

Programming Assignment: Implement RadixSort using queues¹

Problem Statement:

In this assignment you will implement a new sorting algorithm called Radix Sort for a collection of strings. Radix sort algorithm can be used on integers or strings where the elements can be expressed with positional notation: Digits in numbers, characters of strings. We will sort the items in the lexicographic order (as they will appear in a standard dictionary). A sequence such as "b, e, c, f, d, ba" would be lexicographically sorted as "b, ba, c, d, e, f".

The strategy followed here sorts the strings according to the last character first, then the second last character, then the third last and so on using auxiliary queues.

Specifically you will write a function called `radixSortStrings` that takes a list of strings, sorts them alphabetically and returns the sorted list. You will be using multiple queues, and an ordered Dictionary to implement this sorting algorithm. For this assignment, we will assume that all the strings contain only alphabets.

Here's an overview of the steps of the algorithm the function needs to follow (see the worked out sample below showing the queues after various steps of the algorithm):

1. Strings from the input list are placed in a queue called 'mainQueue' and the length of the largest string is calculated, say `maxLength`.
2. An `orderedDictionary` of 27 auxiliary empty queues is created, called `auxQueueDictionary` with keys being the space character (' ') and the 26 lower-case alphabets. The keys are added in a specific order starting with the space character, followed by the 26 characters in alphabetical order. This will ensure that when we iterate over the ordered dictionary, the keys are retrieved in this very order.
3. Next a for-loop is set up to perform the following steps that will sort the strings according to the character at the given index (with index starting from `maxLength-1` down to 0 in steps of 1):
 - a. Strings from the `mainQueue` are dequeued one by one, and enqueued onto the auxiliary queue corresponding to the character of the string at current index. If any string is shorter than the current index, simply getting the character at the current index by using `string[index]` syntax will result in `IndexError` exception. The algorithm uses the helper function `charAt` provided to get the character at current index. This function simply returns ' ' when the requested index is > length of the string. Using the `charAt` function, all shorter strings are enqueued to

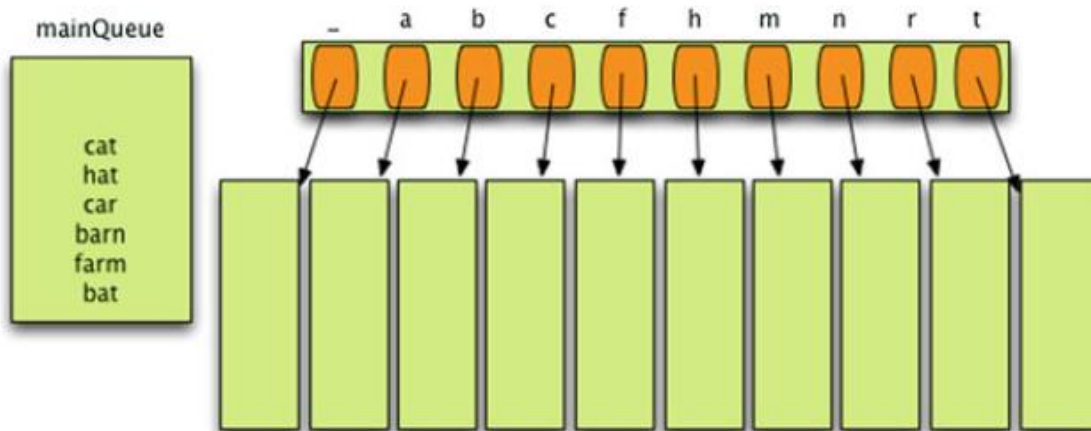
¹ Adapted from Chapter 4 Section 4.11.3 from Data Structures and Algorithms with Python (Undergraduate Topics in Computer Science) By Kent D. Lee, Steve Hubbard

the aux queue for ‘ ‘, strings having ‘a’ at current index are enqueued on the ‘a’ queue and all strings having ‘b’ at current index are enqueued on the ‘b’ queue and so on.

- b. Once all the strings from the mainQueue are dequeued, the auxQueueDictionary is iterated over and all strings are dequeued from each auxiliary queue in sequence and placed on the main queue. Each queue is emptied first before proceeding to the next queue in the auxQueueDictionary. This is repeated until all aux queues are empty.
4. After the maxLength iterations of the for-loop, mainQueue will have strings sorted alphabetically. A new list is created by dequeuing the strings from the mainQueue and returned.

