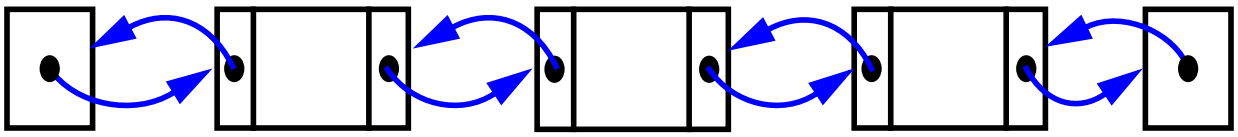
SEKVENSER

- ADT Vektorer
- ADT Posisjoner
- ADT Lister
- Generell ADT Sekvenser



- Idag: Vektor implementert vha tabell og vha 2-veis lenket liste

ADT Vektor

- ADT Vektor (også kallt 'Ranked sequence i læreboka) er en samling av n elementer *S*, ordnet etter rank, med følgende metoder:

  - elemAtRank(*r*):
    Return the element of *S* with rank *r*; an error occurs if $r < 0$ or $r > n - 1$

  - replaceAtRank(*r*,*e*):
    Replace the element at rank *r* with *e* and return the old element; an error condition occurs if $r < 0$ or $r > n - 1$

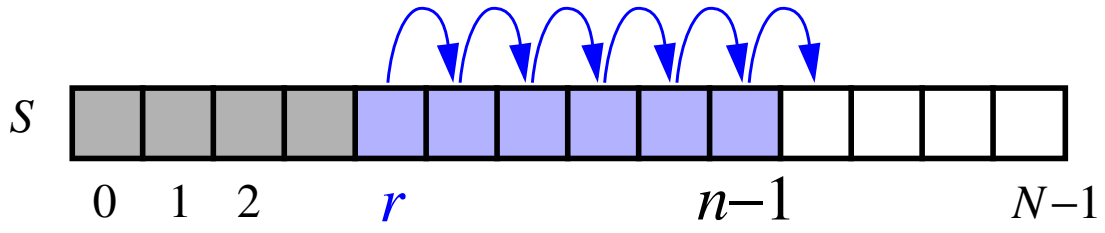
  - insertAtRank(*r*,*e*):
    Insert a new element into *S* which will have rank *r*; an error occurs if $r < 0$ or $r > n - 1$

  - removeAtRank(*r*):
    Remove from *S* the element at rank *r*; an error occurs if $r < 0$ or $r > n - 1$

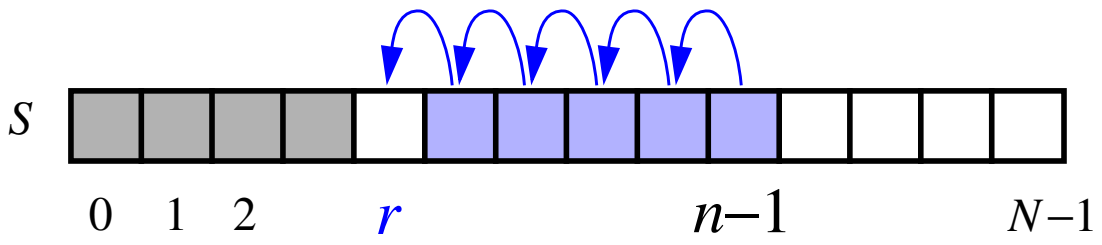
Tabell-basert implementasjon

- Pseudokode:

Algorithm insertAtRank(r,e):
  for $i = n - 1, n - 2, \ldots, r$ **do**
    $S[i+1] \leftarrow s[i]$
  $S[r] \leftarrow e$
  $n \leftarrow n + 1$



Algorithm removeAtRank(r):
  $e \leftarrow S[r]$
  for $i = r, r + 1, \ldots, n - 2$ **do**
    $S[i] \leftarrow S[i + 1]$
  $n \leftarrow n - 1$
  return



