COP2334

PG2 –SORTING, SEARCHING

CONTENTS

Objectives 2
Topics Covered 2
Project Setup 2
PART 1 3
The Menu 3
PART 2 3
Load the file 3
PART 3 4
Bubble Sort 4
PART 4 6
Merge Sort 6
PART 5 8
Binary Search 8
PART 6 10
NuGet & Json.NET 10
Save 10
Rubric 11
Programmer's Challenge 11
Sorting Challenge 11

FULL SAIL
UNIVERSITY

COP2334

OBJECTIVES

Learn to load and parse CSV files. Implement various sorting and search algorithms. Utilize recursion to solve problems. Clone arrays into new lists.

You will be loading in a CSV file that contains unsorted data and you can store that information in a List (each item is a string). The user will be able to sort that data using the different sorting algorithms that we covered in the lecture: Bubble, Merge. Search a sorted list with a binary search algorithm.

Topics Covered

Cloning, bubble sort, merge sort, binary search, CSV, Split, File loading

Project Setup

A **C# .NET Core console application** has been provided for you in your **GitHub repo**. **Use the provided solution.**

NOTE: you MUST add the *new lab 2* methods to the PG2Sorting class.

NOTE: leave the Read methods in Input.cs.

-15: did not add the lab 2 methods to the PG2Sorting class.

PART 1

The Menu

Lab Overview Video

Part 1 Overview

Show a menu to the user so they can select one of the algorithms (bubble, merge and binary search), save a sorted list, and Exit. (Use the ReadChoice method you created in the first lab)

1. Bubble Sort
2. Merge Sort
3. Binary Search
4. Save
5. Exit

GRADING: 20 POINTS

COMMON MISTAKES:

-3: the Exit option does not exit

PART 2

Load the file

Lecture Videos

Reading CSV
Reading CSV Example
Reading CSV Challenge

Lab Overview Video

Part 2 Overview

Write a method to read the file and **return** a list of strings. Open and read the line from the **inputFile.csv** file. The line in the file contains a list of comic book titles separated by commas. Split the string and store each title in a **List of strings**.

NOTE: the inputFile.csv has been added to the project with the copy properties set so that **you do not need to add a path** to read the file. Just use the filename.

GRADING: 10 POINTS

COMMON MISTAKES:

-2: you have a full or relative path to the file that is specific to your machine

-2: you are not closing the file after you read it

-2: you are not parsing the data correctly

-1: you are not cloning the original correctly. Setting List<string> list2 = list1; only points list2 to the same thing that list1 points to.

-1: not converting the array to a list.

-2: did not create a method for reading the file

PART 3

Bubble Sort

Lecture Videos

Bubble Sort
Swapping
Swapping Challenge
Comparing Strings
Comparing Strings Example
Comparing Strings Challenge

Lab Overview Video

Part 3 Overview

Write a method to implement the Bubble sort algorithm. You want to keep the original list unsorted so make sure to clone the list inside of Bubble sort.

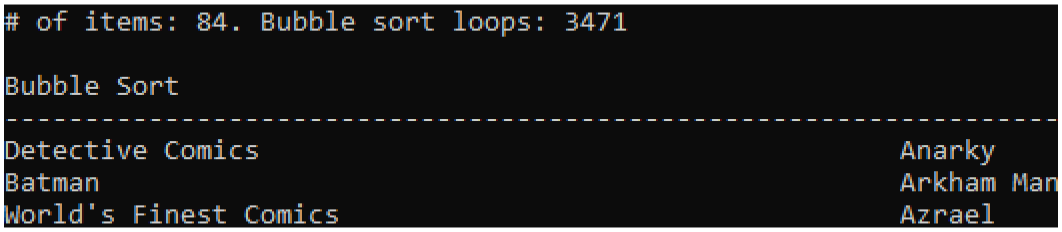
NOTE: **swap** is a method that is not provided by C#. You can create your own method or you can insert the swap logic inside the if. See the lectures slides for how to swap 2 items in a list.

Keep track of how many times the for loop executes for the whole method. At the end of the method, print out the number of items in the list and the number of times the algorithm had to loop.

Your code must follow the pseudocode. Turn this Wikipedia pseudocode into C#:

```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
    repeat
        swapped := false
        for i := 1 to n - 1 inclusive do
            if A[i - 1] > A[i] then
                swap(A, i - 1, i)
                swapped = true
            end if
        end for
        n := n - 1
    while swapped
end procedure
```

EXAMPLE OUTPUT:



```
# of items: 84. Bubble sort loops: 3471

Bubble Sort
-----------------------------------------------------------------
Detective Comics                                        Anarky
Batman                                                  Arkham Man
World's Finest Comics                                   Azrael
```

The items on the left are the unsorted values and the items on the right are the sorted.