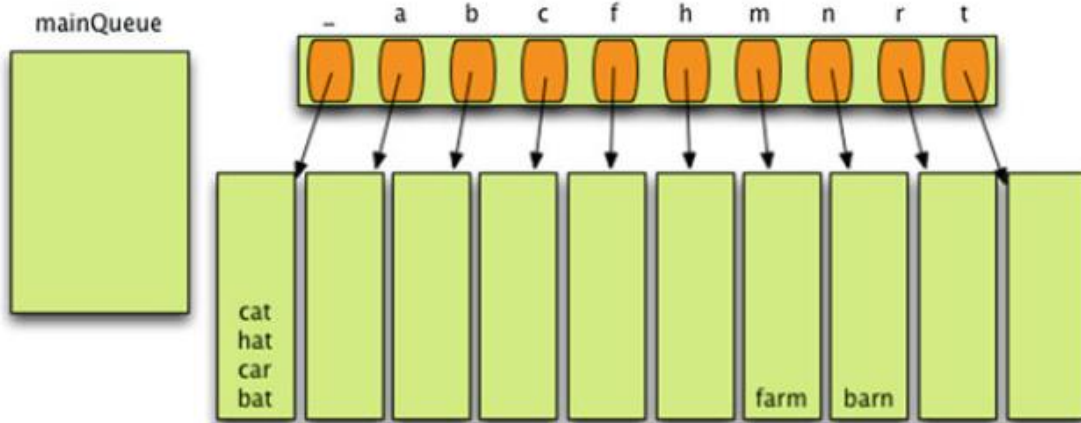
Let’s look at a worked out sample of input list of strings [‘bat’, ‘farm’, ‘barn’, ‘car’, ‘hat’, ‘cat’]. Here’s how the queues will look like at various steps of the algorithm on this input list.

After steps 1 and 2 of the algorithm, input strings will be queued onto the main queue and the ordered dictionary is set up with 27 empty aux queues corresponding to the ‘ ‘ character and the 26 alphabets (In all of the picture, assume the head of the queues are at the bottom and tail is at the top):

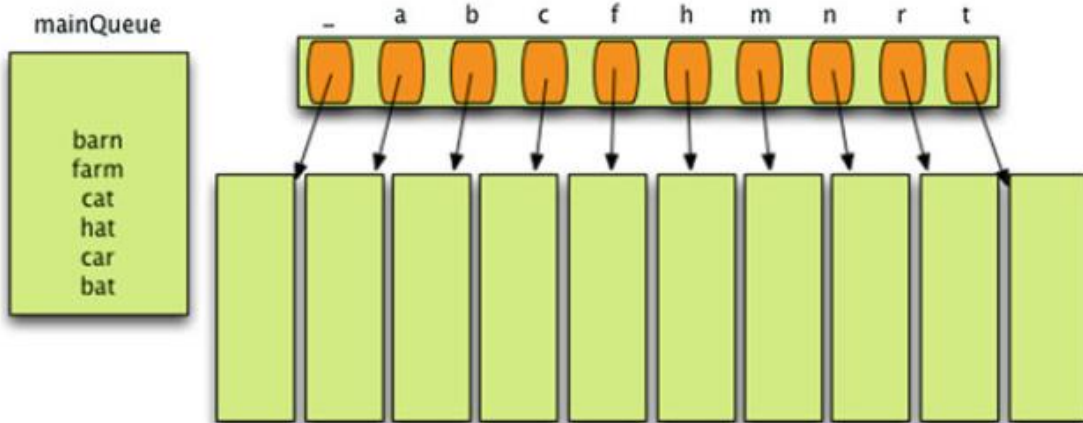


Since the largest string from this sample input list is 4 long, maxLength will be calculated as 4. Next, the steps 3a and 3b will be repeated for index = maxLength - 1 = 3, then index = maxLength - 2 = 2, then 1 and then 0.

