CSE 431/531: Analysis of Algorithms (Summer 2023)
Introduction

Chen Xu

May 30, 2023

Course Information

- **Course Instructor:** Chen Xu
- **Grader:** TBA
- **Course Timing:** Tue/Thu 10:00 - 11:20 a.m. EST, 12 weeks (05/30/2023 - 08/18/2023)
- **Course Webpage:** https://cse.buffalo.edu/ chenxu/teaching/cse431531/
- **Zoom Link:** Please check the announcement on UBLearns Brightspace.
- **Piazza Link:** https://piazza.com/buffalo/summer2023/cse431531

Prerequisites

You are expected to have:

- Basic knowledge of data structure
- Basic logic reasoning skills
- Basic programming skills

These prerequisites form the foundation upon which we will build our understanding of algorithm analysis. If you feel you may need a refresher on these topics, I encourage you to review relevant materials before we delve into the course content.

Contents

This course introduces the principles of algorithm analysis, including time and space complexity, as well as the basic concepts of algorithm design. The topics we will be covering include:

- Time and Space Analysis
- Graph Basics
- Greedy Algorithms
- Divide and Conquer Algorithms
- Dynamic Programming
- Graph Algorithms
- NP-Completeness

These subjects form the foundational knowledge for understanding and designing effective algorithms.

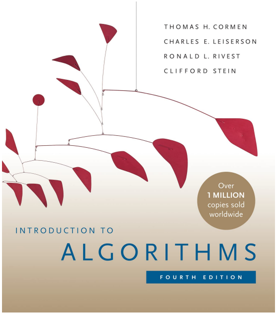
Tentative Schedule

Date	Topic	Slides	Homework	Project
5/30	Intro			
6/1	Asymptotic analysis			
6/6	Recursions and master theorem			
6/8	Graph representation and basic graph algorithms		HW1 released	
6/13	Greedy 1			
6/15	Greedy 2		HW1 deadline	
6/20	Greedy 3		HW2 released	
6/22	Divide and Conquer 1			
6/27	Divide and Conquer 2			Project 1 released

Tentative Schedule

Date	Topic	Slides	Homework	Project
6/29	Divide and Conquer 3		HW2 deadline, HW3 released	
7/4	Holiday			
7/6	Dynamic Programming 1			
7/11	Dynamic Programming 2			
7/13	Dynamic Programming 3		HW3 deadline	
7/18	Dynamic Programming 4		HW4 released	Project 1 deadline
7/20	Graph Algorithms 1			Project 2 released
7/25	Graph Algorithms 2			
7/27	Graph Algorithms 3		HW4 deadline	
8/1	NP-Completeness 1		HW5 released	
8/3	NP-Completeness 2			
8/8	NP-Completeness 3			
8/10	Review		HW5 deadline	Project 2 deadline
8/17	Final exam			

Textbook



- **Introduction to Algorithms (4th edition)**

Grading Weights

- **4 Quizzes:** 20%
- **Final Exam:** 20%
- **5 Homeworks:** 30%
- **2 Projects:** 30%

Guidelines for Quizzes

- Quizzes will be held on UBLearns Brightspace during the lecture time.
- It is usually 30 minutes.
- There will be a variety of problem types including multiple choice, true/false, and short answers, etc..
- Some questions may be based on lecture videos.
- Top 4 scores from the 5 quizzes will be used for grading.

Homework Guidelines - Do's

- Homeworks will be out bi-weekly, start early and complete assignments on time.
- Make sure to understand the problem statement.
- Discussion among classmates is allowed. Mention their names in your submission.
- Seek help on Piazza if needed.

Homework Guidelines - Don'ts

- Do not collaborate in writing the assignment. Write solutions independently.
- Do not submit someone else's work as your own.
- Do not use online resources.

Project Guidelines

- Cannot copy code from other sources. Must be implemented by yourself.
- We use Moss for similarity detection in programs. https://theory.stanford.edu/ aiken/moss/

Final Exam Guidelines

- The exam will be held via Zoom.
- It is a closed book, timed exam.
- The face camera is required during the exam period.

Late Policy

- Each student has one late credit.
- This allows a one-time extension of 3 days on homework assignments only. (Not projects)
- Late submissions beyond this will receive a score of 0.

