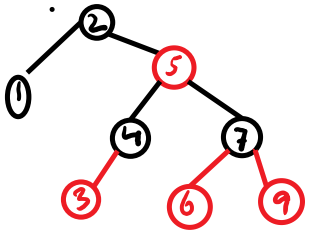


CS 201 Data Structure II (L2 / L5)

Muhammad Qasim Pasta

qasim.pasta@sse.habib.edu.pk

Class Norms:

L2

Mark your attendance using biometric machines

- Chit-chat during the lectures – Don’t
- Receiving calls – leave the class
- Ask questions – Do’s
- Sleeping in the class – twice in a month
- Coming late – Sometimes
- Leaving the class and coming back – without causing disturbance
- Request for early leaves – 15 minutes , once in a month
- Working on laptop – for taking notes
- Checking phones – 3 to 5 times
- …

Class Norms:

L5

Mark your attendance using biometric machines

- Chit-chat during the lectures – Don’t
- Receiving calls – leave the class – Do’s
- Ask questions – Do’s
- Sleeping in the class – Do’s
- Coming late – Do’s
- Leaving the class and coming back – non disruptive manner
- Request for early leaves – not often, not more than 15 minutes
- Working on laptop – only taking for notes, not to solve assignment
- and checking phones – 2 to 4 times
- …

