CSC 6580 – Design and Analysis of Algorithms

Overview and Background

Instructor:

Name: Dr. Abusayeed Saifullah
Office address: 5057 Woodward Ave. Suite 14110.2
Office hours: Monday 3:00pm—4:00pm
Phone: 313 577 2831
Email: saifullah@wayne.edu

Teaching Assistant:

Name: Xiangrui Li
Office address: 5057 Woodward Avenue, Rm. 2211 (2nd Floor)
Office hours: 11:00am – 12:00pm, Tuesday and Thursdays or by appointment
Email: fx7219@wayne.edu

Course Description:

Best case, worst case, and expected case complexity analysis; asymptotic approximations; solutions of recurrence equations; probabilistic techniques; divide-and-conquer; the greedy approach; dynamic programming; branch and bound; NP-completeness; parallel algorithms.

Supplementary information for the course is available at http://canvas.wayne.edu. Log on with your Access ID for class notes, lecture slides, class announcements, the course syllabus, and other information for the course.

Credit Hours: 3 Credit Hours (Lecture)

Prerequisite:

CSC 3110: Algorithm Design & Analysis (or equivalent undergraduate "algorithms and data structure" course) is the official "hard" course prerequisite. A moderate level of mathematical maturity is required in this course. A significant portion of the homework and exams will require formal proofs.

Text(s) Book:

Introduction to Algorithms. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein. MIT Press, 3rd edition (2009). ISBN 9780262033848. (**Required**)

Computers and Intractability: A Guide to the Theory of NP-Completeness, Garey and Johnson, 1979. (Optional reference; on reserve at UG Library)

The Design and Analysis of Computer Algorithms, Aho, Hopcroft, and Ullman, 1974. (Optional reference; on reserve at UG Library)

Course Contents:

Randomized algorithms and probabilistic analysis (e.g., quicksort and skiplists); dynamic programming; greedy algorithms; amortized analysis; advanced data structures (e.g., Fibonacci heaps); graph algorithms; intractability theory; and approximation algorithms. A schedule of topics and reading assignments may be found on the Canvas website. Please check this site often for any changes to the schedule or announcements.

Course Learning Objectives:

Upon successful completion of this class, the student will be able to:

#	CSC 6580 Course Learning Objectives
1	Use formal proof techniques to argue about the correctness of an algorithm.
2	Apply asymptotic analysis for the (worst-case, best-case, or average-case) analysis of algorithm time or space complexity.
3	Analyze randomized algorithm and/or data structures by using probability theory to derive asymptotic bounds on expected run times.
4	Use optimal substructure to develop either dynamic programming or greedy algorithm solutions for optimization problems.
5	Employ different methods of amortized analysis (aggregate analysis, accounting, and potential method) when analyzing data structure operations.
6	Synthesize new graph algorithms and algorithms that employ graph computations as key components, and analyze them.
7	Define NP-completeness and synthesize new proofs of NP-completeness.

Assessment:

Two Midterm Exams: 40%
Final Exam: 40%
Homeworks (approximately 5): 20%

Grading Scale:

90-100%	A	65-69%	C+
85-89%	A-	60-64%	C
80-84%	B+	55-59%	C-
75-79%	B	54 or Below	F
70-74%	B-		

Grading Policies:

Grades will not be curved for the course, and you will receive the grade that you earn through your performance on the homeworks and exams. For letter grading, if a total score (in scale of 100) is non-integer, then it will be converted to the nearest integer. There will be no individual exceptions to the grading policy, and, therefore grades of a **C** or **F** are possible (even for graduate students). Please be sure that you have the appropriate background (i.e., you did sufficiently well in your undergraduate algorithms course) for the course. In addition, the course material, homeworks, and exams are expected to be quite challenging and time-consuming; you should ensure that you fully devote the proper amount of time to understanding the topics from lectures and homeworks.

Any re-grading request must be made within one week upon receiving the grade. A regrading request made after one week upon receiving the grade will not be entertained.

