CS 209
Data Structures and Mathematical Foundations

04 / 01 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions?/Comments?
- Finish implementation of Stack
- Radix Sort
- Implement a Queue
- Implement Radix Sort

Queues and Stacks

- A queue is a data structure that has the following characteristics
  - It is linear
  - Uses FIFO (first in, first out) processing
- Queue operations
  - Enqueue – add an item to the *rear* of the queue
  - Dequeue – remove an item from the *front* of the queue
  - Empty – returns true if the queue is empty
- What's significant about a queue is that there are no insert in anywhere or remove from anywhere in the queue. The only place to add something to the queue is the rear, and the only place to remove something from the queue is the front.

Queues and Stacks

- A stack is a data structure that has the following characteristics
  - It is linear
  - Uses LIFO (last in, first out) processing
- Stack operations
  - Push – add an item to the *top* of the stack
  - Pop – remove an item from the *top* of the stack
  - Empty – returns true if the stack is empty
  - Peek – retrieve information about the item on *top* of the stack without removing it
- The only allowable ways to put an item into and to get an item from the stack is via push and pop. There are no insert in anywhere or remove from anywhere in the stack.
- What if there was no peek? Is it redundant --- could a series of the existing operations achieve the same functionality?

Queues and Stacks

- Let's implement a stack with a python list
  - what will be our instance variables?

Queues and Stacks

- Let's use that Queue class in an interesting sorting method called Radix Sort.

Queues and Stacks

- Radix Sort.
  - consider the job of sorting (base 10) integers
    - let's limit them to 0-99 for now (all >=0 and <= 100)
  - we handled this in several ways already
  - a totally different way is the following way
    - start with an unsorted list
    - separate the numbers to be sorted into 10 different bins based on their 1's digit
    - then, get the numbers out of the bins and make a new list (take numbers from bin 0, then add on to the end of the list the numbers in bin 1, ... and so on, finally adding to the end of the list the numbers from bin 9)
    - then separate this list into 10 bins based on the 10's digit.
    - get the numbers out of the bins like before to get a sorted list.

Queues and Stacks

- Radix Sort.
  - example on board with the following:
    - { 12, 15, 88, 76, 63, 21, 22, 1, 52, 2, 23, 5 }

Queues and Stacks

- Radix Sort.
  - example on board with the following:
    - { 12, 15, 88, 76, 63, 21, 22, 1, 52, 2, 23, 5 }
  - each of the 10 bins were queues
  - also if you want to sort numbers that have a maximum of n digits in them, then you need n passes through the sort and each pass is a higher place in the number
  - we could use an list of 10 queues
    - each index to the list corresponds to the digit in whatever place we're working on

Queues and Stacks

- Let's create a Queue class with a linked list
- Then let's implement Radix Sort using that Queue class