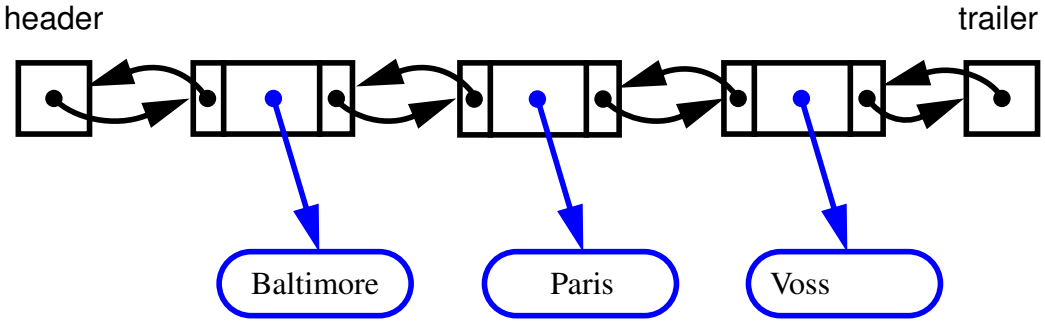
Tabell-basert implementasjon

- Kjøretid (tidskompleksitet) for metodene:

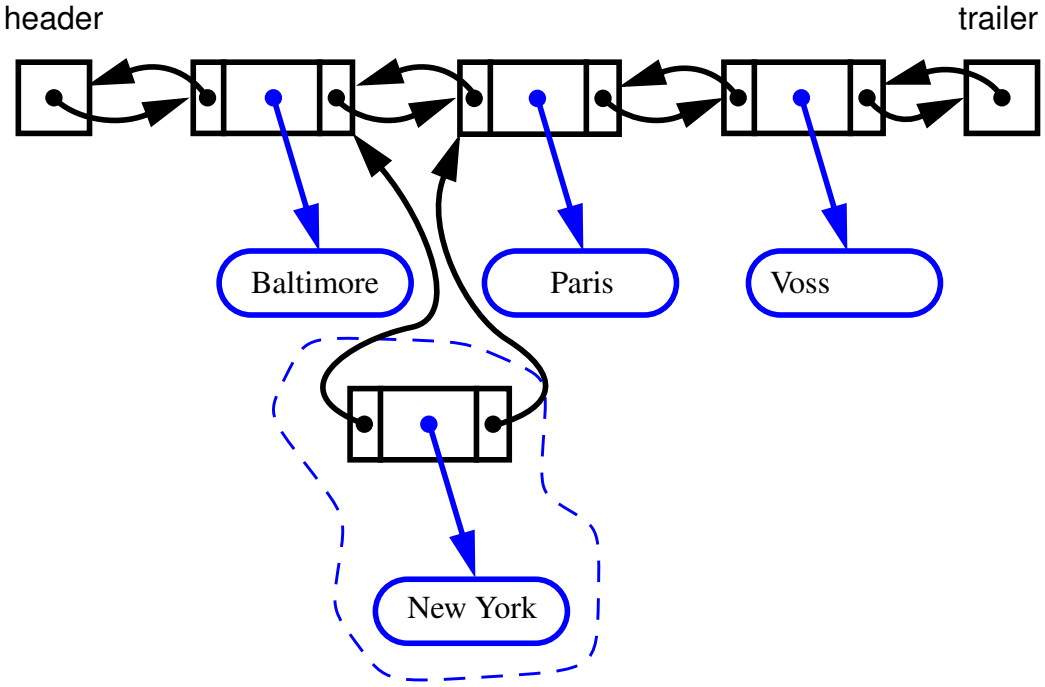
Metode	Tid
size	$\boldsymbol{O}(1)$
isEmpty	$\boldsymbol{O}(1)$
elemAtRank	$\boldsymbol{O}(1)$
replaceAtRank	$\boldsymbol{O}(1)$
insertAtRank	$\boldsymbol{O}(n)$
removeAtRank	$\boldsymbol{O}(n)$

Implementasjon vha 2-veis lenket liste

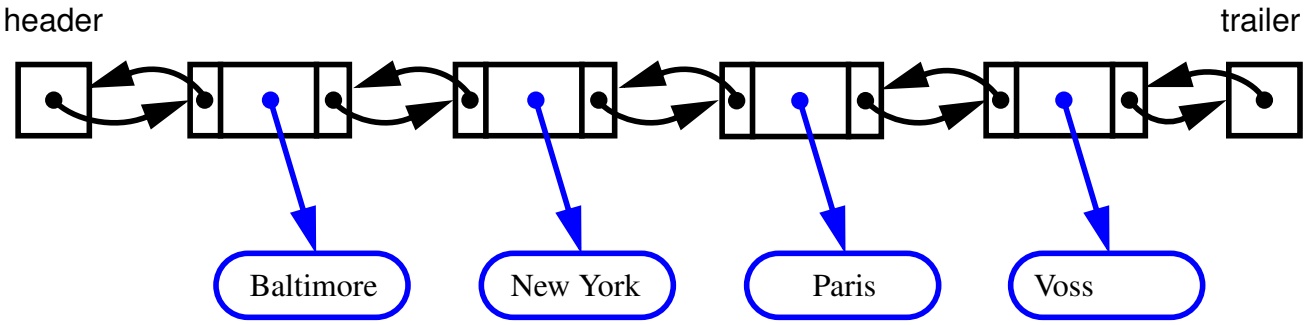
- listen før innsetting:



