(http://csmatters.org) 5 - 2

0b101 - 0b10

Searching

Unit 5. Data Manipulation

Revision Date: Sep 08, 2019
Duration: 2 50-minute sessions

Lesson Summary

Pre-Lesson Preparation

- Teachers will need to have a piece of paper with a unique number on it for all but one student in the class.
- Students will need access to the datasets and Python skeleton code in the Lesson Resources folder.
- Teachers will need to print out the "Search Comparison Worksheet" for each student.

Summary

Students investigate data organization, simulate linear and binary searches, and write pseudocode and Python for linear and binary search methods.

Outcomes

- Students will demonstrate how combining data sources and classifying data are part processing data
- Students will describe challenges to structuring large data sets for analysis
- Students will identify how the order of data influences which methods are appropriate for searching the data.
- Students will describe standard search algorithms in pseudocode and in Python.
- Students will Compare different algorithms for *efficiency* when searching for an item.

Overview

Session 1

1. Getting Started (5 min) - Journal
2. Class Discussion and Activities (25 min) - Introduction to linear and binary search algorithms
3. Coding Linear Search (20 min)

Session 2

1. Getting Started (5 min) - Think-Pair-Share
2. Coding Activity (30 min) - Write pseudocode and implement binary search
3. Compare Searches (15 min) - Fill out worksheet to compare search algorithms

Source

Phone book presentation adapted from a lesson taught by Dr. Rheingans in CMSC 201 at the University of Maryland, Baltimore County

Learning Objectives

CSP Objectives

- *EU CRD-2 - Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.*
  - LO CRD-2.C - Identify input(s) to a program.
  - LO CRD-2.D - dentify output(s) produced by a program.
  - LO CRD-2.F - Design a program and its user interface.
  - LO CRD-2.J - Identify inputs and corresponding expected outputs or behaviors that can be used to check the correctness of an algorithm or program.
- *EU DAT-2 - Programs can be used to process data, which allows users to discover information and create new knowledge.*
  - LO DAT-2.A - Describe what information can be extracted from data.
  - LO DAT-2.C: - Identify the challenges associated with processing data.
  - LO DAT-2.E - Explain how programs can be used to gain insight and knowledge from data.
- *EU AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.*
  - LO AAP-2.A - Express an algorithm that uses sequencing without using a programming language.
  - LO AAP-2.L - Compare multiple algorithms to determine if they yield the same side effect or result.
  - LO AAP-2.M - For algorithms: a. Create algorithms. b. Combine and modify existing algorithms.
  - LO AAP-2.O - For algorithms involving elements of a list: a. Write iteration statements to traverse a list. b. Determine the result of an algorithm that includes list traversals.
  - LO AAP-2.P - For binary search algorithms: a. Determine the number of iterations required to find a value in a data set. b. Explain the requirements necessary to complete a binary search.
- *EU AAP-3 - Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.*
  - LO AAP-3.A - For procedure calls: a. Write statements to call procedures. b. Determine the result or effect of a procedure call.
- *EU AAP-4 - There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.*
  - LO AAP-4.A - For determining the efficiency of an algorithm: a. Explain the difference between algorithms that run in reasonable time and those that do not. b. Identify situations where a heuristic solution may be more appropriate.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.

Common Core Math:

- F-BF.1-2: Build a function that models a relationship between two quantities

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure

NGSS Practices:

- 2. Developing and using models
- 3. Planning and carrying out investigations
- 5. Using mathematics and computational thinking

Key Concepts

Students will:

- Demonstrate how combining data sources and classifying data are part processing data
- Describe challenges to structuring large data sets for analysis
- Be able to identify how the order of data influences which methods are appropriate for searching the data.
- Be able to describe standard search algorithms in pseudocode and in Python.
- Compare different algorithms for *efficiency* when searching for an item.

Essential Questions

- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- What considerations and trade-offs arise in the computational manipulation of data?
- What opportunities do large data sets provide for solving problems and creating knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In Lesson Resources Folder:

- Search Comparison Worksheet to use at the end of Session 2
- DataSets folder that contains the six datasets for the Search Comparison Worksheet
- SearchCode.py contains skeleton code for a numeric search program
- Binary Search in LISP example to show on the screens.

Lesson Plan

Session 1

Getting Started (5 min)

Journal: What would be the best way to organize a collection of DVDs so that you could find the one you want very quickly? Would you need a different method for a radio station with thousands of DVDs? Discuss.

Class Discussion and Activities (25 min)

Linear Search Discussion (5 min)

As a class, discuss the following questions:
What if you are looking for a specific song. What is the most effective way to look for something if it is in an unordered, unsorted collection?
What are some challenges to organizing large sets of information so they can be processed?
Why does it help to classify data when you are trying to sort it or search through it?

