Lab 02: Bag ADT and LCS

CS 0445: Data Structures

TAs: Jon Rutkauskas
Brian Nixon

http://db.cs.pitt.edu/courses/cs0445/current.term/

Sep 16, 2019
University of Pittsburgh, Pittsburgh, PA

Goals

⟸ **BAG**

- To practice using the Bag ADT to solve a real-world problem (Longest Common Subsequence).

What is a Bag?

- finite, unordered collection of items in which duplicates are permissible
  - finite: There is a limit to how many items can exist in the bag
    - i.e. the bag can become full
  - unordered: The bag makes no promises regarding the order of elements contained in the bag
    - e.g. {1,2,3,4,5} == {2,3,1,4,5}
  - duplicates: multiple instances of two (or more) equivalent items can be contained within the bag
    - e.g. {1,2,3,3,4,5} != {1,2,3,4,5}

What can you do with a bag? (Part 1)

- public boolean add(E newEntry): adds an element
- public E remove(): removes and returns an arbitrary element
- public E remove(E anEntry): removes and returns a single occurrence of a specified element [if possible]
- public void clear(): empties the bag
- public int getCurrentSize(): returns the number of elements in the bag

What can you do with a bag? (Part 2)

- public boolean isEmpty(): determines if the bag is empty
- public boolean contains(E anEntry): determines if the specified element is contained in the bag
- public int getFrequencyOf(E anEntry): determines the number of instances of a specified element
- public E[] toArray(): returns all objects inside the bag, including all instances of any particular element

Understanding the Longest Common Subsequence Problem

What is a Subsequence?

- A **subsequence** is a sequence *derived* from a parent sequence that maintains the order of the parent sequence
  - *derived* by the deletion of elements from the parent sequence
- e.g. (Strings are Sequences of Characters)
  - Parent: “ABCDEFG” (note: != “ACBDEFG”)
  - Valid Subsequence: A**B**C**D**E**FG** **-> “BDFG”**
  - Not Valid Subsequence (1): A**B**C**DE**F**G** **-> “BEDG”**
  - Not Valid Subsequence (2): **A**BCDE**FG -> “AFGH”**
  - What rules were violated in (1) and (2)?

What is the LCS Problem

- Given two sequences, *x* and *y*, can you find the longest **subsequence** *common* to both *x* and *y*
  - *common* in that this **subsequence** can be *derived* from both *x* and *y*
- e.g. What is the longest common **subsequence**?
  - (“ABCDEFG”,”AGDEBFGC”) **-> “ADEFG”**
  - Note: “ABC”, “ABFG”, etc. are also common **subsequences**, but not the longest

How can we use a Bag to solve LCS (General Idea)

- Given two strings (first and second), we use the bag to store subsequences of the **first** string, *beginning with the first string itself*
- While the bag is not empty, we remove an arbitrary element
  - If the element is a candidate to be the **new** LCS, we replace the current LCS with it
    - **Otherwise**, if the element is a candidate to **parent** a new LCS, we break down the element into further, shorter subsequences and place those subsequences into the bag to be checked later
- The key here is beginning with the **longest possible** LCS at the **start**

How can we use a Bag to solve LCS (Detailed Logic)

- Create an empty bag
- Put the first string into the bag
- Set the longest subsequence to the empty string
- While the bag isn't empty
  - Remove a test string from the bag
  - If the longest subsequence is shorter than the test string
    - If the test string is a subsequence of the second string
      - Set the longest subsequence to the test string
    - Otherwise, if the test is at least 2 longer than the longest subsequence
      - Generate new strings from the test by removing each single character
      - Put each of the new strings in the bag
  - Print the bag of strings that need to be checked
- Print out the longest subsequence

Your Tasks

- Download the provided code from the course website (http://db.cs.pitt.edu/courses/cs0445/current.term/) and read all lab instructions
- Understand the brute-force algorithm given and how the methods provided in the Bag ADT can be used to implement it
  - Remember: Here, you are acting as a client, not implementing a Bag
  - Feel free to look up the Java API for reference on Strings.
  - Ask questions!
- Complete `TODO` portions of code in `LongestCommonSubsequence.java` as instructed
- Test your work