Computer Science I
Selection Sort

CSCI-141
Homework 7

09/25/2020

1 Problem

Compose a program that reads numbers from a file, uses the function *selection_sort* to sort the numbers, and has comments that answer the questions in this assignment.

The *selection_sort* algorithm involves the following *invariant*. The sequence of values can be cut into two sub-sequences such that the first sub-sequence is in order *and* all the elements are less than or equal to the elements of the second sub-sequence. To satisfy this invariant, initially, the first sub-sequence is empty. To extend the first sub-sequence while preserving the invariant, we find the smallest value of the second sub-sequence and move it to the end of the first sub-sequence.

Here is an example of selection sorting:

	$45, 7, 5, 24, 12, 1$
1	$7, 5, 24, 12, 45$
$1, 5$	$7, 24, 12, 45$
$1, 5, 7$	$24, 12, 45$
$1, 5, 7, 12$	$24, 45$
$1, 5, 7, 12, 24$	45
$1, 5, 7, 12, 24, 45$	

The pseudocode[1] below suggests how the selection_sort algorithm might be implemented. The function `find_min_from` is assumed to return the index of the smallest value in the list for the specified range.

```
procedure selection_sort(lst)
    for mark in 0 to len(lst) - 1
        swap(lst, mark, find_min_from(lst, mark))
```

1.1 Constraints

- Use iteration to implement the algorithm (i.e., use *for* or *while* loops).
- Sort the list of numbers in-place; create no extra lists.
- Implement the specified algorithm and no other.
- Do not use any built-in Python sort functions.

1.2 Program Tasks

The solution must do the following:

---

1. Pseudocode is a form of description that looks like some programming language but is not an actual language. This permits the description of an algorithm with less concern for syntactic correctness.

1. Prompt for an input file name. The file will consist of integer values, one per line.
2. Open the file, read the file, store the data in a list, and close the file.
3. Print the list.
4. Call the `selection_sort` function passing the list.
5. Reprint the list, which is now sorted.

The file must be named `selection_sort.py`. Develop and use your own input files to test the solution.

1.3 Questions To Answer

Answer the following questions after implementing the program. **Write and number answers to the questions in the file docstring that contains your name at the start of the program file.**

1. In what kind of test case does `insertion_sort` perform better than `selection_sort`? Clearly describe the test case.

2. Why does `selection_sort` perform worse than `insertion_sort` in that test case?

Copy the `insertion_sort` **program file** from lecture, and customize the function to make it work with files. Run the `insertion_sort` code with the same file inputs to compare performance and answer the above questions.

1.4 Grading

- 40%: Performance
  Graders will use a test program that runs tests on the program. Failure to name the file or the function properly will cause the test harness to fail, and you will lose points for that.
- 25%: Sorting code implementation
  You will get NO credit for submitting a non-selection-sort implementation.
- 20%: File processing and `main` code
- 15%: Answers to questions in the file module docstring
  You will lose points for failure to put the answers into the correct place in the file.
- [**New**] **Deductions off the Top** are up to 10% for Style, Layout, Documentation. The code should follow the course coding standard on the resources page. Deductions will be for missing author name, missing docstrings for functions, missing the docstring for the module itself, and having an incorrect file name or function name.

1.5 Submission

Zip the solution file named `selection_sort.py` into a file named `hw07.zip`, and upload the zip file to the myCourses dropbox for this assignment.