Exams:

All exams are in-class, and closed-book and closed-notes. We will have two (non-cumulative) midterm exams (one in mid-February and the other in mid March); the dates of these exams will be announced early in the semester. We will have a cumulative final exam. The location and time of the final exam for this course is set by the WSU Final Exam Schedule which is **Monday, April 27, 2019 from 10:15 am – 12:15 pm**; the exam will be held in the same location as class. Missing any exam without prior approval by the instructor will result in a zero for that exam.

Class Attendance/Participation:

Lecture attendance is *mandatory*. If you need to miss a lecture (for a valid reason recognized by the University), you must notify me in advance of the lecture. A course in which students attend and actively participate in the discussion of ideas is always much more enjoyable and stimulating. I plan to reward those who participate in class by increasing their final grade by up to half a letter grade. I also reserve the right to similarly decrease the grade of those who routinely come to class late, skip class, surf the internet, talk, text/chat, or sleep in class, etc.

Homework Policy:

Homework assignments will be given every two weeks and are primarily for your practice in the material. (Please note that homeworks only constitute 20% of your total grade). Students can expect most homework assignments to be very time-consuming; however, if you spend time to learn the concepts from the homeworks, exam questions should be much easier to answer. Collaboration during exams or in homeworks or unacceptable collaboration or plagiarism on homeworks will also be considered cheating. A student caught cheating will receive a grade of "F" for the course and face charges with the University Judicial Officer. For more on what constitutes ``acceptable collaboration'' in this course and academic misconduct in general, see the collaboration policy document in Canvas under ``Syllabus", the University's Student Code of Conduct https://doso.wayne.edu/pdf/student-code-of-conduct.pdf), and *Academic Dishonesty* section in this syllabus (below).

Homework assignments are due in class at the beginning of lecture on the due date given. *No late homeworks will be accepted without prior approval*.

Chapter 3

Growth of functions

- $O \approx \leq$
- $\Omega \approx \geq$
- $\Theta \approx =$
- $o \approx <$
- $\omega \approx >$

Common functions.

- e.g., n, lg n, n^2, n!

Chapter 4

- **Chapter 4:** Divide and Conquer
- Divide the problem into a number of subproblems that are smaller instances of the same problem.
- Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
- Combine the solutions to the subproblems into the solution for the original problem.

