CS211 - Data Structures and Algorithms

Lab 3

Instructor: Dr. Sharaf Hussain
E-mail: shhussain@uit.edu

Semester: Fall, 2021

1 Objective

The purpose of this lab session is to practice Sets, Maps, and Multi-Dimensional Array that we discussed in class week 3.

2 Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

- Submit lab work in a single .py file on MS Teams. (No other format will be accepted)
- Lab work file name should be saved with your roll number (e.g. 19b-001-SE_LW.py)
- Submit home work in a single .py file on MS Teams. (No other format will be accepted)
- Lab work file name should be saved with your roll number (e.g. 19b-001-SE_HW.py)

3 Exercises

Task 1

Implement the Set ADT in python as given below,
A set is a container that stores a collection of unique values over a given comparable domain in which the stored values have no particular ordering.

- `Set()`: Creates a new set initialized to the empty set.
- `length()`: Returns the number of elements in the set, also known as the cardinality. Accessed using the `len()` function.
- `contains(element)`: Determines if the given value is an element of the set and returns the appropriate boolean value. Accessed using the in operator.
- `add(element)`: Modifies the set by adding the given value or element to the set if the element is not already a member. If the element is not unique, no action is taken and the operation is skipped.
- `remove(element)`: Removes the given value from the set if the value is contained in the set and raises an exception otherwise.
- `equals(setB)`: Determines if the set is equal to another set and returns a boolean value. For two sets, A and B, to be equal, both A and B must contain the same number of elements and all elements in A must also be elements in B. If both sets are empty, the sets are equal. Access with == or ! =.

- `isSubsetOf(setB)`: Determines if the set is a subset of another set and returns a boolean value. For set A to be a subset of B, all elements in A must also be elements in B.
- `union(setB)`: Creates and returns a new set that is the union of this set A and set B. The new set created from the union of two sets, A and B, contains all elements in A plus those elements in B that are not in A. Neither set A nor set B is modified by this operation.
- `intersect(setB)`: Creates and returns a new set that is the intersection of this set A and set B. The intersection of sets A and B contains only those elements that are in both A and B. Neither set A nor set B is modified by this operation.
- `difference(setB)`: Creates and returns a new set that is the difference of this set A and set B. The set difference, $A - B$, contains only those elements that are in A but not in B. Neither set A nor set B is modified by this operation.
- `iterator()`: Creates and returns an iterator that can be used to iterate over the collection of items.

Additional Tasks

a Modify the `Set()` constructor to accept an optional variable argument to which a collection of initial values can be passed to initialize the set. The prototype for the new constructor should look as follows:

```
def Set( self, *initElements = None )
```

It can then be used as shown here to create a set initialized with the given values:

```
s = Set( 150, 75, 23, 86, 49 )
```

b Add a new operation to the Set ADT to test for a proper subset. Given two sets, A and B,A is a proper subset of B, if A is a subset of B and A does not equal B.

Task 2

2. Implement the following Map ADT in python,
A map is a container for storing a collection of data records in which each record is associated with a unique key. The key components must be comparable.

- `Map()`: Creates a new empty map.
- `length ()`: Returns the number of key/value pairs in the map.
- `contains ( key )`: Determines if the given key is in the map and returns True if the key is found and False otherwise.
- `add(key,value)`: Adds a new key/value pair to the map if the key is not already in the map or replaces the data associated with the key if the key is in the map. Returns True if this is a new key and False if the data associated with the existing key is replaced.
- `remove(key)`: Removes the key/value pair for the given key if it is in the map and raises an exception otherwise.
- `valueOf(key)`: Returns the data record associated with the given key. The key must exist in the map or an exception is raised.
- `iterator ()`: Creates and returns an iterator that can be used to iterate over the keys in the map.

Additional Tasks

a Add a new operation `keyArray()` to the Map class that returns an array containing all of the keys stored in the map. The array of keys should be in no particular ordering.

b Design and implement the iterator class `_SetIterator` for use with the Set ADT implemented using a list.

c Design and implement the iterator class `_MapIterator` for use with the Map ADT implemented using a list.

Task 3

Implement the following MultiArray ADT in python,
A multi-dimensional array consists of a collection of elements organized into multiple dimensions. Individual elements are referenced by specifying an n-tuple or a subscript of multiple components, $(i_1, i_2, i_3, \ldots, i_n)$, one for each dimension of the array. All indices of the n-tuple start at zero.

- `MultiArray`$(d_1, d_2, d_3, \ldots, d_n)$: Creates a multi-dimensional array of elements or ganized into n-dimensions with each element initially set to None. The number of dimensions, which is specified by the number of arguments, must be greater than 1. The individual arguments, all of which must be greater than zero, indicate the lengths of the corresponding array dimensions. The dimensions are specified from highest to lowest, where d_1 is the highest possible dimension and d_n is the lowest.

- `dims()`: Returns the number of dimensions in the multi-dimensional array. `length(dim)`: Returns the length of the given array dimension. The individual dimensions are numbered starting from 1, where 1 represents the first, or highest, dimension possible in the array. Thus, in an array with three dimensions, 1 indicates the number of tables in the box, 2 is the number of rows, and 3 is the number of columns.

- `clear(value)`: Clears the array by setting each element to the given value.

- `getitem` $(i_1, i_2, i_3, \ldots, i_n)$: Returns the value stored in the array at the element position indicated by the n-tuple $(i_1, i_2, i_3, \ldots, i_n)$: . All of the specified indices must be given and they must be within the valid range of the corresponding array dimensions. Accessed using the element operator: $y = x[1, 2]$.

- `setitem`$(i_1, i_2, i_3, \ldots, i_n, value)$: Modifies the contents of the specified array element to contain the given value. The element is specified by the n-tuple $(i_1, i_2, i_3, \ldots, i_n)$. All of the subscript components must be given and they must be within the valid range of the corresponding array dimensions. Accessed using the element operator: $x[1, 2] = y$.

  a Develop the index equation that computes the location within a 1-D array for element (i, j) of a 2-D array stored in column-major order.

  b Complete the implementation of the `MultiArray` class by implementing the helper method `_computeFactors()`.

Home Tasks

1. Write a function that extracts the sales data from a text file and builds the 3-D array used to produce the various reports in Section 3.4. Assume the data file has the format as described in the chapter.

2. Write a menu-driven program that uses your function from the previous question to extract the sales data and can produce any of the following reports:

   a Each of the four types of reports described in the chapter.

   b The sales for a single store similar to that shown in Section 3.4 with the data sorted by total sales.

   c The total sales for each store sorted by total sales from largest to smallest.

   d The total sales for each item sorted by item number.