Ricardo Godoy

Lab Report

The purpose of this program is to implement a Min Heap using an array. The program implements a Min Heap using a text file with numbers separated by comas. The program first reads the text file containing the numbers and then inserts them into the min heap array. After that is performed, the program will sort the array using heap sort and will display the unsorted and sorted array.

In my MinHeap class I included three functions: insert, percolate\_up, and extract-min. Respectively, these functions insert an element into a heap, then percolates up the element that was inserted, and it extracts the minimum value of the heap. Percolating\_up performs a series of swapping between the node and the node’s parent until the min heap conditions is met.

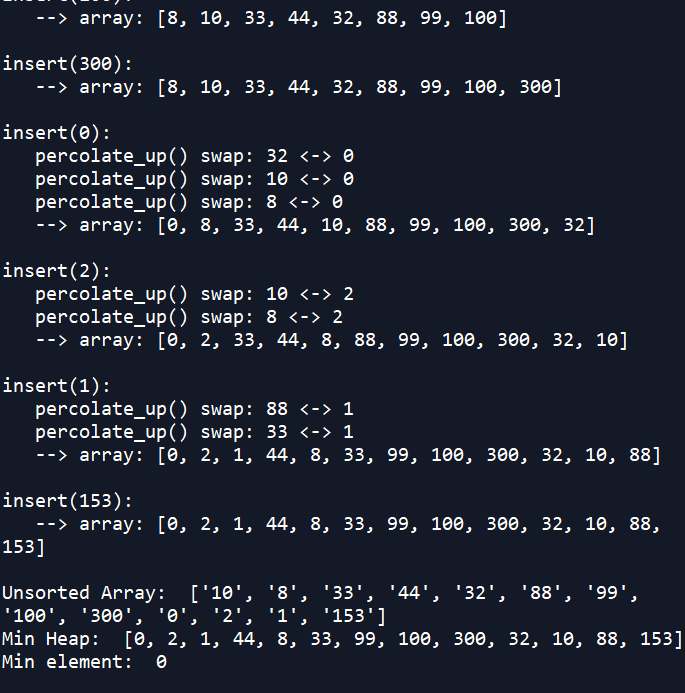
In addition to that, I also implemented a min\_heap\_percolate\_down function that takes the node index, the heap list, and the list size as parameters, and percolates down the elements on the min heap.

The read\_file functions reads the file previously created and inserts the numbers into an array. After that, the heap\_sort function will sort the numbers in the array, in order to meet the conditions of the min heap. This last function will call the min\_percolate\_down function to keep the Min Heap array implemented correctly.

In the first test, I decided to use this list of numbers:

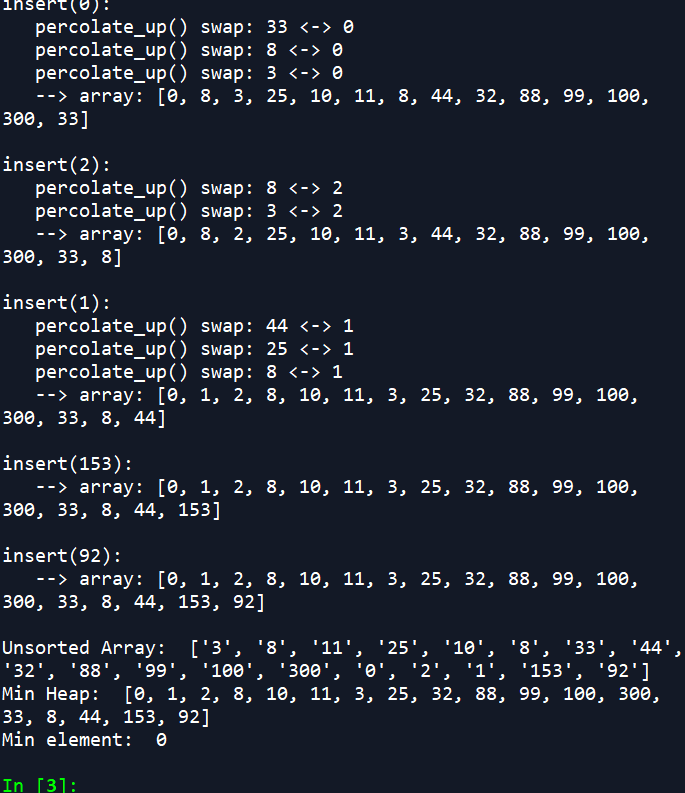
10,8,33,44,32,88,99,100,300,0,2,1,153

Here is the result:



Then, the second test was with these numbers:

3,8,11,25,10,8,33,44,32,88,99,100,300,0,2,1,153,92



In this lab, I learned how to implement not only Min Heaps, but Max Heaps. Along with that, I also learned what are the different uses of min heaps and the uses of it.