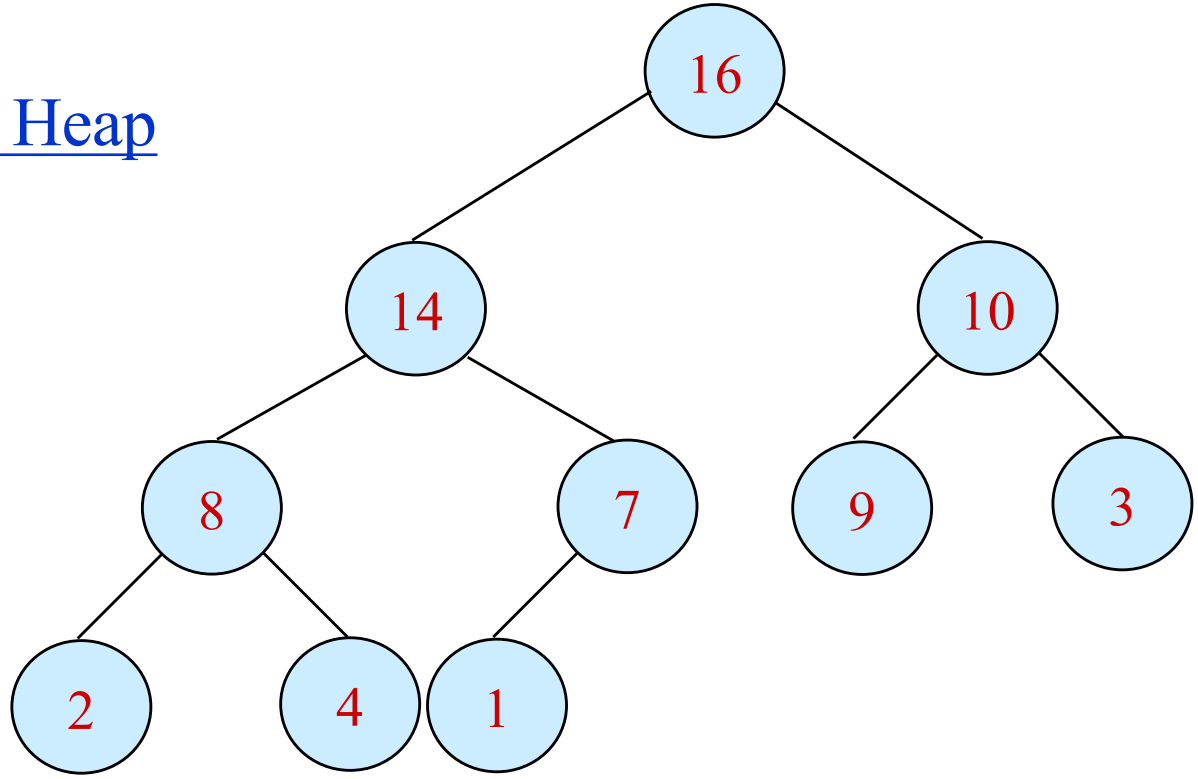
Chapter 5

- **Chapter 5:** Probabilistic Analysis and Randomized Algorithms.
  - Randomized Algorithm: Algorithm that "flips coins".
  - Probabilistic Analysis: Use of probability to analyze algorithms running-time.

Chapter 6: Heapsort

- **Chapter 6:** Heapsort.
  - Can use heaps to:
    - Sort in $\Theta(n \lg n)$ time.
    - Implement a priority queue in $\Theta(\lg n)$ time.

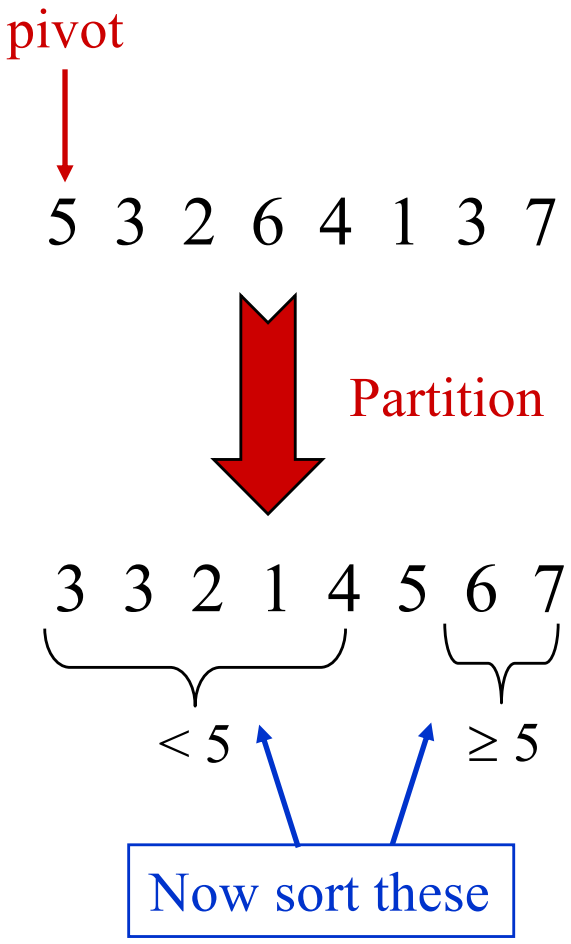
A Heap



Chapter 7: Quicksort

Chapter 7: Quicksort.

- Partition based on **pivot**:



Can choose pivot at **random**.

