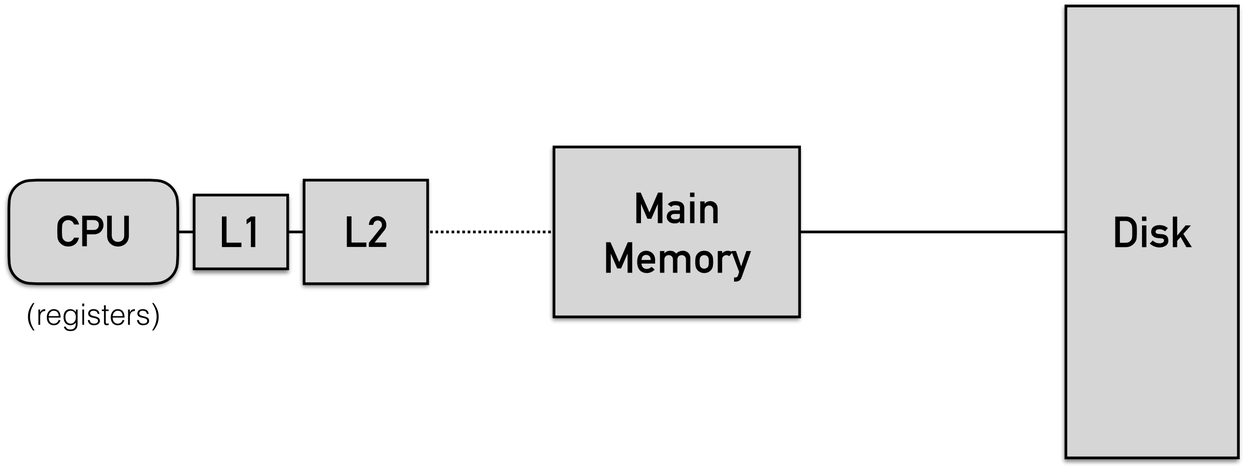
Beyond CS 256:
CSCI 333: Storage Systems

Focuses on interactions between storage and performance

- the memory hierarchy, storage hardware and its influence on storage software designs
- data structures; performance models; and system measurement/evaluation

Prerequisite: CS 237



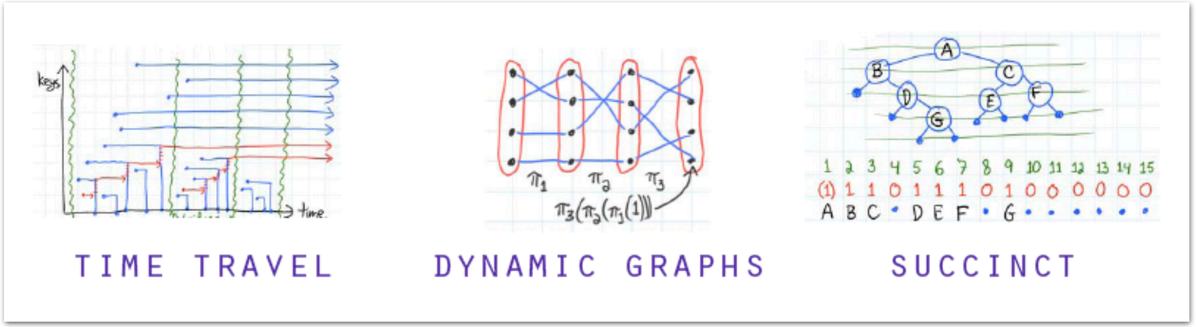


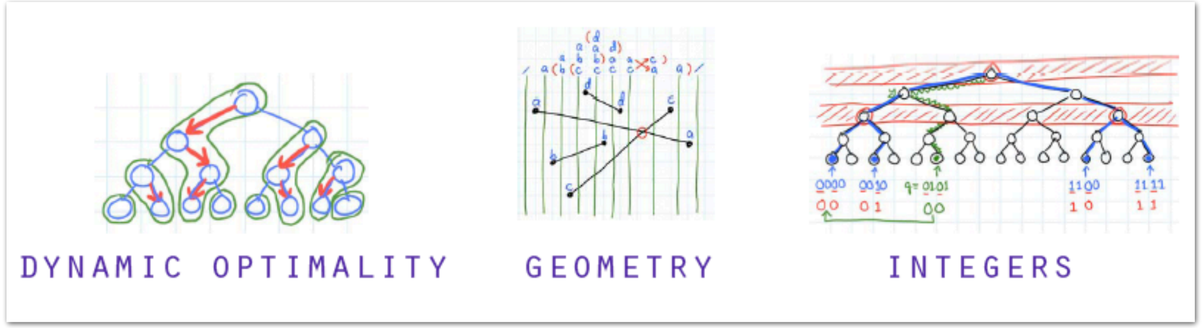
Beyond CS 256:
CSCI 356(T): Advanced Algorithms

Focuses on advanced concepts in algorithm design, algorithm analysis and data structures

- Randomized, approximation algorithms
- Geometric algorithms, advanced graph algorithms
- Combinatorial problems, linear programming, etc

