Pilhas

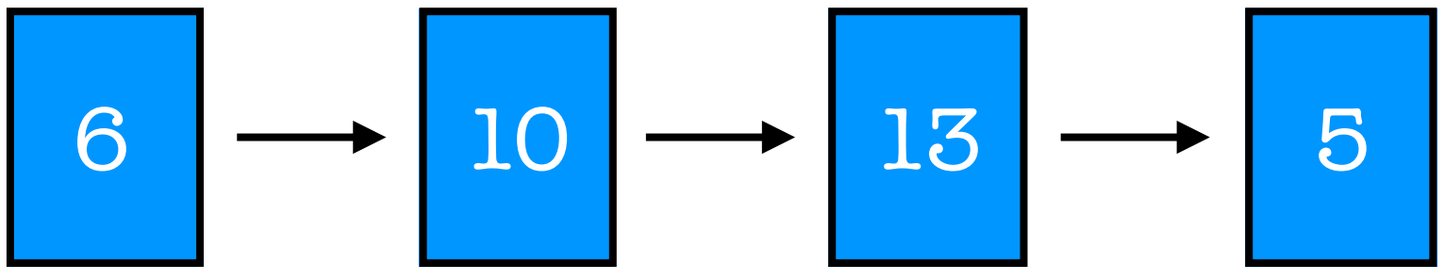
TADs

(lineares)

Filas

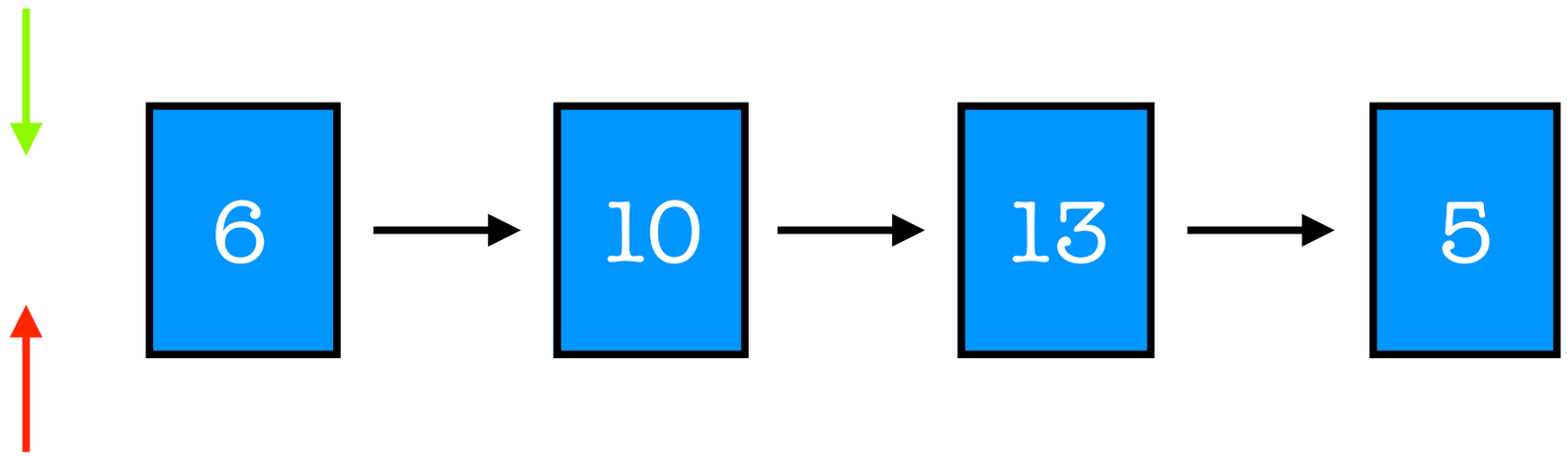
TADs
(lineares)