- **Worst-Case:** $\Theta(n^2)$
- **Average-Case:** $\Theta(n \lg n)$
- Usually **very** fast

Chapter 8: Sorting in Linear Time

- $\Omega(n \lg n)$ lower bound for comparison sorts.

- Can do it in $\Theta(k + n) = \Theta(n)$ time, where "digits" range over $1, \ldots, k$.

A **constant**

Chapter 9: Medians and Order Statistics

- i^{th} order statistic ≡ i^{th} smallest element.
  - Min: $i = 1$.
  - Max: $i = n$.
  - Median:

$$i = \begin{cases} (n+1)/2 & \text{if } n \text{ is odd} \\ n/2 \text{ or } n/2+1 & \text{if } n \text{ is even} \end{cases}$$

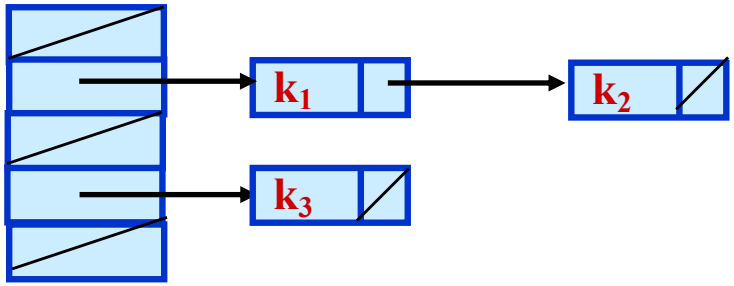
- Can compute i^{th} order statistic in $\Theta(n)$ time.