Prerequisite: CS 256





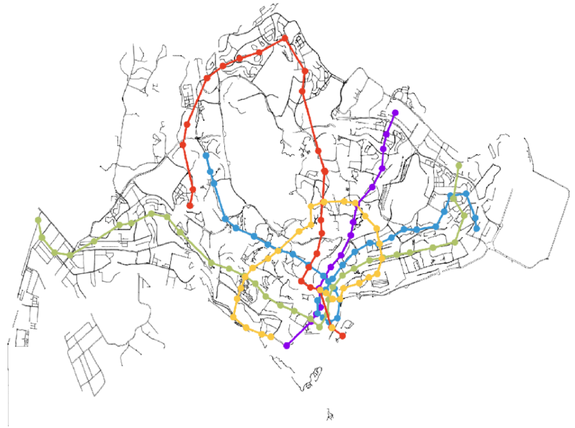
Source: https://courses.csail.mit.edu/6.851/spring12/

Beyond CS 256:
CSCI 357: Algorithmic Game Theory

Topics in game theory/mechanism design from a computational perspective

- how to design algorithms that incentivize truthful behavior, that is, where the participants have no incentive to cheat?
- Overarching goal is to understand and analyze selfish behavior and whether it can or should influence system design

Prerequisite: CS 256

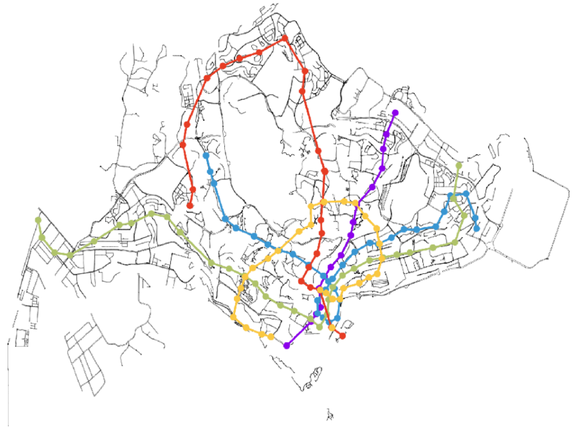


Source: https://courses.csail.mit.edu/6.851/spring12/

Beyond CS 256:
CSCI 357: Algorithmic Game Theory

"The next time we bemoan people exploiting loopholes to subvert the intent of the rule makers, instead of asking 'What's wrong with these people?" let's instead ask, `What's wrong with the rules?' and then adopt a scientifically principled approach to fixing them. "

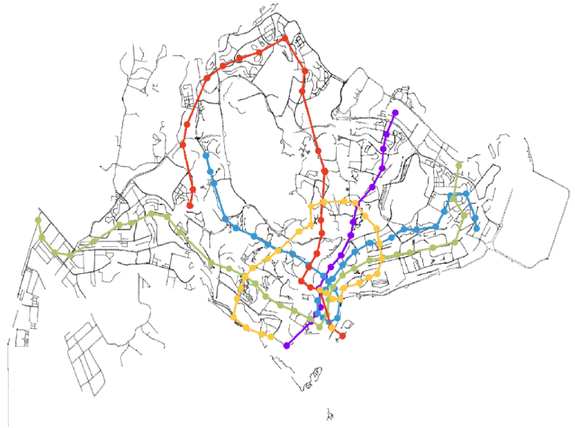
— Hartline and Kleinberg



Beyond CS 256:
CSCI 357: Algorithmic Game Theory

Examples of topics:

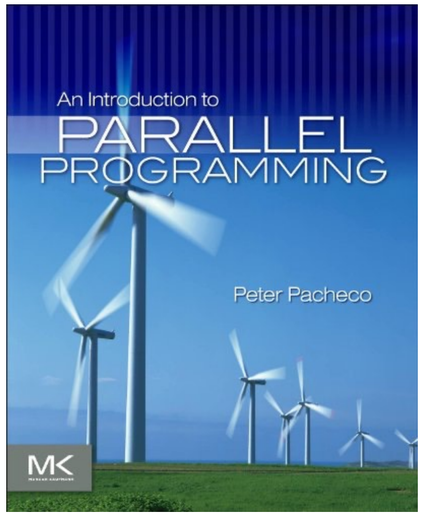
- auction design, efficiency of equilibria, network games
- two-sided markets, crowdsourcing markets
- incentives in computing applications such as file sharing, cryptocurrencies, etc
- computational social choice: selfish voting, resource allocation, etc.



Beyond CS 256:
CSCI 338: Parallel Programming

- How parallelism works
- Programming on a parallel machine
- GPUs
- High-performance computing clusters

CS256 not a prerequisite (237 is)



Course Evaluations: the Script

- Every term, Williams asks students to participate in end-of-semester course evaluations. Your feedback will help improve this course for other students taking it in the future, and help shape the computer science curriculum.
- You may skip questions that you don't wish to answer, and there is no penalty for choosing not to participate. All of your answers are confidential and I will only receive a report on your responses after I have submitted all grades for this course. While evaluations are open, I will receive information on how many students have filled out the evaluations, but I won't know which of you have and haven't completed them. I won't know which responses are associated with which student unless you identify yourself in the comments.
- To access the online evaluations, log into Glow (glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "Course Evaluations." Click on this and then follow the instructions on the screen. If you have trouble finding the evaluation, you can ask a classmate or reach out to Institutional Research at ir@williams.edu. You can complete the online evaluation through the end of reading period. If you haven't filled it out by the beginning of reading period, you will start receiving email reminders from Institutional Research.

Course Evaluations: Specifics

- Make sure you **fill out CS 256 course evaluations on GLOW**
- Right after this (during remaining class time) if you can!
- Student feedback is essential to improving the course

- “Blue sheets” are after the normal SCS questions I believe (hopefully they are clearly marked)
- “Blue sheets” only seen by me; the rest is seen by both me and the administration

All the Best
at
Williams and Beyond

pls fill out course evals

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)