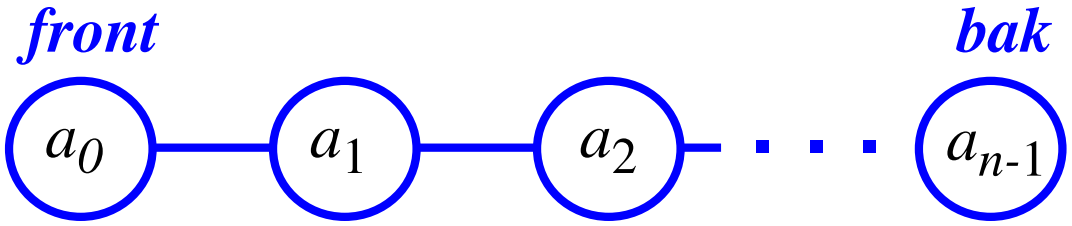
ADTer: Stabel, Kø og 2-sidig Kø

- Data strukturer, dvs konkrete implementasjoner: Tabell, lenket liste, 2-veis lenket liste.
- Tidligere har vi sett: Stabel implementert vha tabell. Idag: ADT stabel implementert ved å 'adaptere' ADT 2-sidig kø.
- Kø implementert vha tabell og vha lenket liste, samt ved å adaptere ADT 2-sidig kø.
- 2-sidig kø implementert vha 2-veis lenket liste.

ADT Kø (eng: queue)

- En kø følger first-in-first-out (FIFO) prinsippet.
- Elementer kan innsettes når som helst, men kun elementet som har befunnet seg lengst i køen kan taes ut.
- Elementene settes inn bakerst (eng:rear) (enqueue) og fjernes fra fronten (dequeue)

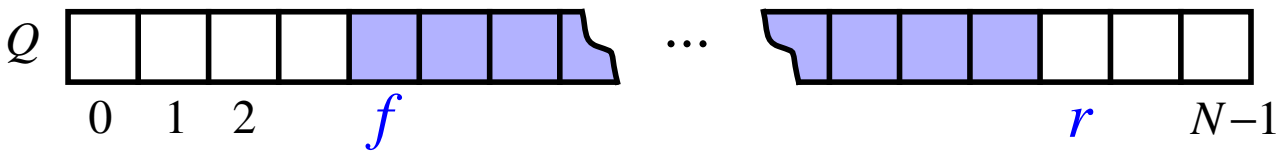


Abstrakt Data Type Kø

- To metoder:
  - enqueue(*o*): Insert object *o* at the rear of the queue
  - dequeue(): Remove the object from the front of the queue and return it; an error occurs if the queue is empty
- Følgende støtte-metoder kan også defineres:
  - size(): Return the number of objects in the queue
  - isEmpty(): Return a boolean value that indicates whether the queue is empty
  - front(): Return, but do not remove, the front object in the queue; an error occurs if the queue is empty

En tabell-basert kø

- Implementer en kø vha array som brukes sirkulært
- Max.størrelse N, f.eks. N= 1,000.
- Køen består av en N-element array Q og to integer variable:
  - f, index til front element
  - r, index til element rett etter det bakerste
- "normal konfigurasjon"



- "sirkulær" konfigurasjon

- Hva innebærer f=r ?