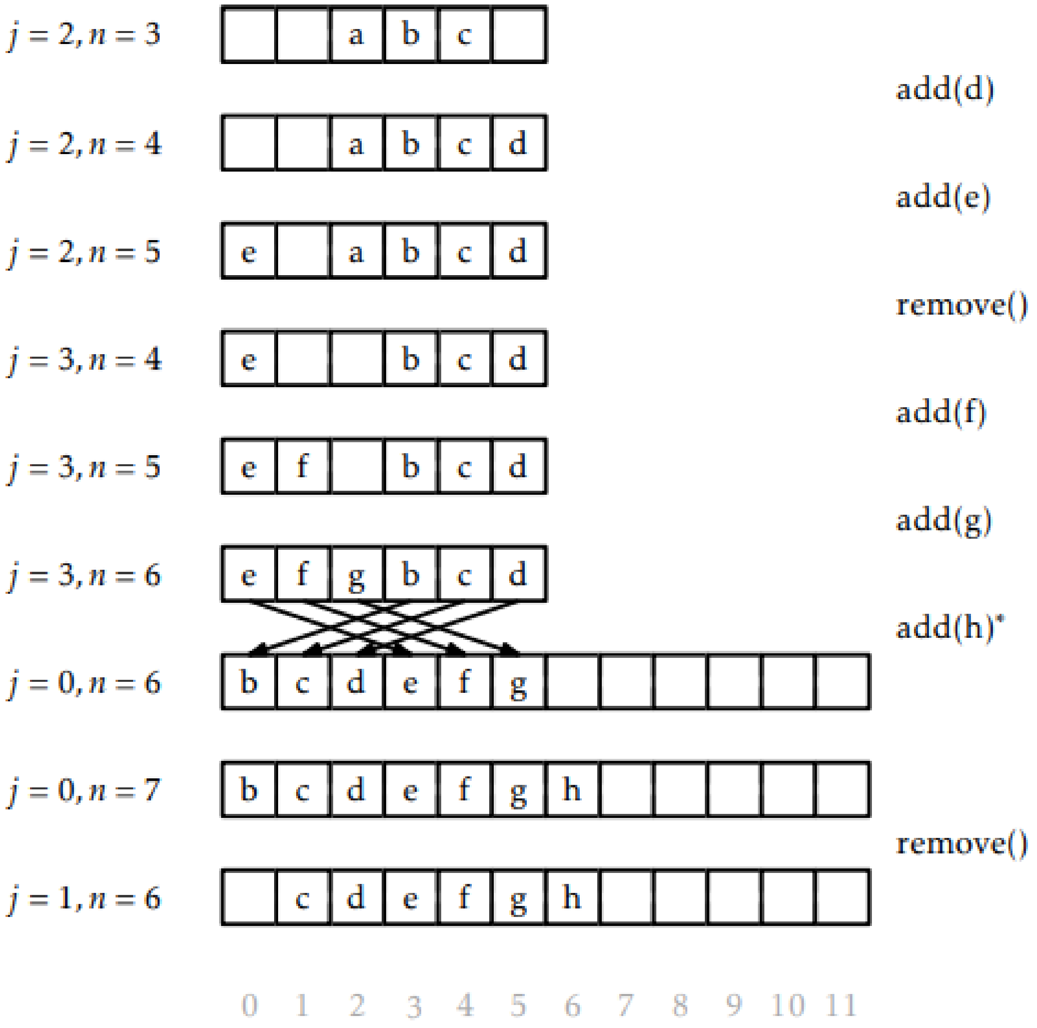
ADT in textbook

ADT in the textbook	Purpose
2.1: ArrayStack	List ADT using array (resizable)
2.3: ArrayQueue	Queue using Array (resizable)
2.4: ArrayDeque	Queue using array with efficient add and remove
2.5: DualArrayDeque	Same as ArrayDequeue with two stacks
2.6: RootishArrayStack	Store n elements using $\sqrt{n}$ arrays

ArrayQueue

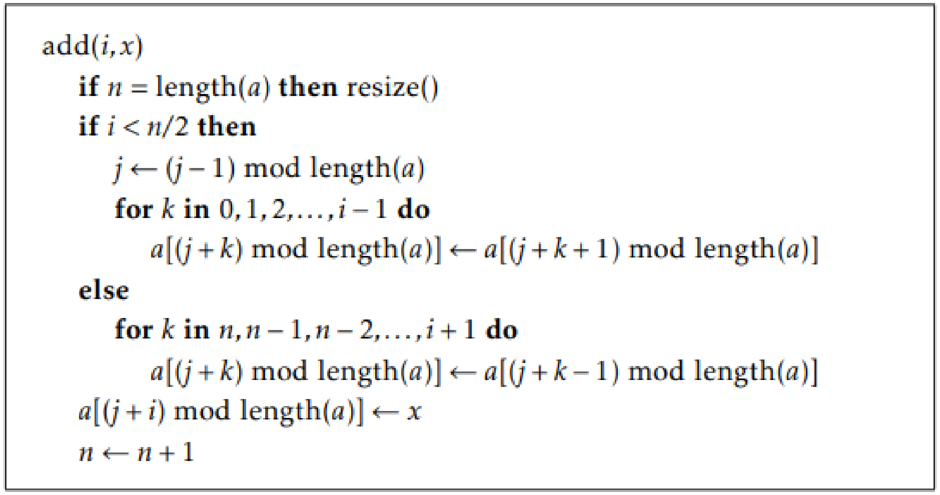
```
add($x$)
    if $n + 1 > \text{length}(a)$ then resize()
    $a[(j + n) \bmod \text{length}(a)] \leftarrow x$
    $n \leftarrow n + 1$
    return $true$
```

```
remove()
    $x \leftarrow a[j]$
    $j \leftarrow (j + 1) \bmod \text{length}(a)$
    $n \leftarrow n - 1$
    if $\text{length}(a) \geq 3 \cdot n$ then resize()
    return $x$
```

