Köer

Koffman & Wolfgang

- kapitel 4

Köer

Method	Behavior
`boolean offer(E item)`	Inserts `item` at the rear of the queue. Returns **`true`** if successful; returns **`false`** if the item could not be inserted.
`E remove()`	Removes the entry at the front of the queue and returns it if the queue is not empty. If the queue is empty, throws a `NoSuchElementException`.
`E poll()`	Removes the entry at the front of the queue and returns it; returns **`null`** if the queue is empty.
`E peek()`	Returns the entry at the front of the queue without removing it; returns **`null`** if the queue is empty.
`E element()`	Returns the entry at the front of the queue without removing it. If the queue is empty, throws a `NoSuchElementException`.

”Översta” elementet är det som har väntat längst

- FIFO = First-In-First-Out

Köer är ett vanligt alternativ till stackar:

- används i operativsystem för att hålla koll på processer som väntar på en resurs, t.ex. en skrivarkö
- flera algoritmer använder köer för att implementera *bredden-först*-sökning

Djupet-först vs bredden-först

- Djupet-först-sökning implementeras med en stack
- Bredden-först-sökning implementeras med en kö
- Alternativa sökalgoritmer, t.ex. ”bäst-först”-sökning kan implementeras med en prioritetskö

LinkedList är en Queue

Java har ett gränssnitt för köer:

- java.util.Queue är en utökning av java.util.Collection
- java.util.LinkedList implementerar java.util.Queue

För att skapa en kö i Java:

```
Queue<String> names = new LinkedList<String>();
```

- detta skapar en LinkedList, men eftersom variabeln är deklarerad som en Queue så kan vi bara använda kömetoderna

Java har **inget** gränssnitt för stackar:

- java.util.Stack är en klass, inte ett gränssnitt
- **DÅLIG** design!

Att implementera en kö

Vi kan implementera kön som en *dubbellänkad lista*:

- det är så det är gjort i Java, eftersom LinkedList är en dubbellänkad lista

Men det är egentligen onödigt – det går precis lika bra med en *enkellänkad lista*:

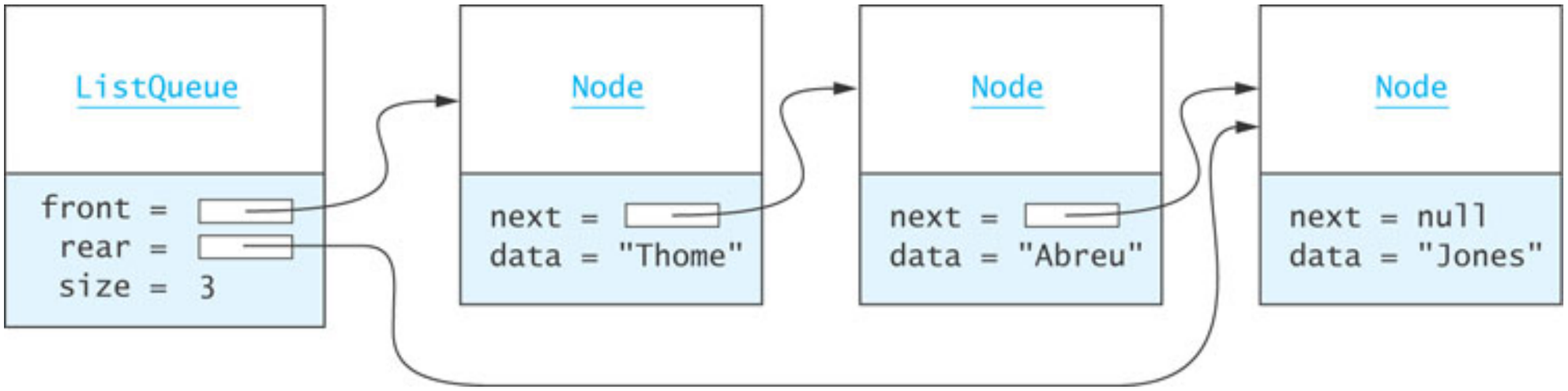
- vi måste ha en pekare till huvudet och en pekare till svansen
- vi stoppar in element i svansen och tar ut dem i huvudet

Det funkar också att implementera som ett *cirkulärt fält*:

- vi måste ha två heltalsvariabler – som säger var början resp. slutet av kön är
- precis som en ArrayList måste vi kunna omallokera fältet när kön blir för stor

En kö som en enkellänka lista

Figur 4.7, sid 207:



Insättningar görs sist i listan:

- ”Jones” är alltså det senast tillagda elementet
- vi behöver peka om `rear.next` och sedan `rear` till den nya noden

Element tas bort från början av listan:

- ”Thome” är alltså det element som har väntat längst
- vi behöver peka om `front` till `front.next`

En kö som ett fält

Vi kan implementera en kö som ett fält.

Antingen med ”översta” elementet först:

- insättning i slutet = O(1)
- borttagning i början = O(n)

Eller med ”översta” elementet sist:

- insättning i början = O(n)
- borttagning i slutet = O(1)

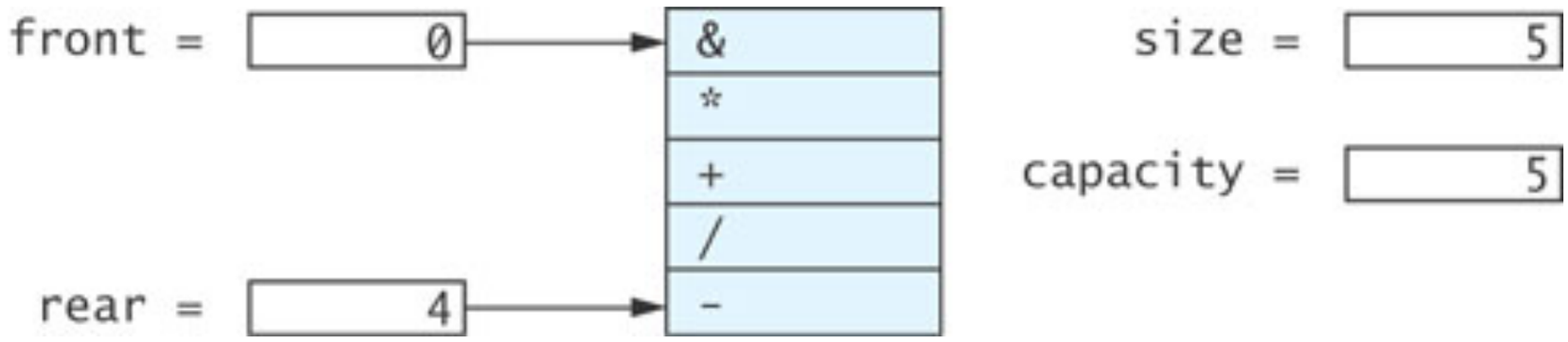
Dvs, alldeles för ineffektivt.

Lösningen är att använda ett cirkulärt fält.

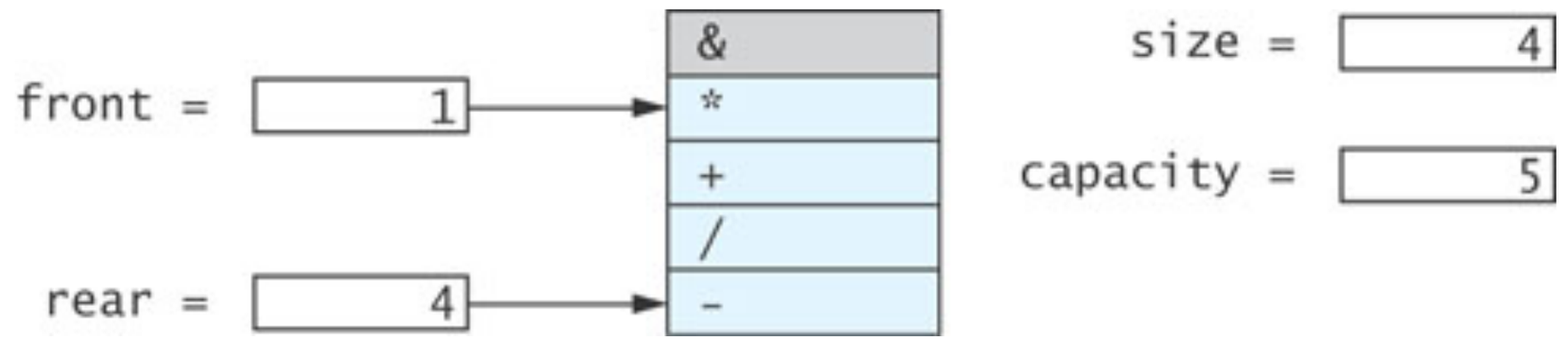
En kö som ett cirkulärt fält

Vi behöver två index, för början och slutet.

Figur 4.8, sid 210 – en kö som har fyllt hela fältet:

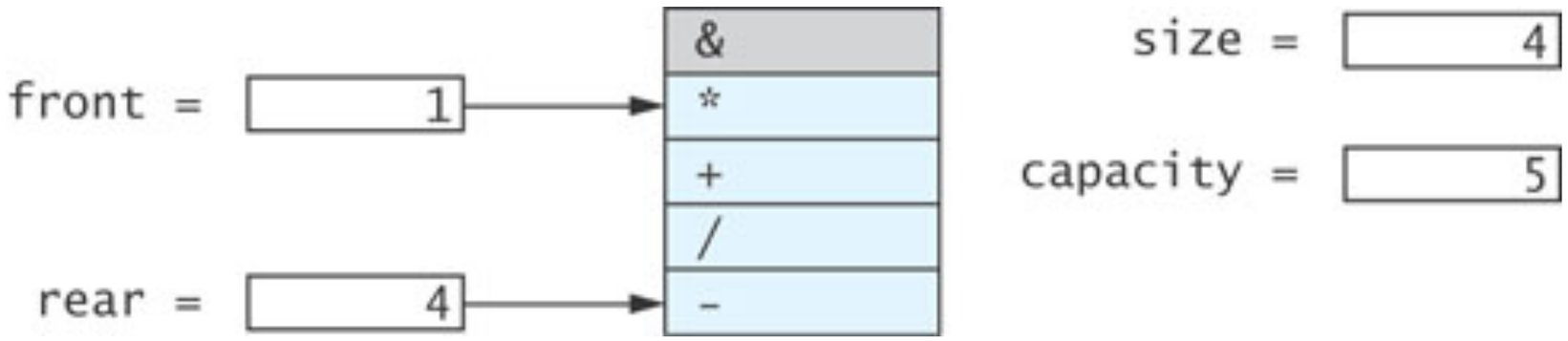


Figur 4.9 – när ett element har tagits bort:

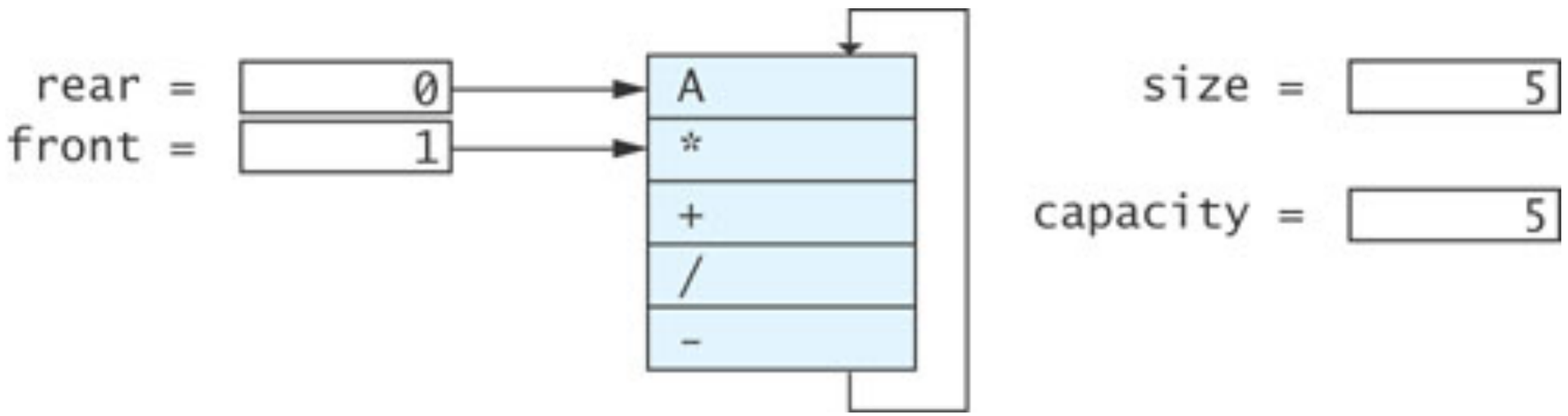


Ett cirkulärt fält

Hur lägger vi till ett nytt element när ändan är sist?



Jo, vi flyttar ändan först i fältet!



Exempel

Exempel

```
ArrayQueue q = new ArrayQueue(5);
```

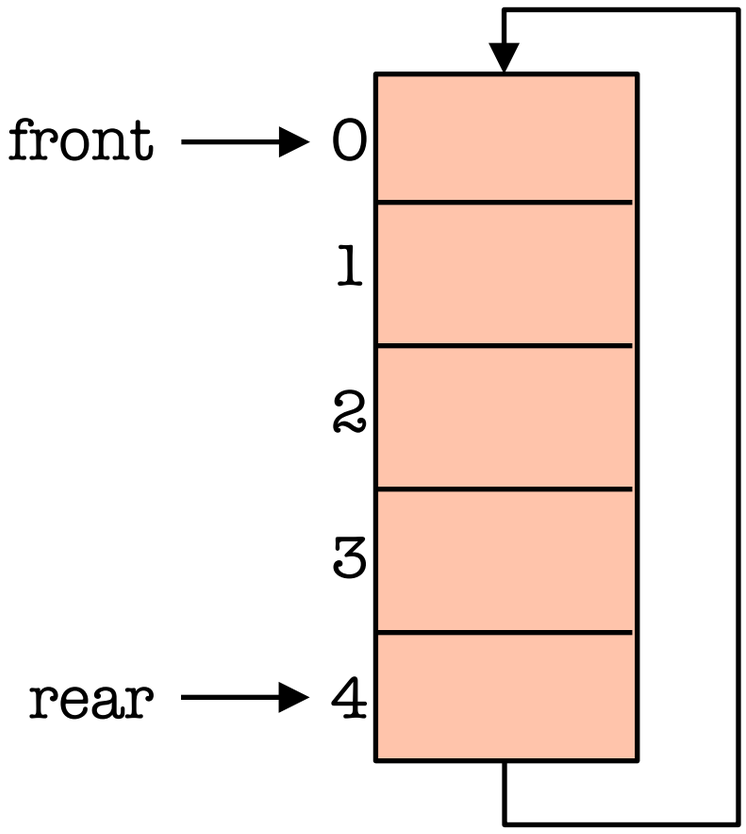
Exempel

```
ArrayQueue q = new ArrayQueue(5);
```

```
public ArrayQueue(int initCapacity) {
        capacity = initCapacity;
        theData = (E[]) new Object[capacity];
        front = 0;
        rear = capacity - 1;
        size = 0;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
```



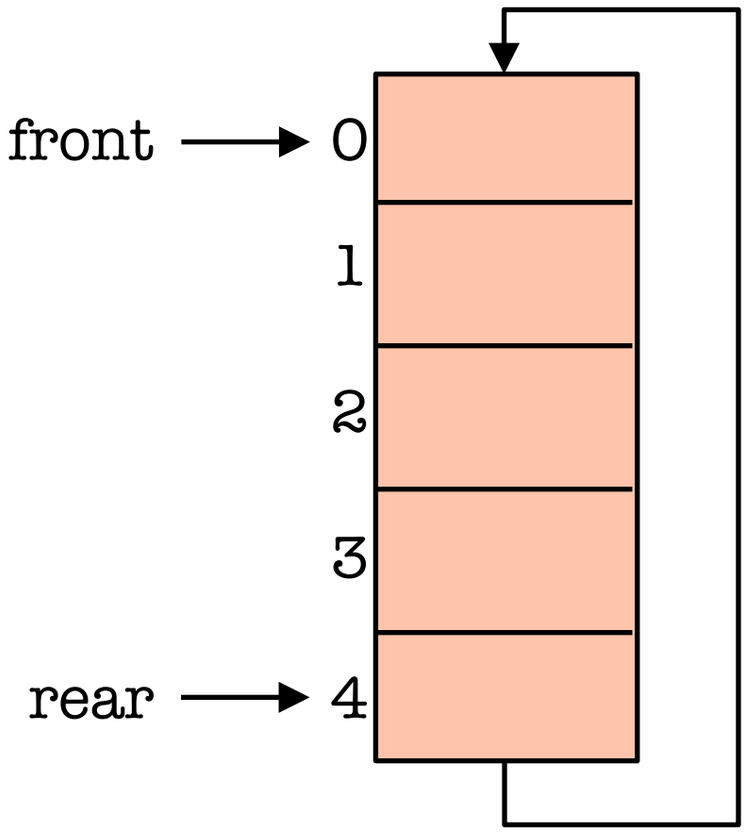
size = 0

capacity = 5

```
public ArrayQueue(int initCapacity) {
        capacity = initCapacity;
        theData = (E[]) new Object[capacity];
        front = 0;
        rear = capacity - 1;
        size = 0;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
```