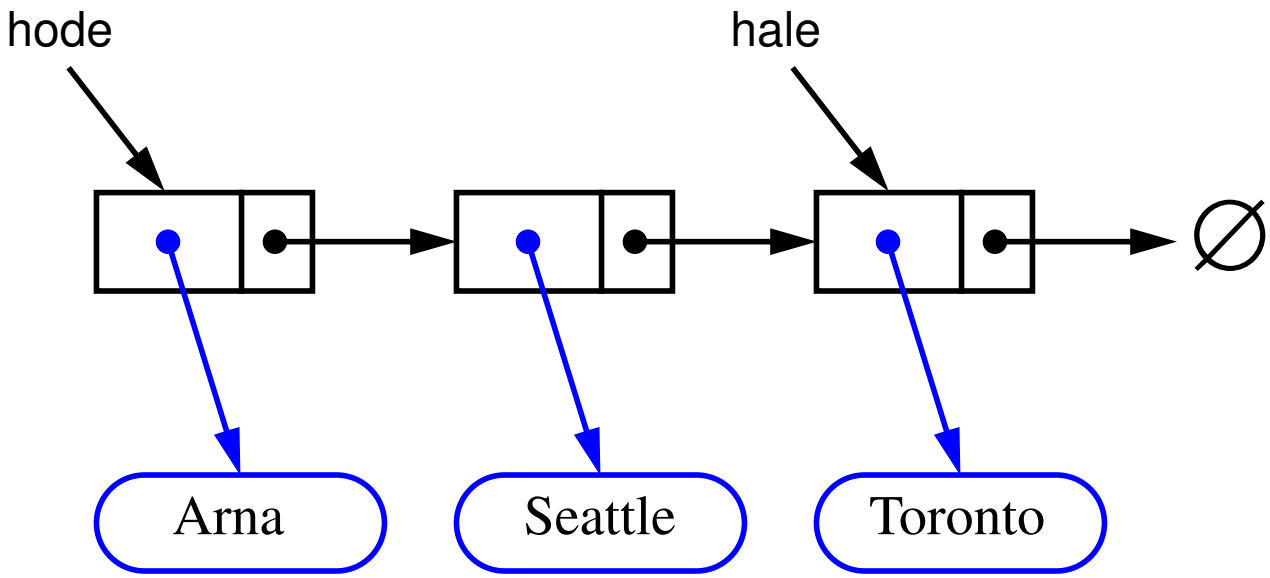
Tabell-basert kø

- Pseudokode

```
Algorithm size():
  return (N - f + r) mod N

Algorithm isEmpty():
  return (f = r)

Algorithm front():
  if isEmpty() then
    throw a QueueEmptyException
  return Q[f]

Algorithm dequeue():
  if isEmpty() then
    throw a QueueEmptyException
  temp ← Q[f]
  Q[f] ← null
  f ← (f + 1) mod N
  return temp

Algorithm enqueue(o):
  if size = N - 1 then
    throw a QueueFullException
  Q[r] ← o
```

