Algorithms and Data Structures for Data Science
CS 277 in Review

CS 277

Brad Solomon

May 1, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Final Exam Retakes

The final exam will have **four** linked exams:

The actual final (clearly labeled)

Three makeup exams (Labeled Exam 1 Retake, Exam 2 Retake, …)

You **must** take the final. You can choose to take one of the others

So its a big cumulative exam — how do I prepare?

Please fill out end-of-semester evaluations

Feedback is important for the development of the class

This is especially important for this class as it is very new and growing

An optional final reflection form: Mini-Project Reflection Feedback Form

Python Fundamentals

Can you read and understand Python code?

- Conditionals
- Loops
- Functions

What did the corresponding labs focus on?

- lab_fundamentals
- lab_debug

Asymptotic Efficiency

Can you work out the Big O of a block of code given the code?

Can you determine the relative size (or speed) of different Big Os?

Can you simplify a Big O to its dominant term?

Lists

When are lists an appropriate data structure to use?

Order matters

Insertion speed or when repeatedly finding minimum

(Unsorted) (Sorted)

What did the corresponding labs focus on?

lab_parsing

lab_cipher

Arrays

Running time, how to use, implementation details

Add()

Insert()

Remove()

__getitem__()

__len__()

Linked Lists

Running time, how to use, implementation details

Add()

Insert()

Remove()

__getitem__()

__len__()

Arrays vs Lists

	Singly Linked List	Array
Look up given an input **position**		
Search given an input **value**		
Insert/Remove at **front**		
Insert/Remove at **arbitrary** location		

Stacks and Queues

Running time, how to use, implementation details

Sets and Set Similarity

What distinguishes a set from a list?

How do we measure set similarity?

Hash Tables

Running time, how to use, implementation details

Closed vs Open Hashing

Probing strategies seen in class:

What is SUHA? What does it mean?

Hash Tables

Lots of conceptual questions!

What happens when my hash table gets close to full? Can it get full?

What is SUHA? What does it mean?

How should I implement a table for given constraints?

Sketches

Bloom Filters

K-minimum value

Minhash Sketch

Sketches

Lots of conceptual and practical questions!

mp_sketching all about real-world use cases of sketches

Should be able to build a bloom filter manually

Should be able to state what set is larger based on minima values

Should be able to judge which sets are most similar given min hashes

Sorting Algorithms

Running time, how to use, implementation details

Selection Sort

Insertion Sort

Merge Sort

Quick Sort

Sorting Algorithms

	Best Case Time	Worst Case time	Best Case Space	Worst Case Space
SelectionSort	$O(n^2)$	$O(n^2)$	$O(1)$	$O(1)$
InsertionSort	$O(n)$	$O(n^2)$	$O(1)$	$O(1)$
MergeSort	$O(n \log n)$	$O(n \log n)$	$O(n)$	$O(n)$
QuickSort	$O(n \log n)$	$O(n^2)$	$O(\log n)$	$O(n)$

Recursion

Base Case: What is the smallest sub-problem? What is the trivial solution?

Recursive Step: How can I reduce my problem to an easier one?

Combining: How can I build my solution from recursive pieces?

Trees and graphs are all about **recursion**!

Binary Search

Running time, how to use, implementation details

Both lists and trees have a binary search function!

Be able to walk through a binary search by hand

Trees

Running time, how to use, implementation details

Binary Tree

BST

AVL

K-dimension tree

Huffman

Tree Traversals

An exercise in recursion!

- Pre-order, in-order, post-order just a matter of code order
- Should be able to recreate a traversal manually

Depth-first Traversal vs Breadth-first Traversal

- We did not cover this in great detail
- The questions I can ask are somewhat limited

Nearest-neighbor Search

AVL Tree

Gets its own slide because its so important!

- Fixes all the issues we have with a BST
- Understand the rotations (what they do, when to use)

Understand the tree ADT for this!

Huffman Tree

Gets its own slide even though we barely covered it

I'd *like* to ask questions about the Huffman tree but we didnt cover concepts

I can reasonably ask you to know how:

Produce a Huffman code from a tree

Build a (small) Huffman tree from a set of frequencies

Graphs

Running time, how to use, implementation details

Edge List

Adjacency Matrix

Adjacency List

$|V| = n, |E| = m$

Expressed as O(f)	Edge List	Adjacency Matrix	Adjacency List
Space	m (or n+m)	n^2	n+m
insertVertex(v)	Impossible or 1*	n*	1*
removeVertex(v)	m**	n	deg(v)***
insertEdge(u, v)	1	1	1*
removeEdge(u, v)	m	1	min(deg(u), deg(v))
getEdges(v)	m	n	deg(v)
areAdjacent(u, v)	m	1	min(deg(u), deg(v))

Graph Traversals

Both conceptual and practical knowledge required

- When should I use BFS or DFS?
- What can I output from a traversal? Why should I?

Be able to manually perform a traversal if given key info.

Shortest Path

Dijkstras is only shortest path you need to know

Graph runtimes usually depend on implementation details!

Be able to manually perform a traversal if given key info.

Minimum Spanning Tree

Understand Kruskal and Prim algorithm details

Graph runtimes usually depend on implementation details!

Be able to manually perform the algorithm if given key info.

Be able to state the runtime for a given piece of code

Questions?

For Wednesday's lecture:

Post questions on Piazza by mid-day tomorrow.