Programming, Problem Solving, and Algorithms

CPSC 203, 2024 W2
(January – April 2025)
Ian M. Mitchell
Lecture 14A

Traveling Salesperson Wrap-up

- TSP is a canonical example of an “NP-complete” problem
  - We suspect (but have not proved) that problems in NP require exponential time to solve exactly (at least on traditional computers)
  - The “complete” part refers to the fact that we have ways of converting this problem into other NP-complete problems and vice-versa: Solving one would solve them all
- So if I can convert my problem into TSP then I am in trouble?
  - Not necessarily: Maybe there is a way to convert it into a simpler problem
  - Not necessarily: There are many approximation / heuristic algorithms that might do a decent job
  - Not necessarily: Quantum computing might save you
  - Not necessarily: You now have evidence that your boss needs lower expectations

We did what now?

- CPSC 203: Programming, Problem Solving, and Algorithms
- Calendar description:
  - Analysis of increasingly complex algorithmic problems, using a modern programming language and a variety of approaches.
  - Problem decomposition and abstraction guide explorations of topics from applied algorithms.
  - Examples: Voronoi Diagrams, Markov Chains, Bin Packing, and Graph Search.
- I hope that you learned three different types of things:
  - High level concepts
  - Abstract data types and algorithms
  - Practical tools and techniques

Week 1: Introductions

- Topics:
  - Teaching team and approach
  - Course topic overview
- Concepts:
  - Learn by doing
  - Abstraction, simplicity, elegance
  - Computational / resource complexity
- Abstract data types and algorithms:
  - “Big-O” notation
  - Representing numbers, colors
- Tools and techniques:
  - Markdown
  - Terminal
  - PrairieLearn and other course tools

Week 2: Python Review

- Topics:
  - Basic Python data types and programming constructs
- Concepts:
  - Evaluation rules
  - Mutable vs immutable (hashable) data and function side-effects
- Abstract data types and algorithms:
  - Varieties of “arbitrary length” data types
- Tools and techniques:
  - JupyterLab (in PL workspace)
  - Simple data types (int, float, bool, str, None)
  - Collection data types (list, tuple, dictionary, set, …) and indexing
  - Exceptions
  - (List) comprehensions

Week 3: Efficiency and Python Classes

- Topics:
  - More Python review
  - Abstracting and implementing the process of knitting washcloths
- Concepts:
  - Classes combine data (“attributes”) with functions (“methods”)
- Abstract data types and algorithms:
  - Classes containing classes
- Tools and techniques:
  - VSCode (in PL workspace) text editor and environment
  - Decorator “@dataclass” creates some default methods (including constructor)
  - Python type hints (in function signatures and class definitions)
  - PIL for creating images

Week 4: Classes and DataFrames

- Topics:
  - Implementing knitting handcrafts with visualization
  - Tabular data and Pandas DataFrames
- Concepts:
  - Modifying existing data (such as flipping knitting blocks)
  - Bugs from aliases and mutable data
- Abstract data types and algorithms:
  - DataFrames: Creating, indexing, filtering, mapping, extracting, iterating
- Tools and techniques:
  - Python copy.deepcopy() to avoid aliasing
  - CSV files
  - Pandas

Week 5: Web scraping

- Topics:
  - Getting data from web pages
- Concepts:
  - HTML as a format for web pages
- Abstract data types and algorithms:
  - Looping over collections
- Tools and techniques:
  - Parsing HTML with billboard and BeautifulSoup libraries
  - Converting data from collection of objects into a DataFrame
  - Plotting DataFrame information with matplotlib

Week 6: Version Control and Git

- Topics:
  - Version control for managing code
- Concepts:
  - Repositories, working copy, commits, branching, merging
- Abstract data types and algorithms:
  - N/A
- Tools and techniques:
  - Git
  - GitHub

Week 7: Reading Week

- Deep breath in, hold, and then out…

Week 8: New Data Structures

- Topics:
  - Moving beyond basic collection / series operations (select element, iterate over all to map, filter, reduce, …)
  - Pointillistic painting
- Concepts:
  - Voronoi diagram to partition a plane based on nearest center
- Abstract data types and algorithms:
  - Images as grids
- Tools and techniques:
  - Start using a locally installed tech stack
  - PIL for manipulating photos

Week 9: Graphs

- Topics:
  - Efficient implementation of Voronoi partition / image blobby images
- Concepts:
  - Nodes and neighbors
  - Breadth first search using a queue
- Abstract data types and algorithms:
  - Queues: FIFO and LIFO (stacks)
  - Graphs
  - Breadth first search (BFS) iterative version
- Tools and techniques:
  - Python deque to implement FIFO queue and BFS