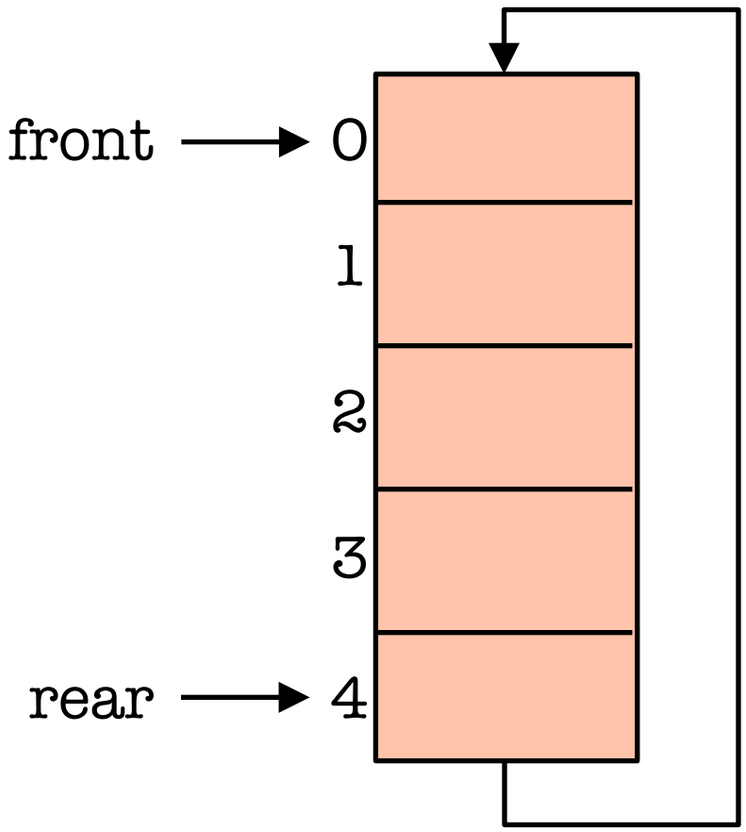


size = 0

capacity = 5

Exempel

```
ArrayQueue q = new ArrayQueue(5);
```

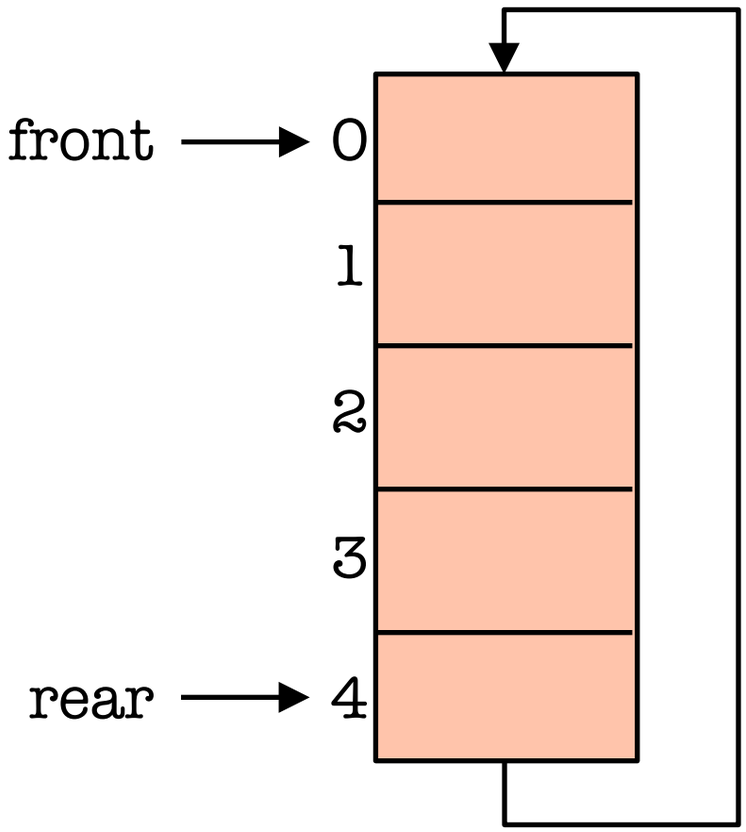


size = 0

capacity = 5

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
```

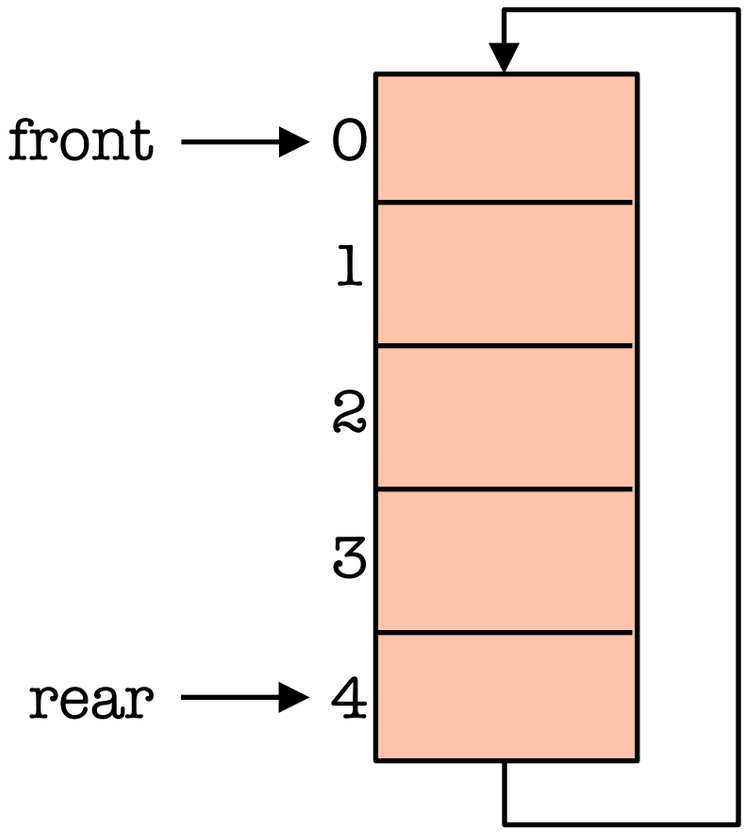


size = 0

capacity = 5

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
```



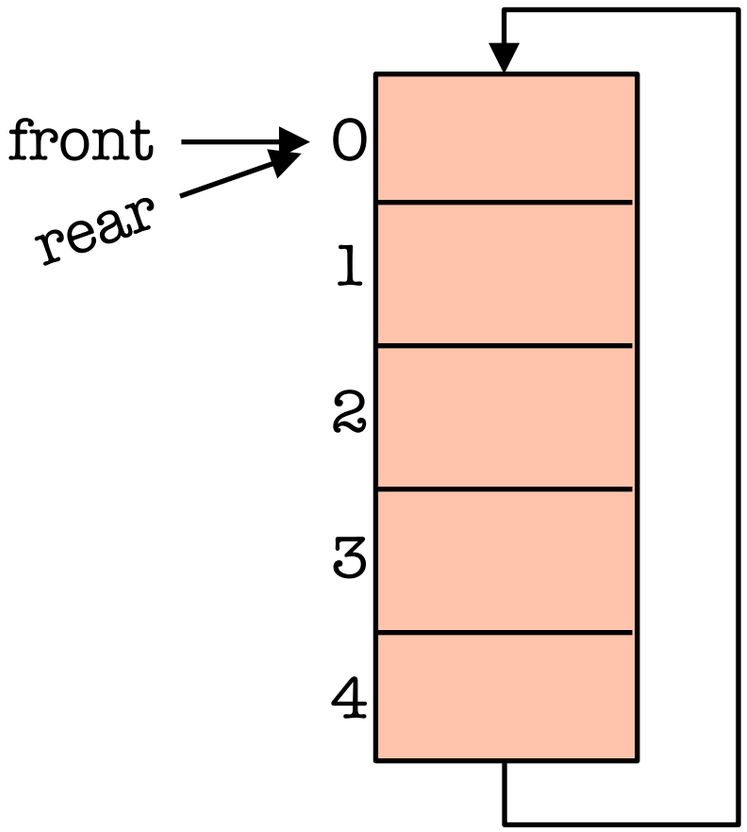
size = 0

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
```



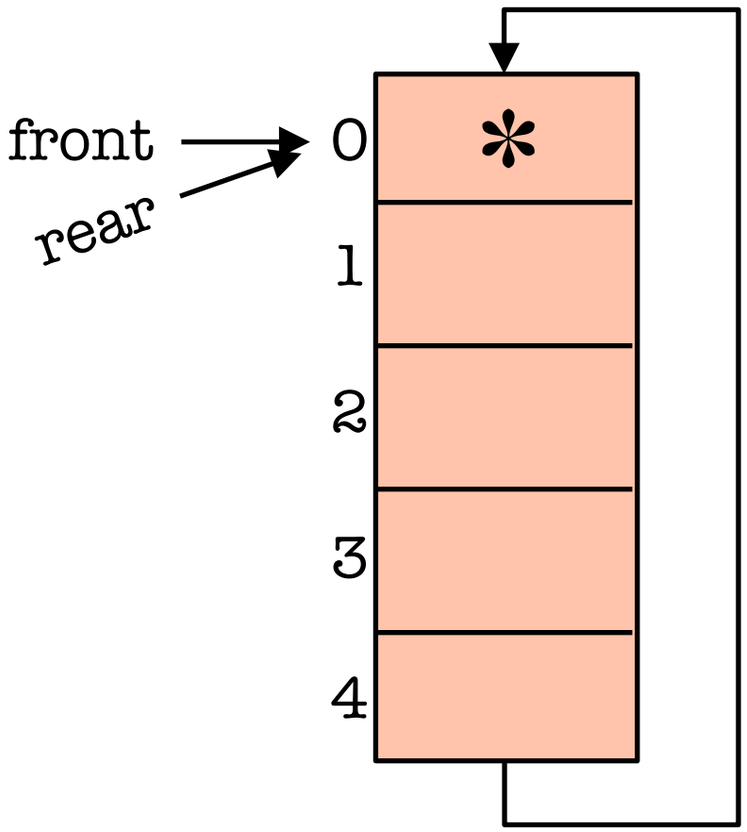
size = 1

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
```



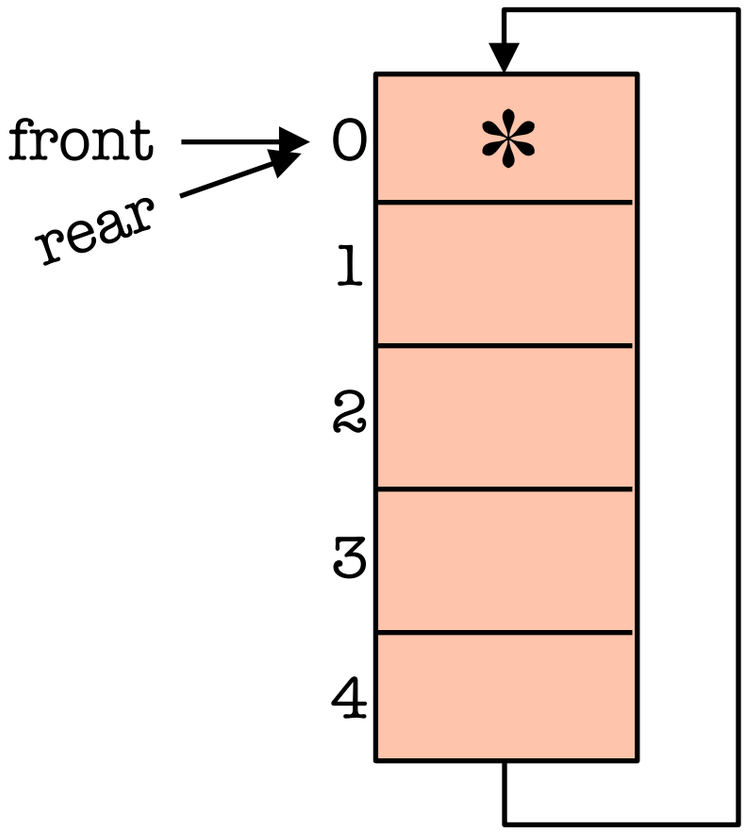
size = 1

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
```



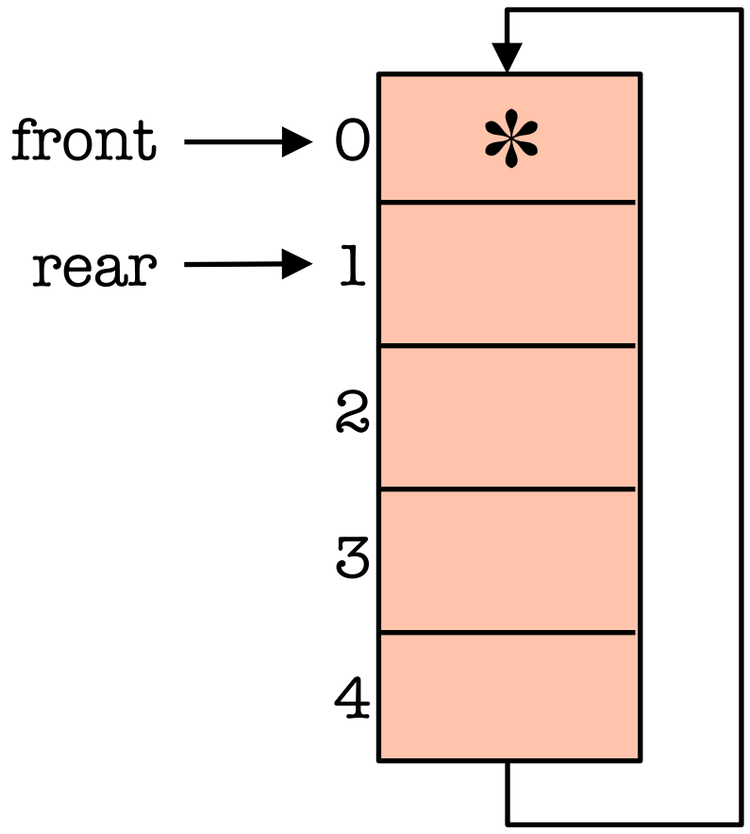
size = 1

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
```



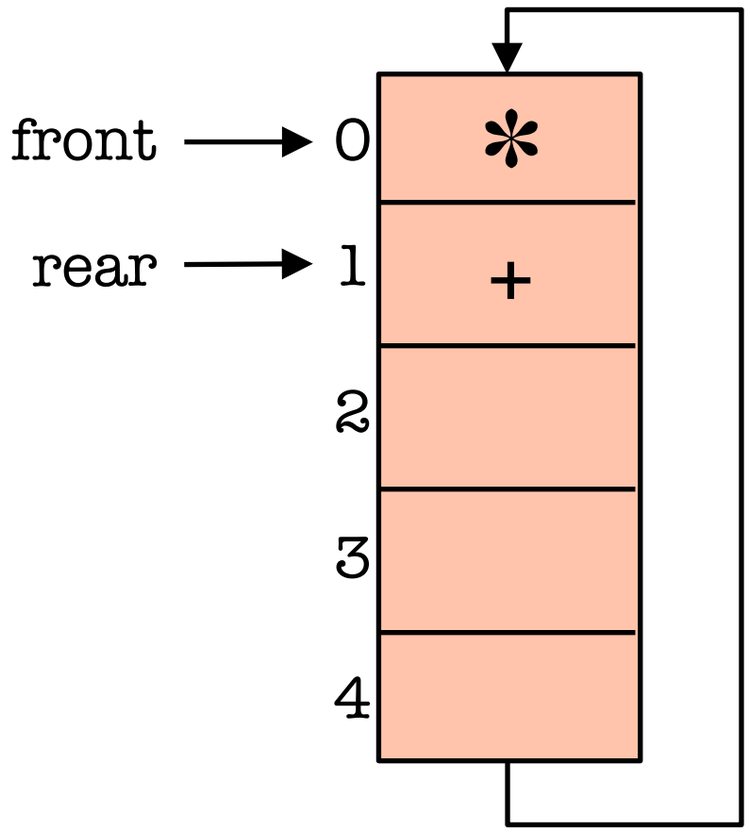
size = 2

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
```



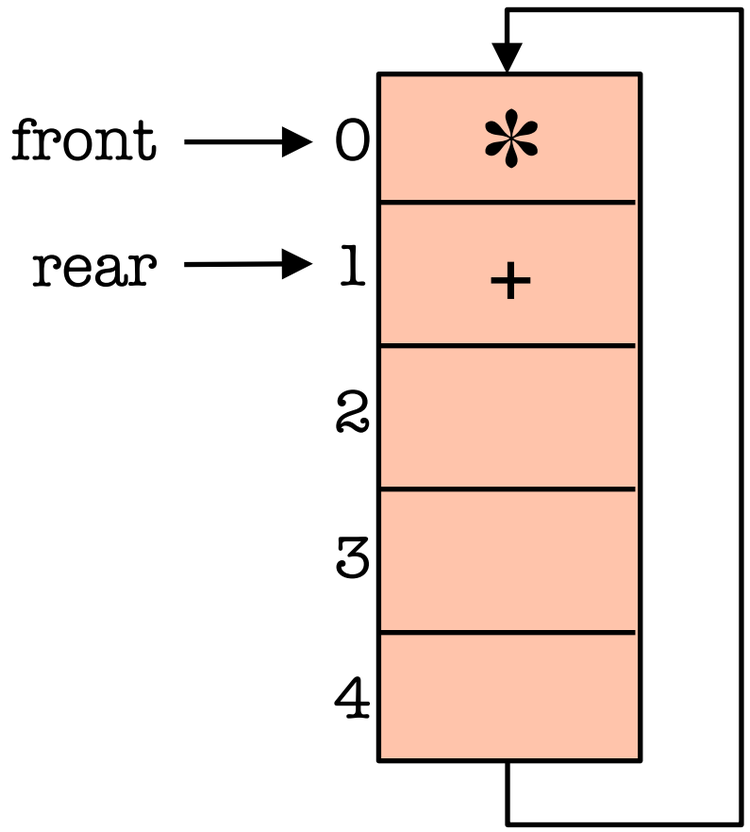
size = 2

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
```