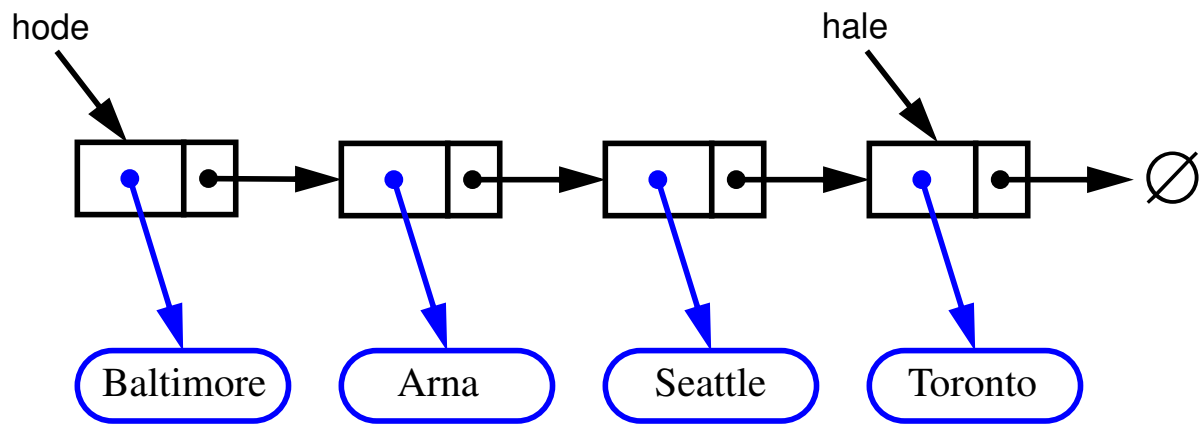
Kø implementert vha lenket liste

- noder kjedet sammen i en lineær ordning
-

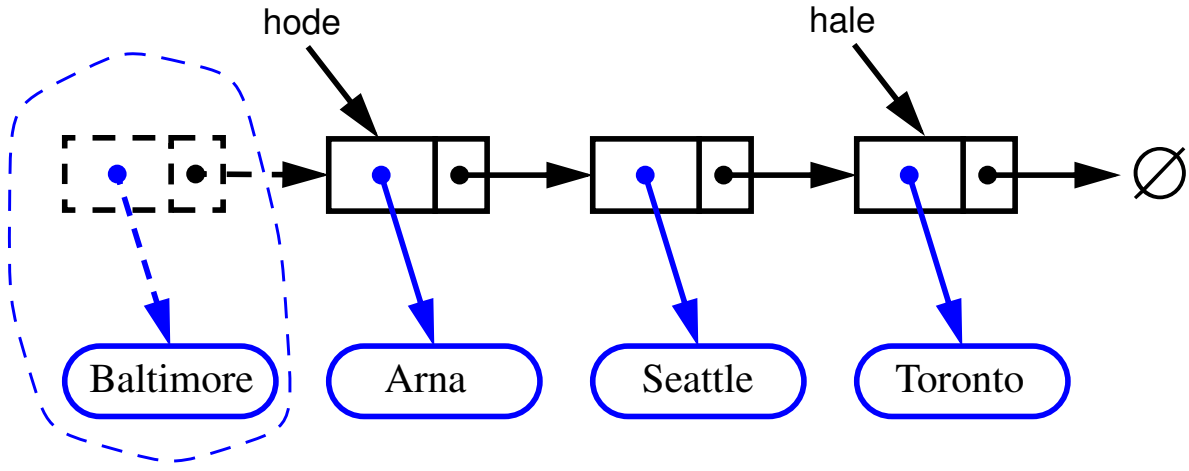


- hode på listen er foran i køen, og halen på listen er bakerst i køen
- hvorfor ikke motsatt?

