CS 234 Fall 2019
Naomi Nishimura

Assignment 2

Coverage: Python session 2; Modules 3 and part of 4.

This assignment consists of a written component and a programming component. Please read the course website carefully to ensure that you submit each component correctly. For full marks, you are required not only to have a correct solution, but also to adhere to the requirements of the assignment question and the style guide, including aspects of the design recipe.

All the questions in the written component should be submitted as a single pdf file with the name `a2written.pdf`. Although scanned documents are acceptable, please keep in mind that only legible writing will be marked (subject to the markers' discretion) and that MarkUs forces a limit of 5000 KB per file, which might be exceeded by high resolution scans.

Written component

For full marks, you are expected to provide a brief justification of any answer you provide. For asymptotic notation, informal arguments are sufficient (that is, using formal definitions is not required).

W1. *[10 marks]* Suppose you wish to determine the number of data items in an ADT Ranking such that the data item is equal to its rank. We will consider two ways of solving the problem: using ADT operations and augmenting the ADT.

(a) *[4 marks]* Write a pseudocode function Equals that consumes a ranking R and produces the number of data items in R equal to their ranks. Your function should use the ADT operations for an ADT Ranking. In this question, you do not know how the ADT has been implemented.

(b) *[2 marks]* Now consider augmenting the ADT Ranking to include an operation Matches that produces the number of data items equal to their ranks. Briefly describe the algorithm that you would use for a contiguous implementation (where the array index is equal to the rank) and the algorithm that you would use for the linked implementation (where the position of a linked node in a linked list is equal to the rank). Your algorithms should refer directly to the implementation; they should not use other ADT operations or the algorithms that implement them.

You do not need to use pseudocode in this question.

(c) *[4 marks]* Give the worst-case running times of your solutions to subquestion W1b as well as the worst-case running time for your solution to subquestion W1a, once for each of the implementations given in subquestion W1b (all in asymptotic notation, using Θ, as a function of n, the number of data items being stored). For each of the four running times you give for this subquestion, give a brief justification of your answer. For the ADT operations, you may use worst-case costs discussed in class for each implementation.

W2. *[10 marks]* Suppose that the ADT Swaplist (defined in Assignment 1) has been implemented as a linked list such that each linked node contains both a data item and a position as well as a link to the next linked node. You cannot make any assumption about the order in which the linked nodes are stored. **Note: Although in the programming question you may only change links between nodes, for this question, you may change links between nodes and you may also assign new values to nodes.**

(a) *[2 marks]* Give a brief description of the algorithm used to implement the operation SWAP*(S, Pos)*. You may use a few sentences; pseudocode is not required.

(b) *[2 marks]* Give the **best**-case cost of your solution to subquestion W2a using asymptotic notation (Θ) as a function of n, the number of items in the swaplist. Briefly justify your answer.

(c) *[2 marks]* Give a brief description of the algorithm used to implement the operation LOOK_UP*(S, Pos)*. You may use a few sentences; pseudocode is not required.

(d) *[2 marks]* Give the **worst**-case cost of your solution to subquestion W2c using asymptotic notation (Θ) as a function of n, the number of items in the swaplist. Briefly justify your answer.

(e) *[2 marks]* Now consider the solutions to subquestions W2b and W2d expressed in asymptotic notation (Θ) as a function of both n, the number of items in the swaplist, and k, the value of *Pos*. If the solutions would be different, explain how and why. If the solutions would be the same, explain why.

Programming component

Please read the course website carefully to ensure that you are using the correct version of Python and the correct style. For full marks, you are required not only to have a correct solution, but also to adhere to the requirements of the assignment question and the style guide, including aspects of the design recipe.

Although submitting tests is not required, it is highly recommended that you test your code. For each assignment question, create a testing file that imports your submission and tests the code. Do not submit your testing file.

You may import any of the following files: `check.py` and `linked.py`.

P1. *[20 marks]* In this question, you will implement the ADT Swaplistplus as a circular linked list with a single pointer to the last item in the list.

The ADT Swaplistplus is similar to the ADT Swaplist introduced in Assignment 1, but also contains a new operation BIG_SWAP. **Note: You must use the specified data structure to receive marks on this assignment. In particular, do not modify the file `swaplist.py` from Assignment 1. Each of your operation implementations should make direct use of the data structure instead of using other ADT operations; you should not, for example, implement BIG_SWAP by copying the work done for Assignment 1. Marks will be deducted for any implementation that deviates from these guidelines. Remember that the goal here is for you to practice working directly with a data structure in the role of the provider; the**

purpose of the assignment is not to complete the exercises in whatever way you can, but rather in a manner that exercises the ideas discussed in class.

Name	Returns	Changes
CREATE*(Sequence)*	a new swaplistplus storing the data items in *Sequence*, where the data item in position *Pos* in *Sequence* is placed in position *Pos* in the swaplistplus (positions in both *Sequence* and the swaplistplus start at 0); *Sequence* is nonempty	
LENGTH*(S)*	number of items stored in swaplistplus S	
LOOK_UP*(S, Pos)*	produces the item in position *Pos* in swaplistplus S, where $0 \leq Pos <$ LENGTH*(S)*	
SWAP*(S, Pos)*		swaps items in positions *Pos* and $Pos + 1$ in swaplistplus S, where $0 \leq Pos$ and $Pos + 1 <$ LENGTH*(S)*
BIG_SWAP*(S, One, Two)*		swaps items in positions *One* and *Two* in swaplistplus S, where $0 \leq One < Two <$ LENGTH*(S)*

Once you have created a swaplistplus using `Swaplistplus()`, you should not create any new nodes or change the values in nodes. In particular, in your implementations of `swap` and `big_swap`, you should change links between nodes, not assign new values to nodes.

You should use the module `linked.py` on the course website for the nodes to use in your circular linked list.

Using the provided interface `swaplistplusinterface.py`, implement all the methods of the ADT. You will not be tested on `__str__`, but you might find it useful for debugging. Notice that `__eq__` is required, as it will be used in testing your code.

Submit your work in a file with the name `swaplistplus.py`.