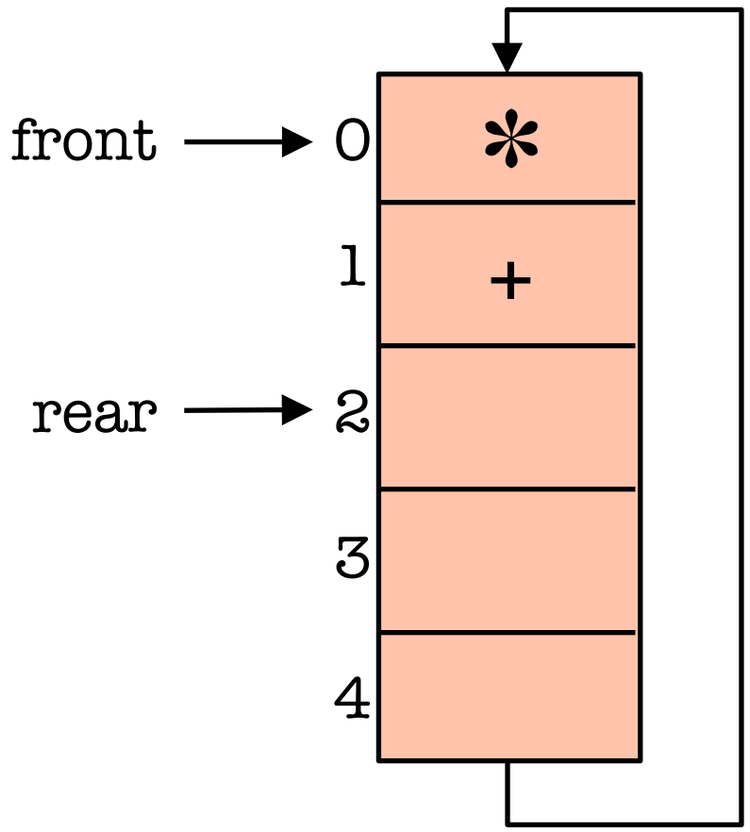
size = 2

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
q.offer("/");
```



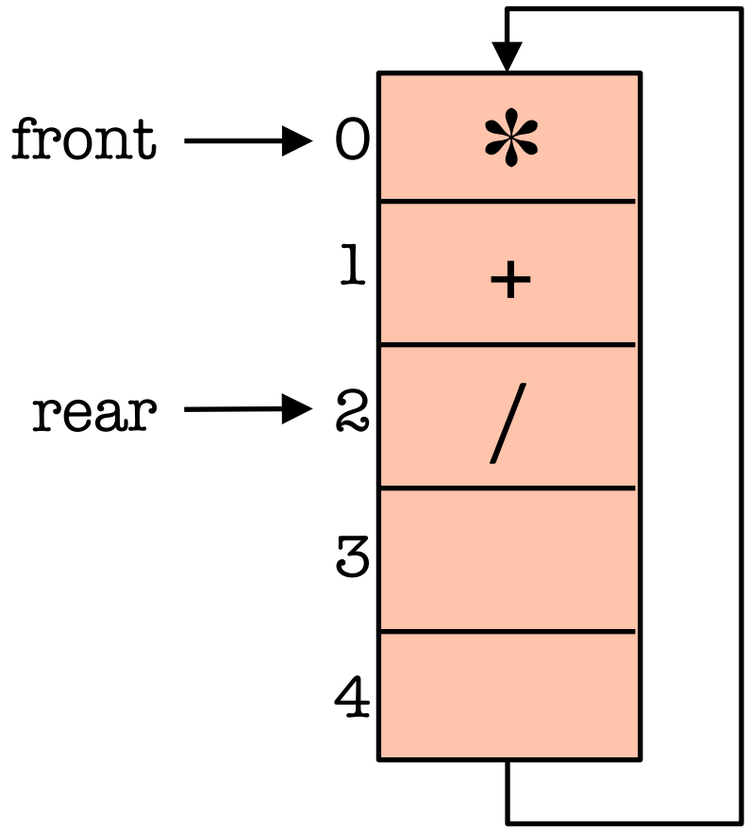
size = 3

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
```



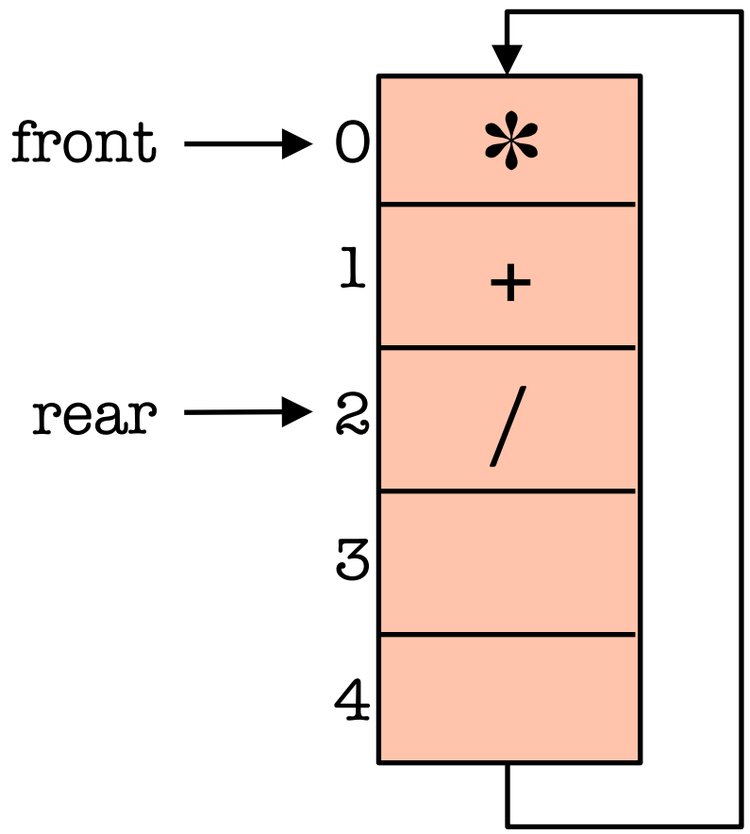
size = 3

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
q.offer("/");
q.offer("-");
```



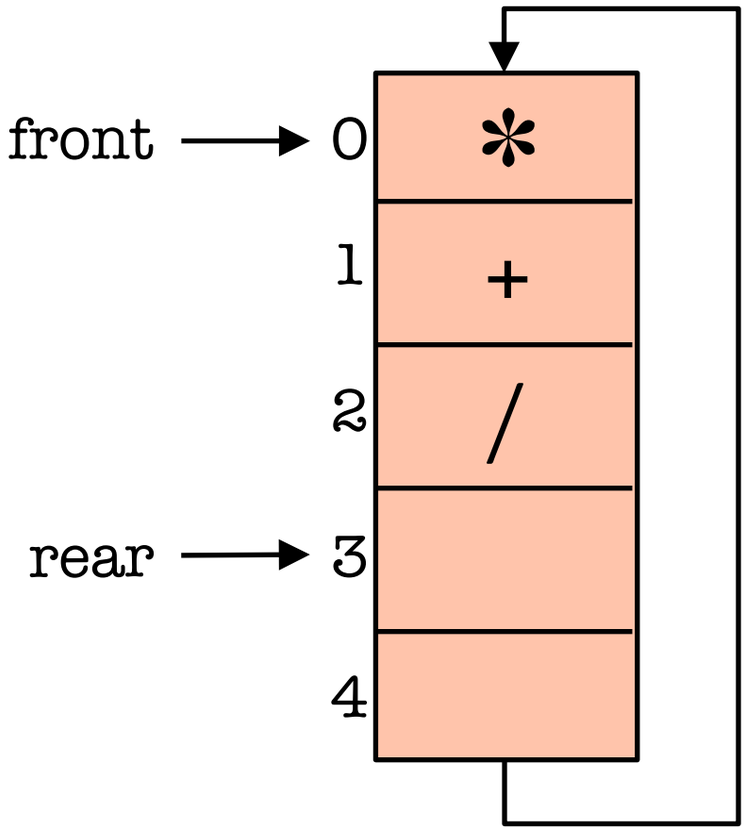
size = 3

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”-”);
```



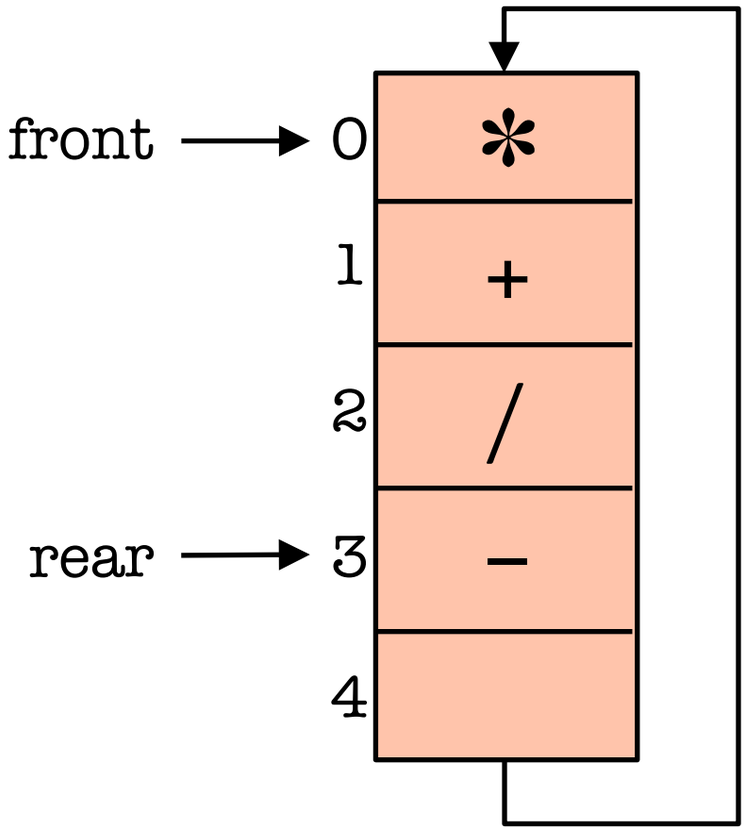
size = 4

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”-”);
```

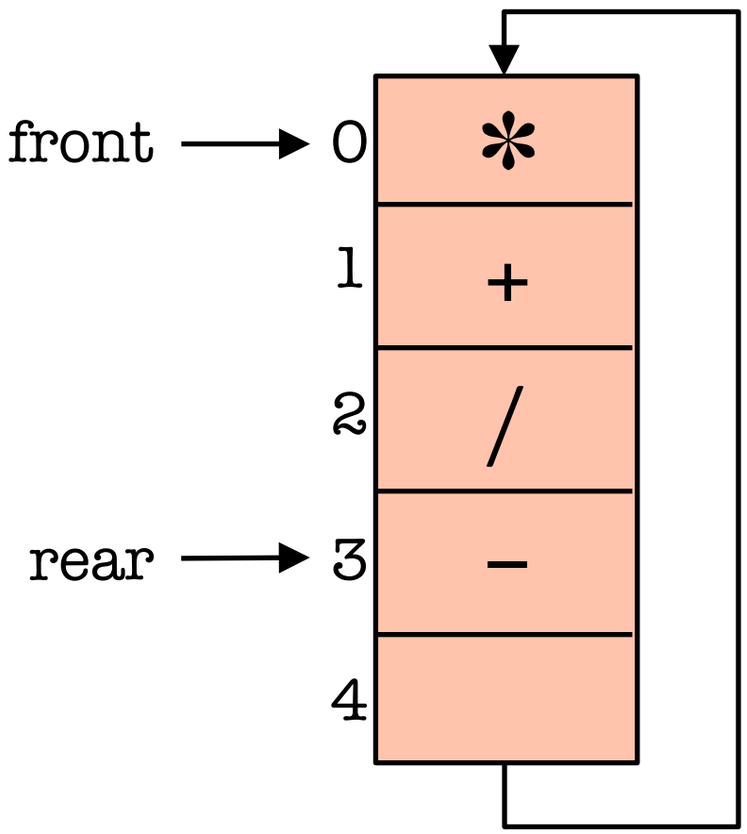


size = 4

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



size = 4

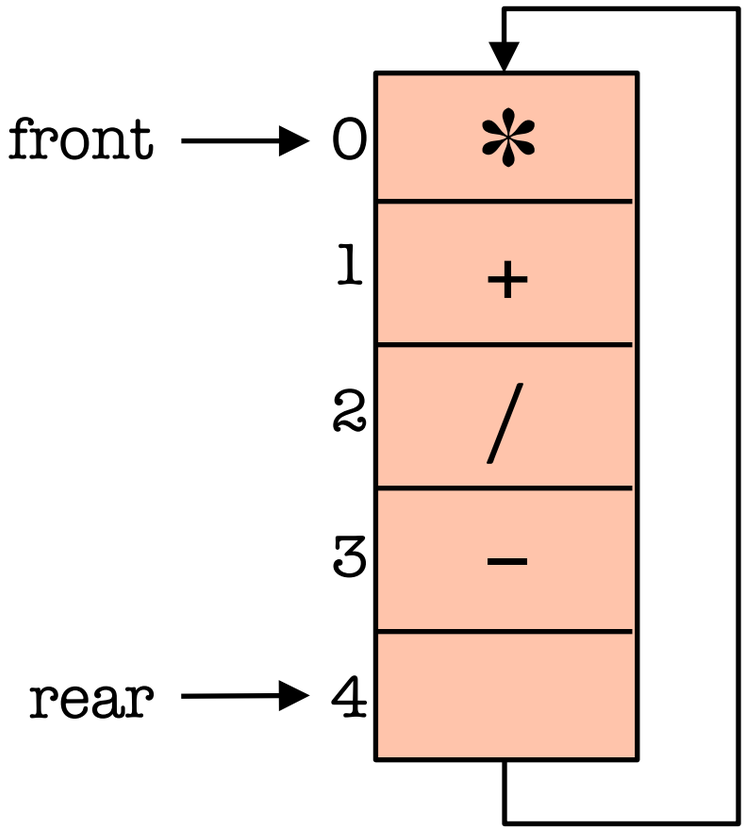
capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
q.offer("/");
q.offer("-");
q.offer("A");
```



