List ADT

Array Based
1. Textbook — ListFAB (Fixed Array Based)
2. List RA (Lab 2)

pros: simple, DIA for get

cons: fixed size, pre allocation of memory, shifting for add & remove, cost of resize

simply linked structure

3. List SLS (Lab 3)
4. (Lab 4)

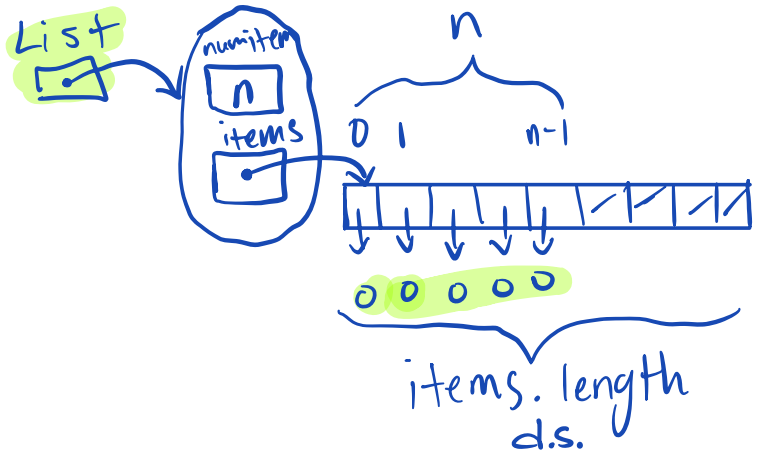
Linked struct.

pros: still simple, on demand allocation

cons: No DIA for get, linked traversal, More memory per item

- Time Complexity
- Space Complexity
  - ↓
  - Not consider memory for variable

List FAB/RA



list obj

head numItems

ref + int + items.length
+ ref = (items.length + 2)
× 4 bytes

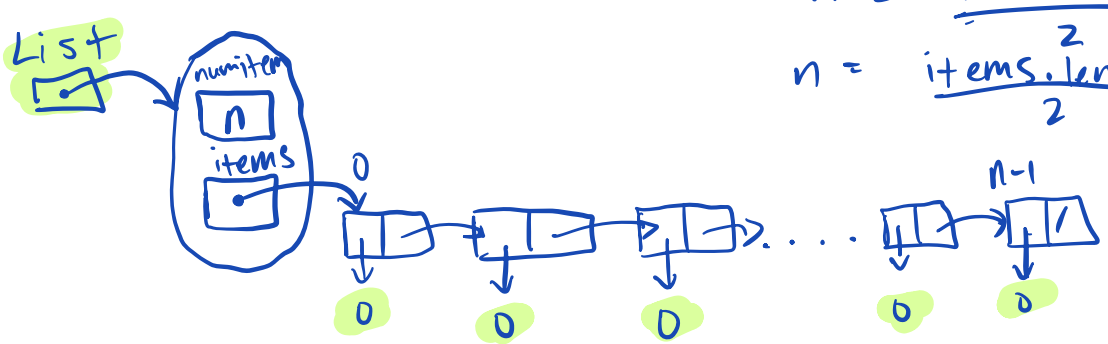
ListRA listj;

(items.length+2)×4bytes

items.length ⟷ 2n

List SLS list;
(2n+2) x 4 bytes
if $n < \frac{items.length}{2}$ ListRA
$n > \frac{items.length}{2}$ ListSLS
$n = \frac{items.length}{2}$ Same

List SLS



list obj

head numItems

ref + int + n x (ref + ref) = (2n + 2) x 4 bytes

List FAB / shift items[] = items[i+1] (i-1) List SLS / link traversal curr = curr.getNext();

get	0			Bc 0	n Bc 0		Wc n-1	
					1		n-1	
	Bc nBc		Wc	Ac	Bc	nBc	Wc	Ac
	n n+1 n-2		0	$\frac{n}{2}$	0	1	n	$\frac{n}{2}$
add	0 1 2	n			0	1	n	
	Bc nBc		Wc	Ac	Bc	nBc	Wc	Ac
	n-1 n-2		0	$\frac{n-1}{2}$	0	1	n-1	$\frac{n+1}{2}$
remove	0 1 2	n-1			1	2	n	

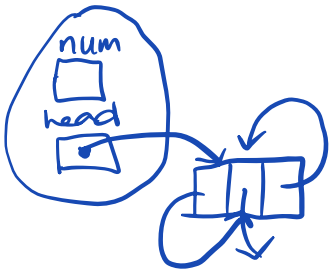
$1+2+3+\ldots+n-2+n-1+n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

$\frac{\text{Sum}}{\#\text{ cases}} = \frac{\frac{n}{2}}{n+1} = \frac{n}{2}$

DNode

emptylist

1 item list




```
public class DNode
{   private Object item;
    private DNode next;
    private DNode back;

  public DNode(Object item...)
  {
      this.item = item;
      next = back = this;   #memory addr of node
  }
}
```

0 → head
1 → head.getNext()
2 → head.getBack()