Appendix

# -\*- coding: utf-8 -\*-

"""

Created on Wed Nov 21 09:15:44 2018

@author: rgodoy

"""

# Course: CS2302

# Author: Ricardo Godoy

# Assignment: Lab 5 A

# T.A: Saha, Manoj

# Instructor: Diego Aguirre

# Date of last modification: 11/27/18

#The purpose of this lab is to implement a Min Heap. This program reads a text file with numbers

#separated by a coma, inserts them into a Heap, and sorts them accordingly.

class MinHeap:

def \_\_init\_\_(self):

self.heap\_array = []

def insert(self, k):

# add the new value to the end of the array.

print("insert(%d):" % k)

self.heap\_array.append(k)

# percolate up from the last index to restore heap property.

self.percolate\_up(len(self.heap\_array) - 1)

def percolate\_up(self, node\_index):

while node\_index > 0:

# compute the parent node's index

parent\_index = (node\_index - 1) // 2

# check for a violation of the min heap property

if self.heap\_array[node\_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]

print(" percolate\_up() swap: %d <-> %d" % (self.heap\_array[parent\_index], self.heap\_array[node\_index]))

temp = self.heap\_array[node\_index]

self.heap\_array[node\_index] = self.heap\_array[parent\_index]

self.heap\_array[parent\_index] = temp

# continue the loop from the parent node

node\_index = parent\_index

def extract\_min(self):

if len(self.heap\_array) < 0:

return None

# save the min value from the root of the heap.

min\_elem = self.heap\_array[0]

# move the last item in the array into index 0.

replace\_value = self.heap\_array.pop()

if len(self.heap\_array) > 0:

self.heap\_array[0] = replace\_value

# percolate up to restore min heap property.

self.percolate\_up(0)

# return the min value

print("Min element: ", min\_elem)

return min\_elem

#Percolates down the elements on the min heap

def min\_heap\_percolate\_down(node\_index, heap\_list, list\_size):

child\_index = 2 \* node\_index + 1

value = heap\_list[node\_index]

while child\_index < list\_size:

# Find the max among the node and all the node's children

min\_value = value

min\_index = -1

i = 0

while i < 2 and i + child\_index < list\_size:

if heap\_list[i + child\_index] > min\_value:

min\_value = heap\_list[i + child\_index]

min\_index = i + child\_index

i = i + 1

if min\_value == value:

return

# Swap heap\_list[node\_index] and heap\_list[max\_index]

temp = heap\_list[node\_index]

heap\_list[node\_index] = heap\_list[min\_index]

heap\_list[min\_index] = temp

node\_index = min\_index

child\_index = 2 \* node\_index + 1

# Reads file and inserts items into heap

def read\_file(filename, h, input\_list):

f = open(filename)

file = f.readline()

numbers = file.split(",")

for number in numbers:

input\_list.append(number)

h.insert(int(number))

print(' --> array: %s\n' % h.heap\_array)

# Sorts the list of numbers using the heap sort algorithm

def heap\_sort(numbers):

# Heapify numbers list

i = len(numbers) // 2 - 1

while i >= 0:

min\_heap\_percolate\_down(i, numbers, len(numbers))

i = i - 1

i = len(numbers) - 1

while i > 0:

# Swap numbers[0] and numbers[i]

temp = numbers[0]

numbers[0] = numbers[i]

numbers[i] = temp

min\_heap\_percolate\_down(0, numbers, i)

i = i - 1

def main():

h = MinHeap()

input\_list = []

filename = "HeapFile.txt"

read\_file(filename, h, input\_list)

print("Unsorted Array: ", input\_list)

print("Min Heap: ", h.heap\_array)

h.extract\_min()

main()