GRADING: 20 POINTS

COMMON MISTAKES:

-10: did not follow the pseudo-code for bubble sort

-10: did not track the loop count and print it out.

-1: you should print the sorted results side-by-side with the unsorted

-5: Bubble sort can be more efficient. The inner for loop should track whether a swap happens. If the inner loop does not swap, then you can break out of the outer loop.

-5: the bubble sort can be optimized more according to the pseudo-code. You can shorten the for loop by 1 after the for loop completes. Store the length of the list in a variable and subtract 1 from it after the for loop. This would mean 1 fewer item to compare each time you run the for loop.

-1: in BubbleSort, you should set swapped = false right before the for loop.

-1: the while condition in bubble sort is incorrect. You need to loop while a swap has happened.

-2: did not create a method for Bubble sort

PART 4

Merge Sort

Lecture Videos

Merge Sort
Merge Sort Split Step
Merge Sort Merge Step
Split Challenge

Lab Overview Video

Part 4 Overview

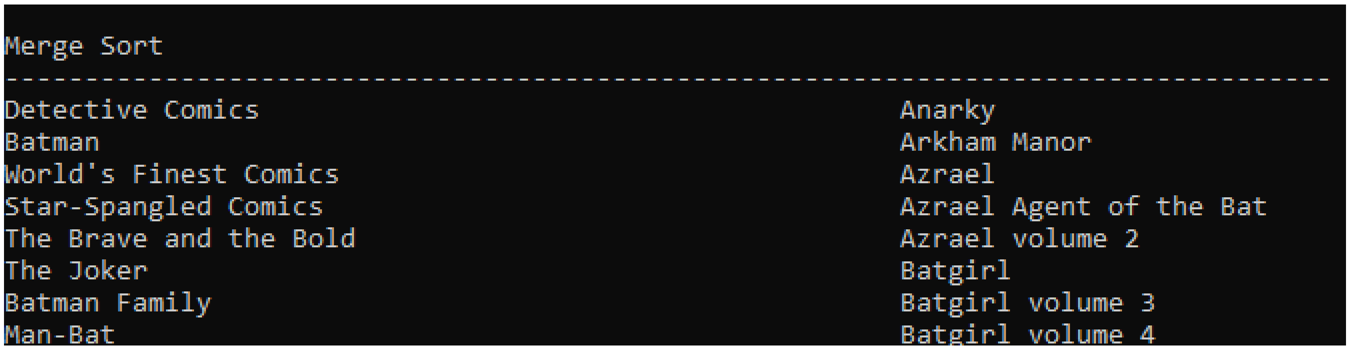
Write a method to implement the Merge sort algorithm. Merge sort will return a new sorted list so there is no need to clone the list that is passed in to MergeSort.

Your code must follow the pseudocode. Turn this Wikipedia pseudocode into C#:

```
function merge_sort(list m) is
    // Base case. A list of zero or one elements is sorted, by definition.
    if length of m ≤ 1 then
        return m

    // Recursive case. First, divide the list into equal-sized sublists
    // consisting of the first half and second half of the list.
    // This assumes lists start at index 0.
    var left := empty list
    var right := empty list
    for I = 0 to length(m) do
        if i < (length of m)/2 then
            add m[i] to left
        else
            add m[i] to right

    // Recursively sort both sublists.
    left := merge_sort(left)
    right := merge_sort(right)

    // Then merge the now-sorted sublists.
    return merge(left, right)
```

```
function merge(left, right) is
    var result := empty list

    while left is not empty and right is not empty do
    {
        if first(left) ≤ first(right) then
            add first(left) to result
            remove first from left
        else
            add first(right) to result
            remove first from right
    }

    // Either left or right may have elements left; consume them.
    // (Only one of the following loops will actually be entered.)
    while left is not empty do
    {
        add first(left) to result
        remove first from left
    }
    while right is not empty do
    {
        add first(right) to result
        remove first from right
    }
    return result
```

EXAMPLE OUTPUT:



```
Merge Sort
----------------------------------------------------------------------------------
Detective Comics                                        Anarky
Batman                                                  Arkham Manor
World's Finest Comics                                   Azrael
Star-Spangled Comics                                    Azrael Agent of the Bat
The Brave and the Bold                                  Azrael volume 2
The Joker                                               Batgirl
Batman Family                                           Batgirl volume 3
Man-Bat                                                 Batgirl volume 4
```

GRADING: 20 POINTS

COMMON MISTAKES:

-10: did not follow the pseudo-code for merge sort

-2: the exit condition needs to be if the count of the list <= 1.

-2: did not write a method for Merge Sort

-1: you should print the sorted results side-by-side with the unsorted

PART 5

Binary Search

Lecture Videos

Binary Search
Binary Search Algorithm

Lab Overview Video

Part 5 Overview

Write a method to implement the Binary Search algorithm (use a recursive approach).

