Timothy P. McCrary
11/15/2018

CS 2302 Lab 5 – Option A

INTRODUCTION

For this lab, the problem we are trying to solve is how to implement a min heap and heap sort. To prove our code works we just need to make sure it runs on a list.

PROPOSED SOLUTION DESIGN AND IMPLEMENTATION

In this lab, I decided to separate the code into different files in order to keep code modular and easy to read (also because the professor stated to separate the code segments with different field). The min heap object got its own class and file, the heap sort got its own file, and a main runner file was created to call the other functions and methods.

First, the partial implementation of the mine heap data structure needed to be completed. The insert method needed to be completed with the percolate_up method, which was based off the Zybooks implementation, however, it was modified for a min heap instead of a max heap. All that needed to be changed was to check if an item is greater than the parent, and if it that was true, nothing needed to be done as it met the properties of the heap. In the same file (but not class), a function was created to build a min heap object with a given list and return the built min heap. Furthermore, one more function needed to be created that took numbers separated by comas from a file and put them in the list, which then can be used to build a min heap.

Second, the heap sort code needed to be created. In a new file, heap sort and percolate_down were created based off the implementation from the Zybooks. Only variable names needed to be changed to make code clear, as heap sort sorts the items in ascending order, making the implementation the same for both the max and min heaps.

Thirdly, a runner file needed to be created to test the min heap object and heap sort. The two files were imported and tested on a main function. Nothing needed to be inputted by the user due to the list being the only changing input, so info about the min heap was printed (i.e. a list before built to a min heap, min heap after heap sort, etc.)

EXPERIMENTAL RESULTS

Five tests that changed the list input were conducted on this lab to test certain cases. Test one used a list with 1,000 numbers in order to check that the code worked in the first place. Test two used a hard-coded list with 10 numbers. With test one and two proving the code could run with a list from a file or a hard-coded, test three gives an empty list, seeing if the code can handle empty values. The fourth test used a list that was not formatted correctly, testing to see how the code handles an incorrect input. The fifth and final test was given a file that doesn't exist.

TEST 1 – LIST FROM FILE WITH 1,000 NUMBERS

```
_________TEST 1_________: Uses file with 1,000 numbers.
List before converting to min heap:
 [468, 4, 583, 263, 848, 84, 74, 152, 766, 53, 530, 147, 749, 980, 668, 480, 288, 218, 479, 296, 100, 678, 51, 812, 333, 921, 318, 227
After converting to min heap:
 [1, 4, 2, 6, 10, 3, 14, 15, 46, 12, 19, 5, 7, 40, 18, 54, 34, 66, 81, 45, 21, 25, 22, 11, 13, 23, 8, 48, 56, 62, 28, 138, 58, 142, 52
After using heap sort:
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 3

Running time after building min heap then sorting it = 0.031252145767211914 seconds
```

TEST 2 – HARD CODED LIST WITH 10 NUMBERS

```
_________TEST 2_________: Uses hard coded list.
List before converting to min heap:
 [12, 4567, 2, 4, 1, 5678, 34, 43, 6, 7]
After converting to min heap:
 [1, 2, 12, 6, 4, 5678, 34, 4567, 43, 7]
After using heap sort:
 [1, 2, 4, 6, 7, 12, 34, 43, 4567, 5678]

Running time after building min heap then sorting it = 0.0 seconds
```

TEST 3 – EMPTY LIST

```
_________TEST 3_________: Uses empty file.
List before converting to min heap:
 []
After converting to min heap:
 []
After using heap sort:
 []

Running time after building min heap then sorting it = 0.0 seconds
```

TEST 4 – INCORRECTLY FORMATTED FILE

```
_________TEST 4_________: Uses file not formatted.
ERROR: File not formatted correctly.
List before converting to min heap:
 []
After converting to min heap:
 []
After using heap sort:
 []

Running time after building min heap then sorting it = 0.0 seconds
```

TEST 5 – NON-EXISTING FILE

```
_________TEST 5_________: Uses file that doesn't exist.
ERROR: File does not exist.
List before converting to min heap:
 []
After converting to min heap:
 []
After using heap sort:
 []

Running time after building min heap then sorting it = 0.0 seconds
```

CONCLUSION

In this lab I learned how to use and implement a min heap with heap sort. Furthermore, I relearned how to uses different files to make code more organized and easier to work with. All in all, this was a good lab that was enjoyable to work on.

