# CSE 373

## JUNE 2ND – EXAM REVIEW

# ASSORTED MINUTIAE

- **Exam Review – Today 4:30 – 6:00 EEB 105**
- **HWs 5 and 6 back this weekend**
- **Submit regrade requests for before exam time**
- **Old patches gone through, recheck grades**
- **Extra assignment due tonight at midnight**
  - No late days allowed
  - “Closes” at 12:30, but anything after 12:00 is up to my judgement

# ASSORTED MINUTIAE

- **Course evaluations**
  - Very important to this class and this department
  - Above all, they're very important to me
  - Should only take ~5 minutes, and it's very valuable feedback

# TODAY'S LECTURE

- **Exam Review**
  - Important topics
  - Exam is comprehensive, but review will focus on the new material

# EXAM FORMAT

- 1:50 to complete 12 problems
- First question is short answer, which has many parts of varying difficulty, it is not likely to be the easiest
- Runtime and debugging questions
- Technical questions
- Algorithm Design question

# EXAM FORMAT

- **We will be our most strict grading yet, don't make any assumptions that aren't explicit**
- **Analysis work needs to be thorough and concrete, recurrences and summations will likely be required**
- **Show all of your work. Many algorithms are trivial to solve by hand. Just providing "the solution" will not earn points. Algorithms are about process.**

# EXAM FORMAT

- **A time crunch is likely**
  - There are many topics that need to be covered
  - Get down things that you know, and if you don’t make progress move on and come back

# TOPICS

- **Definitions**
  - ADT – Abstract Data Type – Describes a certain set of functionality and behavior
    - e.g. PriorityQueue
  - Data structure – Theoretical storage method that implements an ADT.
    - e.g. Heap
  - Implementation – Low-level design decisions that are often language dependent
    - e.g. Using an array for the heap

# TOPICS

- **Stacks and Queues**
  - LIFO and FIFO ordered storage respectively
  - Can be implemented with arrays or linked lists
  - Understand the desired behavior and how to implement these structures

# TOPICS

- **Priority Queues**
  - Insert(key, priority)
  - findMin()
  - deleteMin()
  - changePriority()

# TOPICS

- **Heaps**
  - Usually array implementations
  - Heap property
  - Complete trees
  - Runtimes and buildHeap()

# TOPICS

- **Algorithm analysis**
  - bigO, bigOmega, bigTheta
  - c and $n_0$
  - Asymptotic behavior
  - Memory analysis
  - Recurrences
  - Summations

# TOPICS

- **Dictionary**
  - ADT- insert(k,v), find(k) delete(k)
  - Many possible underlying data structures
  - Different runtimes (and support)

# TOPICS

- **Binary search trees**
  - Best and worst case
  - Traversals
- **Balance property – AVL**
  - Rotations and correctness

# TOPICS

- **Hashtables**
  - Linear, quadratic, secondary hashing
  - Separate chaining
  - Load factor and resizing
  - Primary and Secondary clustering
  - Runtime and memory constraints

# TOPICS

- **Graphs**
  - Notation G(V,E)
  - Traversals
  - Topological Sorts
  - Properties
    - Directed v. Undirected
    - Dense v. Sparse
    - Weighted v. Unweighted
    - Cyclic v. Acyclic

# TOPICS

- **Graphs**
  - Algorithms
    - Dijkstra's – path finding
    - Prim's and Kruskal's – Minimum spanning trees
  - Know their runtimes and the data structures they rely on for those runtimes…

# TOPICS

- **Iterators**
  - hasNext(), next()
  - Can iterate over any domain
  - Usually helpful to get connected and relevant data together
  - Can break up processing for each call, rather than doing all the processing at once
  - May not always be advised

# TOPICS

- **Union find**
  - ADT – Disjoint sets
  - Partitions
  - Weighted Union
  - Path compression
  - Uptree – single array representation

# TOPICS

- **Sorting**
  - Insertion and Selection
  - Heap, Merge and Quick
  - Bucket and Radix
- **Properties**
  - Comparison sorts
  - Stable
  - In place
  - Interruptible (top k)

# TOPICS

- **Analysis**
  - Lower bound for comparison sorts
  - Memory usages for sorting
  - Best and worst case runtimes

# TOPICS

- **Testing**
  - White box v. Black box
  - Identifying edge cases
  - Difficulties and techniques
- **Debugging**
  - Programming process
  - Understanding code and potential problems

# TOPICS

- **Memory**
  - Temporal and Spatial localities
  - Pages and their use
  - Tiered caching
  - Impact on cloud computing

# TOPICS

- **Algorithm Design**
  - How can you approach the problem?
    - Guess and check (Approximation)
    - Brute Force (Linear Work)
    - Divide and Conquer
    - *Greedy algorithms (make best decision for a local sub-problem)*
    - Randomization, Las Vegas and Monte Carlo
    - Preprocessing

# TOPICS

- **Algorithm Design**
  - Find an approach to the problem that finds the solution
  - Understand what the edge cases are
  - Be able to analyze best-case, worst-case and memory usage of your algorithm
  - Randomization is okay if you can show it's faster than a more clever solution.

# STRATEGIES

- **Go through the exam from easiest to hardest**
  - Problems in the middle may be the easiest
- **Be as thorough as possible, if you think it's relevant and correct, include it**
- **Algorithm Design problem is as much about analysis as it is about clever solutions, so don't leave that done poorly**
- **Think about what things make certain algorithms tricky – highly likely for this final**

# FINAL WORDS

- **Great quarter!**
- **Stressful week**
  - Nothing feels better than walking out of an exam and…
  - Filling out course evaluations!
- **Course has been tough**
  - But you have learned a lot
  - and you're going to show us on Tuesday

# FINAL WORDS

- Good luck!
- Have a nice summer!