Fjerne element i Hode



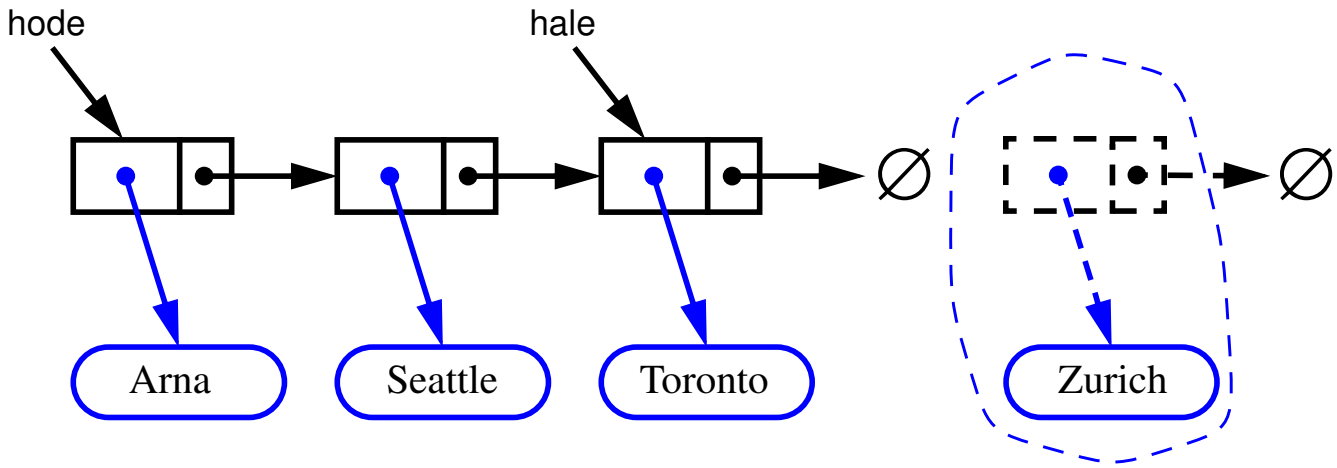
- flytt hodepeker fram



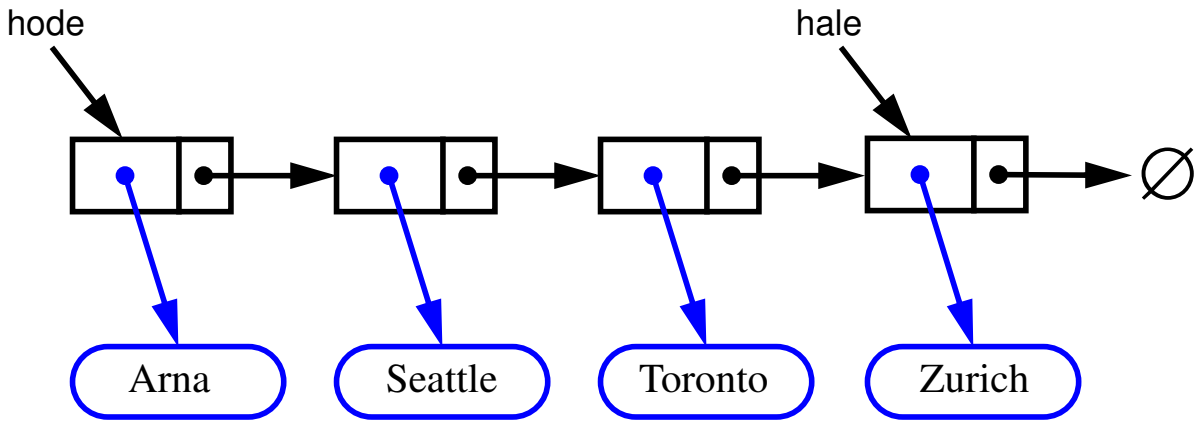
- Sette inn et element i Hode er ikke så lett

Sette inn i Halen

- lag ny node



- legg noden inn i kjeden og flytt halepeker



- hva med å fjerne element i halen?

To-sidige køer

- En to-sidig kø, (en: double-ended queue = deque), tillater innsetting og fjerning både foran og bak.
- Deque Abstrakt Data Type
  - insertFirst(*e*): Insert e at the deginning of deque.
  - insertLast(*e*): Insert e at end of deque
  - removeFirst(): Removes and returns first element
  - removeLast(): Removes and returns last element
- Støtte-metoder kan være:
  - first()
  - last()
  - size()
  - isEmpty()

Implementering av stabler og køer vha Deques

- Stabler med Deques:

Stack Method	Deque Implementation
size()	size()
isEmpty()	isEmpty()
top()	last()
push(e)	insertLast(e)
pop()	removeLast()

- Køer med Deques:

Queue Method	Deque Implementation
size()	size()
isEmpty()	isEmpty()
front()	first()
enqueue()	insertLast(e)
dequeue()	removeFirst()