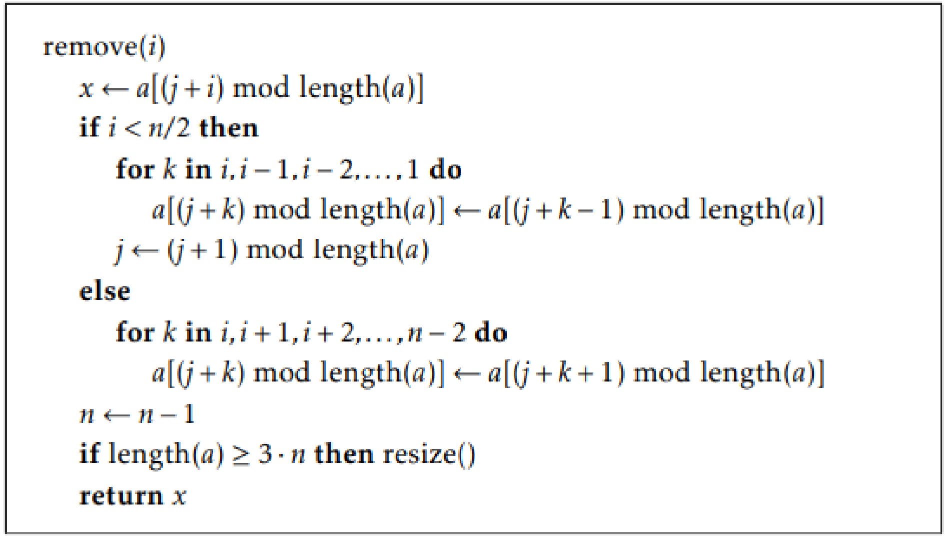


$j = 2, n = 3$			a	b	c								
$j = 2, n = 4$			a	b	c	d							add(d)
$j = 2, n = 5$	e		a	b	c	d							add(e)
$j = 3, n = 4$	e			b	c	d							remove()
$j = 3, n = 5$	e	f		b	c	d							add(f)
$j = 3, n = 6$	e	f	g	b	c	d							add(g)
$j = 0, n = 6$	b	c	d	e	f	g							add(h)*
$j = 0, n = 7$	b	c	d	e	f	g	h						
$j = 1, n = 6$		c	d	e	f	g	h						remove()
	0	1	2	3	4	5	6	7	8	9	10	11	

ArrayDequeue



```
add($i,x$)
    if $n = \text{length}(a)$ then resize()
    if $i < n/2$ then
        $j \leftarrow (j-1) \bmod \text{length}(a)$
        for $k$ in $0,1,2,\ldots,i-1$ do
            $a[(j+k) \bmod \text{length}(a)] \leftarrow a[(j+k+1) \bmod \text{length}(a)]$
    else
        for $k$ in $n,n-1,n-2,\ldots,i+1$ do
            $a[(j+k) \bmod \text{length}(a)] \leftarrow a[(j+k-1) \bmod \text{length}(a)]$
    $a[(j+i) \bmod \text{length}(a)] \leftarrow x$
    $n \leftarrow n+1$
```



```
remove($i$)
    $x \leftarrow a[(j+i) \bmod \text{length}(a)]$
    if $i < n/2$ then
        for $k$ in $i,i-1,i-2,\ldots,1$ do
            $a[(j+k) \bmod \text{length}(a)] \leftarrow a[(j+k-1) \bmod \text{length}(a)]$
        $j \leftarrow (j+1) \bmod \text{length}(a)$
    else
        for $k$ in $i,i+1,i+2,\ldots,n-2$ do
            $a[(j+k) \bmod \text{length}(a)] \leftarrow a[(j+k+1) \bmod \text{length}(a)]$
    $n \leftarrow n-1$
    if $\text{length}(a) \geq 3 \cdot n$ then resize()
    return $x$
```

