# Review

## CS 624 — Analysis of Algorithms

May 7, 2024

# Final Exam Information

## Schedule

The final exam is Tuesday, May 14, 6:30pm–9:30pm (see WISER).

More details:

- Written exam. Bring a pen or pencil.
- Probably 5–6 questions.
- You may bring up to 40 pages of **handwritten notes**.
  (That is, 40 pieces of paper, up to letter size.)
  **No printouts, no photocopies.**
- No other resources: no books, no computers, no cellphones/smartphones/tablets, no friends.
- The exam will make up 40% of your final grade.
- You must pass the final exam to pass the course.

# Final Exam Questions

Content:

- new material since midterm 2
- topics from midterm 1 and midterm 2

Kinds of questions:

- run an algorithm, do a calculation
- solve a problem, invent an algorithm using familiar pieces
- analyze the running time or other property of an algorithm
- prove/argue something

# New Topics

- graph traversals
  - definitions of graphs, trees, paths, cycles, etc
  - breadth-first search (BFS)
  - depth-first search (DFS)
  - properties of BFS, DFS
  - topological sort
  - finding SCCs
- maximum flow
  - flow networks, flow, cuts,
  - residual graph, augmenting path
  - application: the marriage problem
- complexity classes
  - decision vs optimization problems
  - polynomial time reductions, $\leq_P$
  - the definition of class P
  - the definitions of class NP
  - the definitions of NP-hard and NP-complete
- known NP-complete problems:
  - SAT (really, CNF-SAT, 3CNF-SAT)
  - vertex cover (VC)
  - clique, independent set
  - subset sum
- how to prove a problem is NP-complete
  - SAT $\leq_P$ 3-SAT
  - 3-SAT $\leq_P$ VC
  - VC $\leq_p$ independent set $=_p$ clique

# Old Topics 1

- correctness of algorithms
  - proof by loop invariant
- asymptotic analysis / function growth
  - the definitions of $O, \Theta, \Omega$
  - solving recurrences to find bounds
    - using substitution + induction
    - using recursion trees
    - using master theorem
- sorting algorithms
  - InsertionSort, MergeSort, HeapSort, QuickSort
  - implementation of sorting algorithms, auxiliary algorithms
  - sorting as binary decision tree
- heaps
  - heap definitions, invariants
  - algorithms for heap operations
  - using heap operations
- medians and order statistics
  - algorithm based on QuickSort
- binary search trees
  - BST definitions, invariants
  - algorithms for BST operations
  - using BST operations
- general math knowledge and techniques
  - algebraic manipulation
  - proofs by induction
- invention of simple algorithms

# Old Topics 2

- binary search trees (BST)
  - BST definitions, invariants
  - the algorithms that implement BST operations
  - using BST operations
- dynamic programming
  - properties that enable DP
    - optimal substructure
    - overlapping subproblems
  - examples of DP problems and solutions: LCS, change-making, optimal BST, etc
  - top-down vs bottom-up memoization
  - analysis of run time, space
- greedy algorithms
  - properties that permit greedy solution
    - optimal substructure
    - “greedy choice” property
  - examples of greedy algorithm problems and solutions: Huffman codes, change-making (sometimes), activity selection (count)
  - analysis of run time, space
- amortized analysis
  - variants: aggregate vs accounting vs potential
  - examples of amortized analysis binary counter, dynamic tables (growable arrays), queues from stacks

# Notes on Answering Questions

- DO make sure to answer the question
  Suggestion: make a checklist of what to provide.
- DO show your work (can be helpful for partial credit)
- DON'T recite definitions that are in the book/lectures/etc
  - Merely remembering the things I've told you or the book has told you starts you off at 0%.
  - You get points for using them to answer new questions.
  - If it helps you to write it down, maybe put it in scratch paper.
- but DO refer to them and use them as appropriate
  - eg, "Since $f = O(n^2)$, that means there is $n_0$ and $c$ such that…"
  - eg, "Since $A$ is NP-complete, there is a polynomial-time reduction $f$ from SAT to A, so we can apply $f$ to…"

# Notes on Answering Questions

- DO make your strategy clear first
- DO mention properties/consequences relevant to the problem
  - eg, "After running RANDOMIZED-SELECT$(A, m)$, the $m$ least elements of the array are in $A[1..m]$."
  - eg, "BFS explores the entire component connected to the start node, and at the end every node in that component is colored black."
- DO describe your data, not just the actions your algorithm takes
  - eg, "$L$ is an array that maps vertex indexes to component labels."

# Course Evaluation

Link on course web page.

> *This evaluation is designed to provide information about the quality of this course to the instructor and department.*
>
> *All submissions will be anonymous and these questionnaires will be treated as confidential information. The instructor will not review your responses until after the final grades for this course have been submitted to the Registrar.*