- lag ny node for innsetting:



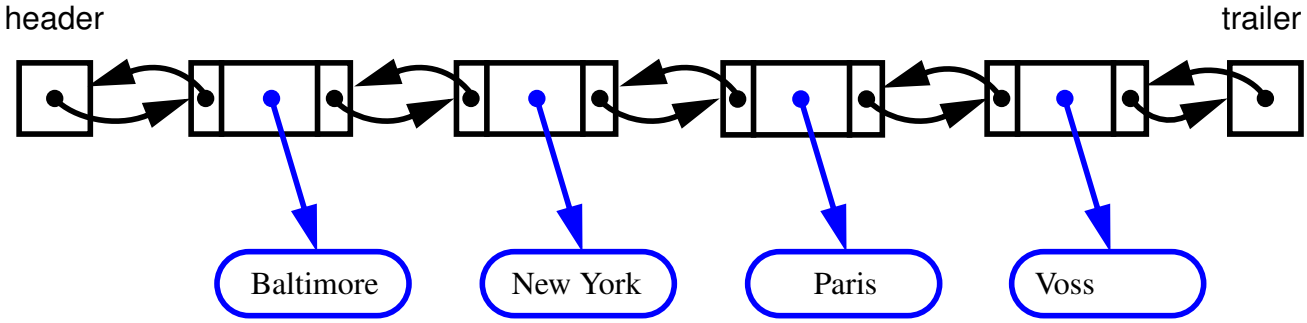
- listen etter innsetting:



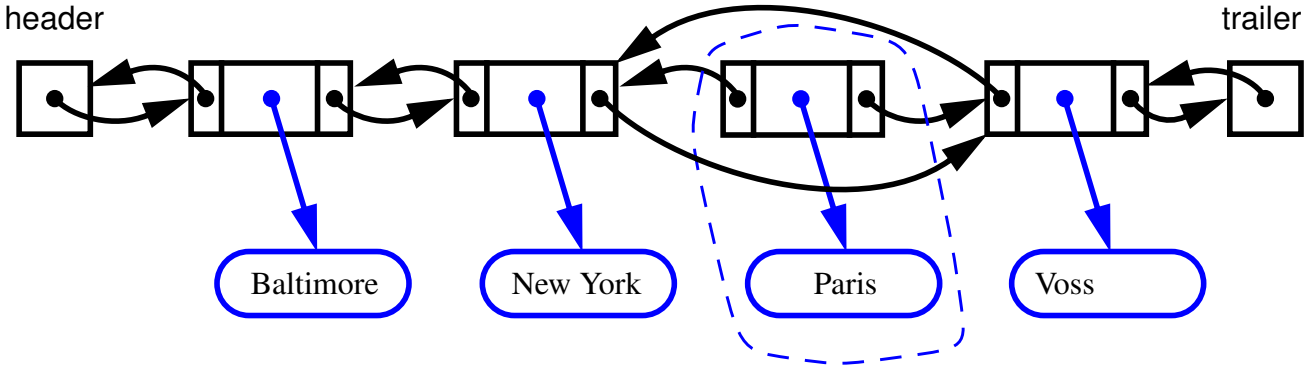

```
public void insertAtRank (int rank, Object element)
   throws BoundaryViolationException {
      if (rank < 0 || rank > size())
         throw new BoundaryViolationException("invalid rank");
      DLNode next = nodeAtRank(rank); // the new node
                        //will be right before this
      DLNode prev = next.getPrev(); // the new node
                        //will be right after this
      DLNode node = new DLNode(element, prev, next);
      // new node knows about its next & prev. Now
      // we tell next & prev about the new node.
      next.setPrev(node);
      prev.setNext(node);
      size++;
   }
```