ArrayDeque: Fast Deque Operations Using an Array §2.4

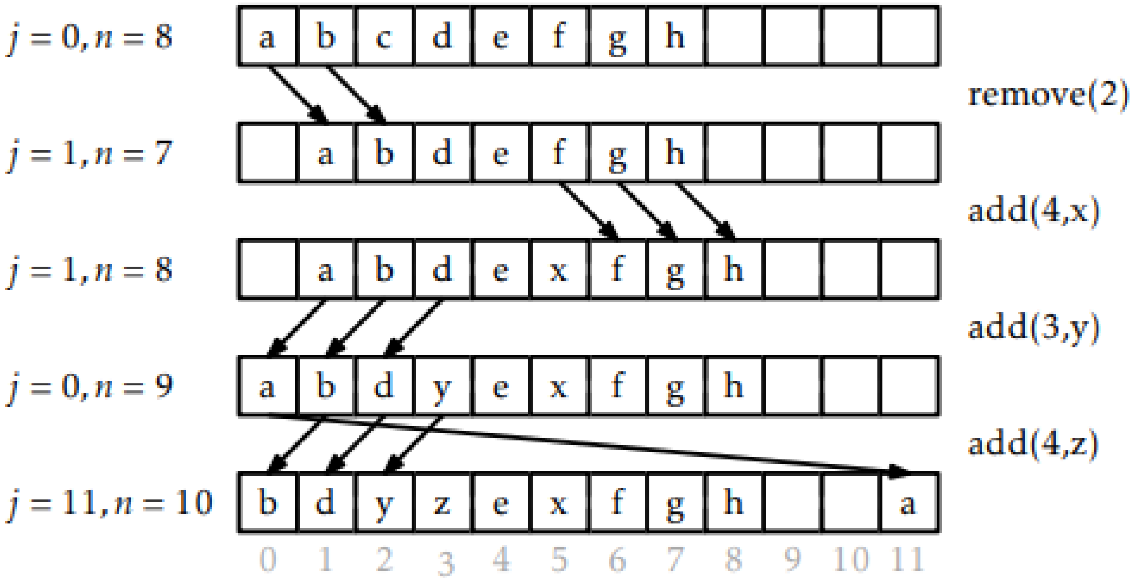


Figure 2.3: A sequence of add(i,x) and remove(i) operations on an ArrayDeque. Arrows denote elements being copied.

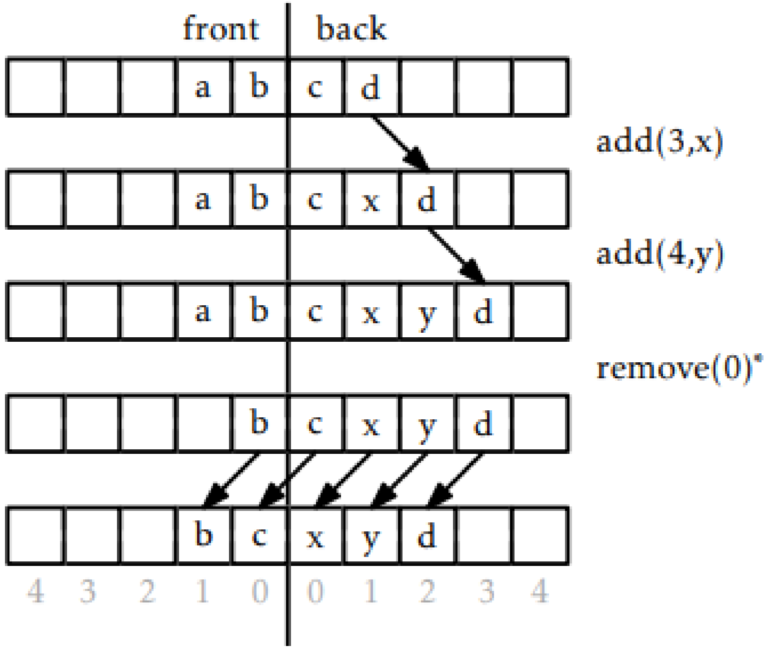
DualArrayDeque

```
initialize()
    $front \leftarrow$ ArrayStack()
    $back \leftarrow$ ArrayStack()
```

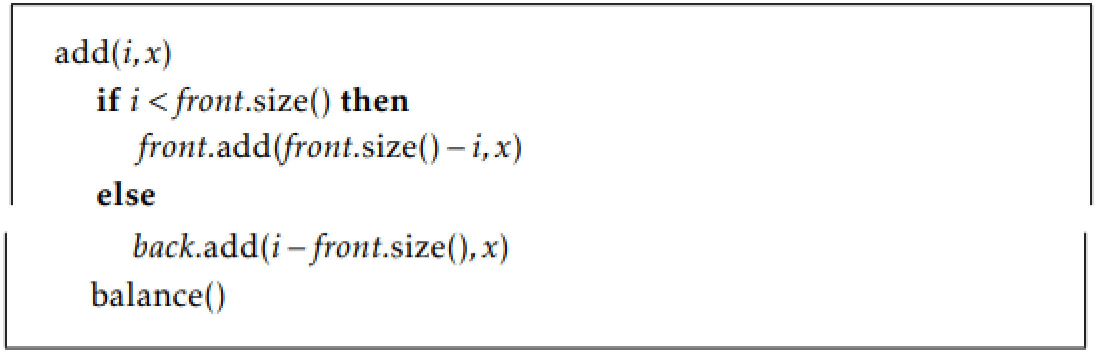
```
size()
    return $front$.size() + $back$.size()
```

```
get($i$)
    if $i < front$.size() then
        return $front$.get($front$.size() $- i - 1$)
    else
        return $back$.get($i - front$.size())
```