Week 10: Catch-up week

- Woulda, shoulda, coulda…
- Take 10 minutes and fill out your course evaluation

Week 11: Mary had a little lamb

- Topics:
  - Music generation
- Concepts:
  - Markov chains
  - State space of dynamic systems
- Abstract data types and algorithms:
  - Markov chains as directed weighted graphs
  - Graphs in adjacency matrix or adjacency list forms
- Tools and techniques:
  - Extracting Markov chain probabilities from data sample(s)

Week 12: State spaces

- Topics:
  - Sudoku
- Concepts:
  - Representing a sudoku state
- Abstract data types and algorithms:
  - Graphs of collections
  - Depth first search (DFS) iterative version
  - Dijkstra’s algorithm for shortest path
  - Priority queue
- Tools and techniques:
  - DFS and stacks using deque
  - Priority queue using heapq

Week 13: Maps (the regular kind)

- Topics:
  - Street maps
- Concepts:
  - Representing geographic data in graph or tabular form
- Abstract data types and algorithms:
  - TSP
- Tools and techniques:
  - Extracting and graphing map features using OSMnx and GeoPandas
  - Extracting and graphing a shortest path using NetworkX
  - Constructing a TSP solver on a street map
  - Solving TSP by brute force iteration over all permutations

Project 1: Spotify (if only)

- Topics:
  - Download playlist and song data from Spotify
- Concepts:
  - Use an API to directly access remote data
- Abstract data types and algorithms:
  - Tabular data and DataFrames
- Tools and techniques:
  - Spotipy library for Spotify’s web API
  - Pandas
  - matplotlib

Project 2: Cellular Automata

- Topics:
  - Conway's Game of Life and more general cellular automata
- Concepts:
  - Systems with dynamic state
  - Simple but implicitly defined graphs
- Abstract data types and algorithms:
  - Render loop for animation
- Tools and techniques:
  - Class constructors
  - Simple game engine PyGame for dynamic graphics
  - PIL for animated gifs

Project 3: Maps and Path Planning

- Topics:
  - Planning a path through a downloaded streetmap
- Concepts:
  - Geographical Information Systems (GIS)
- Abstract data types and algorithms:
  - Graphs
  - GeoDataFrames
  - Depth first search (DFS)
- Tools and techniques:
  - OSMnx for working with OpenStreetMap data
  - GeoPandas for geophysical data manipulation and visualization
  - NetworkX for graph algorithms
  - Debugging somebody else's code

To Summarize

- New abstract data types
  - Tabular data
  - Graphs
  - Queues
  - Cellular automata and Markov chains
  - State spaces of dynamic systems
- Python data types
  - Built-in: List, dictionary, set, tuple
  - Other modules: deque, defaultdict, heapq
  - Classes
  - DataFrames
  - Many representations of graphs
- Algorithms:
  - Multi-dimensional iteration
  - BFS, DFS, Dijkstra's algorithm
  - Traveling salesperson problem
- Tools & techniques
  - Markdown
  - Terminal
  - VSCode
  - Local tech stack and conda
  - Git and GitHub
  - JupyterLab and Jupyter notebooks
  - Pandas, PIL, matplotlib and too many other Python libraries to mention

To Really Summarize

- Get the data representation right
  - Once the data is in the correct form, solving the problem is usually easy (at least conceptually – execution time / memory may be an issue)
  - A lot of your code will be transforming data from one representation to another
  - Be careful that transformations do not (unintentionally) add, delete or corrupt the information that the data is representing
- Abstraction allows you to leverage the work of others
  - Try to transform your data into a standard form (collection, table, graph, …)
  - Try to transform your algorithm into a standard problem (map, filter, reduce, iteration, BFS, DFS, TSP, …)
- If you learn nothing else: Use version control!

Where to go from here?

- CPSC 203 is designed to be a terminal course for students who are **not** pursuing a computer science major
  - It contains a mixture of material from CPSC 121, 210 and 221
- There are numerous courses which you can take with CPSC 203
  - CPSC 302 / 303 / 406: Numerical methods
  - CPSC 330: Applied machine learning
  - CPSC 368: Databases in data science
  - DSCI 310: Reproducible and Trustworthy Workflows for Data Science
  - DSCI 320: Visualization for Data Science
- Not programming courses, but worth serious consideration
  - CPSC 430: Computers and Society
  - DSCI 430: Fairness, Accountability, Transparency and Ethics (FATE) in Data Science
- Also: Data science minor and (new next fall) major

Where to go from here?

- Anywhere you want!
  - You know Python, Jupyter Notebooks, the terminal, conda VSCode, Git/GitHub, …
  - You have a local development stack on your own machine
- https://xkcd.com/353
  - import matplotlib.pyplot.xkcd