APPENDIX

MAINRUNNER.PY

```
#
*********************************************************************************************
*******************************
# NAME: Timothy P. McCrary
# CLASS: CS 2302
# LAB 5 OPTION A
# INSTRUCTOR: Diego Aguirre
# TA: Manoj Pravaka Saha
# DATE: 11/25/2018
# PURPOSE: To understand and use a min heap.
#
*********************************************************************************************
*******************************

import time
import MinHeap
from MinHeapSort import heap_sort


def main():
    # __________TEST
1_____________________________________________________________________________________
_____
    print('\n_________TEST 1_________: Uses file with 1,000 numbers.')
    # Creates list from file. This is the test we will be using.
    input_list = MinHeap.file_to_list("Numbers.txt")
    print('List before converting to min heap:\n', input_list)

    start_time = time.time()

    # Builds min heap from list then prints list.
```

```
    min_heap = MinHeap.build_min_heap(input_list)
    print('After converting to min heap:\n', min_heap.heap_array)

    # Sorts min heap using heap sort then prints list once sorted.
    heap_sort(min_heap.heap_array)
    print('After using heap sort:\n', min_heap.heap_array)

    print("\nRunning time after building min heap then sorting it = %s seconds" %
(time.time() - start_time))

    # __________TEST
2_____________________________________________________________________________
_____________
    print('\n__________TEST 2__________: Uses hard coded list.')
    # Creates list from file. This is the test we will be using.
    input_list = [12, 4567, 2, 4, 1, 5678, 34, 43, 6, 7]
    print('List before converting to min heap:\n', input_list)

    start_time = time.time()

    # Builds min heap from list then prints list.
    min_heap = MinHeap.build_min_heap(input_list)
    print('After converting to min heap:\n', min_heap.heap_array)

    # Sorts min heap using heap sort then prints list once sorted.
    heap_sort(min_heap.heap_array)
    print('After using heap sort:\n', min_heap.heap_array)

    print("\nRunning time after building min heap then sorting it = %s seconds" %
(time.time() - start_time))

    # __________TEST
3_____________________________________________________________________________
_____________
    print('\n__________TEST 3__________: Uses empty file.')
    # Creates list from file. This is the test we will be using.
    input_list = MinHeap.file_to_list("Empty.txt")
    print('List before converting to min heap:\n', input_list)

    start_time = time.time()

    # Builds min heap from list then prints list.
    min_heap = MinHeap.build_min_heap(input_list)
    print('After converting to min heap:\n', min_heap.heap_array)

    # Sorts min heap using heap sort then prints list once sorted.
    heap_sort(min_heap.heap_array)
    print('After using heap sort:\n', min_heap.heap_array)

    print("\nRunning time after building min heap then sorting it = %s seconds" %
(time.time() - start_time))

    # __________TEST
4_____________________________________________________________________________
_____________
    print('\n__________TEST 4__________: Uses file not formatted.')
    # Creates list from file. This is the test we will be using.
    input_list = MinHeap.file_to_list("Not Formatted.txt")
    print('List before converting to min heap:\n', input_list)

    start_time = time.time()

    # Builds min heap from list then prints list.
```

```
    min_heap = MinHeap.build_min_heap(input_list)
    print('After converting to min heap:\n', min_heap.heap_array)

    # Sorts min heap using heap sort then prints list once sorted.
    heap_sort(min_heap.heap_array)
    print('After using heap sort:\n', min_heap.heap_array)

    print("\nRunning time after building min heap then sorting it = %s seconds" %
(time.time() - start_time))

    # __________TEST
5_____________________________________________________________________________
_____________
    print('\n__________TEST 5__________: Uses file that doesn\'t exist.')
    # Creates list from file. This is the test we will be using.
    input_list = MinHeap.file_to_list("Does Not Exist.txt")
    print('List before converting to min heap:\n', input_list)

    start_time = time.time()

    # Builds min heap from list then prints list.
    min_heap = MinHeap.build_min_heap(input_list)
    print('After converting to min heap:\n', min_heap.heap_array)

    # Sorts min heap using heap sort then prints list once sorted.
    heap_sort(min_heap.heap_array)
    print('After using heap sort:\n', min_heap.heap_array)

    print("\nRunning time after building min heap then sorting it = %s seconds" %
(time.time() - start_time))


if __name__ == '__main__':
    main()
```

