Notes on the Midterm

- **Value:** 25% of your grade
- **Format:**
  - in-person, closed book
  - you will be allowed to use one A4 page of notes (front and back). If you type it, but please avoid too small fonts.
  - you will be provided with questions and papers to write your solutions.
- **When:**
  - 10:00am (Friday, Feb 23)
  - 1:20 hour for the exam
- **Helpful Resources:**
  - Our slides, recorded videos and your own lecture notes
  - Questions in our quizzes and assignments are good examples

Possible topics in Midterm

- Essentially everything discussed before our class in Feb 16 (i.e., not covering binary search trees), but the following topics are focused:
  - Asymptotic notations, their definitions, and their relations;
  - Relations between functions using asymptotic notations;
  - Runtime of loops. Given a loop or nested loop, determine its asymptotic runtime as a function of some variable *n* using asymptotic notations.
  - Loop invariants: You will be given a loop invariant, α, and you will use it to prove the correctness of a loop. (Coming up with loop invariants is an important skill, but not one that I will evaluate you on in a timed midterm.);
  - Linear data structures (i.e., arrays, linked lists, stacks, queues), possible implementations for their operations, and corresponding running time;
  - Priority queues (abstract data type) and implementation as sorted or unsorted list.
  - Binary heaps: their properties (e.g., heights, index rules for child/parent), representations (e.g., array vs tree), and implementations of their operations (e.g., insert, extract_max, increase_key, heapsort) and running time;
  - Designing efficient algorithms for some problems employing the data structures/procedures (e.g., tree traversal) we discussed so far.
  - Tree traversals and their applications to design algorithms (i.e., in-order, pre-order, and post-order traversals).

Not covered in Midterm

- Formal proofs for asymptotic statements
- Formal proofs of amortized analysis
- Ordered tree representation
- Binary Search Trees