Computational Problem Solving CSCI-603
LinkedHashTable Lab 7

1 Implementation

For this lab, you will implement a linked hash set that remembers the order of insertion. The documentation for the classes you will write, and the source code for the `Set` class, are provided in the following link:

`https://www.cs.rit.edu/~csci603/Labs/07-LinkedHashSet/code.zip`

Your data structure design is required to look like the diagram in Figure 1. That is, you build a chained hash table with keys only, no values, and the entry nodes have two additional references: *previous* and *link* (next) node links. This means that the ordering is done using the technique of a doubly linked list that overlays on top of the existing nodes needed for the bucket chains.

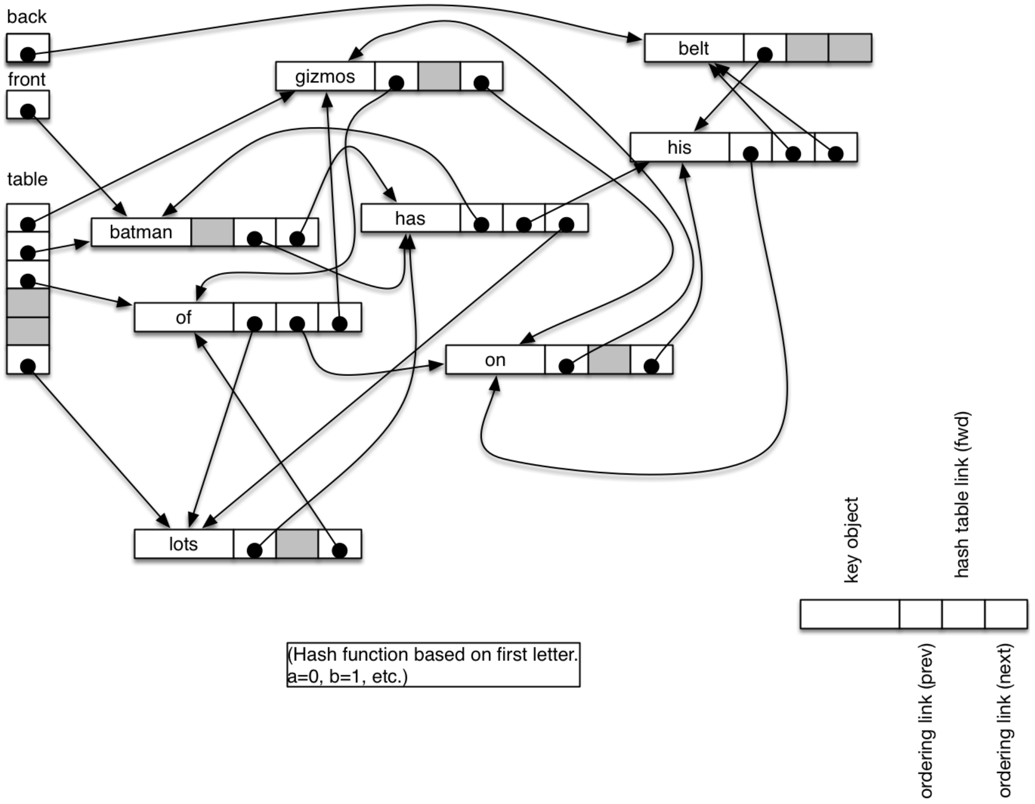

Figure 1: The linked hash table design. Structure shown is developed from entering the following words to the set: `batman has lots of gizmos on his belt`

Note that you have to write an iterator for your set (instead of the `keys()` method you saw in lecture for the dictionaries). The file **sample_iter.py** contains examples to show you how to do it. There are many online resources as well.

1.1 Design Constraints

Methods running time:

1. The `add`, `contains`, `remove` and methods should all run in $O(1)$ time (assuming not much clustering due to excessive collisions). This means no linear searches besides the small chains of entries at specific "bucket" locations in the hash table.
2. The `len` method must run in $O(1)$.
3. The iterator, `str`, and `repr` methods are of course linear.

You are not allowed to use dictionaries or any built-in data structure from the Python library except for the list to implement the basic hash table. The data elements must be stored using linked nodes.

Basically you need to follow the design exemplified in **Figure 2**.

Here is the suggested order of implementation of all the `LinkedHashSet` methods:

- `__init__` and `__len__` methods
- `__str__` and `__repr__` methods
- `__iter__` method
- `contains` method
- `add` method
- `remove` method

1.2 Testing

You must construct at least 3 additional non-trivial test cases and add them to the provided **tests.py** file. Each test must be distinct from the others in terms of what it demonstrates.

There is the beginning of a test program file available for you to use for adding test functions.

2 Grading

- 20% Problem Solving
- 20% Design
- 40% Functionality:
  - 6% contains
  - 10% add
  - 10% remove
  - 6% iterator
  - 8% constructor and others
- 15% Testing: (5% each)
- 5% Style: Proper commenting of modules, classes and methods (e.g. using docstring's).

3 Submission

Create a ZIP file named Lab7.zip that contains all your source code **(linkedhashset.py, tests.py)**. Submit the ZIP file to the MyCourses assignment before the due date (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this lab).