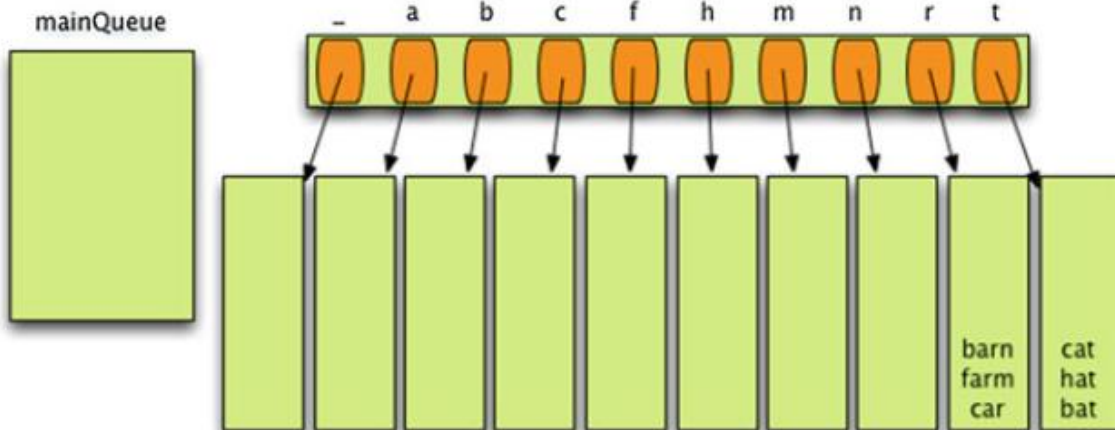
At step 3a for index = 3, all the strings from the mainQueue are dequeued and placed in auxQueues corresponding to the character returned by charAt(string, 3). So you can see that all the shorter strings are placed on to the queue corresponding to ‘ ‘ since that’s what charAt will return for the shorter strings. The string ‘farm’ is placed onto the queue for ‘m’ and string ‘barn’ is placed onto the queue for ‘n’:



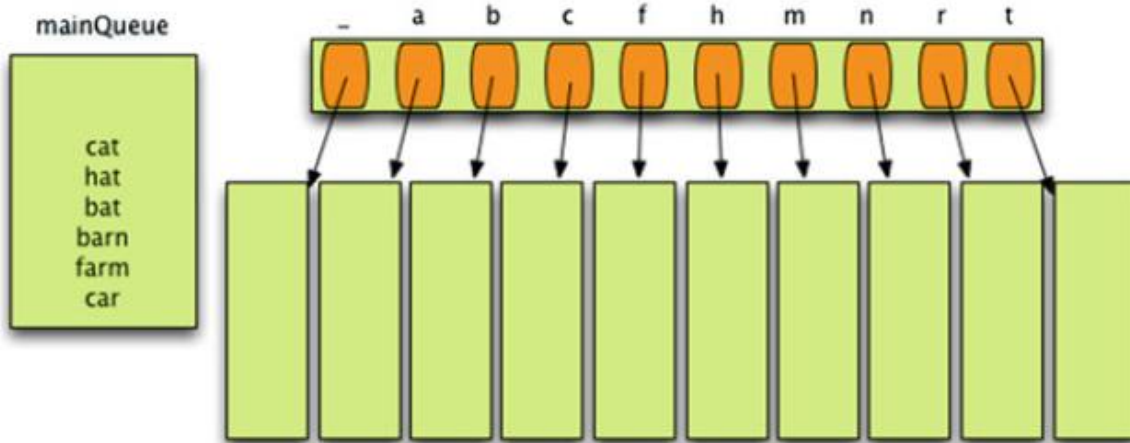
Next, step 3b for index = 3 will dequeue the items from all the aux queues in order and enqueue them on the main queue:



