CS 234, Spring 2024 A. Jamshidpey

ASSIGNMENT 2

Due date: June 19, 2024 16:00 (Waterloo time)

WARNING: To receive credit for this assignment, you checked "I agree" to the academic integrity declaration. Please keep all the rules in mind as you complete your work.

Coverage: Through Module 4

This assignment consists of a written component and a programming component. Please read the instructions on the reference page for assignments carefully to ensure that you submit each component correctly.

Please check the pinned FAQ in the discussion forum for corrections and clarifications.

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).

Read each question carefully to make sure you know whether you are playing the role of the user or the role of the provider. You do not need to state the role explicitly, but you should be aware that your answers will be used to assess your understanding of and adherence to the appropriate role.

W1. *[10 marks]* We will say that a data item *Data* in the ADT Indexed Sequence I is *unique* if there is exactly one copy of *Data* in I.

(a) *[5 marks]* Using the pseudocode interface for the ADT Indexed Sequence (found on the page **Reference material > Indices > ADTs**), write a pseudocode function IS_UNIQUE that consumes an ADT Indexed Sequence I and a data item *Data* and produces *True* if *Data* is unique and *False* otherwise. You may not mutate I and you may not make a copy of I to mutate. Remember that in this question, you are the user of the ADT Indexed Sequence.

(b) *[3 marks]* Assuming that the cost of each of the first three ADT Indexed Sequence operations is in $\Theta(1)$ and the cost of each of the last three operations is in $\Theta(i)$ (where i is the value of *Index*), give the worst-case running time of UNIQUE in Θ notation as a function of c, the capacity of I. Briefly justify your answer. You may find it convenient to add line numbers to your pseudocode. Make sure to refer to each line in your analysis.

(c) *[1 mark]* Would your answer to part (b) change if you were instead considering **best-case running time**? Briefly justify your answer.

(d) *[1 mark]* Would your answer to part (b) change if you were instead considering the worst-case running time as a function of both c (the capacity of I) and n (the number of data items stored in I)? Briefly justify your answer.

W2. *[9 marks]* We now consider augmenting the ADT Indexed Sequence to include an operation that determines if a data item is unique. In this question, we introduce a new ADT, the ADT IndexPlus, which has all the same operations as the ADT Indexed Sequence as well as the operation UNIQUE. You can find the pseudocode interface of the operations of the ADT Indexed Sequence on the page **Reference material** > **Indices** > **ADTs**, and the new operation in Table 1.

All running times should be given in Θ notation as functions of the specified variables, where c is the capacity of I, n is the total number of data items stored in I, and k is the number of copies of *Data* in I.

Name	Preconditions	Produces	Effects
UNIQUE*(I, Data)*	I is an Indexplus, *Data* is a data item	*True* if there is exactly one copy of *Data* in I, else *False*	

Table 1: Pseudocode interface for ADT IndexPlus operation UNIQUE

(a) *[3 marks]* Briefly describe how to implement UNIQUE using two arrays *Indices* and *Values* of the same size, a variable *Cap* storing the size of the arrays, and a variable *Last* containing the index of the last full entry in both arrays or -1 if the arrays are empty. Use a few sentences

to present your algorithm; pseudocode is not required. Your algorithm should be designed to use as little time as possible in the worst case and in the best case.

(b) *[2 marks]* State and briefly justify the worst-case running time of your implementation from part (a) as a function of c, k, and n.

(c) *[2 marks]* State and briefly justify the **best-case** running time of your implementation from part (a) as a function of n.

(d) *[2 marks]* Briefly describe how to implement UNIQUE using a doubly-linked list in which the linked node in position p is at index p. Use a few sentences to present your algorithm; pseudocode is not required.

W3. *[6 marks]* Each subquestion specifies ADTs A and B and operation X. Follow the instructions below for each subquestion:

- Provide a description of the implementation of ADT A using ADT B, including any extra variables that are used and the meanings of their values. Do not add any additional ADTs to the implementation.
- For the implementation you provided, give an algorithm for operation X of the ADT A. You do not need to provide pseudocode, but you should refer to specific operations of ADT B that are needed, including specific inputs to the operations.

You can find the details of the ADT operations on the page **Reference material > Indices > ADTs**.

(a) *[2 marks]* A: Queue; B: Ranking; X: DEQUEUE*(Q)*

(b) *[2 marks]* A: Pair ; B: Grid ; X: ADD_SECOND*(P, Data)*

(c) *[2 marks]* A: Bucket; B: Stack; X: ADD_TO_BUCKET*(B, Data)*

Programming component

Please read the information on assignments and Python 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.

Both programming questions are implementations of the ADT Collection. Use the supplied interface files `collectioncontiguouspairinterface.py` and `collectiondoublylinkedinterface.py` as starting points for the question. You can find the files linked off of **Assessments** > **Assignments** > **Assignment 2**. Each provides a `repr` method and enough of the class definition to ensure that the `repr` method uses the correct field names.

You may need to add additional documentation for methods and the class definition for each implementation. Do not change the headers of the methods, and do not change the `repr` methods, as it will be used in the testing of your code.

You may wish to create additional methods for your convenience, such as helper functions used in the implementations of multiple ADT operations.

Note: You must use the specified data structures to receive marks on the programming questions. Each of your operation implementations should make direct use of the data structure instead of using other ADT operations. 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 the lecture content.

You may import any of the following files: `contiguous.py`, `linked.py`, and `pair.py`. You can find all three files linked off of the page **Reference material** > **Python** > **Modules**.

Be sure to read the instructions on the Python requirements page to ensure that you have imported the files as required.

P1. *[12 marks]* In this question, implement the ADT Collection using a contiguous implementation.

- Each slot in the implementation should store an ADT Pair, where the first value in a pair is a data item in the collection and the second value is the (positive) number of copies of the data item stored in the collection.

- In the array, all full slots must come before any empty slot.
- There is no particular order in which the pairs are stored.

You should use the module `contiguous.py` to create an array and `pair.py` to create an ADT Pair. Make sure to access the array using only the methods provided in `contiguous.py`: you are the provider of the ADT Collection, but the user of `contiguous.py`.

For this question, you can assume that the size of the array will be no greater than 100.

Here are the details of the methods:

- Your `add` method should place a newly-appearing data item in the first empty slot, and increment the number of copies of an already-appearing data item.
- Your `delete_any` method should remove a data item from the last full slot (resulting either in a smaller number of copies or an emptied slot, depending on the number of copies stored at the time of the operation.)

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

P2. *[13 marks]* In this question, implement the ADT Contiguous using a doubly-linked list implementation.

- Each linked node will contain a single copy of a data item.
- There is no particular order in which the data items are stored.

You should use the class `Double` in the module `linked.py` to create the linked nodes in your doubly-linked list. Make sure to access the linked nodes using only the methods provided in `linked.py`: you are the provider of the ADT Collection, but the user of `linked.py`.

Here are the details of the methods:

- Your `add` method should add a new data item at the front of the list.
- Your `delete_any` method should return the data item at the front of the list.

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