Implementering vha 2-veis lenket liste

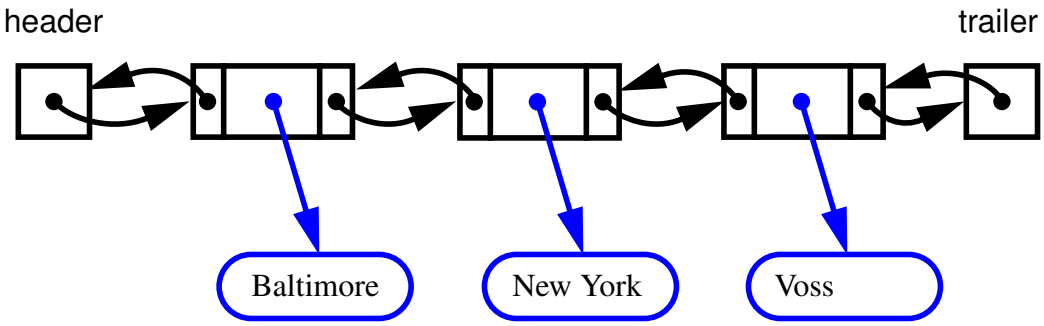
- listen før fjerning av element:



- fjern en node:



- etter fjerning:



Java-Implementasjon

- fjerning av node

```
public Object removeAtRank (int rank)
                    throws BoundaryViolationException {
  if (rank < 0 || rank > size()–1)
     throw new BoundaryViolationException("Invalid
          rank.");
  DLNode node = nodeAtRank(rank); // node to
                                  // be removed
  DLNode next = node.getNext(); // node before it
  DLNode prev = node.getPrev(); // node after it
  prev.setNext(next);
  next.setPrev(prev);
  size--;
  return node.getElement(); // returns the
              // element of the deleted node
 }
```

Java-Implementasjon

- finne node med gitt rank

```
private DLNode nodeAtRank (int rank) {
  // auxiliary method to find the node of the
  // element with the given rank. We make
  // auxiliary methods private or protected.
  DLNode node;
  if (rank <= size()/2) { //scan forward from head
     node = header.getNext();
     for (int i=0; i < rank; i++)
     node = node.getNext();
  }
  else { // scan backward from the tail
     node = trailer.getPrev();
     for (int i=0; i < size()-rank-1 ; i++)
     node = node.getPrev();
  }
  return node;
}
```

Noder

- Lenkede lister kan effektivt utføre ***node-baserte operasjoner***:
  - removeAtNode(Node v) og insertAfterNode(Node v, Object e), tar begge O(1) tid.
- MEN, node-baserte operasjoner kan ikke overføres til tabell-baserte implementasjoner, da tabeller ikke har noder!
- Node er en implementasjonsdetalj.
- **Dilemma**:
  - Hvis vi ikke definerer nodebaserte operasjoner, så gjør vi ikke full nytte av lenkede lister.
  - Hvis vi definerer dem, så bryter vi med grunnideen om ADT, dvs at ADT skal defineres uavhengig av bestemt implementasjon.