Clone the original list and sort the cloned list (call Sort on the list).
Loop over the sorted list.
Call ***your*** binary search method to search the sorted list for each title in the sorted list.
HINT: the index returned from your binary search should match the index.

Show the search title, the index and the index returned by your binary search method.

Your code must follow the pseudocode. Turn this Wikipedia pseudocode into C#:

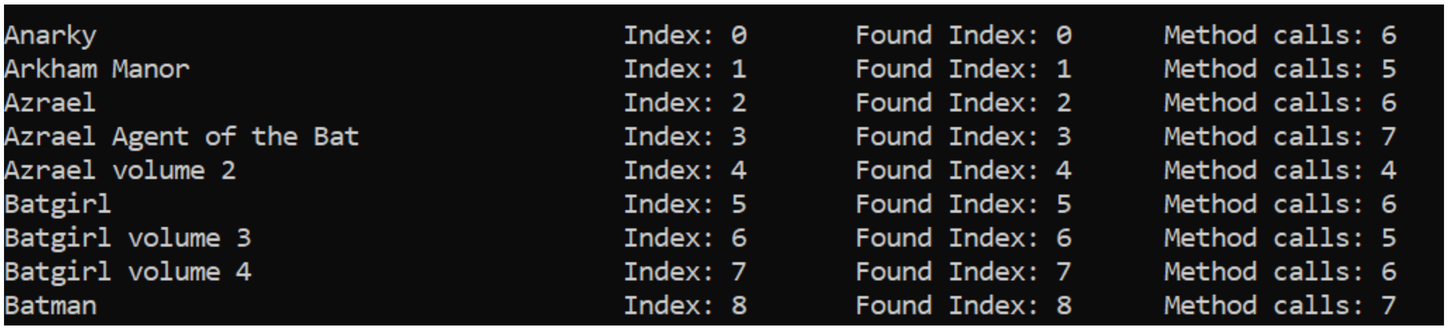
```
// initially called with low = 0, high = N-1. A is a sorted list.
  BinarySearch(A[0..N-1], searchTerm, low, high) {
      if (high < low)
          return -1 // -1 means not found
      mid = (low + high) / 2
      if (searchTerm < A[mid])
          return BinarySearch(A, searchTerm, low, mid-1)
      else if (searchTerm > A[mid])
          return BinarySearch(A, searchTerm, mid+1, high)
      else
          return mid //the searchTerm was found
  }
```

Keep track of how many times the binary search method is called when searching for a word. Print that count at the end of each line.

Format of the output to show for each word:

[word] Index: [index] Found Index: [index returned from your binary search] Method calls: [count of method calls]

EXAMPLE OUTPUT:



```
Anarky                                  Index: 0      Found Index: 0      Method calls: 6
Arkham Manor                            Index: 1      Found Index: 1      Method calls: 5
Azrael                                  Index: 2      Found Index: 2      Method calls: 6
Azrael Agent of the Bat                 Index: 3      Found Index: 3      Method calls: 7
Azrael volume 2                         Index: 4      Found Index: 4      Method calls: 4
Batgirl                                 Index: 5      Found Index: 5      Method calls: 6
Batgirl volume 3                        Index: 6      Found Index: 6      Method calls: 5
Batgirl volume 4                        Index: 7      Found Index: 7      Method calls: 6
Batman                                  Index: 8      Found Index: 8      Method calls: 7
```

GRADING: 20 POINTS

COMMON MISTAKES:

-10: did not follow the pseudo-code for binary search

-10: did not track the method calls for each word

-1: in Binary Search, you should only call the CompareTo method once and store the result instead of calling it twice.

-1: Binary Search should return the index if found or -1 if not found

-5: binary search code was not modified to work with strings and doesn't return the correct index.

-2: the binary search needs an exit condition for when min > max. If this condition happens, then you need to return -1 to indicate that the search item was not found. You should check the condition at the top of the binary search method.

-2: in binary search, you need to calculate the mid like this: min + (max – min)/2 OR (max + min) / 2.

-2: when recursively calling binary search, you need to do mid+1 or min-1 so you are not re-evaluating the mid point again.

-2: the lab requirements for binary search were to loop over the sorted list and call your binary search for each item in the list. Print the word, the index, and the index returned from your binary search.

-2: did not write a method for Binary Search

PART 6

NuGet & Json.NET

Lecture Videos

NuGet

NuGet is the package manager for .NET – it's a place to grab helpful code from 3rd parties. For this lab, you'll need to use NuGet to grab Json.NET.

To add a reference to Json.NET, right-click the References node under your class library project and select "Manage NuGet Packages…". Select the "Browse" link in the top-left of the page that is loaded in the IDE. Enter "Newtonsoft.Json" in the search box. Select the item in the list of search results and in the right panel of the page, select Install.