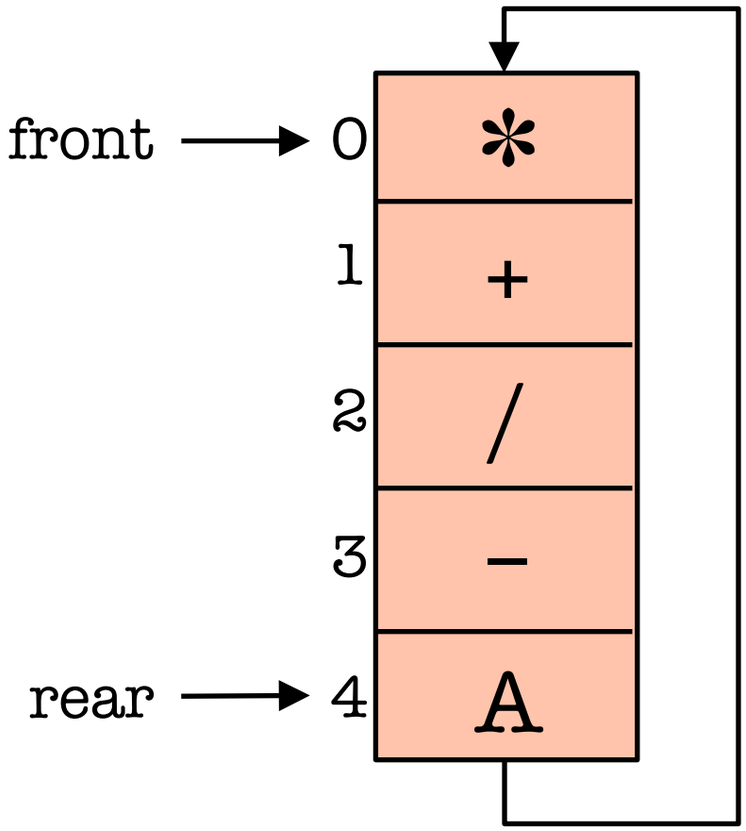
size = 5

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel

```
ArrayQueue q = new ArrayQueue(5);
q.offer("*");
q.offer("+");
q.offer("/");
q.offer("-");
q.offer("A");
```

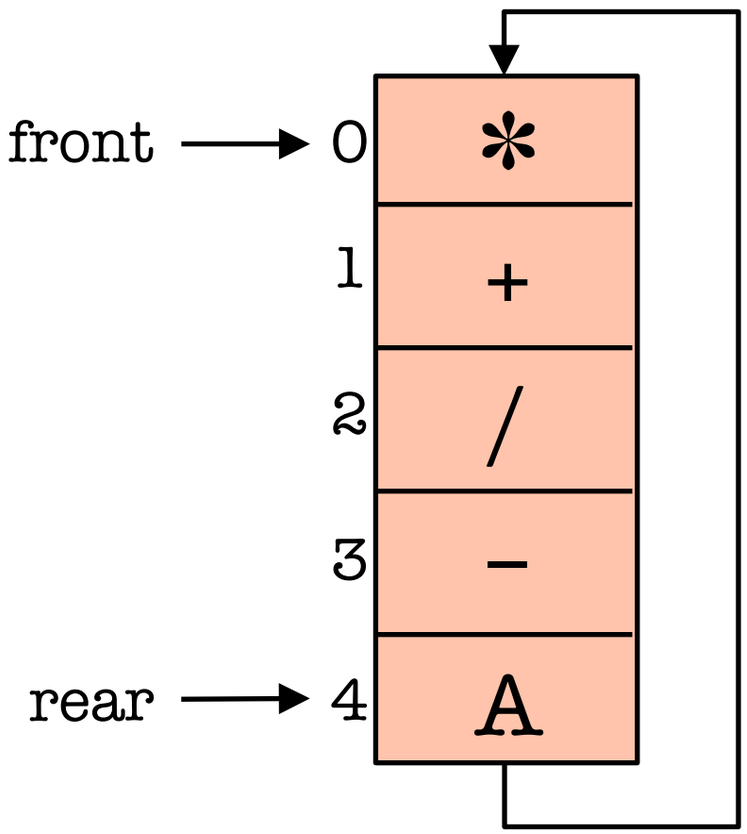


size = 5

capacity = 5

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



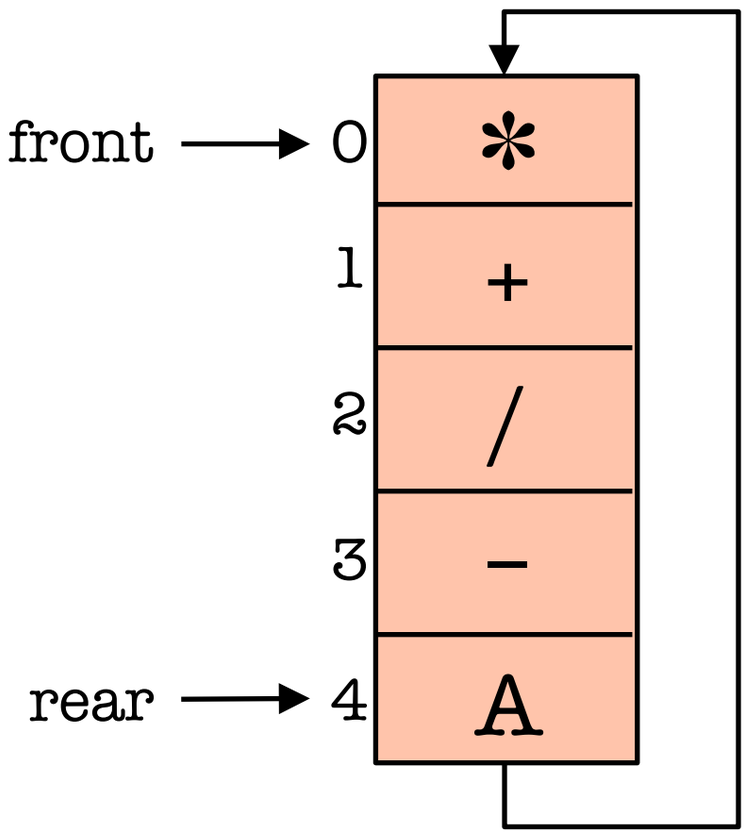
size = 5

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



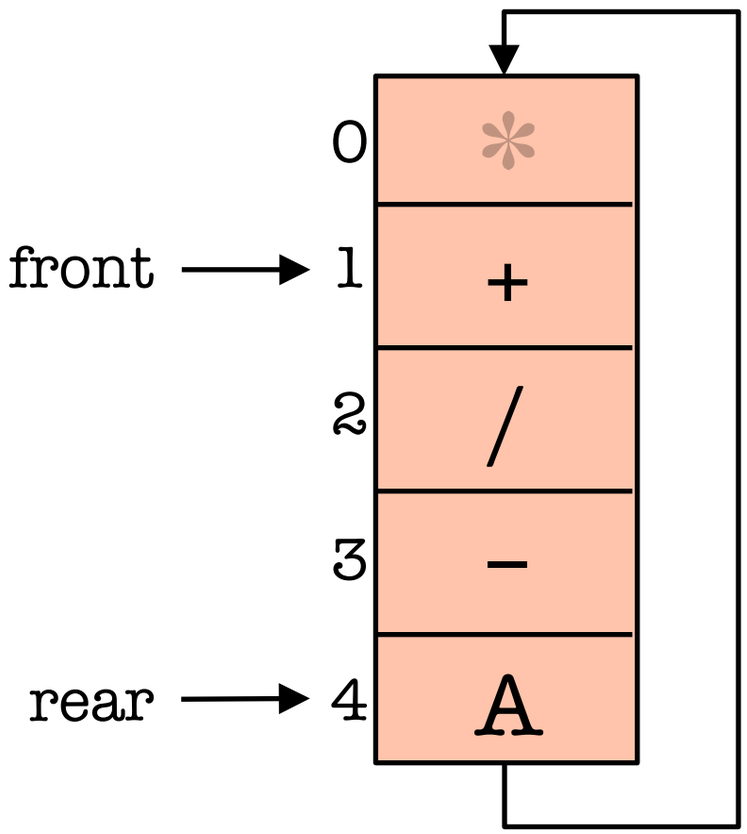
size = 5

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
```

```
public E poll() {
        if (size == 0)
                return null;
        E result = theData[front];
        front = (front + 1) % capacity;
        size--;
        return result;
}
```

Exempel



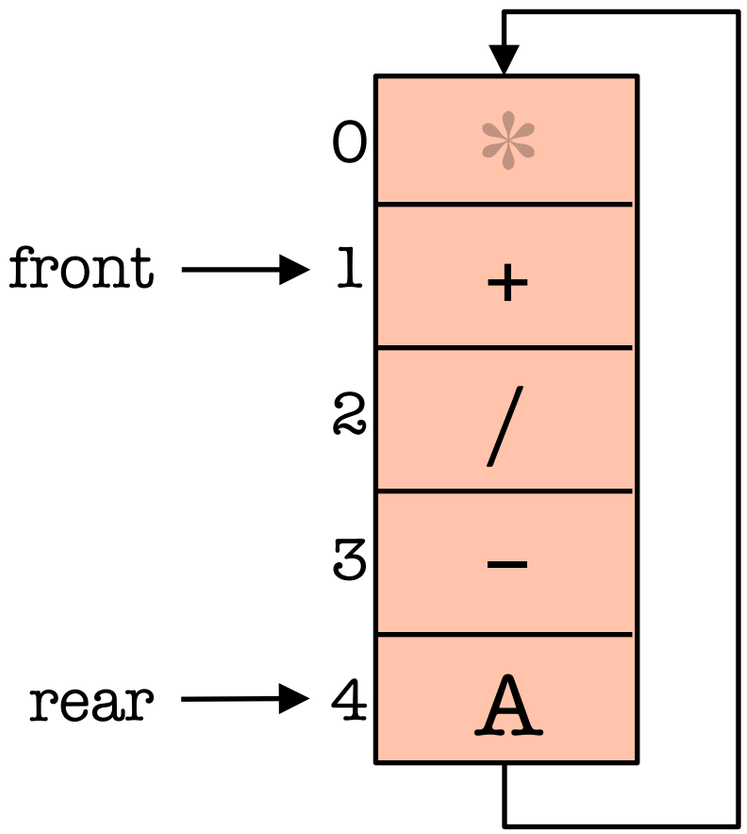
size = 4

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
```

```
public E poll() {
        if (size == 0)
                return null;
        E result = theData[front];
        front = (front + 1) % capacity;
        size--;
        return result;
}
```

Exempel



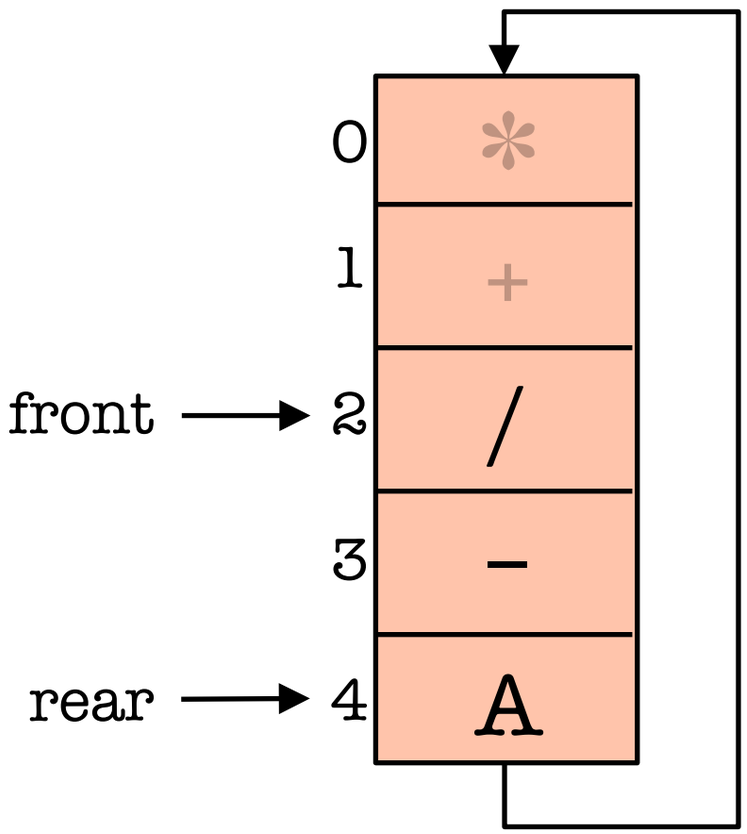
size = 4

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
```

```
public E poll() {
        if (size == 0)
                return null;
        E result = theData[front];
        front = (front + 1) % capacity;
        size--;
        return result;
}
```

Exempel



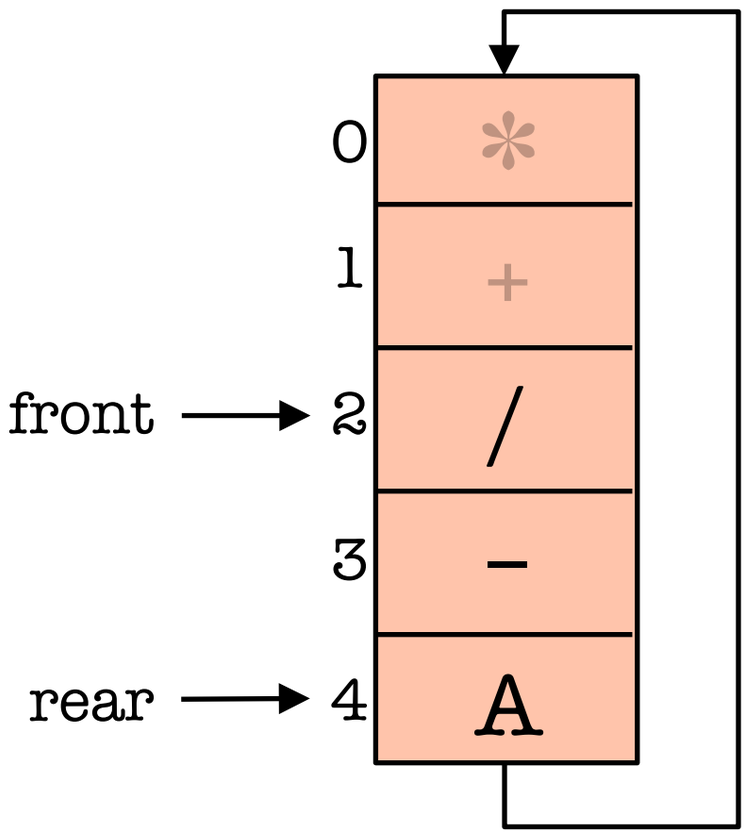
size = 3

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
```

```
public E poll() {
        if (size == 0)
                return null;
        E result = theData[front];
        front = (front + 1) % capacity;
        size--;
        return result;
}
```