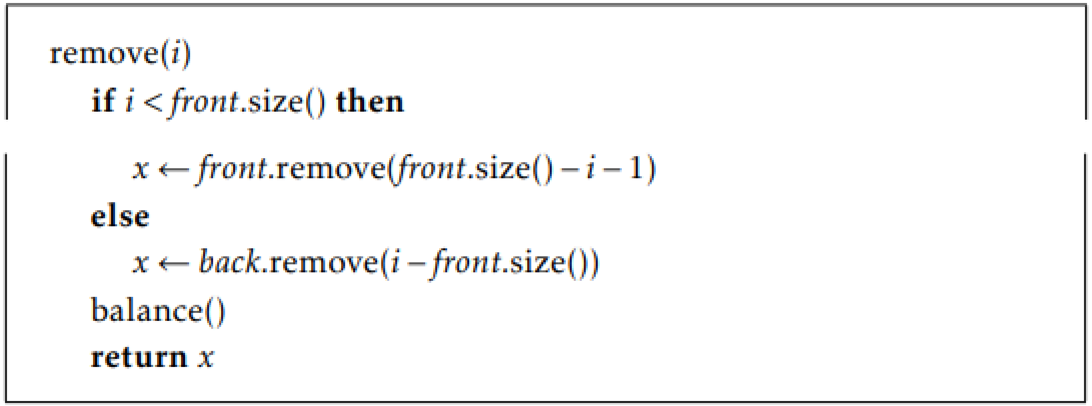
```
set($i, x$)
    if $i < front$.size() then
        return $front$.set($front$.size() $- i - 1, x$)
    else
        return $back$.set($i - front$.size(), x$)
```



