Computing Insertion Sort using Python

Homework #2

By

Mario Pendleton

CS 303 [Algorithms and Data Structures](https://uab.instructure.com/courses/1507655)

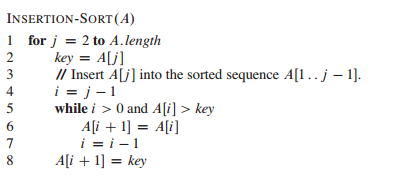
September 8, 2019

### Problem Specification

Implement a method that will sort a given array using the insertion sort algorithm. Write a driver program to test the insertion algorithms implemented. Read the input file “input\_100.txt” for the input numbers and store them in an array. Sort this array using  insertion sort. Test the program for the different size input files provided in Canvas. Record the runtime for insertion sort on various sized arrays by using the provided files.  Comment on how the execution time of insertion sort varies with size of the input array. Use a table or plot to summarize the results and document your observations and explanations in the report.

### Program Design

This program requires an array of data that will be sorted using the insertion sort method. The method was designed in python after the pseudo below.



The following steps were required to develop this program:

1. Re-write the insertion sort pseudo code to a python method (insertionSort(list)).
2. use the a method to read the following txt files and covert them into arrays to be sorted

* input\_100.txt
* input\_1000.txt
* input\_5000.txt
* input\_10000.txt
* input\_50000.txt
* input\_100000.txt
* input\_500000.txt

The following methods were defined within the lab2.py:

1. insertionSort (list)

Method that clones an array then performs an insertion sort on the original array. It then takes the cloned array and performs a python sort (foo.sort()). The method returns True if the insertion sort array is equal to the python sort array. If False, the method will print the python sort results for comparison.

Constructor that creates an instance of an IdealWeight object, setting the feet and inches variables to the user inputs.

1. sampleList (n)

Method to create a random array of n^2 integer characters.

1. load (path): int

Method that opens a txt file and creates an array.

### Testing Plan

Sample string inputs were selected to see the program could sort (a) none integer values, (b) string and integer values, (c) string integers with strings, (d) floats, (e) random list of integers, (f) values loaded from the txt files, (g) empty array, (h) single integer, and (i) single string.

### Test Cases

The test cases are shown in the table below using a MacBook Pro 16GB, 8 core 2.3 GHz Intel Core i9:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case Number | Input Values | Expected Output | Actual Output | Time |
| (a) | ["apple", "cherry", "mango", "banana", "dragon fruit"] | ['apple', 'banana', 'cherry', 'dragon fruit', 'mango'] | ['apple', 'banana', 'cherry', 'dragon fruit', 'mango'] | 0.000711 |
| (b) | [123,"apple",5,6,"green"] | not supported between instances of 'int' and 'str' | not supported between instances of 'int' and 'str' | N/A |
| (c) | ["123","apple","5","53","5a","6","green"] | ['123', '5', '53', '5a', '6', 'apple', 'green'] | ['123', '5', '53', '5a', '6', 'apple', 'green'] | 0.000672 |
| (d) | [10.1,10.9,9.3,7.4,6.49,2.0,1.999,0.01,5.999] | [0.01, 1.999, 2.0, 5.999, 6.49, 7.4, 9.3, 10.1, 10.9] | [0.01, 1.999, 2.0, 5.999, 6.49, 7.4, 9.3, 10.1, 10.9] | 0.000764 |
| (e) | sampleList(n) n = 5 | Random array of 8 integers  [4, 8, 1, 4, 7, 4, 2, 0] | Random array of 8 integers sorted  [0, 1, 2, 4, 4, 4, 7, 8] | 0.000687 |
| (f) | load(path) path = input\_100.txt  [4, 50, 34, 40, 22, 54, 94, 3, 94, 38, 8, 95, 0, 36, 54, 54, 81, 30, 24, 98, 12, 25, 43, 0, 52, 52, 88, 22, 83, 70, 96, 57, 89, 53, 13, 64, 74, 18, 37, 86, 73, 76, 15, 1, 93, 69, 77, 81, 29, 78, 14, 45, 67, 1, 0, 41, 60, 63, 74, 16, 75, 75, 36, 49, 68, 5, 67, 29, 15, 84, 47, 77, 40, 80, 24, 61, 25, 7, 85, 83, 81, 47, 10, 39, 22, 72, 87, 64, 92, 27, 50, 69, 12, 54, 23, 85, 38, 75, 73, 94] | [0, 0, 0, 1, 1, 3, 4, 5, 7, 8, 10, 12, 12, 13, 14, 15, 15, 16, 18, 22, 22, 22, 23, 24, 24, 25, 25, 27, 29, 29, 30, 34, 36, 36, 37, 38, 38, 39, 40, 40, 41, 43, 45, 47, 47, 49, 50, 50, 52, 52, 53, 54, 54, 54, 54, 57, 60, 61, 63, 64, 64, 67, 67, 68, 69, 69, 70, 72, 73, 73, 74, 74, 75, 75, 75, 76, 77, 77, 78, 80, 81, 81, 81, 83, 83, 84, 85, 85, 86, 87, 88, 89, 92, 93, 94, 94, 94, 95, 96, 98] | [0, 0, 0, 1, 1, 3, 4, 5, 7, 8, 10, 12, 12, 13, 14, 15, 15, 16, 18, 22, 22, 22, 23, 24, 24, 25, 25, 27, 29, 29, 30, 34, 36, 36, 37, 38, 38, 39, 40, 40, 41, 43, 45, 47, 47, 49, 50, 50, 52, 52, 53, 54, 54, 54, 54, 57, 60, 61, 63, 64, 64, 67, 67, 68, 69, 69, 70, 72, 73, 73, 74, 74, 75, 75, 75, 76, 77, 77, 78, 80, 81, 81, 81, 83, 83, 84, 85, 85, 86, 87, 88, 89, 92, 93, 94, 94, 94, 95, 96, 98] | 0.001553 |
| (f) | load(path) path = input\_1000.txt | (Data) Insertion sort was successful. No errors found.  True | (Data) Insertion sort was successful. No errors found.  True | 0.099092 |
| (f) | load(path) path = input\_5000.txt | (Data) Insertion sort was successful. No errors found.  True | (Data) Insertion sort was successful. No errors found.  True | 2.29648 |
| (f) | load(path) path = input\_10000.txt | (Data) Insertion sort was successful. No errors found.  True | (Data) Insertion sort was successful. No errors found.  True | 9.41198 |
| (f) | load(path) path = input\_50000.txt | (Data) Insertion sort was successful. No errors found.  True | (Data) Insertion sort was successful. No errors found.  True | 229.453 |
| (f) | load(path) path = input\_100000.txt | (Data) Insertion sort was successful. No errors found.  True | (Data) Insertion sort was successful. No errors found.  True | 904.434601 |
| (f) | load(path) path = input\_500000.txt | (Data) Insertion sort was successful. No errors found.  True | (Data) Insertion sort was successful. No errors found.  True | 22824.3855 |
| (g) | [] | [] | [] | 0.000651 |
| (h) | [13] | [13] | [13] | 0.000657 |
| (i) | [“red”] | [‘red’] | [‘red’] | 0.000670 |

### Analysis and Conclusions

Insertion sort was successfully computed for all listed test cases. Test case (b) failed as expected. Testing confirmed that the algorithm was able to properly sort the known test cases. The known program limitation was confirmed.

The program could be improved by removing the Python sort that verifies if the selection sort was implemented properly.

### References

Textbook, python.org, and examples provided in the assignment.

### 