Exempel



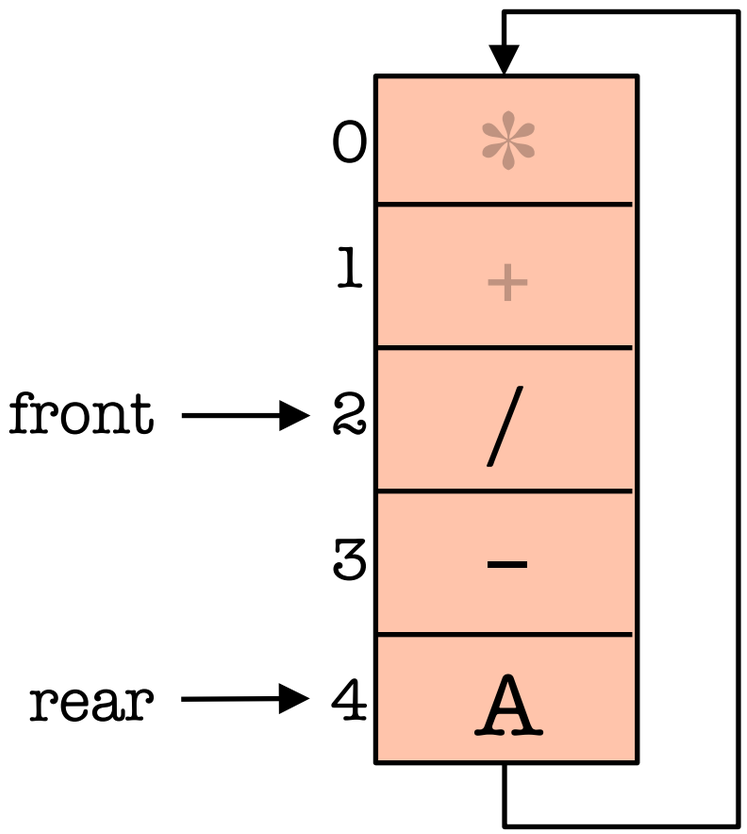
size = 3

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
```

```
public E poll() {
        if (size == 0)
                return null;
        E result = theData[front];
        front = (front + 1) % capacity;
        size--;
        return result;
}
```

Exempel



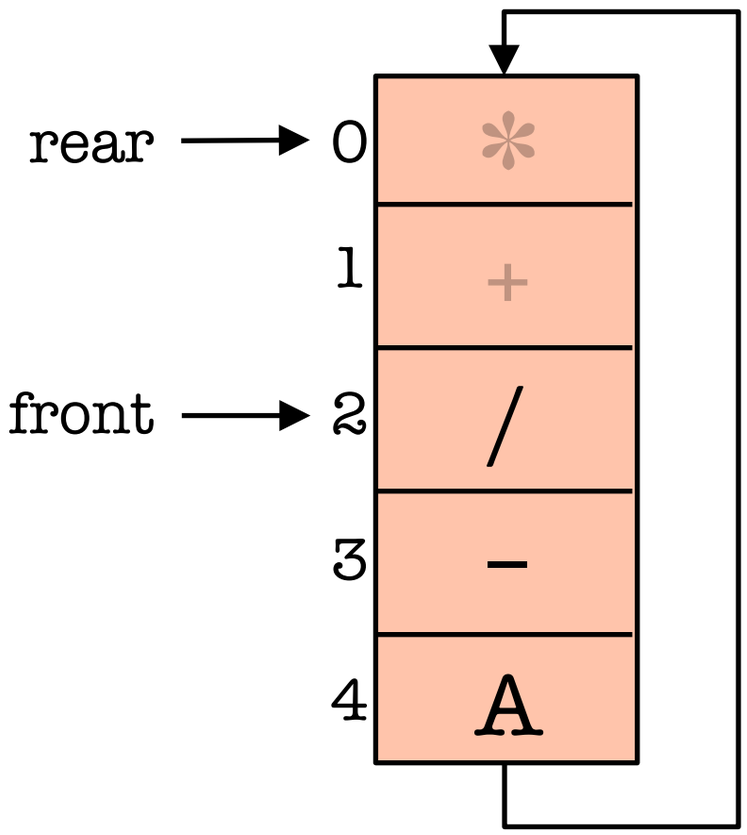
size = 3

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



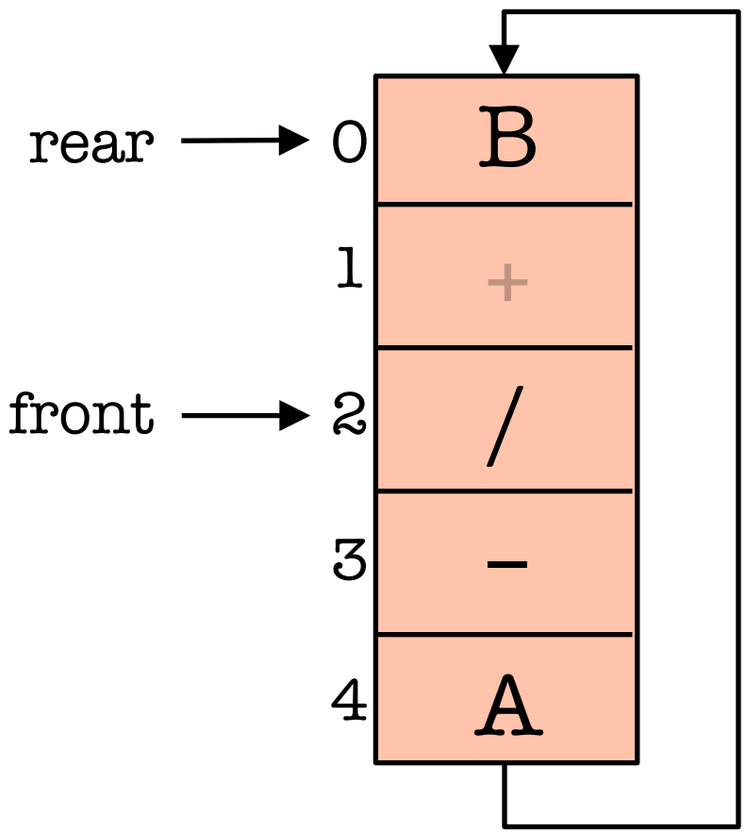
size = 4

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



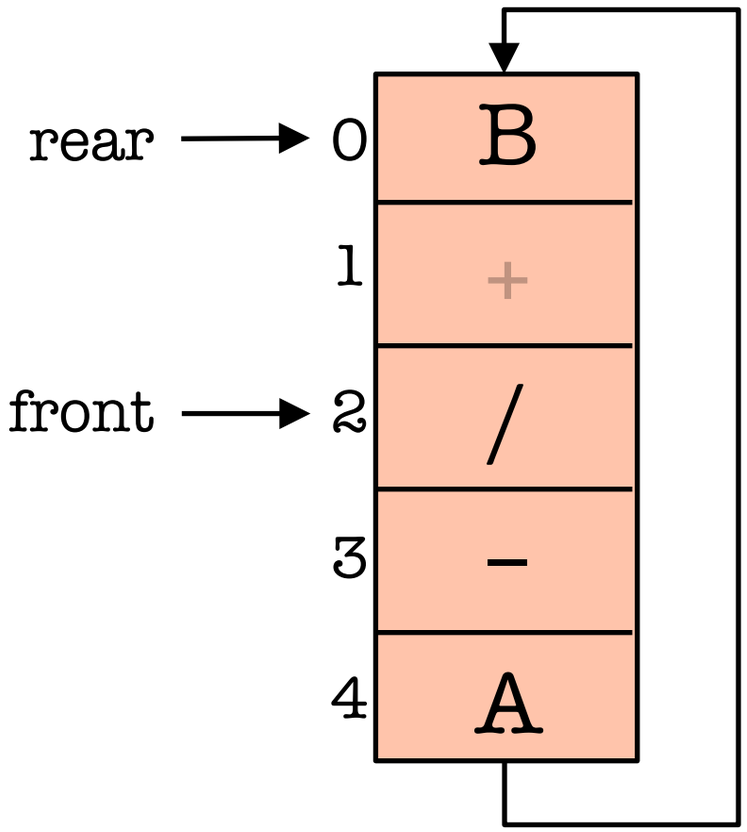
size = 4

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



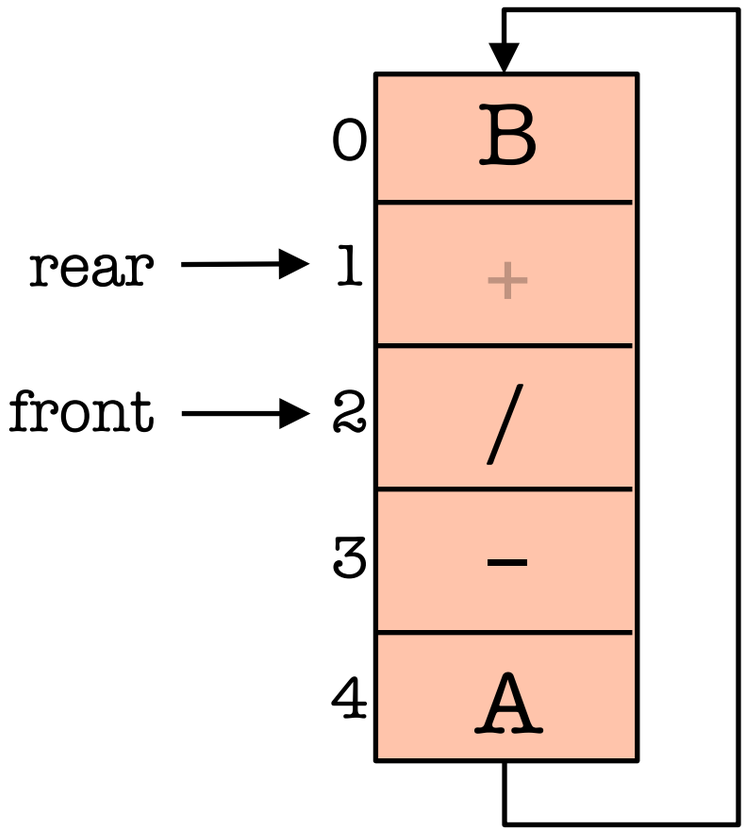
size = 4

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
q.offer(”C”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



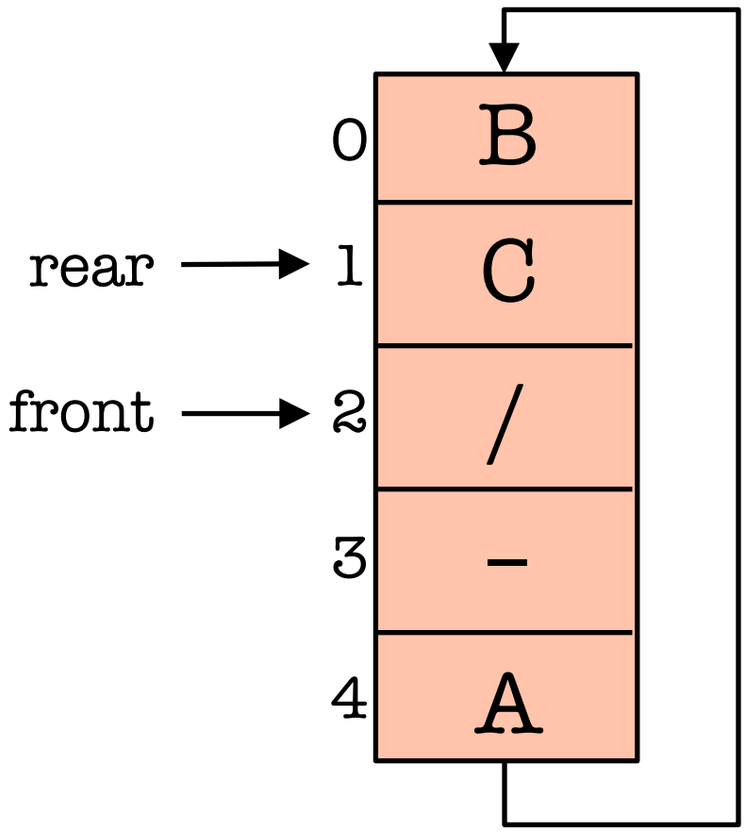
size = 5

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
q.offer(”C”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



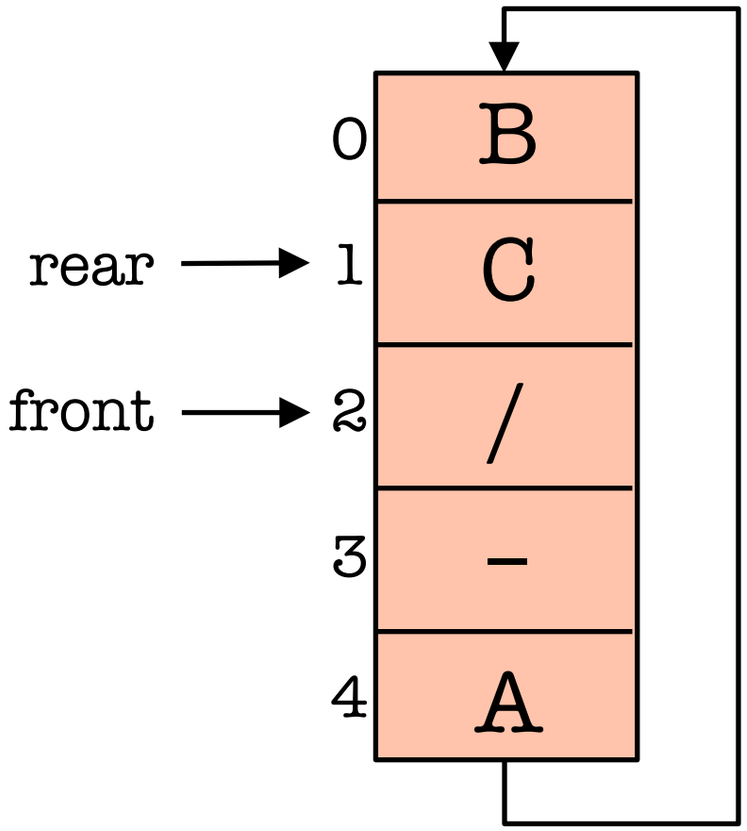
size = 5

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
q.offer(”C”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

Exempel



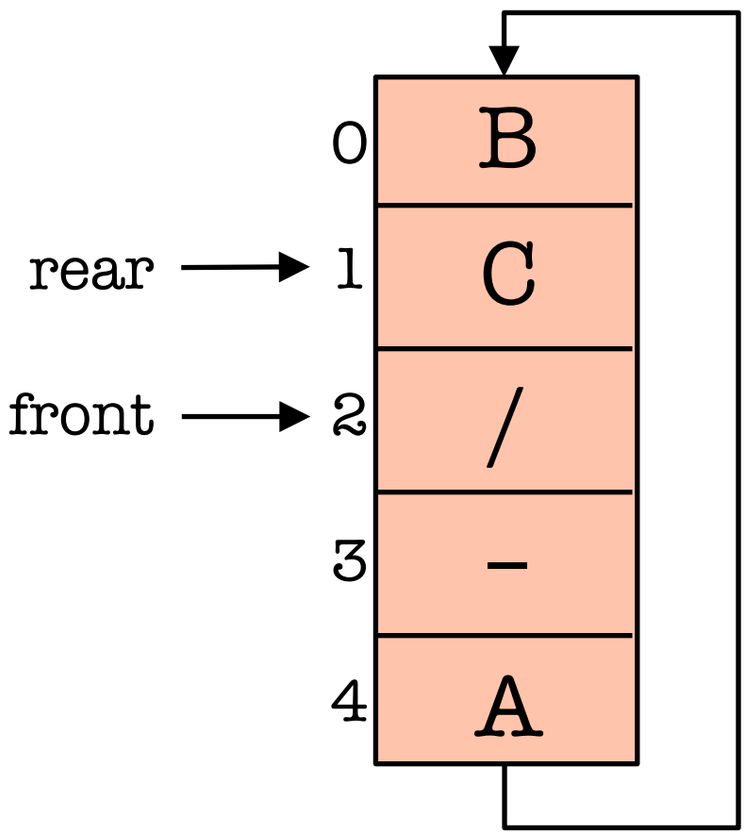
size = 5

capacity = 5

```
ArrayQueue q = new ArrayQueue(5);
q.offer(”*”);
q.offer(”+”);
q.offer(”/”);
q.offer(”–”);
q.offer(”A”);
next = q.poll();
next = q.poll();
q.offer(”B”);
q.offer(”C”);
q.offer(”D”);
```

```
public boolean offer(E item) {
        if (size == capacity)
                reallocate();
        size++;
        rear = (rear + 1) % capacity;
        theData[rear] = item;
        return true;
}
```

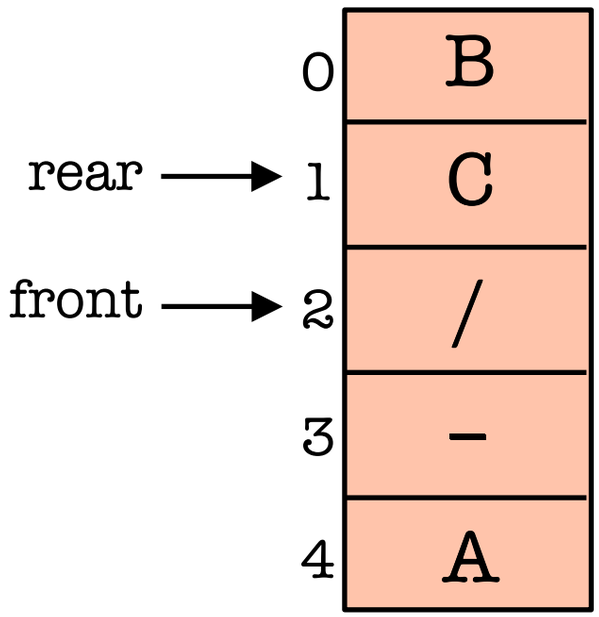
Exempel



size = 5

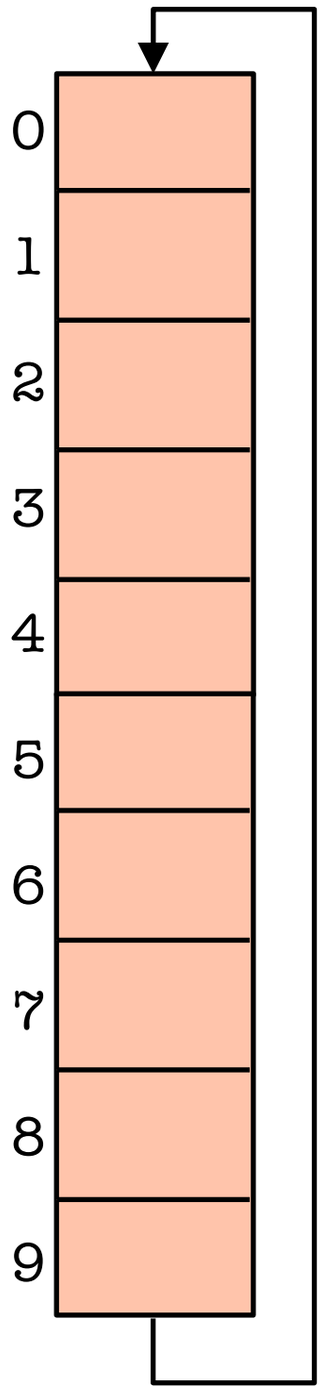
capacity = 5

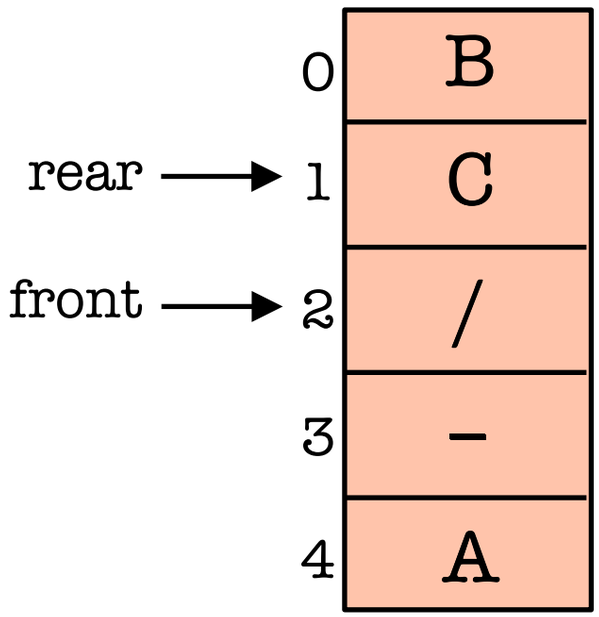
Exempel