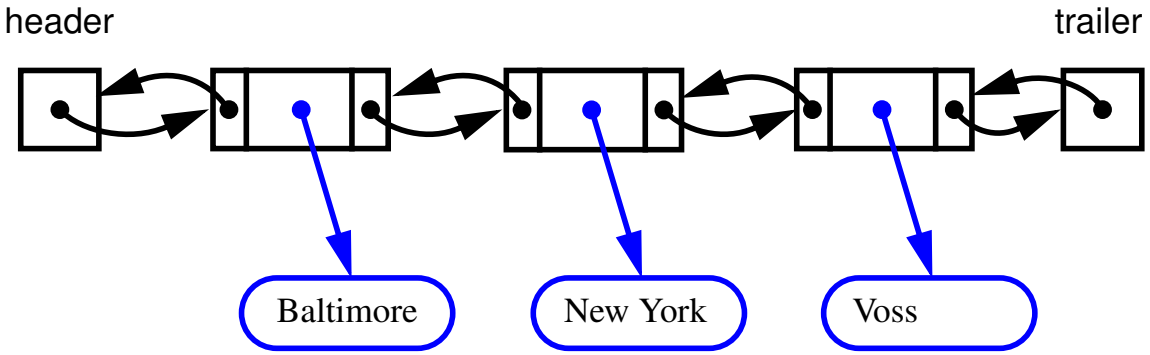
Adaptor Pattern

- Det å bruke en deque til å implementere kø eller stabel er eksempel på adaptor pattern. Adaptor pattern innebærer implementasjon av en ADT vha av en annen ADT.

- 2 mulige applikasjoner:
  - Spesialisere en generell ADT ved å endre noen metoder.
    Eks: implementere stabel ved deque.
  - Spesialisere typen objekter som en ADT bruker.
    Eks: Definer IntegerArrayStack som adapterer ArrayStack til å lagre kun integers.

Implementering av 2-sidige køer vha 2-veis lenkede lister

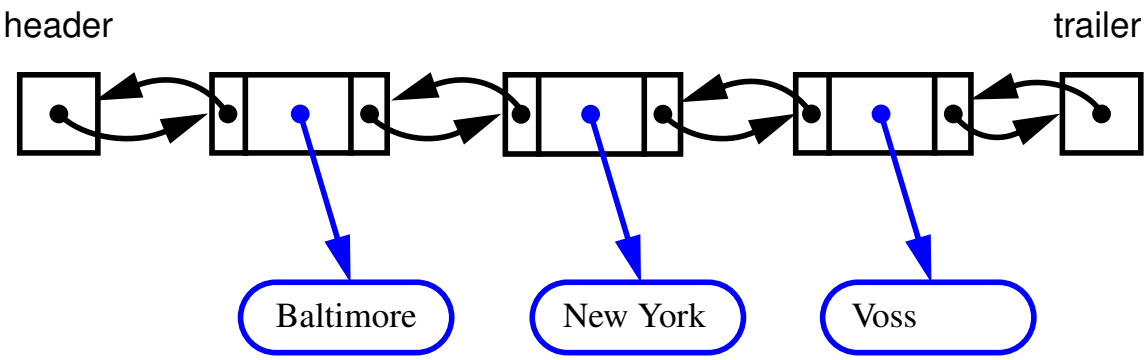
- Fjerning fra halen av 1-veis lenket liste kan ikke gjøres i O(1) tid, dvs konstant tid.
- For implementering av deque, bruker vi 2-veis lenket liste (eng: doubly linked list) med ekstra noder kallt header og trailer.