Save

Lecture Videos

Serializing
Serializing Example
Serializing Challenge

Lab Overview Video

Part 6 Overview

Now you have the information you need to add logic to the menu for the "**Save**" option. **Write a method** to serialize a sorted list to a save file. Take a clone of the unsorted, sort using one of your sort algorithms, then save the sorted list to a json file.

Ask the user for the name of the save file. **Use ReadString to get the name of the file**. **If the name does not have the json extension**, add it to the file name. Look at the Path methods GetExtension, HasExtension, and ChangeExtension to make sure you get the extension set correctly.
You will need to **serialize** the list in JSON format. Use the JSON.net library.

GRADING: 10 POINTS

COMMON MISTAKES:

- -2: not using ReadString to get the file name from the user
- -2: not ensuring the filename has a .json extension.
- -2: not changing the extension correctly
- -4: not serializing a sorted list
- -4: not serializing in JSON format
- -2: did not write a method for saving the data

RUBRIC

FEATURE	VALUE	GRADE
PART 1: The Menu	20	
PART 2: Load the file	10	
PART 3: Bubble Sort	20	
PART 4: Merge Sort	20	
PART 5: Binary Search	20	
PART 6: Saving	10	
TOTAL	100	

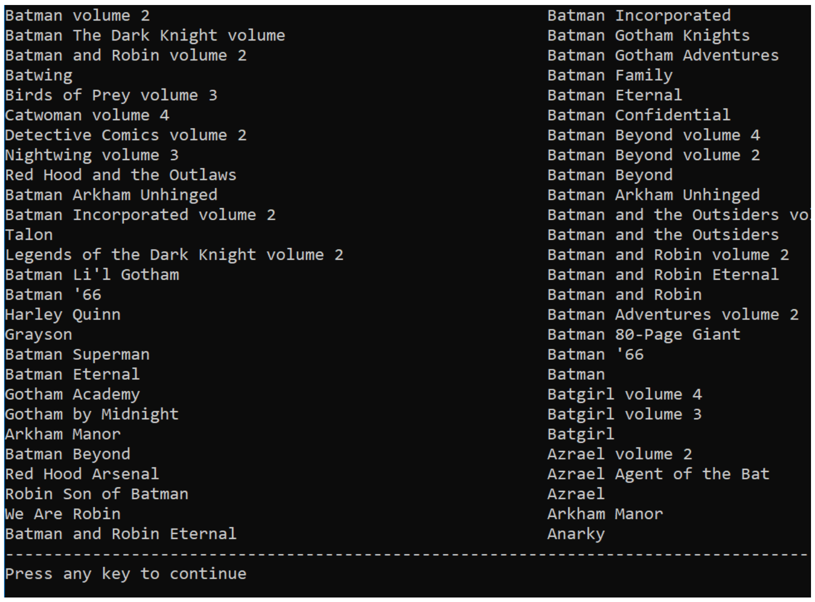
PROGRAMMER'S CHALLENGE

As with every programmer's challenge, remember the following…

1. Do the rubric first. Make sure you have something to turn in for the assignment.
2. When attempting the challenge, don't break your other code.
3. You have other assignments so don't sacrifice them to work on the challenges.

Sorting Challenge

Add the ability to sort the list in the opposite direction. This is an alternating behavior such that one time, the sort is ascending. Then the next time that same sort method is selected, it sorts in descending order. It alternates between ascending and descending.



```
Batman volume 2                                         Batman Incorporated
Batman The Dark Knight volume                           Batman Gotham Knights
Batman and Robin volume 2                               Batman Gotham Adventures
Batwing                                                 Batman Family
Birds of Prey volume 3                                  Batman Eternal
Catwoman volume 4                                       Batman Confidential
Detective Comics volume 2                               Batman Beyond volume 4
Nightwing volume 3                                      Batman Beyond volume 2
Red Hood and the Outlaws                                Batman Beyond
Batman Arkham Unhinged                                  Batman Arkham Unhinged
Batman Incorporated volume 2                            Batman and the Outsiders vo
Talon                                                   Batman and the Outsiders
Legends of the Dark Knight volume 2                     Batman and Robin volume 2
Batman Li'l Gotham                                      Batman and Robin Eternal
Batman '66                                              Batman and Robin
Harley Quinn                                            Batman Adventures volume 2
Grayson                                                 Batman 80-Page Giant
Batman Superman                                         Batman '66
Batman Eternal                                          Batman
Gotham Academy                                          Batgirl volume 4
Gotham by Midnight                                      Batgirl volume 3
Arkham Manor                                            Batgirl
Batman Beyond                                           Azrael volume 2
Red Hood Arsenal                                        Azrael Agent of the Bat
Robin Son of Batman                                     Azrael
We Are Robin                                            Arkham Manor
Batman and Robin Eternal                                Anarky
-----------------------------------------------------------------------------------
Press any key to continue
```