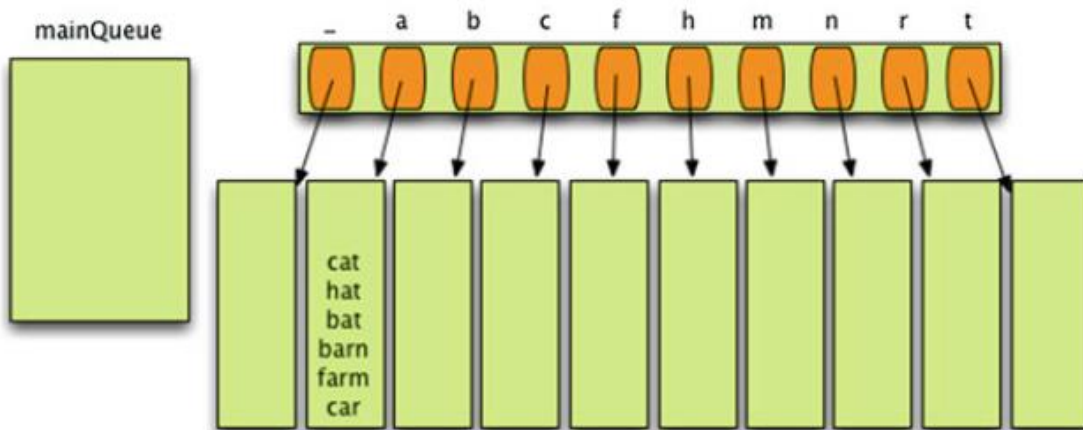
Next in step 3a and 3b will be repeated for index = 2: all the strings will be put on the queues corresponding to the character at index 2. So strings with 'r' at index 2 are enqueued onto the aux queue for 'r' and all the strings with 't' at index 2 are enqueued onto the aux queue for 't', maintaining the order from the mainQueue within each aux queue.



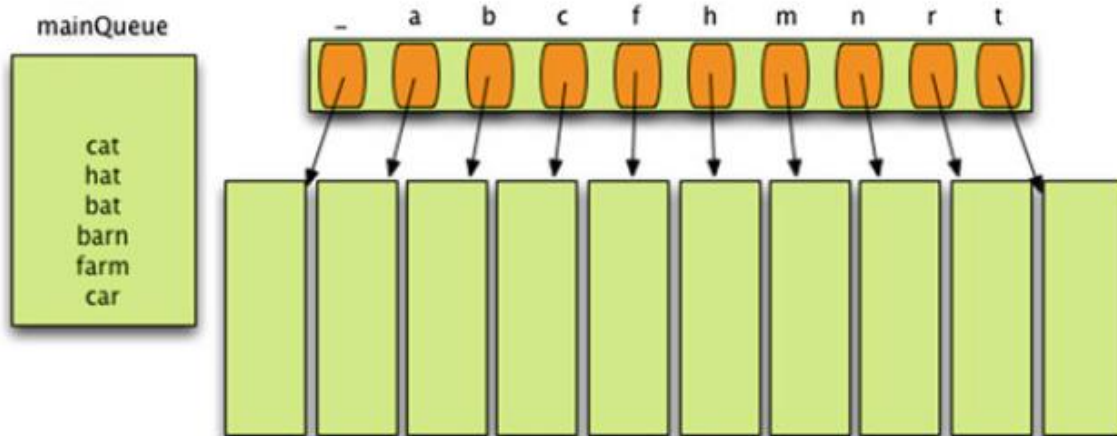
After step 3b for index = 2 the queues will look like:



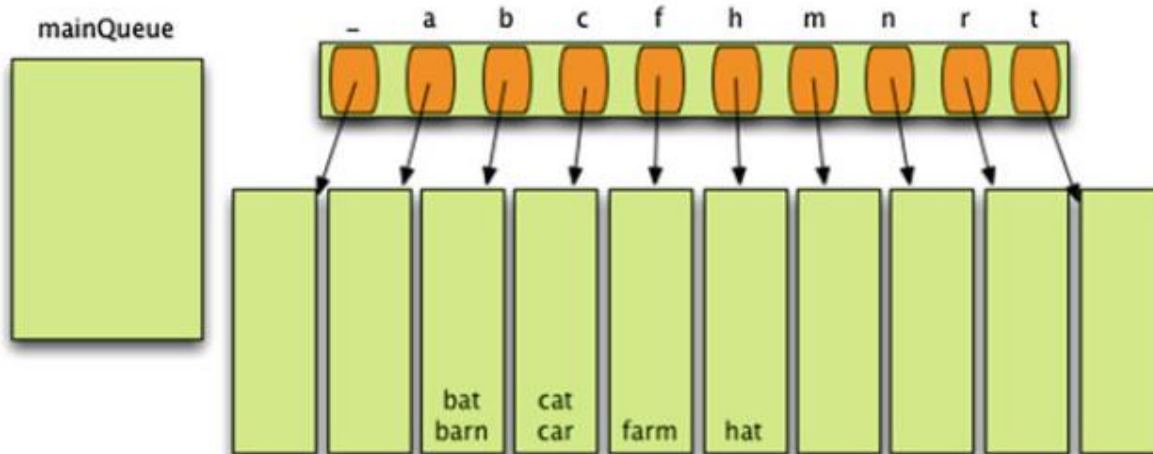
After step 3a for index = 1, all strings get placed in the aux queue for 'a' since all the strings have 'a' at index 1.