MinHeap.py

```
#
*******************************************************************************
*******************************
# NAME: Timothy P. McCrary
# CLASS: CS 2302
# LAB 5 OPTION A
# INSTRUCTOR: Diego Aguirre
# TA: Manoj Pravaka Saha
# DATE: 11/25/2018
# PURPOSE: To understand and use a min heap.
#
*******************************************************************************
*******************************


# Class for min heap.
class Heap:
    def __init__(self):
        self.heap_array = []

    # Based off Zybook implementation modified for a min heap.
    def percolate_up(self, item_index):
        while item_index > 0:
            # compute the parent node's index
            parent_index = (item_index - 1) // 2
```

```
            # check for a violation of the max heap property
            if self.heap_array[item_index] >= self.heap_array[parent_index]:
                # no violation, so percolate up is done.
                return
            else:
                # swap heap_array[node_index] and heap_array[parent_index]
                temp = self.heap_array[item_index]
                self.heap_array[item_index] = self.heap_array[parent_index]
                self.heap_array[parent_index] = temp

                # continue the loop from the parent node
                item_index = parent_index

    # Inserts item into min heap and sorts it.
    def insert(self, k):
        # Item placed at end of min heap list.
        self.heap_array.append(k)

        # This is where the item is sorted in the heap.
        self.percolate_up(len(self.heap_array) - 1)

    # Extracts first item from min heap.
    def extract_min(self):
        if self.is_empty():
            return None

        min_elem = self.heap_array.pop(0)

        return min_elem

    # Returns TRUE if min heap is empty.
    def is_empty(self):
        return len(self.heap_array) == 0


# Function that creates a min heap from a list.
def build_min_heap(input_list):
    min_heap = Heap()
    for item in input_list:
        min_heap.insert(item)

    return min_heap


# Function that stores data from a file into a list. File contents must be separated
by commas.
def file_to_list(file_name):
    int_list = []
    empty_list = []

    try:
        file = open(file_name)
    except FileNotFoundError:
        print('ERROR: File does not exist.')
        return empty_list

    for line in file:
        string_list = line.split(", ")
        for number in string_list:
            try:
                number = int(float(number))
                int_list.append(number)
```

```
            except ValueError:
                print('ERROR: File not formatted correctly.')
                return empty_list

    return int_list
```

MinHeapSort.py

```
#
********************************************************************************
*******************************
# NAME: Timothy P. McCrary
# CLASS: CS 2302
# LAB 5 OPTION A
# INSTRUCTOR: Diego Aguirre
# TA: Manoj Pravaka Saha
# DATE: 11/25/2018
# PURPOSE: To understand and use a min heap.
#
********************************************************************************
*******************************


# Based off Zybook implementation.
def percolate_down(item_index, min_heap, list_size):
    child_index = 2 * item_index + 1
    value = min_heap[item_index]

    while child_index < list_size:
        # Find the max among the item and all the item's children
        max_value = value
        max_index = -1
        i = 0
        while i < 2 and i + child_index < list_size:
            if min_heap[i + child_index] > max_value:
                max_value = min_heap[i + child_index]
                max_index = i + child_index
            i = i + 1

        if max_value == value:
            return

        # Swap heap_list[node_index] and heap_list[max_index]
        temp = min_heap[item_index]
        min_heap[item_index] = min_heap[max_index]
        min_heap[max_index] = temp

        item_index = max_index
        child_index = 2 * item_index + 1


# Based off Zybook implementation.
def heap_sort(min_heap):
    i = len(min_heap) // 2 - 1
    while i >= 0:
        percolate_down(i, min_heap, len(min_heap))
        i = i - 1

    i = len(min_heap) - 1
    while i > 0:
        # Swap min_heap[0] and min_heap[i]
        temp = min_heap[0]
```

```
        min_heap[0] = min_heap[i]
        min_heap[i] = temp

        percolate_down(0, min_heap, i)
        i = i - 1
```

“I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.”

X____Timothy P. McCrary____