```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

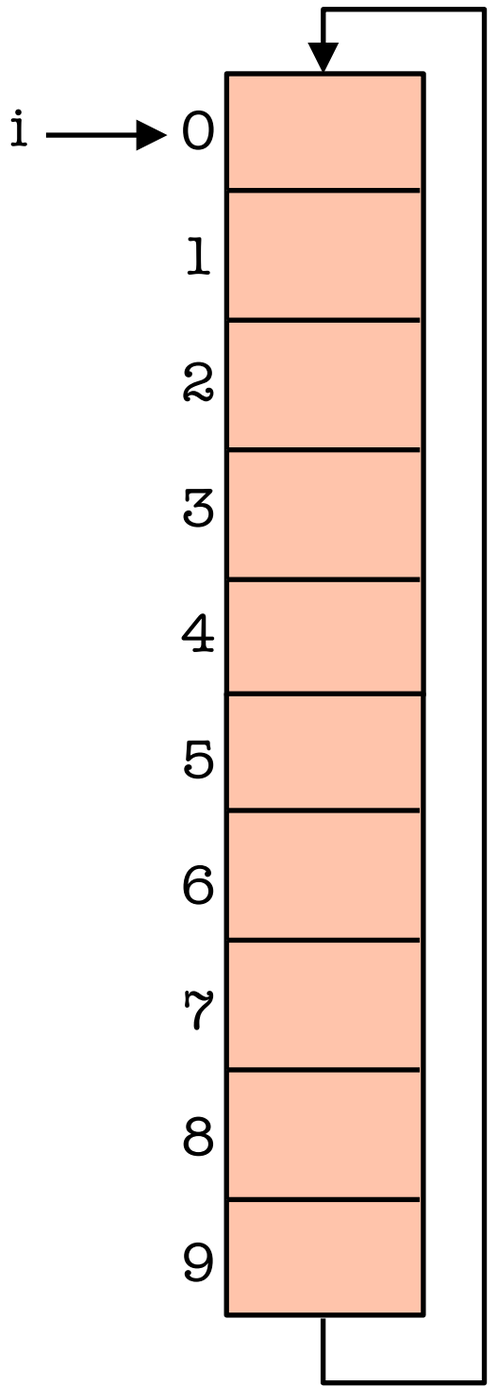
Exempel

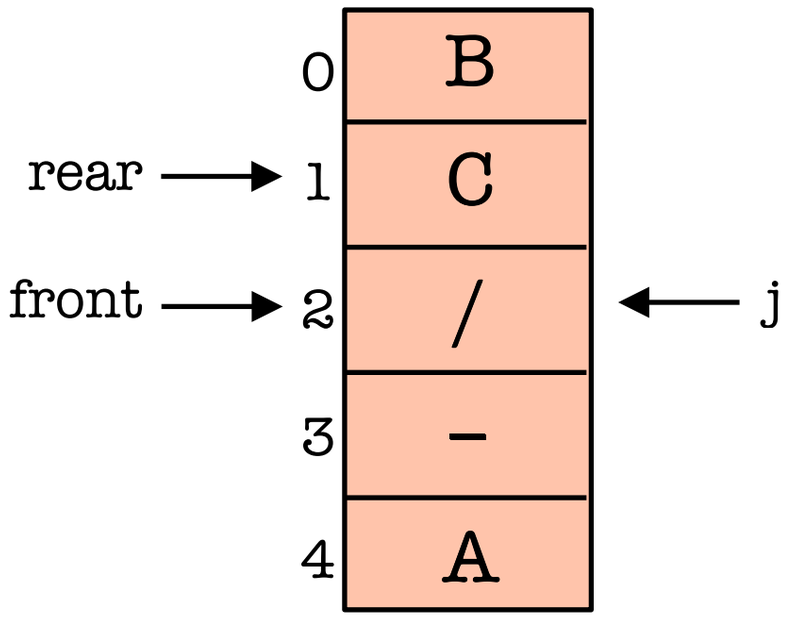





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

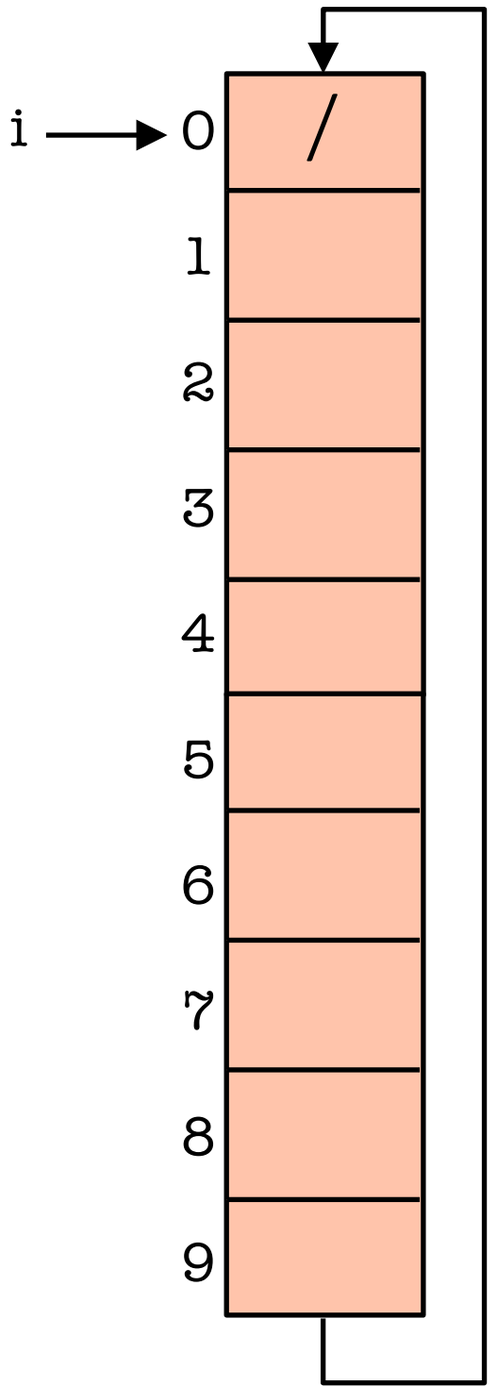
Exempel

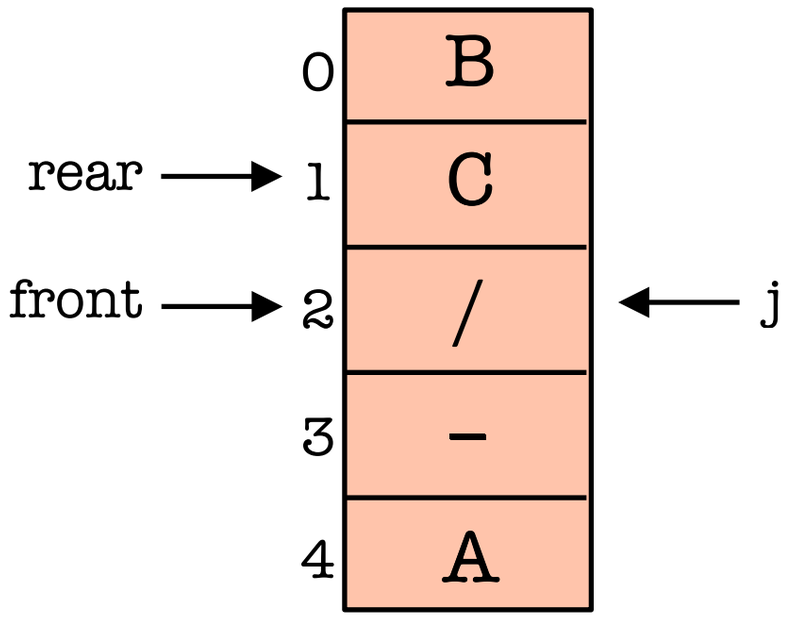





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

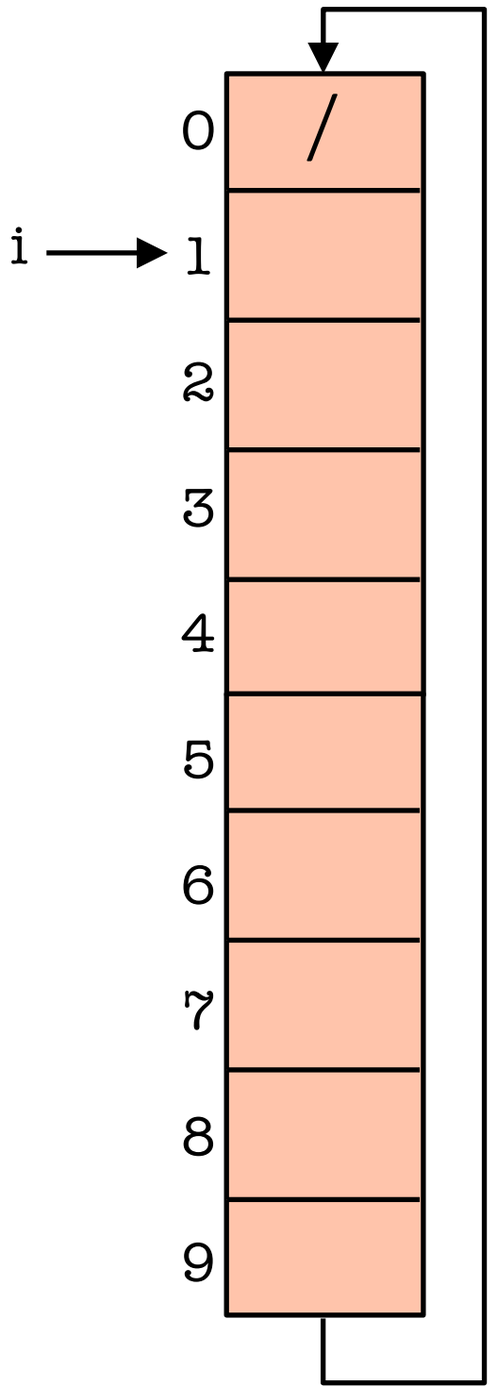
Exempel

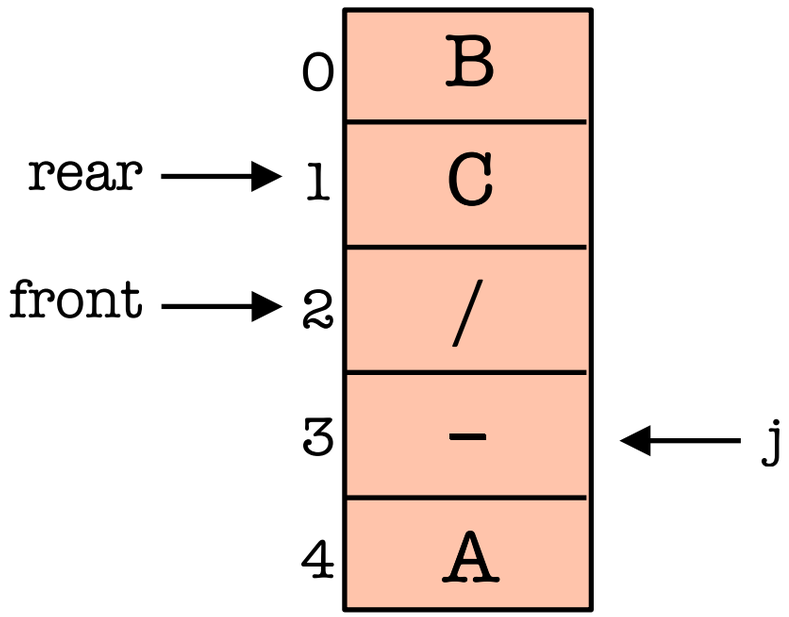





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

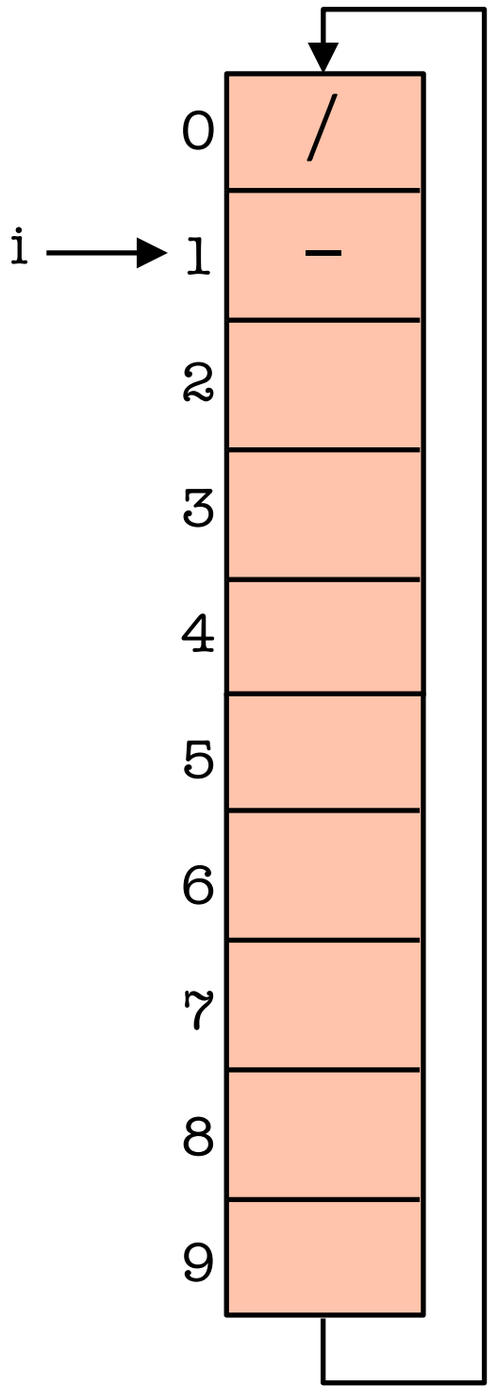
Exempel

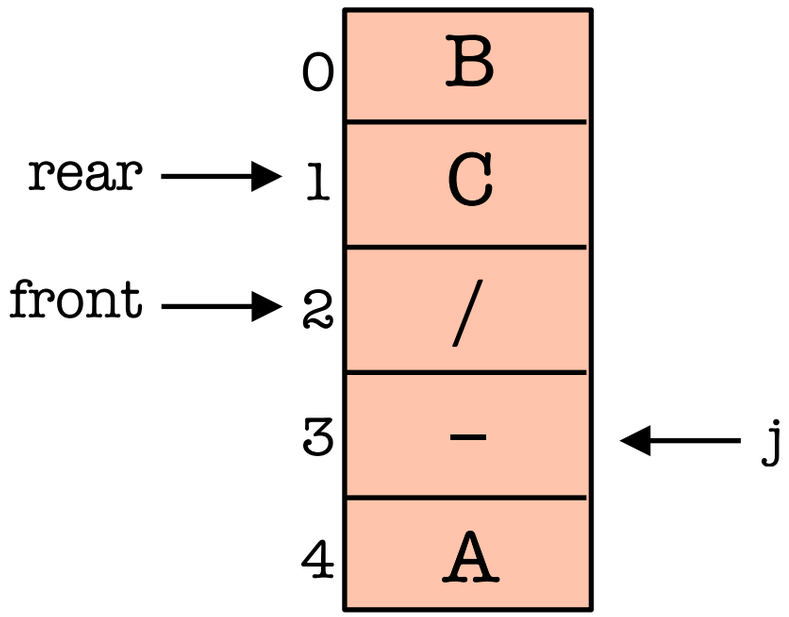





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

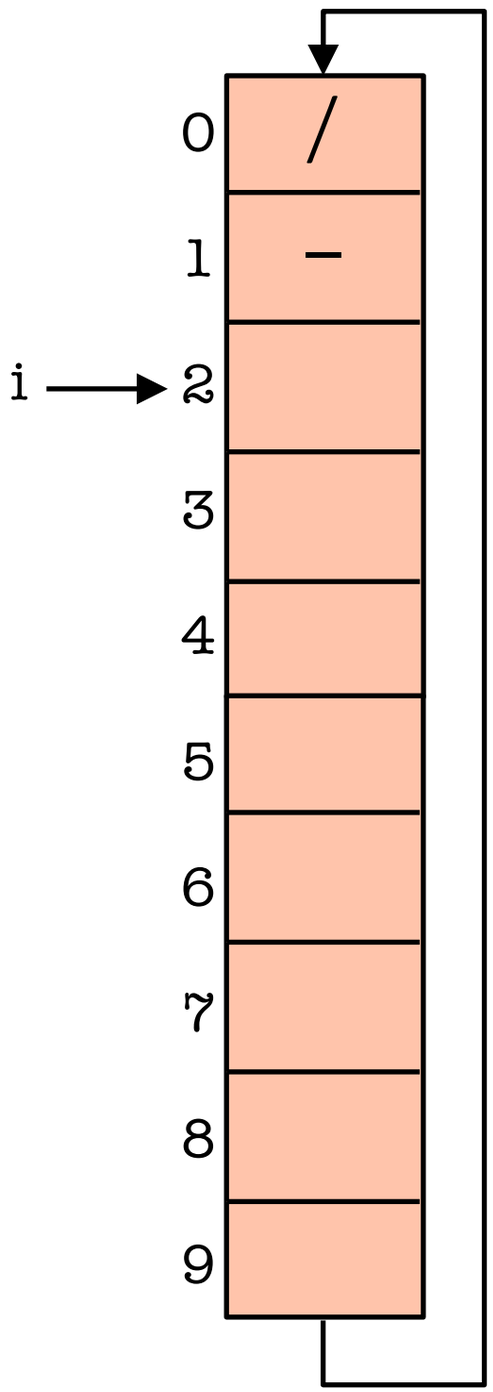
Exempel

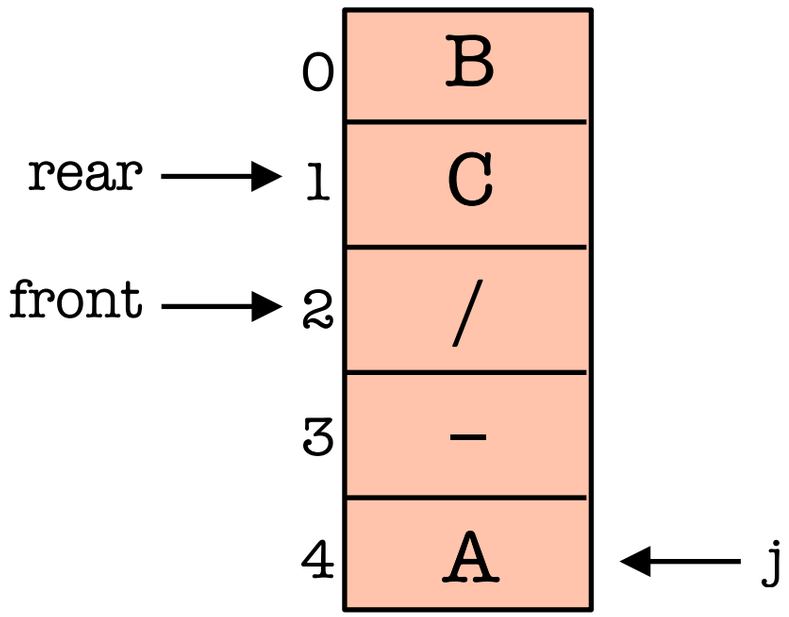





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

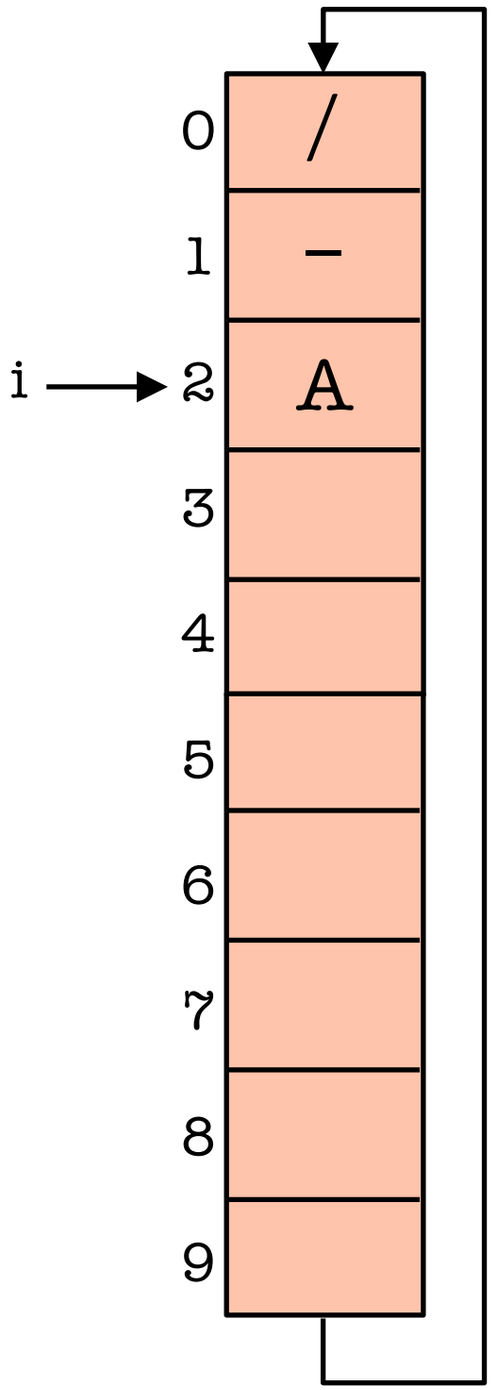
Exempel

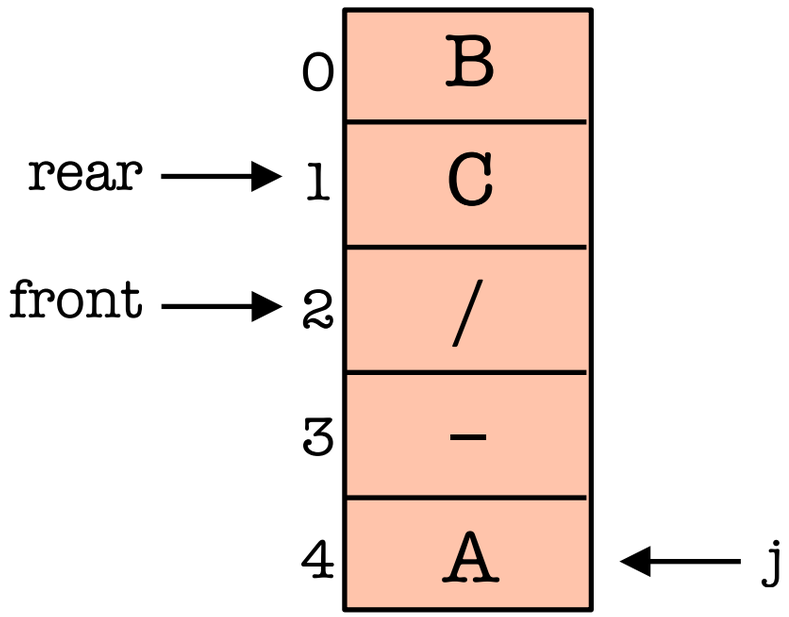





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

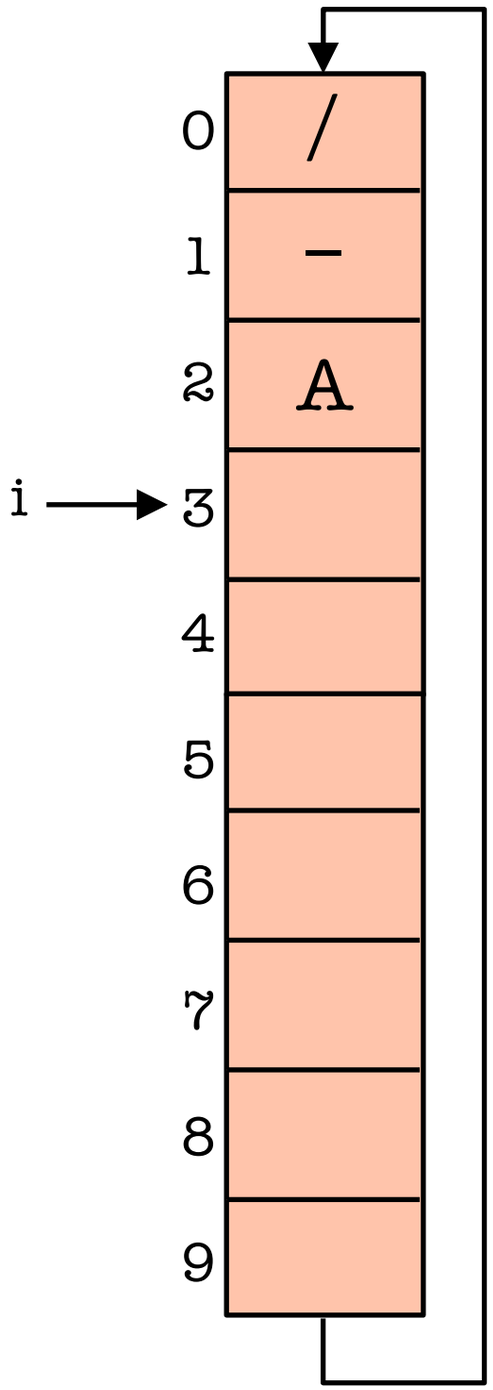
Exempel

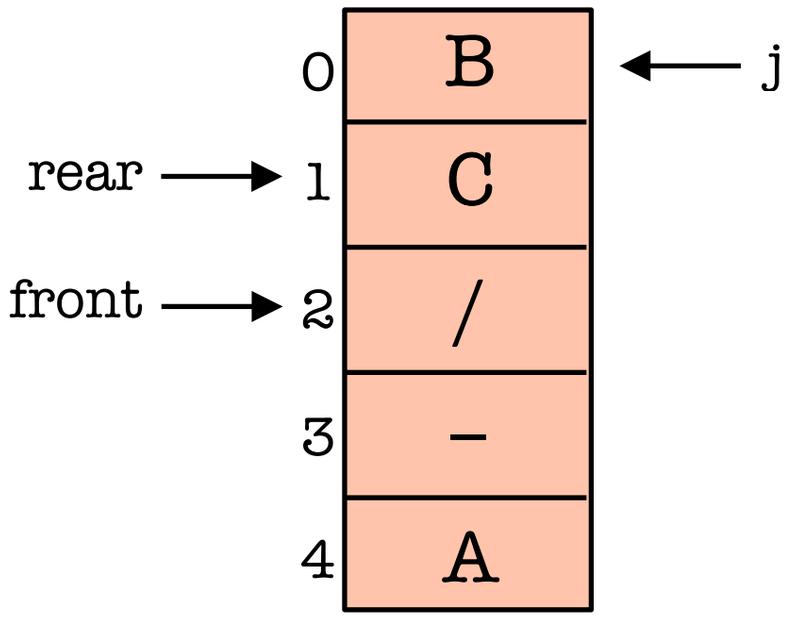





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

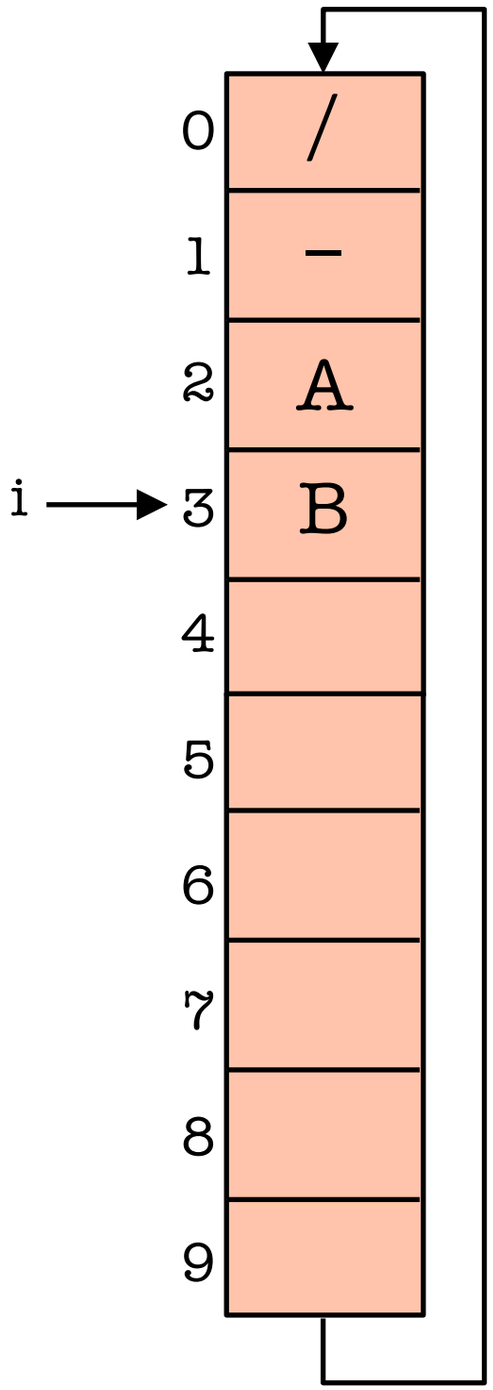
Exempel

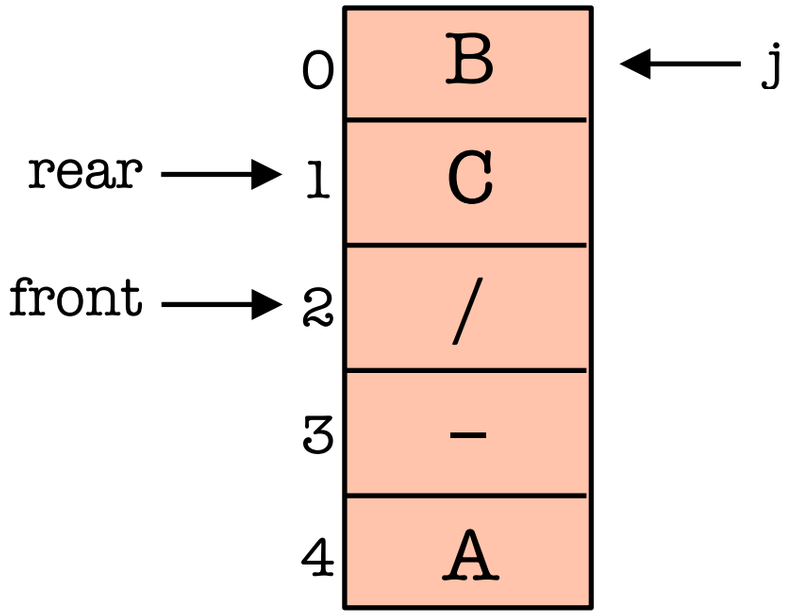





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

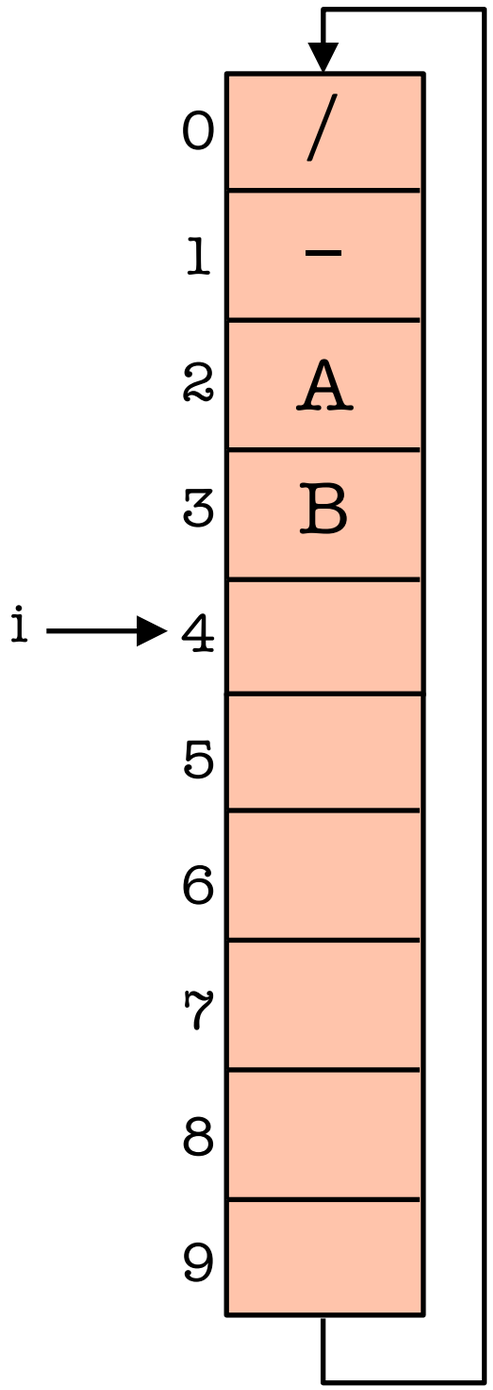
Exempel

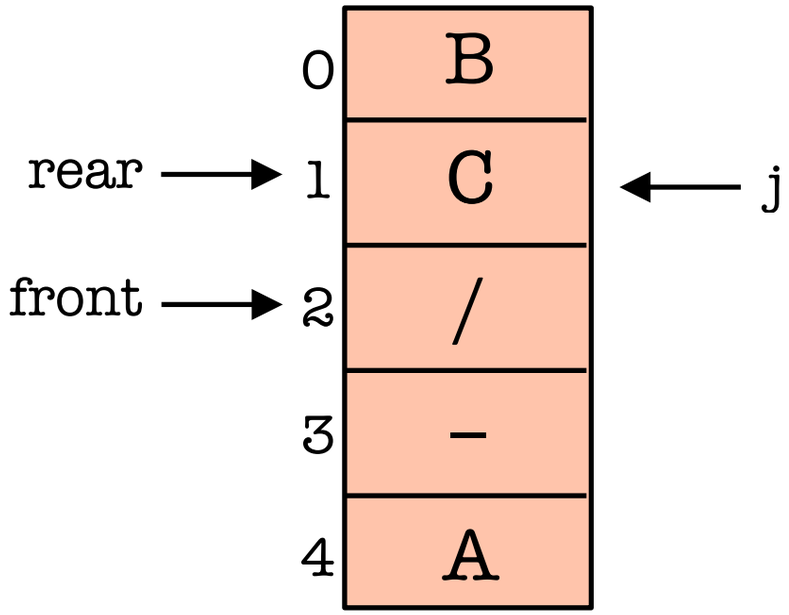





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

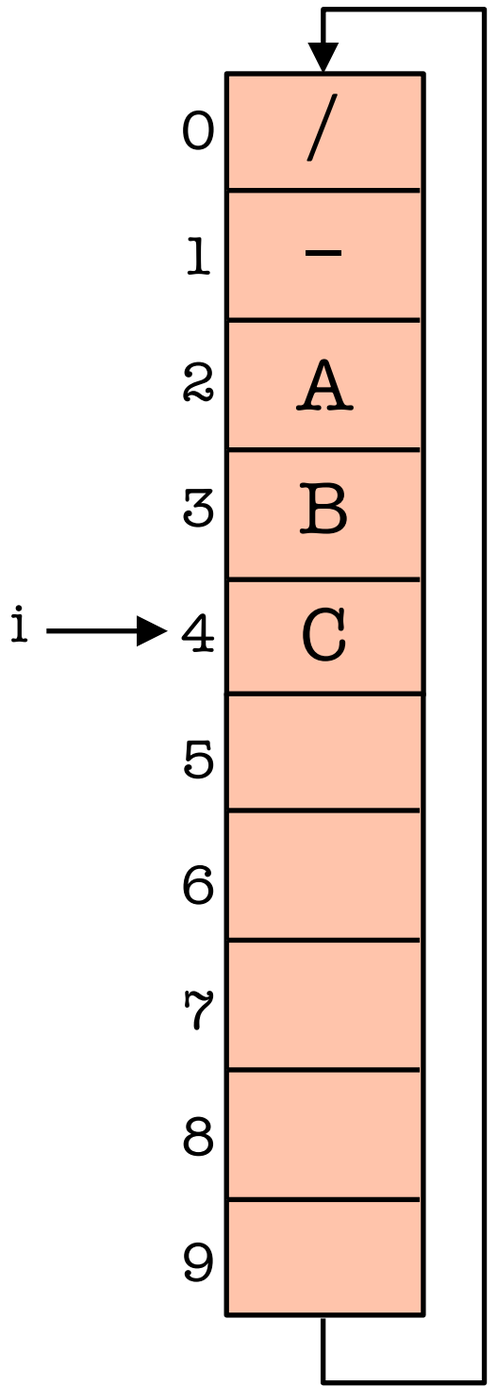
Exempel

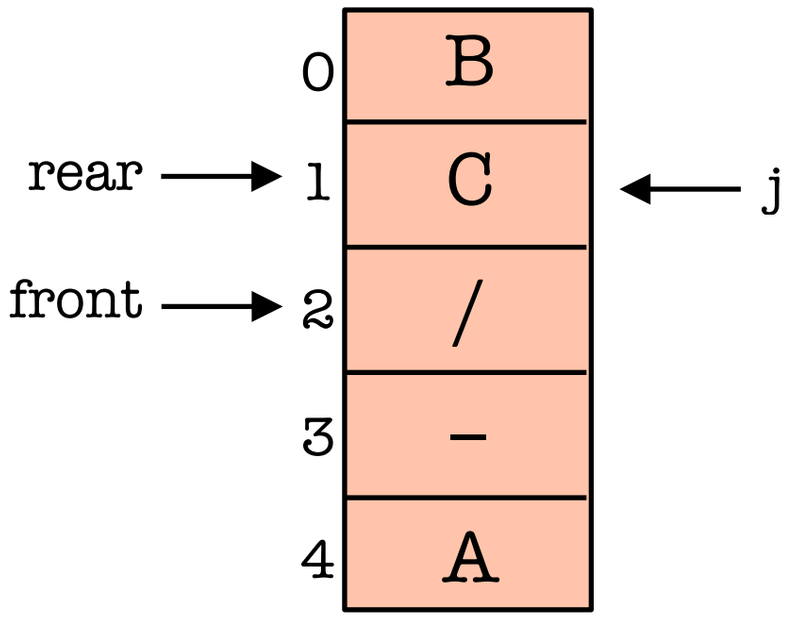





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

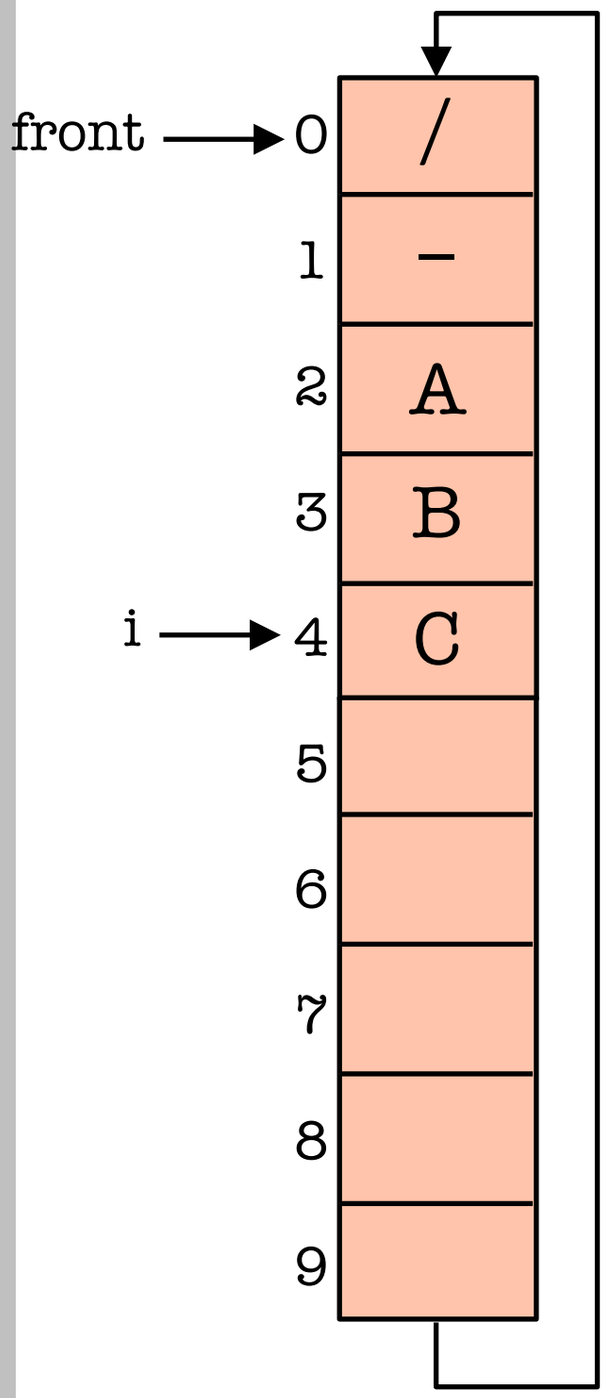
Exempel

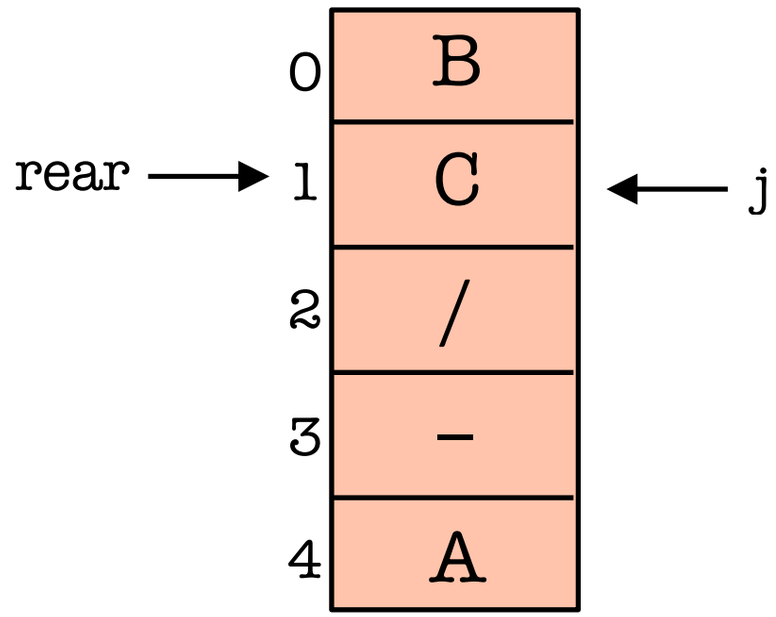