Possible Answers and suggestions for discussion:

- You can search Randomly (How do you stop yourself from repeating yourself?)
- You can search One-by-one, marking off which ones have already been checked.
- You can divide up the problem and each search a pile (multiprocessing)
- Challenges: data entry is time consuming. Large sets of data are more important to keep organized but if you combine information from multiple places they might be organized differently so you have to come up with a single system.
- If you don't classify the information you're sorting or searching such as the album title, song title or other searchable information, it would be very confusing to try to find what you are looking for, especially if you don't know exactly what you are trying to find.
- Having things grouped by artist, genre, or other categories makes it easier to narrow down what you're looking for.

Linear Search Activity (5 min)

1. Pass out pieces of paper with numbers on them to everyone in the class except one person. Students should not share their numbers with anyone.
2. Have everyone stand at the front of the room in a line. The person without a number stands in front and is assigned a number to look for. The class should keep track of how many people they have to ask before they get the number (students that have been checked should sit down).
    - Do this activity a few times. In between, each student should switch with another student a couple of times without showing their paper. (Note: A fun way to do this might be a snowball fight if you can)
    - Ask for a number that does not exist in the set at least once. What happens? How many people does it take before they figure out the number is not in the set?
3. Have everyone sit down but keep their papers.

Binary Search Discussion and Presentation (10 min)

Take out a dictionary (or phone book). Ask the class, how would you search for a particular word/name?

Steps for Binary Search in a book of items to demonstrate to the class:

1. Flip to the middle and pick a word in the middle of the page

2. Is your word higher or lower than this word? If it is higher, “throw out” the lower half of the book. If it is lower, “throw out” the top half. (Not literally unless it is a very old phonebook. Students do love it when you tear up the phonebook, though, and it makes for a very effective demonstration.)
3. Repeat steps 1 and 2 until you find the word.

Why would this not work for an unordered list?

Binary Search Activity (5 min)

1. Have everyone go to the front of the room and get in numerical order by paper. Repeat the search activity using the binary search algorithm.
2. Try with a number not in the list. How can you tell when it doesn’t exist?

Coding Linear Search (20 min)

1. Have students work in pairs to write a code for linear search. The code should:
    1. Read in a csv file that has a list of unordered numbers. (They should input the file name from the user.)
    2. Ask the user what number they want to find and validate that number.
    3. Tell the user whether the number was found. If it was, it should output the number of items it had to look at.
2. Students should save their code for the next day.

Note: Skeleton code (SearchCode.py) is provided in the Lesson Resources Folder.

Session 2

Getting Started (5 min)

Think-Pair-Share

- Ask students to list non-numbered, real-world things that they search for or sort/order in their daily lives.
- Can all data be sorted, or do types of data exist that cannot be sorted? How would you organize and search these types of data?
- Is there always a "correct" solution when sorting data?

Coding Activity (30 min)

Part 1 Pseudocode (10 min)

1. As a class, write the major steps for Binary Search on the board.
2. Identify algorithms that could be combine to make a binary search algorithm.
3. In pairs, have the students pseudocode a binary search algorithm.
4. Discuss the advantages of using established algorithms.
5. When finished, display binary search code in other languages.
   (Scratch https://scratch.mit.edu/projects/23163175/;
   (https://scratch.mit.edu/projects/23163175/;) javascript https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/implementing-binary-search-of-an-array
   (https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/implementing-binary-search-of-an-array) ; LISP (see handout)

Part 2 Coding (20 min)

1. Students should use their pseudocode to write a program for binary search. (It would be useful to build on the skeleton code provided for linear search.) The code should accomplish the same things as linear search: read in a file, get a number, and output if the number is found and how many items were checked.

Comparing Searches (15 min) – (May also be Homework if programs are not finished)

- Pass out the worksheet "Search Comparison Worksheet" from the lesson resources folder.
- Students should run both their linear and binary search programs with the six provided datasets of increasing sizes, (also in the lesson resources folder.)
- As they go through, students should record their results in the worksheet and answer the questions at the bottom.
- Discuss the results as a class.

Options for Differentiated Instruction

For students that have difficulty understanding the concepts of searching for items in a set of data, pair those students with a student who has a firm grasp of the concept for the activities. Have the pair work together for 1A and then have them keep their own paper secure using the extra game sheet (1A'). Simiilarly for 1B - 1B' and 1C - 1C'.

Evidence of Learning

Formative Assessment

Correctness of Python functions for linear search and binary search

Summative Assessment

"Searching Assessment Items.docx" in lesson folder

"Search Comparison Worksheet" in the lesson folder

(http://www.umbc.edu/) (http://www.umd.edu/) (http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)

by University of Maryland, Baltimore County (http://umbc.edu) and University of Maryland, College Park (http://umd.edu).