Chapter 10: Elementary Data Structures

- Stacks, queues, linked lists, arrays, trees, …

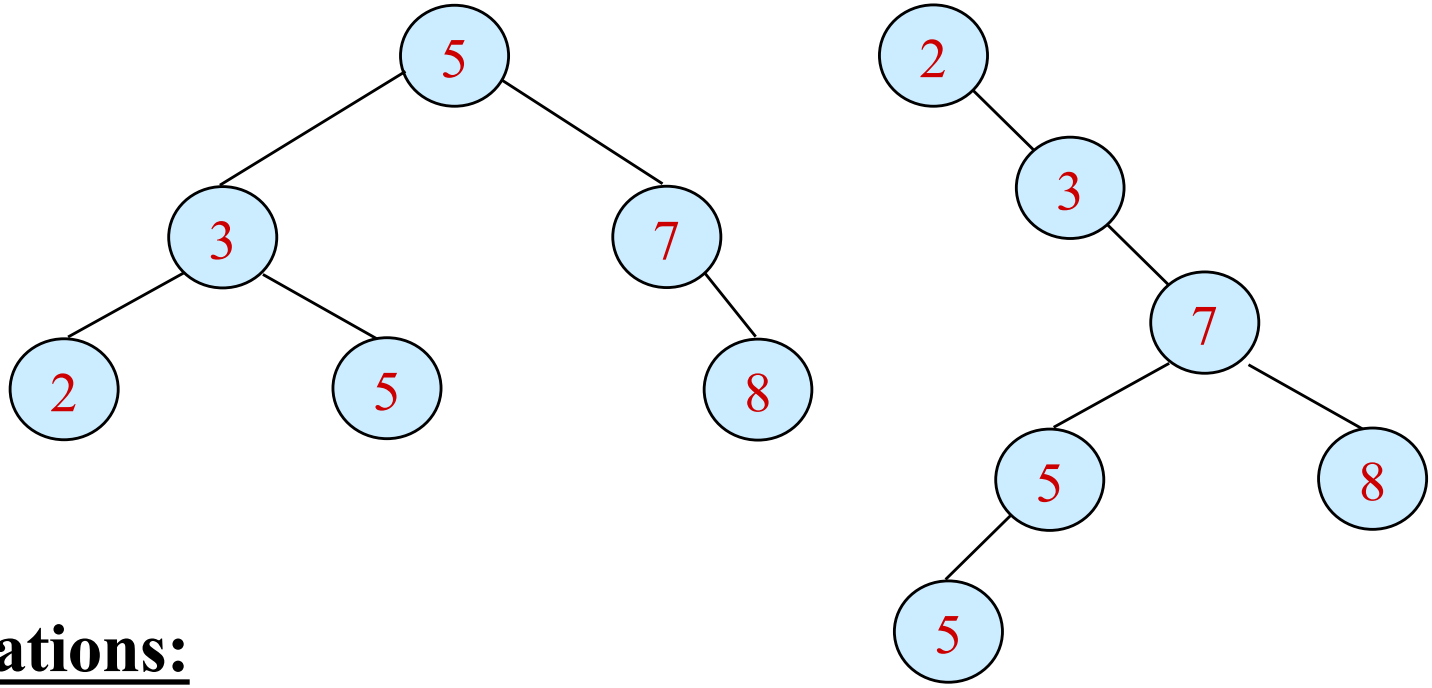
Chapter 11: Hashing

- Many applications require a dynamic set that supports only the dictionary operations INSERT, SEARCH, and DELETE.
- A hash table is an effective data structure for implementing dictionaries.
- Hash functions.
- Collision resolution by chaining



Chapter 12: Binary Search Trees

Might be unbalanced:



Operations:

Search	Successor	Delete
Min	Predecessor	
Max	Insert	

Randomly-Built BSTs: Expected height is O(lg n).

Chapter 13: Red-Black Trees

- **Balanced** BSTs.
- Use **colors** to maintain balance.
- **Red-Black Properties:**
  - Every node is either red or black.
  - Every leaf (NIL) is black.
  - If a node is red, then both its children are black.
  - Every simple path from a node to a descendent leaf contains the same number of black nodes.

Chapter 15: Dynamic Programming

- Dynamic programming, like the divide-and-conquer method, solves problems by combining the solutions to subproblems.
- Exploit optimal substructure property to solve a problem
- Typically applied for optimization problems

Chapter 16: Greedy Algorithms

- A greedy algorithm always makes the choice that looks best at the moment.
- That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.
  - Easy to design
  - Not necessarily an optimal choice globally
  - Some times optimal

Chapter 17: Amortized Analysis

- Analyzing average the time required to perform a sequence of data-structure operations over all the operations performed.

Chapter 18: B-Trees

- Used in disk/storage devices
- Balanced Trees
- Like red-black tree but better in disk I/O operation

Chapter 19: Fibonacci Heaps

- Advanced data structure with very efficient operations.

Chapter 22: Elementary Graph Algs.

- Breadth-first and depth-first search.
  - Lots of properties to know …

- **Applications:**
  - Topological sort.
  - Strongly connected components.

Chapter 23: Minimum Spanning Trees

- Kruskal's algorithm.
- Prim's algorithm.

Chapter 24: Single-Source Shortest Paths

- Dijkstra's algorithm.
- Bellman-Ford algorithm.

Chapter 25: All-Pairs Shortest Paths

- Finding shortest paths between all pairs of vertices in a graph.
- Matrix multiplication
- Dynamic programming

Chapter 26: Maximum Flow

- Flow networks: model many problems, e.g., liquids flowing through pipes, current through electrical networks, and information through communication networks.
- We can think of each directed edge in a flow network as a conduit for the material.
- Each conduit has a stated capacity
- Compute the greatest rate at which we can ship material from the source to the sink

Chapter 34: NP-Completeness

- Turing machine
- NP-Hard
- NP
- NP-Complete

Chapter 35: Approximation Algorithms

- Algorithms with approximation bound.

For Next Class…

- **Evaluate your Background**:
  - Gauge background.
  - Review undergrad algorithms material as necessary.
- **Reading Assignment**:
  - Review carefully Chapters 1, 2, and 3.
  - Refresh your knowledge on O-notation