Algorithm Design

M. T. Goodrich and R. Tamassia

John Wiley & Sons

Solution of Exercise C-2.5

We use the circular array approach (recall the implementation of a queue, Figure 2.4 of the textbook). For that purpose, we define the two indexes f and l to point to the element of rank 0 and the element of rank $n-1$ respectively, where n is the size of the vector (alternatively, l could point to the array position that is next to element of rank $n-1$). In our algorithms, we use modular N arithmetic (N is the size of the array A used to implement the vector).

We give the pseudo-code for the various methods:

Algorithm size():
  return $N + l - f + 1 \text{ mod } N$

Algorithm isEmpty():
return $(l = f - 1)$

Algorithm elemAtRank(r):
  if $r < 0 \vee r \geq \text{size}()$ **then**
    throw InvalidRankException
  return $A[(f + r) \text{ mod } N]$

```
Algorithm replaceAtRank(r,e):
  if r < 0 ∨ r ≥ size() then
    throw InvalidRankException
  o ← A[(f + r) mod N]
  A[(f + r) mod N] ← e
  return o
```

```
Algorithm insertAtRank(r,e):
  s ← size()
  if s = N − 1 then
    throw VectorFullException
  if r < 0 ∨ r ≥ size() then
    throw InvalidRankException
  if r < ⌊s/2⌋ then
    for i ← f, (f + 1) mod N, ..., (f + r − 1) mod N do
      A[(i − 1) mod N] ← A[i]
    A[(f + r − 1) mod N] ← e
    f ← (f − 1) mod N
  else
    for i ← l, (l − 1) mod N, ..., (l − s + r + 1) mod N do
      A[(i + 1) mod N] ← A[i]
    A[(l − s + r + 1) mod N] ← e
    l ← (l + 1) mod N
```

```
Algorithm removeAtRank(r):
  s ← size()
  if isEmpty() then
    throw VectorEmptyException
  if r < 0 ∨ r ≥ size() then
    throw InvalidRankException
  o ← elemAtRank(r)
  if r < ⌊s/2⌋ then
    for i ← (f + r − 1) mod N, (f + r) mod N, ..., (f + 1) mod N, f do
      A[(i + 1) mod N] ← A[i]
    f ← (f + 1) mod N
  else
    for i ← (l − s + 1) mod N, (l − s) mod N, ..., (l − 1) mod N, l do
      A[(i − 1) mod N] ← A[i]
```

$l \leftarrow (l - 1) \bmod N$
return o