Inside Cover of A Practical Guide to Data Structures & Algorithms in Java

we will discuss these in Taxonomy of ADTs, Part II
Disjoint Set of Elements

Partition

Collection of Elements

We'll focus on these for much of this course (and much of the text).

binary relationships

Graphs

(methods involve looking at paths)
Collection of Elements

ex) print queue in order submitted → Manually Positioned

Algorithmically Positioned

ex) queue ordered by the priority for each job
Manually Positioned Collections

- General
- Access
- At ends
- FIFO
- LIFO
- Position: $0, 1, 2, \ldots, n-1$
- Number of elements in the collection: $n$
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 \( p \)
+ constant time to add or remove once element located
after this
insert 13
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

  ystrip

  Changes when doubling array size

  Insertion at ystrip[4] triggers resizing
Circular Array

Queue with at most 4 elements

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>index</th>
</tr>
</thead>
</table>

front  b  c  d  back

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
Array
Dynamic Array
Circular Array
Dynamic Circular Array

Tracked Array

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

index

Singly Linked List
Doubly Linked List
### For time

\[ O(1) \sim O(\min(\frac{P}{n}, n-P)) \sim O(n) \]

<table>
<thead>
<tr>
<th>Key</th>
<th>Array</th>
<th>Circular Array</th>
<th>Dynamic Array</th>
<th>Dynamic-Circular Array</th>
<th>TrackedArray</th>
<th>SinglyLinkedList</th>
<th>DoublyLinkedList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excellent</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Good</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Fair</td>
<td>⬤</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Not required</td>
<td>⬤</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

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