1 Administrative

- Midterm Exam 1: Wednesday 10/26
- HW 01 and 02 solutions posted to Blackboard (tonight)

2 Midterm Exam 1

- Written exam (Bring a pen or pencil!)
- Probably 5 questions
- Allowed to bring printouts of lecNN.pdf files ([auxNN] links)
  - Must not have any other writing/printing on them.
  - Must not write on them during the exam.
  - I will conduct spot checks during the exam.
- No other resources are allowed.
  - No books or other notes.
  - No electronic devices.

Topics:

- correctness of algorithms
  - loop invariants
- asymptotic analysis (growth of functions)
  - the definitions of different bounds ($O$ vs $\Theta$ vs $\Omega$)
  - solve recurrences to find asymptotic bounds
    * using substitution + induction
    * using recursion trees
* using master theorem

• sorting algorithms
  – insertion-sort, merge-sort, heap-sort, quick-sort
  – implementation of sorting algorithms
  – properties of sorting algorithms
  – sorting viewed as binary decision diagram (tree)

• heaps
  – heap definitions, invariants
  – the algorithms that implement heap operations
  – using heap operations

• medians and order statistics
  – algorithm based on quick-sort

• binary search trees
  – BST definitions, invariants
  – the algorithms that implement BST operations
  – using BST operations

• general mathematical knowledge and techniques
  – algebraic manipulation
  – proofs by induction

• invention of simple algorithms

Not covered:

• generating functions
• specific summation formulas
• bucket-sort
• median-finding algorithm with $O(n)$ worst-case time
• dynamic programming

3 Dynamic Programming

3.1 Slide 11

\[
\text{MakeChange(int[]} \text{ CoinValues, int goalValue)} ::=\]
// solutions : int[]
solutions ← new int[0..goalValue] initially ∞
solution[0] ← 0

for k ← 1 to goalValue:
    // Invariant: for every j in [1..k-1], solutions[j] contains
    // the optimal number of coins needed to sum up to j cents
    for cv in CoinValues
        if cv ≥ k
            newSolution ← solutions[k-cv] + 1
            solutions[k] ← min(solutions[k], newSolution)
        endif
    end for
end for
return solutions[goalValue]

3.1.1 continue

Two ways to fill in solutions array:

- “bottom-up”
- “top-down” / “on demand” / “memoization”