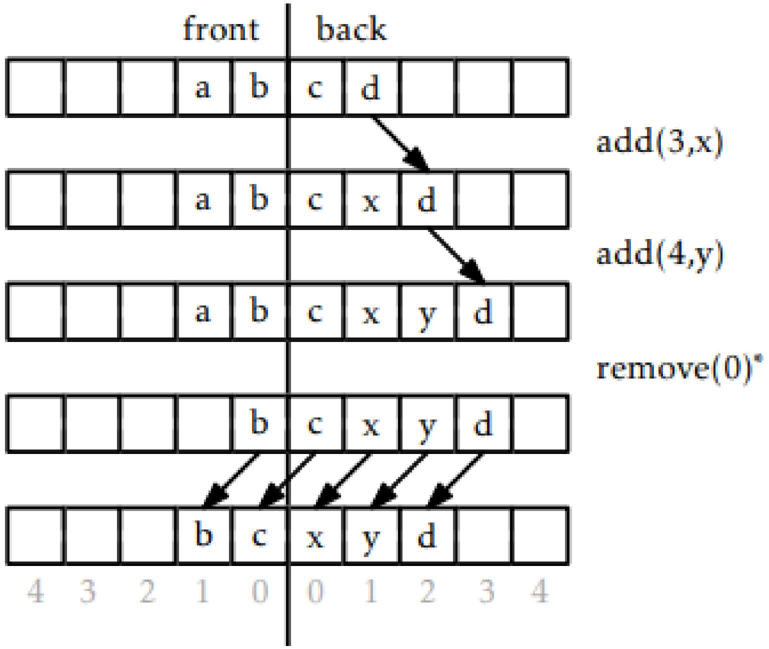
DualArrayDeque



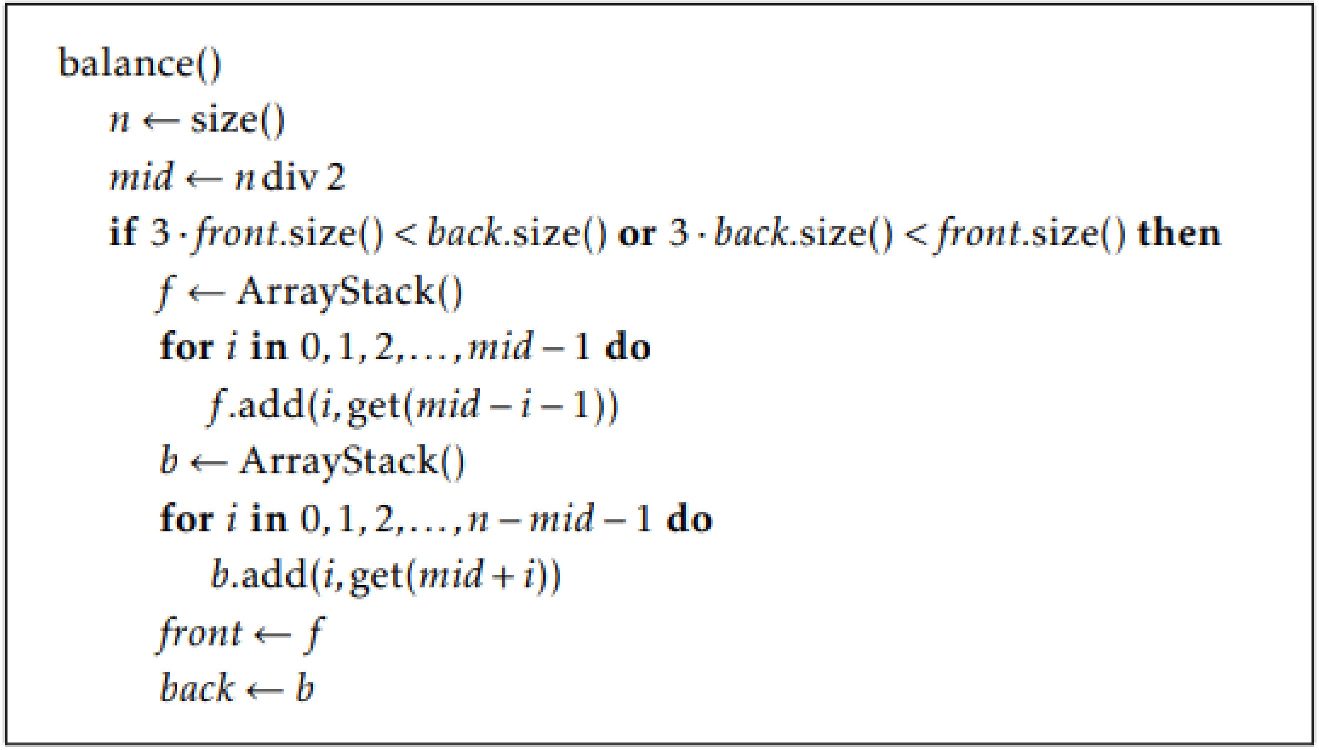
```
add($i, x$)
    if $i < front$.size() then
        $front$.add($front$.size() $- i, x$)
    else
        $back$.add($i - front$.size()$, x$)
    balance()
```



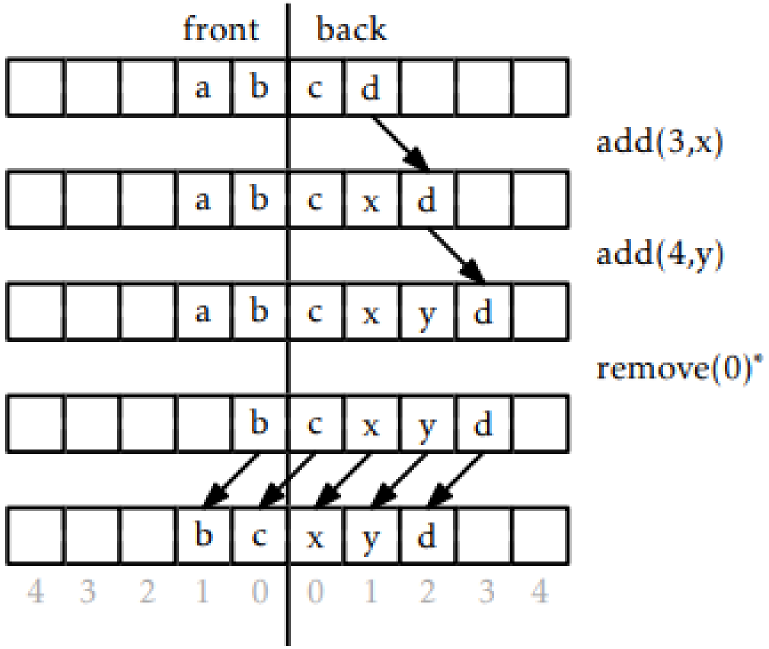
```
remove($i$)
    if $i < front$.size() then
        $x \leftarrow front$.remove($front$.size() $- i - 1$)
    else
        $x \leftarrow back$.remove($i - front$.size())
    balance()
    return $x$
```



