

# Final Review

---

## 1. REVIEW TOPICS

- Java basics
- Sorting and searching
  - linear and binary search
  - bubble sort, insertion sort, selection sort
- Linked lists
  - lists vs. arrays
  - operations on lists and analysis
  - singly LL, doubly LL, circular lists
- Program analysis
  - growth rate: big-Oh, big-Theta
  - finding the order of growth of an expression
  - analyzing running times of algorithms
  - comparing (running time of) algorithms
- Recursion
  - simple recursion examples
  - towers of Hanoi
  - blob counting, flow, maze
  - generating permutations, subsets, subset sum
- Stacks and queues
  - functionality
  - implementation with vectors and lists
- Searching with stacks and queues
  - the general framework
  - breadth-first search and depth-first search
  - trade-offs between DFS, BFS
  - examples: missionary and cannibals puzzle, maze
- Recursive solutions with marking
  - flow, blob count, maze, percolation
- Maps and hashing
  - operations supported by a map
  - hashing and collisions with chaining, open addressing
  - load factor and performance
  - what is expected of a good hash function
- Trees
  - computing height, level, size
  - traversals (post-order, pre-order)
- Binary search trees
  - definition
  - in-order traversal

- height as function of  $n$  (size)
- basic operations: search, insert, delete
- more operations: min, max, successor, predecessor
- sorting with a binary search tree
- Comparison of hash tables and binary search trees.
- Python

## 2. COURSE OUTCOMES

After this class you should be comfortable with the fundamental computer science algorithms and data structures, be able to use them to model and solve a problem, discuss their efficiency, be able to go from concepts to details, from theory to practice and implement a problem from scratch, and be able to debug your code.

More precisely,

- Knowledge of the fundamental data structures (arrays, vectors, lists, stacks, queues, trees, binary search trees, maps, hash tables) and basic algorithmic techniques (recursion; divide-and-conquer; backtracking, breadth- and depth-first search).
- Ability to analyze the asymptotic performance of fundamental data structures and discuss which structure is better in what circumstances and what are the trade-offs.
- Ability to use a data structure without knowing its implementation, just its interface.
- Knowledge of the efficient implementation of the fundamental data structures and what lies behind Java class like Stack, Vector, etc when you use them in your code.
- Familiarity with the general ideas for sorting (insertion sort, selection sort, bubble sort, merge-sort) and searching (linear search, binary search, binary search trees, hashing)
- Problem solving: the ability to approach a problem and break it into simpler blocks
- And last but not least, the ability to implement your code in Java, search the Java doc files, debug and get it to work.