Satellite Data Processing

In this assignment, we have to write a utility for people at Pakistan Space & Upper Atmosphere Research Commission (SUPARCO).

They have a ton of data received from different Pakistan's satellites moving in orbit. Each satellite sends a large number of observations about various sensors periodically. A received at the SUPARCO stations, located not much far from our campus, a utility known as MerleDixon (or shortly as Dixon), receives these observations and stores them in a flat-file. Due to the large data volume, all data stored in a single file.

Unfortunately, due to a bug in the receiving utility, all irrelevant data is also stored. Now, they are looking for a utility to achieve two goals:

1. to separate data for each satellite,

2: to store only relevant data.

The team has a large number of files for which utility should work. Due to this, the utility should be simple and optimized.

Presently, Dixon is storing the data in the following format:

S A D

Each row in the file contains three values separated by a space: S, A, and D. S is the satellite number for which data received, and A identifies the action. There are two kinds of actions that can be performed on the data: I: Insert and D: delete. D is the actual data that should be stored or deleted for a specific satellite.

For example, the following data shows that we have two satellites: 1 and 2. In the first line, we have to insert/store 5 for satellite 1, and in the second line, we have to insert/store 4 for satellite 2. The last line is asking to delete data for satellite 1. For delete, the value can be 0 or 1: 0 to remove the oldest value received for the satellite and 1 to remove the latest value received for the satellite. In this example, we have to delete the oldest value received for satellite 1. In the next line, we have to delete the latest value received for satellite 2.

1 I 5 2 I 4 1 I 10 1 D 0 2 D 1

The architecture of the utility decided to use List ADT to store this data. The utility will create a separate instance of List for each satellite. The SUPARCO has five satellites, but we don't have prior information about the number of observations store in the file for each satellite. Therefore, the utility should create a new instance whenever new satellite data appears in the file.

Implementation

Storing Satellite Data:

The basic interface of the List ADT implemented as SatelliteData class with its operations is given below. One of the constraints is that all operations, except Sort(), must be implemented in O(1) [Amortized]. The Sort() operation must be implemented in O(n log(n)). The details of each function are provided as comments in the respective functions.

Processing Data:

The implementation to process each file need to be implemented as SatelliteUtility class. The details of each function are provided as comments in the respective functions.

You are free to add further function as per requirements. Don't add test code in your submission.

Your Task:

You have to implement the functionality of both classes: SatelliteData and SatelliteUtility. You can add further classes/functions if required. Your implementation must be

```
class SatelliteData:
    # stores an element at the front of queue
    # must be implemented in O(1)
def InsertAtFront(self, e):
        pass

    # removes an element from the front of queue
    # must be implemented in O(1)
def RemoveFront(self):
        pass

    # stores an element at the end of the
queue     # must be implemented in O(1)
def InsertAtEnd(self, e):
        pass

    # removes an element from the end of the queue
    # must be implemented in O(1)
def RemoveAtEnd(self):
        pass


    # returns an element from the queue at i_th position
    # must be implemented in O(1)
def Get(self, i):
        pass

    # sort the data
    # # must be implemented in O(n log(n) )
def Sort(self):
        pass

    # returns the number of elements in the queue
```

```
    # must be implemented in O(1)
def Count(self):
        pass

    # returns the number of elements can be store by the queue
    # must be implemented in O(1)
def Capacity(self):
        pass

    # returns the entire data as a list
def GetAll(self):
        pass
 class
SatelliteUtility:
    def __init__(self, filename):
        # store instance for each satellite
self.satellites = []
        # store all data from file to this list
self.allLines = []

    # instance must be initilized with a filename
    # filename represents the file containing data
    # to be processed
def ReadData(self):
        # read data from file and
pass

    # this function should read lines from
    # text file and store in self.allLines list
    # each entry in the list represent one line
```

```
    def ProcessData(self):
self.ReadData()          for aLine in
self.allLines:              # parse the line to
get data for S A D             # store data for
respective satellite             pass
     def
GetSummary(self):
        # returns a list of tuples where each tuple shows
        # a satellite with number of data for each entry
        # for example: [(1,100),(2,200)] shows 2 satellites
# 1 and 2 with 100 and 200 data elements, respectively
pass
```