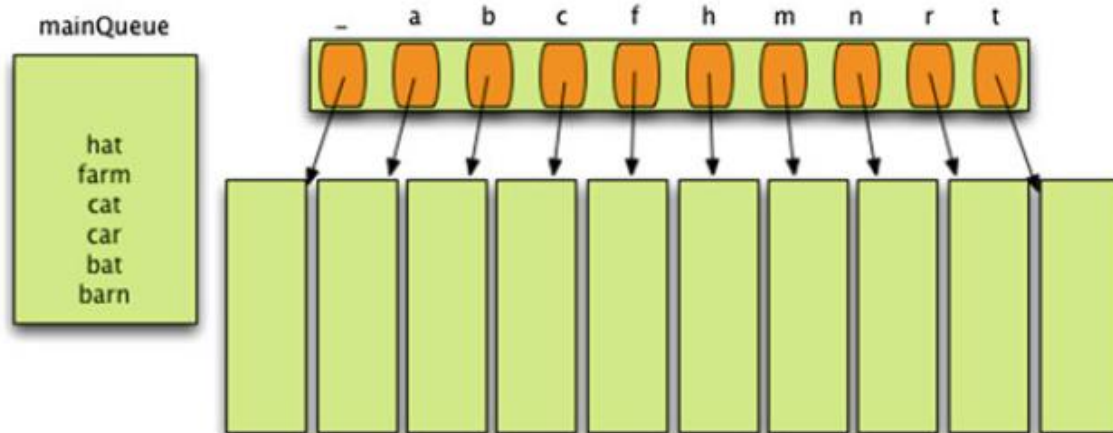
After step 3b for index = 1:



After step 3a for index = 0, strings are placed in queue corresponding to the first character:




After step 3b for index = 0.



Finally step 4 will dequeue items from the mainQueue to form the final sorted list: ['barn', 'bat', 'car', 'cat', 'farm', 'hat'].

Download the startup files **myqueue.py** and **orderedDictionary.py**. These files have implementation of the Queue and OrderedDictionary classes to be imported and used. Download **radixSortString.py** and implement the helper functions: **setupAuxQueuesDictionary** and **updateQueue** and implement the **radixSortStrings** function. The main function has some test code. Here's how the test code run will look like:



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
gs-final.py
Sorting lists of strings using radix sort using queues

Before sorting:
['cat', 'hat', 'CAR', 'barn', 'farm', 'bat']
After sorting:
['barn', 'bat', 'car', 'cat', 'farm', 'hat']

Before sorting:
['cat', 'hat', 'Cat', 'car', 'BARN', 'bat', 'one', 'ZOO', 'cow']
After sorting:
['barn', 'bat', 'car', 'cat', 'cat', 'cow', 'hat', 'one', 'zoo']

Before sorting:
['Initialize ', 'a', 'Dictionary', 'that holds', 'the', 'queues', 'corresponding
to', 'the', 'individual', 'letters']
After sorting:
['a', 'corresponding to', 'dictionary', 'individual', 'initialize', 'letters', '
queues', 'that holds', 'the', 'the']

Before sorting:
['Iron Man', 'Captain America', 'Black Widow', 'Hulk', 'Thor', 'Clint Barton', '
Loki']
After sorting:
['black widow', 'captain america', 'clint barton', 'hulk', 'iron man', 'loki', '
thor']

>>>
Ln: 63 Col: 33
```

Complete the implementation of the **radixSortStrings** function and the helper functions. Submit the file as **first_last_radixSortStrings.py**. Be sure to add the top comment block giving details such as your name, date, assignment name etc.

Note:

1. Make sure the sorting is case-insensitive.
2. Make sure you add the keys to the ordered dictionary in the right sequence.
3. Use the **charAt** function to make sure you get a character at every index value no matter how long a string is and that you don't end up with an **IndexError** exception.