Academic Integrity

- Violations of academic integrity are not tolerated.
- Violations result in failing the course and being reported to the departmental AI committee.

What is an Algorithm?

- Quote from Donald Knuth: "An algorithm is a finite, definite, effective procedure, with some input and some output."
- Algorithm as a set of instructions designed to accomplish a specific task.

Computational Problems and Algorithms

- Computational problem defines a task to be performed, typically specifying the relationship between given inputs and the desired output.
- Algorithms come into play as step-by-step procedures to solve these computational problems. Given the inputs, an algorithm will follow its set of instructions to produce the desired output.

Pseudo-code

- Pseudo-code: Human-readable algorithm representation with simplified programming syntax.

```
factorial(n):
if n = 0 then
    Output: 1
else
    Output: n * factorial(n - 1)
end if
```

Algorithms, Computer Programs, and Artificial Intelligence

- Computer programs are practical implementations of algorithms.
- AI is a collection of advanced algorithms to mimic human behavior.

Example 1: Binary Search Algorithm

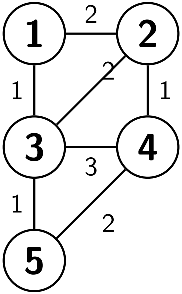
- Input: Sorted array [2, 3, 4, 10, 40] and target number 10.
- Output: Index of target number in the array: 3.

Example 2: Selection Sort Algorithm

- Input: Unsorted array [64, 25, 12, 22, 11].
- Output: Sorted array [11, 12, 22, 25, 64].

Example 3: Dijkstra's Algorithm

- Input: Graph with 5 nodes and the following node distances:



Start node is 1, and end node is 5.

- Output: Shortest path from node 1 to node 5: [1, 3, 5] with total distance 2.

Theoretical Analysis of Algorithms

- Correctness
- Running time
- Memory usage

Why study the running time (efficiency) of an algorithm?

1. feasible vs. infeasible
2. efficient algorithms: less engineering tricks needed, can use languages aiming for easy programming
3. pursue fundamental theoretical limits

Example: Insertion Sort

- Input: [6, 5, 4, 3, 2, 1]
- Output: [1, 2, 3, 4, 5, 6]

Pseudo code for Insertion Sort

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

Execution of Insertion Sort

At the end of j-th iteration, the first j numbers are sorted.

1. Input: [6, 5, 4, 3, 2, 1]
2. [6] [5, 4, 3, 2, 1]
3. [5, 6] [4, 3, 2, 1]
4. [4, 5, 6] [3, 2, 1]
5. [3, 4, 5, 6] [2, 1]
6. [2, 3, 4, 5, 6] [1]
7. Output: [1, 2 ,3 ,4 ,5 ,6]

Correctness

- Does the algorithm always return the sorted array?
- Yes, because after iteration j of outer loop, $A[1..j]$ is the sorted array for the original $A[1..j]$.

Running time

- Q: What is the size of input?
- A: Here the size is the number of integers.
- Q: Does the condition of the input affect the running time?
- A: For the insertion sort algorithm: if input array is already sorted in ascending order, then algorithm runs much faster than when it is sorted in descending order.
- Q: What is the worst-case scenario?
- A: When we don't miss a single comparison in Line 5.

Asymptotic O notation

- Informal way to define O-notation:
- $3n^3 + 2n^2 - 18n + 1028 \to 3n^3 \to n^3$
- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$
- $\frac{n^2}{100} - 3n + 10 \to \frac{n^2}{100} \to n^2$
- $\frac{n^2}{100} - 3n + 10 = O(n^2)$

Asymptotic Analysis of Insertion Sort

- When the worst case happens, we need to compare:
- $\sum_{j=2}^{n} O(j) = O(\sum_{j=2}^{n} j) = O(\frac{n(n+1)}{2} - 1) = O(n^2)$ times.
- Can we do better than insertion sort asymptotically?
- Yes: merge sort, quicksort and heap sort take $O(nlogn)$ time

Some facts

- Basic operations that takes $O(1)$ constant time, i.e. reading and writing $A[j]$, perform one comparison, arithmetic operations.
- Each integer n has *clogn* bits, $c \geq 1$ large enough
- We often handle integers within range $[-n^c, n^c]$, it is convenient to assume this takes $O(1)$.