Positional Collection (Chapter 9, pages 107-119)

- position 0 1 ... n-1

array-based implementation

+ constant time access to position p
- must shift elements
- to add or remove near middle (linear time)

list-based implementation

- must traverse list to reach position p
- linear time to get to middle
+ constant time to add or remove once element located
array-based

- Some space (and time) overhead to maintain a tracker

+ Space usage can be as small as roughly \( n \) references when \( n \) elements in collection

- Must select a size when allocating (you can resize)

list-based

+ Negligible overhead to track an element

- Even in a singly-linked list use 3 refs per node (element, next, type)

  Roughly 3n references to hold same \( n \) elements

+ Naturally can dynamically resize
Positional Collection Data Structures

- Array position p stored in a[p]

- Dynamic Array
  (This is what a Java arraylist is!)

Initial size

Insertion at ystrip[4]
triggers resizing

Changes when doubling array size
Circular Array
Queue with at most 4 elements

Offset of 1, let variable \( \text{start} = 1 \) to mark index for position 0

decqueue (to remove a)

Translation between index + position computed by:

\[
\text{index} = \text{position} + \text{start} \mod \text{array size}
\]

\[
\text{position} = \text{index} - \text{start} \mod \text{array size}
\]
Each dequeue implicitly changes positions of remaining elements by incrementing start

offset into underlying array for position 0

start = 8

size = 7

Queue: f, g, h, i, j, k, l
Array
Dynamic Array
Circular Array
Dynamic Circular Array

Tracked Array

application program must maintain a reference to the tracker returned by addTracked

Singly Linked List
Doubly Linked List
For time:
\[ O(1) \approx O(\min(p, n-p)) \approx O(n) \]

Table 9.1 in *A Practical Guide to Data Structures and Algorithms in Java*
Quicksort

First let's review two things from last class.

![Diagram of quicksort](image)

Figure 12.1
A circular array representing the positional collection \((w, x, y, \text{null}, z)\) where \(\text{start}=5\).

*implementation in text guarantees that array slots not in use are null.*

I called this offset last class.
Tracked Array holding \(<w, x, y, \emptyset, z>\)

redirect used to navigate to next element in iteration order from a tracked element even when it has been deleted

underlying index (not position) stored. Then only need to update when element is physically moved