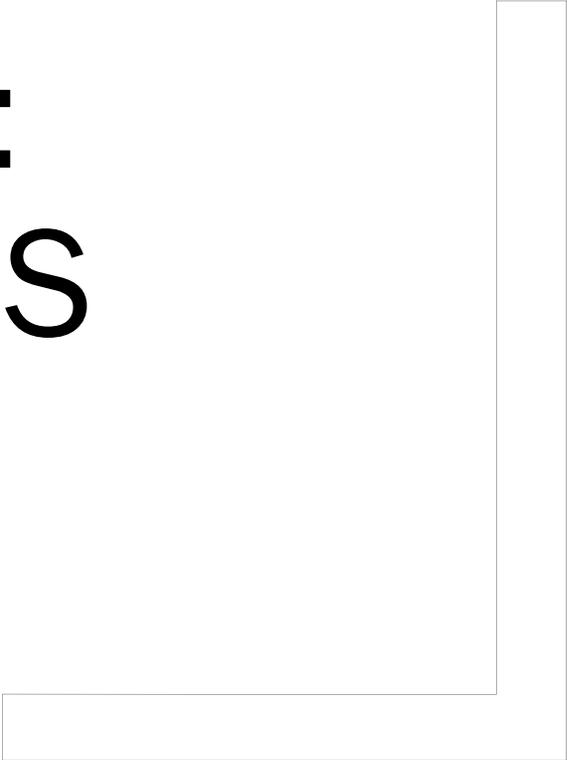


# CSCI 3100: ALGORITHMS

Kate Holdener, Ph.D.  
[cs.slu.edu/~holdener/csci3100](http://cs.slu.edu/~holdener/csci3100)



Discrete Math:

- Proofs
- Induction
- Sets
- Sums and Products
- Algorithms
- Complexity
- Graph theory



Data Structures:

- Trees
- Stacks
- Queues
- Graphs
- Dictionaries

# What we will be doing this semester

- Analyze why a given algorithm works for a given problem
- Prove that a given algorithm works correctly
- Write algorithms to problems in pseudocode
- Study various algorithms on familiar data structures
- Implement some of the algorithms in real code
  - *Git repository*
  - *More about this later*
- Group activities in class
- Class participation is important (10% of your grade)

# Course specifics

- Course web site: [cs.slu.edu/~holdener/csci3100](http://cs.slu.edu/~holdener/csci3100)
- Read the syllabus
- Some syllabus highlights:
  - *Grading*
  - *Textbook*
  - *Attendance*
    - Fall/Spring semester \$21,850 (12-18 credit hours)
    - This class for the semester:  $(\$21850/18)*3 = \$3641.6667$
    - One class:  $\$3641/43 = \$84.69$
- Academic integrity

# Things you should know:

## Ch. 1, 2, 3, 6, 7, 10, 12

- Fundamental data structures
- Asymptotic notation: big-O
- Sorting algorithms
- Binary search trees
- Writing proofs
- Induction

# Find 7

- Do not shuffle your cards
- Keep your cards face down
- Goal: find a card with a number 7 on it.
- Constraints:
  - *Look at one card at a time*
  - *Keep track of the number of cards you looked at.*
  - *Pretend, you are charged \$1 for each card you look at.*

# Algorithm complexity

- Pair up and talk about:
  - *The exact steps you took to find the card*
  - *How many cards you looked at*
  - *What is the best case scenario (min number of cards you'd need to look at to find 7)?*
  - *What is the worst case scenario (max number of cards you'd need to look at to find 7)?*

# Bounds

- $O(f(n))$  – Big-O – tight upper bound on the growth of the algorithm
- $o(f(n))$  – little-o – loose upper bound on the growth of the algorithm
- $\Omega(f(n))$  – big-Omega – tight lower bound on the growth of the algorithm
- $\omega(f(n))$  – little-Omega – loose lower bound on the growth of the algorithm

# Bounds on the “find 7” problem

- Big-O:  $n$
- Little-o:  $n^2$
- Big-Omega:  $n$
- Little-Omega:  $1$
- $\Theta(f(n)) = g(n)$  iff
  - $g(n)$  is  $O(f(n))$  AND
  - $g(n)$  is  $\Omega(f(n))$
- $\Theta(f(n))$ :  $n$

# Merge Sort

- Keep your cards face down
- Goal: cards are sorted
- Algorithm:
  - *Flip over one card at a time and insert into the new deck in a sorted position*
  - *Keep the sorted deck face down*
  - *Keep track of the number of cards you looked at*
  - *Pretend you are charged \$1 for each card you look at*

# Pair up and talk about

- How you inserted the card in the sorted position
- How many cards you looked at
- What if the deck was larger, how would you insert the card in the sorted position?
- What is big-O of this algorithm

# Assignments

- Read the syllabus
- Initial Assessment Survey (extra credit)
- Review chapters with pre-requisite material