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

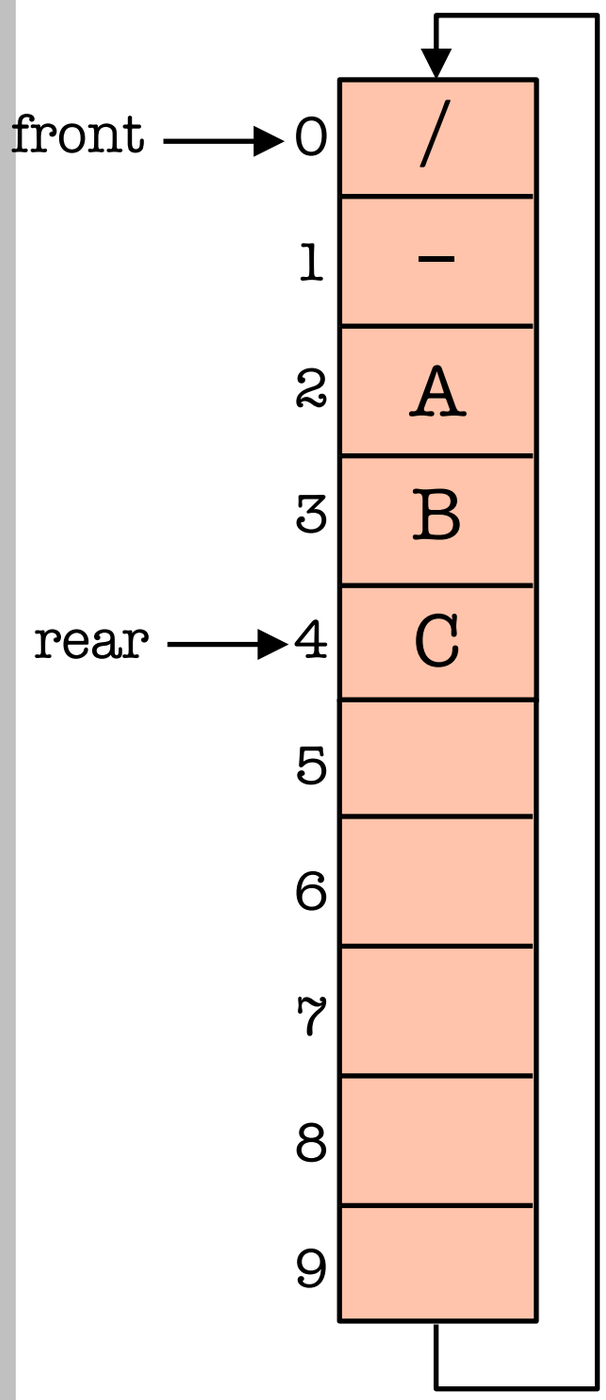
Exempel

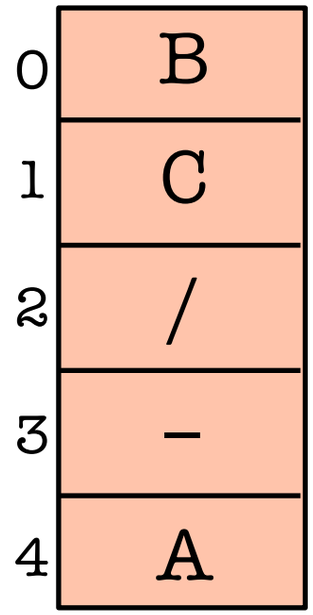





```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

Exempel






```
private void reallocate() {
      int newCapacity = 2 * capacity;
      E[] newData = (E[]) new Object[newCapacity];
      int j = front;
      for (int i = 0; i < size; i++) {
            newData[i] = theData[j];
            j = (j + 1) % capacity;
      }
      front = 0;
      rear = size - 1;
      capacity = newCapacity;
      theData = newData;
}
```

Deque (double-ended queue)

En deque är både en stack och en kö, samtidigt:

- `addFirst(E)`, `addLast(E)`
- `pollFirst()`, `pollLast()`

java.util.Deque är ett gränssnitt; det finns två färdiga implementeringar:

- `java.util.ArrayDeque<E>`
- `java.util.LinkedList<E>`

java.util.Deque

tabell 4.3, sid 218

Method	Behavior
`boolean offerFirst(E item)`	Inserts `item` at the front of the deque. Returns **`true`** if successful; returns **`false`** if the item could not be inserted.
`boolean offerLast(E item)`	Inserts `item` at the rear of the deque. Returns **`true`** if successful; returns **`false`** if the item could not be inserted.
`void addFirst(E item)`	Inserts `item` at the front of the deque. Throws an exception if the item could not be inserted.
`void addLast(E item)`	Inserts `item` at the rear of the deque. Throws an exception if the item could not be inserted.
`E pollFirst()`	Removes the entry at the front of the deque and returns it; returns **`null`** if the deque is empty.
`E pollLast()`	Removes the entry at the rear of the deque and returns it; returns **`null`** if the deque is empty.
