Data Structures and Algorithms in Python

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 6 Stacks, Queues, and Deques

Hints

Reinforcement

R-6.1) Use a paper and pencil with eraser to simulate the stack.

R-6.2) If a stack is empty when pop is called, its size does not change.

R-6.3) Transfer items one at a time.

R-6.4) First check if the stack is already empty.

R-6.5) Use two loops.

R-6.6) Give a recursive definition.

R-6.7) Use a paper and pencil with eraser to simulate the queue.

R-6.8) If a queue is empty when dequeue is called, its size does not change.

R-6.9) Each successful dequeue operation causes that index to shift circularly to the right.

R-6.10) Consider how the queue might be configured within the underlying array.

R-6.11) Read the documentation of collections.deque, if needed.

R-6.12) Use a paper and pencil to simulate the deque.

R-6.13) You may use the return value of a removal method as a parameter to an insertion method.

R-6.14) You may use the return value of a removal method as a parameter to an insertion method. Think about how to effectively use the stack for temporary storage.

Creativity

C-6.15) Pop the top integer, but remember it.

C-6.16) Use a new instance variable to store the capacity limit.

C-6.17) Use an expression such as [**None**] $*$ k to build a list of k **None** values.

C-6.18) You will need to do three transfers.

C-6.19) After finding what's between the $<$ and $>$ characters, the tag is only the part before the first space (if any).

C-6.20) Use a stack to reduce the problem to that of enumerating all permutations of the numbers $\{1,2,\ldots,n-1\}$.

C-6.21) Use the stack to store the elements yet to be used to generate subsets and use the queue to store the subsets generated so far.

C-6.22) Use a stack.

C-6.23) You can still use R as temporary storage, as long as you never pop its original contents.

C-6.24) Rotate elements within the queue.

C-6.25) Consider using one stack to collect incoming elements, and another as a buffer for elements to be delivered.

C-6.26) Think of using one stack for each end of the deque.

C-6.27) Think of how you might use Q to process the elements of S twice.

C-6.28) Use a new instance variable to store the capacity limit.

C-6.29) You might start by combining the code of a dequeue followed by an enqueue, and then simplify.

C-6.30) Think of the queues like boxes and the integers like red and blue marbles.

C-6.31) Lazy and Crazy should only go across once.

Projects

P-6.32) Suggested instance variables are described in the book.

P-6.33) What is the index of a new element? What is the index of the old element that is lost?

P-6.34) You will need to use a stack.

P-6.35) How does this functionality compare to a deque?

P-6.36) Keep information about the purchase shares and prices in a queue, and then match those against sales. Care must be taken if only part of a purchase block is sold.

P-6.37) Start one stack at each end of the array, growing toward the center.