Doubly-linked lists

```typescript
struct Node<T>:
    prev: ref Node<T>
    value: T
    next: ref Node<T>
```

There's some $j$ such that $n.prev[j]$ = $n$

There's some $k$ such that $n.next[k]$ = $n$

Circular doubly-linked list:

$n.next.prev$ = $n$

$n.prev.next$ = $n$
Doubly-linked lists

```c
struct Node<T>:
    prev: ref Node<T>
    value: T
    next: ref Node<T>
```

- Doubly-linked list:
  - n.next.prev = n
  - n.prev.next = n

Circular doubly-linked list:
  - n.next.prev = n
  - n.prev.next = n
Doubly-linked lists

**Struct Node<T>:**

- prev: ref Node<T>
- value: T
- next: ref Node<T>

- Doubly-linked list:
  - If `n.next ≠ nil` then `n.next.prev = n`
  - If `n.prev ≠ nil` then `n.prev.next = n`

- Circular doubly-linked list:
  - `n.next.prev = n`
  - `n.prev.next = n`
Doubly-linked lists

```rust
struct Node<T>:
    prev: ref Node<T>
    value: T
    next: ref Node<T>
```

- Doubly-linked list:
  - If \( n . \text{next} \neq \text{nil} \) then \( n . \text{next}.\text{prev} = n \)
  - If \( n . \text{prev} \neq \text{nil} \) then \( n . \text{prev}.\text{next} = n \)
  - There’s some \( j \) such that \( n . \text{prev}^j = \text{nil} \)
  - There’s some \( k \) such that \( n . \text{next}^k = \text{nil} \)
Doubly-linked lists

```plaintext
struct Node<T>:
    prev: ref Node<T>
    value: T
    next: ref Node<T>
```

- **Doubly-linked list:**
  - If \( n \).next \( \neq \) nil then \( n \).next.prev = \( n \)
  - If \( n \).prev \( \neq \) nil then \( n \).prev.next = \( n \)
  - There’s some \( j \) such that \( n \).prev^j = \( \text{nil} \)
  - There’s some \( k \) such that \( n \).next^k = \( \text{nil} \)

- **Circular doubly-linked list:**
  - \( n \).next.prev = \( n \)
  - \( n \).prev.next = \( n \)
DLL insertion

To insert o after n (DLL):

```python
    o.prev = n
    o.next = n.next
    o.prev.next = o
    if o.next != nil:
        o.next.prev = o
```
To insert $o$ after $n$ (DLL):

$$
\begin{align*}
o.\text{prev} &= n \\
o.\text{next} &= n.\text{next} \\
o.\text{prev}.\text{next} &= o \\
\text{if } o.\text{next} \neq \text{nil}: \\
&\quad o.\text{next}.\text{prev} = o
\end{align*}
$$

To insert $o$ after $n$ (CDLL):

$$
\begin{align*}
o.\text{prev} &= n \\
o.\text{next} &= n.\text{next} \\
o.\text{prev}.\text{next} &= o \\
o.\text{next}.\text{prev} &= o
\end{align*}
$$
Lists as sequences

How do we:

- lookup an element by position?
- insert an element?
- insert an element at position?
- delete an element at position?
Lists as sequences

How do we:

- lookup an element by position?
- insert an element?
- insert an element at position?
- delete an element at position?

How long does it take?
Arrays as sequences

Assume we leave extra space at the end:

```c
struct Vector<T>:
    size: \mathbb{N}
    data: Array<T>
```

How do we:

- lookup an element by position?
- insert an element?
- insert an element at position?
- delete an element at position?
Arrays as sequences

Assume we leave extra space at the end:

```cpp
struct Vector<T>:
    size: \( \mathbb{N} \)
    data: Array<T>
```

How do we:

- lookup an element by position?
- insert an element?
- insert an element at position?
- delete an element at position?

How long does it take?