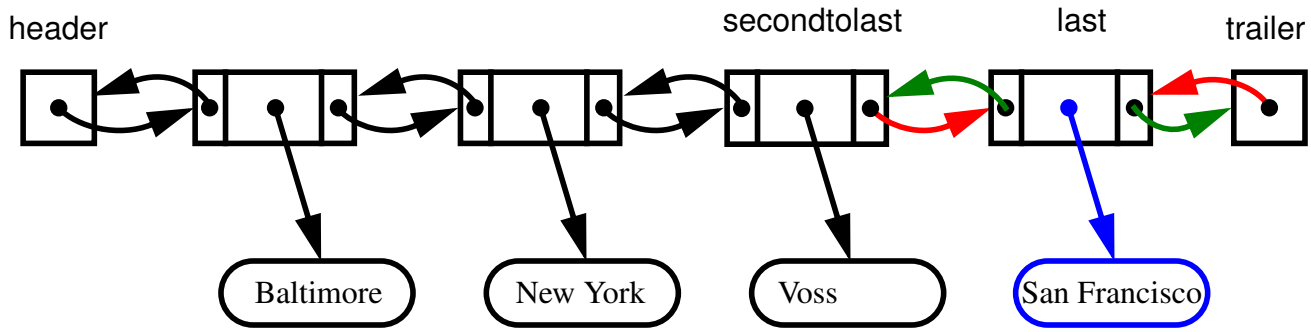
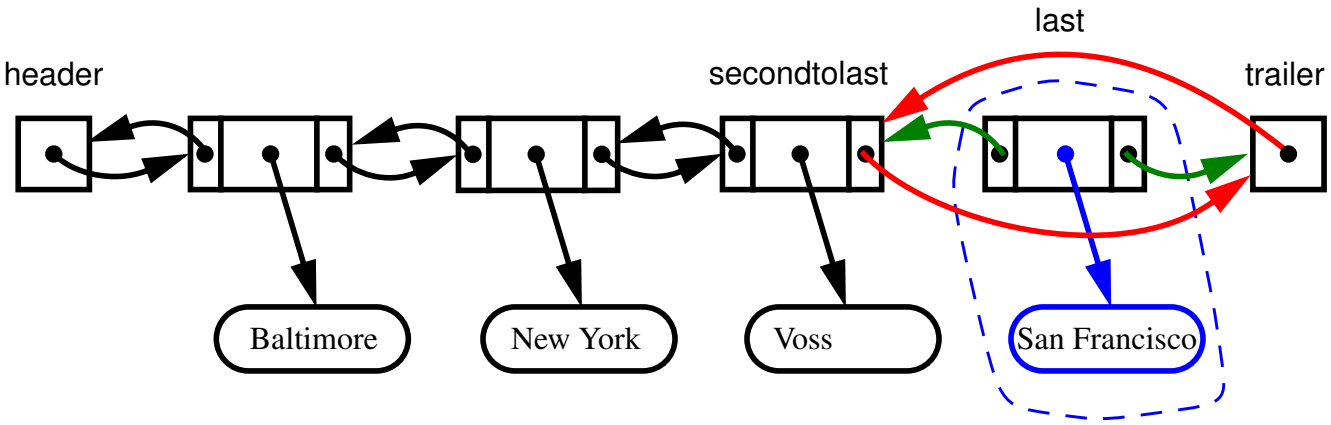
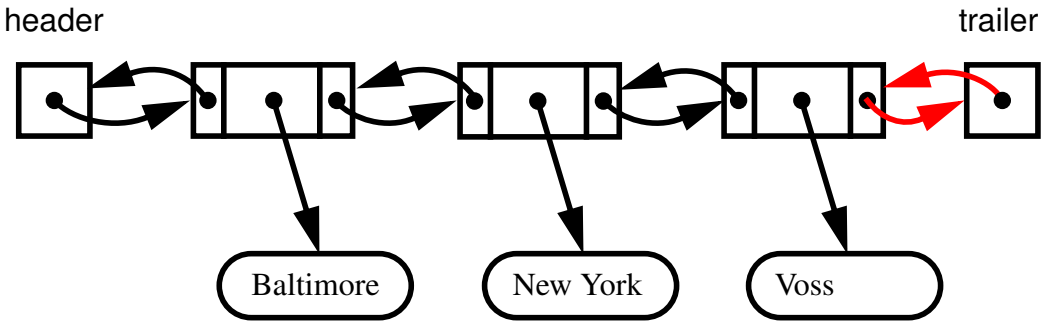


- Noder har next og prev lenker. Følgende metoder (NB: Ikke en ADT, men en data struktur):
  - setElement(Object e)
  - setNext(Object newNext)
  - setPrev(Object newPrev)
  - getElement()
  - getNext()
  - getPrev()
- Alle deque metoder har nå O(1) kjøretid.

Implementering av 2-sidige køer vha 2-veis lenkede lister

- Hva gjør header og trailer nodene?
  - Header noden går foran første element i listen, og har en null prev lenke.
  - Trailer noden kommer etter siste element i listen, og har en null next lenke.
- Header og trailer nodene er "dummy" noder som ikke lagrer elementer
- Diagram:



Implementering av 2-sidige køer vha 2-veis lenkede lister

- Visualisering av removeLast().