`E removeFirst()`	Removes the entry at the front of the deque and returns it if the deque is not empty. If the deque is empty, throws a `NoSuchElementException`.
`E removeLast()`	Removes the item at the rear of the deque and returns it. If the deque is empty, throws a `NoSuchElementException`.
`E peekFirst()`	Returns the entry at the front of the deque without removing it; returns **`null`** if the deque is empty.
`E peekLast()`	Returns the item at the rear of the deque without removing it; returns **`null`** if the deque is empty.
`E getFirst()`	Returns the entry at the front of the deque without removing it. If the deque is empty, throws a `NoSuchElementException`.
`E getLast()`	Returns the item at the rear of the deque without removing it. If the deque is empty, throws a `NoSuchElementException`.
`boolean removeFirstOccurrence(Object item)`	Removes the first occurrence of item in the deque. Returns **`true`** if the item was removed.
`boolean removeLastOccurrence(Object item)`	Removes the last occurrence of item in the deque. Returns **`true`** if the item was removed.
`Iterator<E> iterator()`	Returns an iterator to the elements of this deque in the proper sequence.
`Iterator<E> descendingIterator()`	Returns an iterator to the elements of this deque in reverse sequential order.

Tabell 4.4, sid 219

Exempel på deque-metoder:

Deque Method	Deque d	Effect
`d.offerFirst('b')`	b	'b' inserted at front
`d.offerLast('y')`	by	'y' inserted at rear
`d.addLast('z')`	byz	'z' inserted at rear
`d.addFirst('a')`	abyz	'a' inserted at front
`d.peekFirst()`	abyz	Returns 'a'
`d.peekLast()`	abyz	Returns 'z'
`d.pollLast()`	aby	Removes 'z'
`d.pollFirst()`	by	Removes 'a'