DualArrayDeque



```
balance()
    $n \leftarrow$ size()
    $mid \leftarrow n$ div 2
    if $3 \cdot front$.size() $<$ $back$.size() or $3 \cdot back$.size() $<$ $front$.size() then
        $f \leftarrow$ ArrayStack()
        for $i$ in $0, 1, 2, \ldots, mid - 1$ do
            $f$.add($i$, get($mid - i - 1$))
        $b \leftarrow$ ArrayStack()
        for $i$ in $0, 1, 2, \ldots, n - mid - 1$ do
            $b$.add($i$, get($mid + i$))
        $front \leftarrow f$
        $back \leftarrow b$
```



Linked List: Introduction

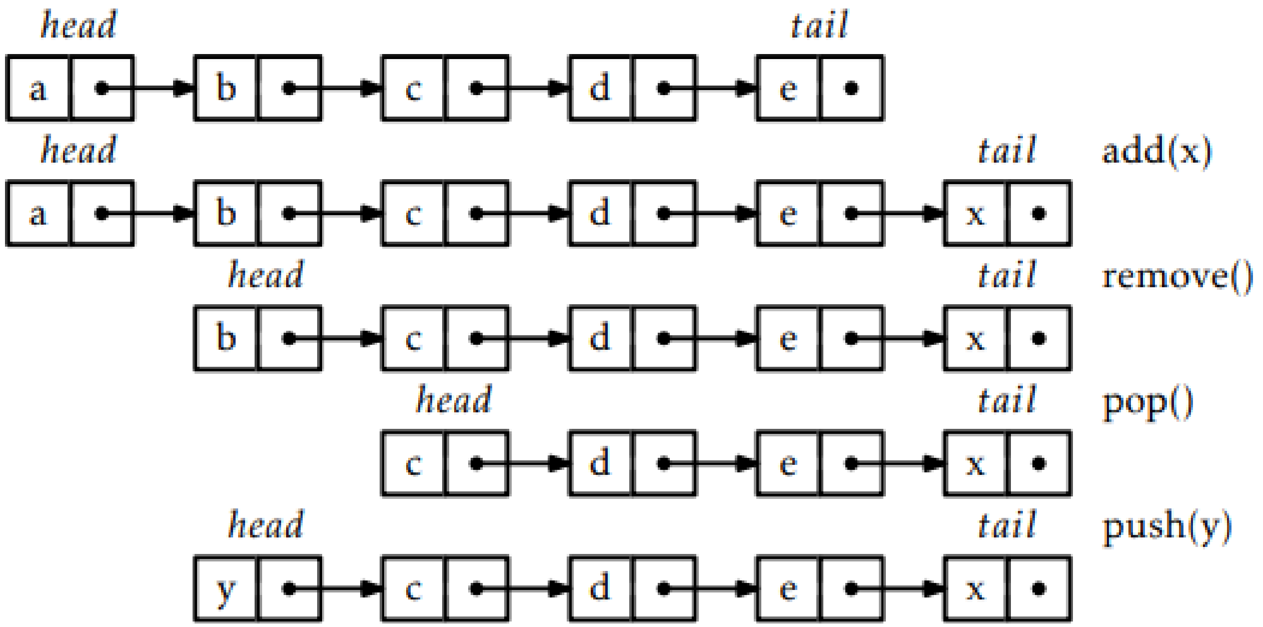
- Using references (pointers), nodes are connected to each other

- Comparison with Arrays:
  - get(i) and set(i,x)
  - Add(i,x) and remove(i)
- Type of Linked List
  - Single Linked List
  - Double Linked List

SLList: List using Single Linked List

- initialize()
- add_front(x)
- add_end(x)
- remove_front()
- remove_end()

- Stack using SSList
  - push(x)
  - pop()
  - top()



head
tail
a
b
c
d
e
head
tail
add(x)
a
b
c
d
e
x
head
tail
remove()
b
c
d
e
x
head
tail
pop()
c
d
e
x
head
tail
push(y)
y
c
d
e
x