Fra Noder til Posisjoner

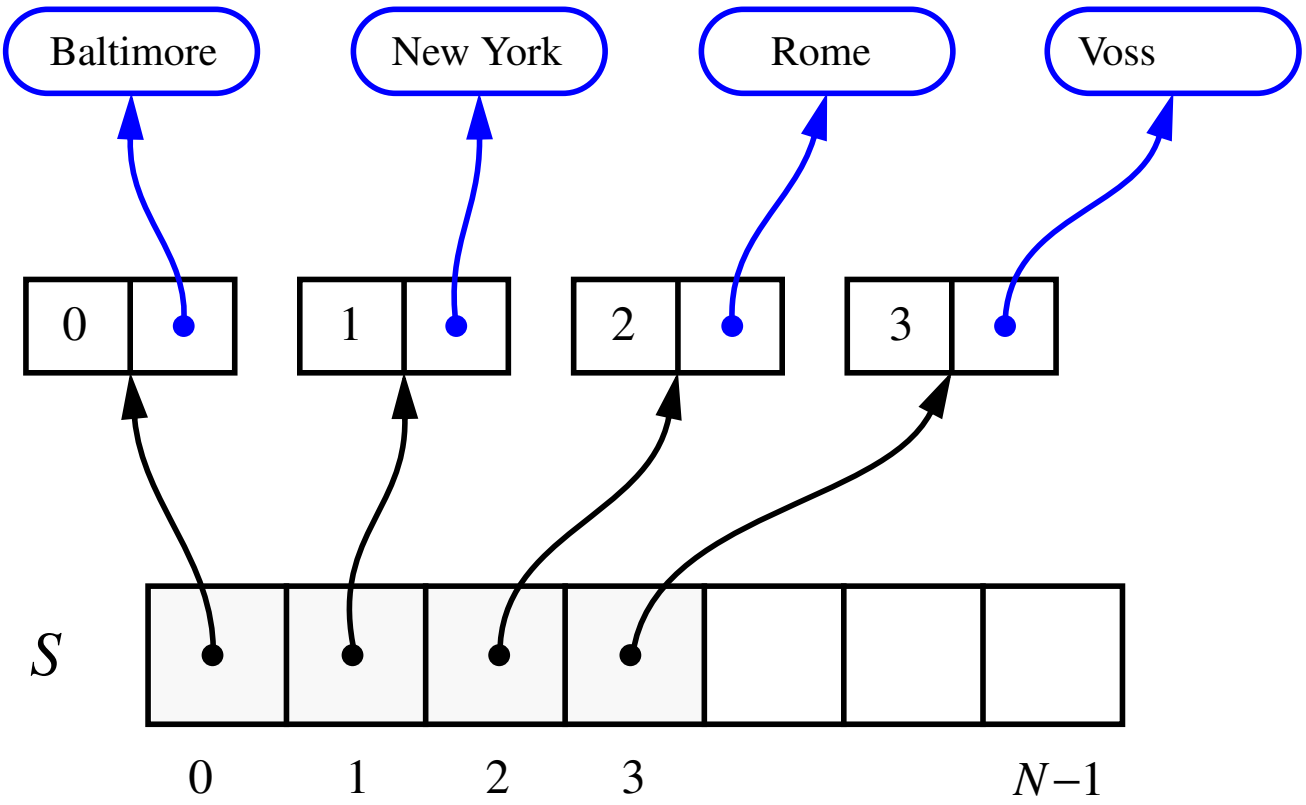
- Vi introduserer ADT ***Posisjon*** (eng: position), som intuitivt er "stedet" et element befinner seg.
- Posisjoner har kun en metode:
  element(): Return the element at this position .
- En posisjon defineres relativt til andre posisjoner (foran/bak relasjon)
- Posisjoner er ikke knyttet til et element eller en rank

ADT LISTE

- ADT Liste (også kallt 'Positional sequence' i læreboka) er en ADT med posisjon-baserte metoder
- generiske metoder size(), isEmpty()
- query metoder isFirst(p), isLast(p)
- accessor metoder first(), last(), before(p), after(p)
- update metoder swapElements(p,q), replaceElement(p,e), insertFirst(e), insertLast(e), insertBefore(p,e), insertAfter(p,e). remove(p)
- implementert vha 2-veis lenket liste vil hver av disse metodene ta O(1) tid.

Sekvens ADT

- Kombinerer ADTer Vektor og List (multippel arv av grensesnitt)
- Nye metoder bygger bro mellom rank og posisjon:
  - atRank(r) returns a position
  - rankOf(p) returns an integer rank
- En tabell-basert implementasjon må bruke objekter for å representere posisjoner:



Sammenlikning av sekvens-implementasjoner

Operasjoner	Array	List
size, isEmpty	$O(1)$	$O(1)$
atRank, rankOf, elemAtRank	$O(1)$	$O(n)$
first, last	$O(1)$	$O(1)$
before, after	$O(1)$	$O(1)$
replaceElement, swapElements	$O(1)$	$O(1)$
replaceAtRank	$O(1)$	$O(n)$
insertAtRank, removeAtRank	$O(n)$	$O(n)$
insertFirst, insertLast	$O(1)$	$O(1)$
insertAfter, insertBefore	$O(n)$	$O(1)$
remove	$O(n)$	$O(1)$

Iterator

- Er en abstraksjon av det å 'gå gjennom alle elementene i en gitt samling elementer'.
- Iterator enkapsulerer ideene om 'sted' og 'neste'.
- Utvider ADT posisjon
- Generiske og spesialiserte iteratorer.
- ***ObjectIterator***
  - hasNext()
  - nextObject()
  - object()
- ***PositionIterator***
  - nextPosition()
- Nyttige metoder som returnerer iteratorer:
  - elements()
  - positions()