Calvin Tracy

Professor Kopp

Data Structures 222

06 May 2018

Final Project Bubble Sort

My code will be vastly different from any other students, I have gone through tutoring outside of class, and have learned some new python techniques to use in my programs which you will see in the source code. However I do keep it as simple as possible, for example, I read my assigned file into my program and sorted it into separate lists, each data field having its own index. Date at index 0 max temp at index 2 and min temp at index two.

From this point I wrote the code which converted the max and min temp to Fahrenheit and appended it to another list called data\_Fields, which I emptied out in the for loop so that every date with its associated temperatures was in its own list which I added to another list called data\_File. Had I not emptied out the data\_Fields list every iteration it would have added the dates and temperatures all in on giant list instead of seperate lists, this is not what we wanted and would make it much harder to sort through the data when we get to the userInput() function.

The userInput() function simply asks the user to enter a date to see the temperatures for that year until they enter ‘Done.’ Pretty straight forward. Before we call this function however, we must first sort through the list. During my testing sorting through a list of 5-10 dates taken from the file which was read into the program that had around 24,000 records, each method sorted very fast. Sorting the entire record of 24,000 was an entirely different story however.

The bubbleSort() took almost 3 minutes to sort the file while the shellSort took around 3 seconds. A massive difference in timing. While the bubbleSort() ‘bubbles’ up each number, switching them if it finds an index less than the one it is currently on, it is inefficient in timing. We can see the number of comparisons by simply setting up a ‘count’ variable and increase it by one at the end of the loop. Timing is inefficient because it goes through every single record index by index classifying it as n2. The shellSort() however goes with another method choosing to divide the list and round it down, allowing it to find data and index locations much faster and increasing the sorting efficiency this way.

As for my searching method, I choose a bit of an unorthodox method, I decided to created a dictionary. Initializing each month 1-12 with a key value of 0. I then passed in alist, and then data\_File (which has the date max and min temp) into alist. My for loop goes through the length of alist, and created a variable ‘month’ to equal the index of the month in alist. The next few lines calculate the average temp with simple math and use of indexes. I chose this method because I wanted to try something new and different than what we had learned in class. I like having all of my dates to have their key values specific to them and easily found in a dictionary.

In conclusion, I chose my code and its methods out of personal preference, I like using different techniques as I believe I can make my program more efficient, and allowing me to try new this will help me become a better programer. Learning new ways to code is a challenging way to help yourself become better and I will do everything I can to be the best I can be as a programming student, and eventually as a programmer for a company. I will make myself a valued asset one that merits recognition.