- Ordem de **inserção**
- Ordem de **remoção**

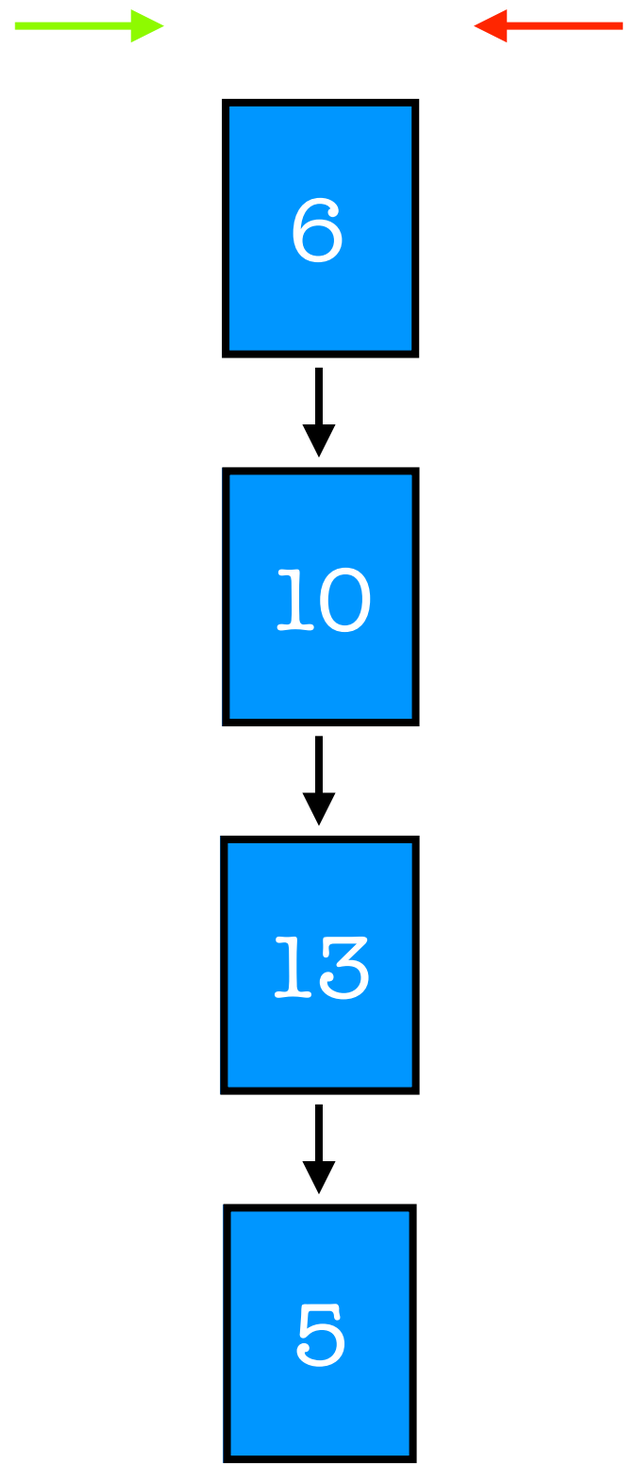


Pilhas

TADs
(lineares)



Pilhas



pilhas

```
template <typename T>
class Stack {
    private:
        vector<T> dados;
    public:
        unsigned size (void) {
            return dados.size();
        }
        const T & peek (void) {
            return dados.back();
        }
        void push (T chave) {
            dados.push_back(chave);
        }
        void pop (void) {
            dados.pop_back();
        }
};
```

pilhas

```
template <typename T>
class Stack {
    private:
        list<T> dados;
    public:
        unsigned size (void) {
            return dados.size();
        }
        const T & peek (void) {
            return dados.back();
        }
        void push (T chave) {
            dados.push_back(chave);
        }
        void pop (void) {
            dados.pop_back();
        }
};
```

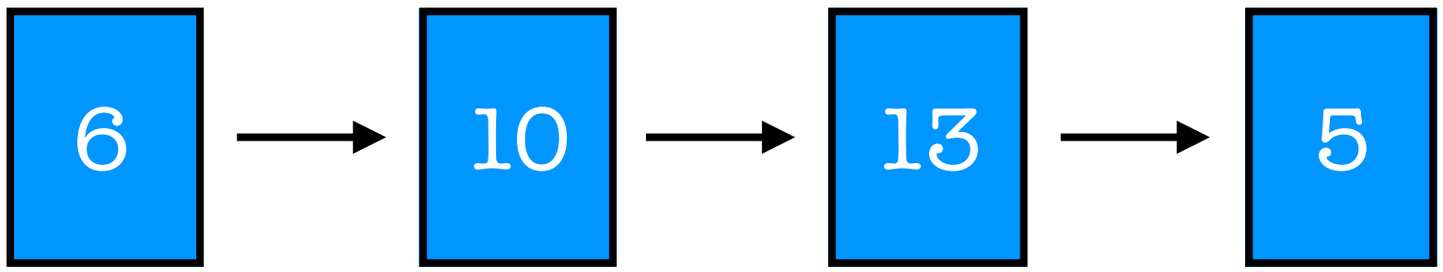
pilhas

```
template <typename T, typename C>
class Stack : private C {
    private:
        using C::back;
        using C::push_back;
        using C::pop_back;
    public:
        using C::size;
        using C::operator=;
        T & peek (void) {
            return back();
        }
        void push (T chave) {
            push_back(chave);
        }
        void pop (void) {
            pop_back();
        }
};
```


ANY
QUESTIONS?
imgflip.com

TADs
(lineares)

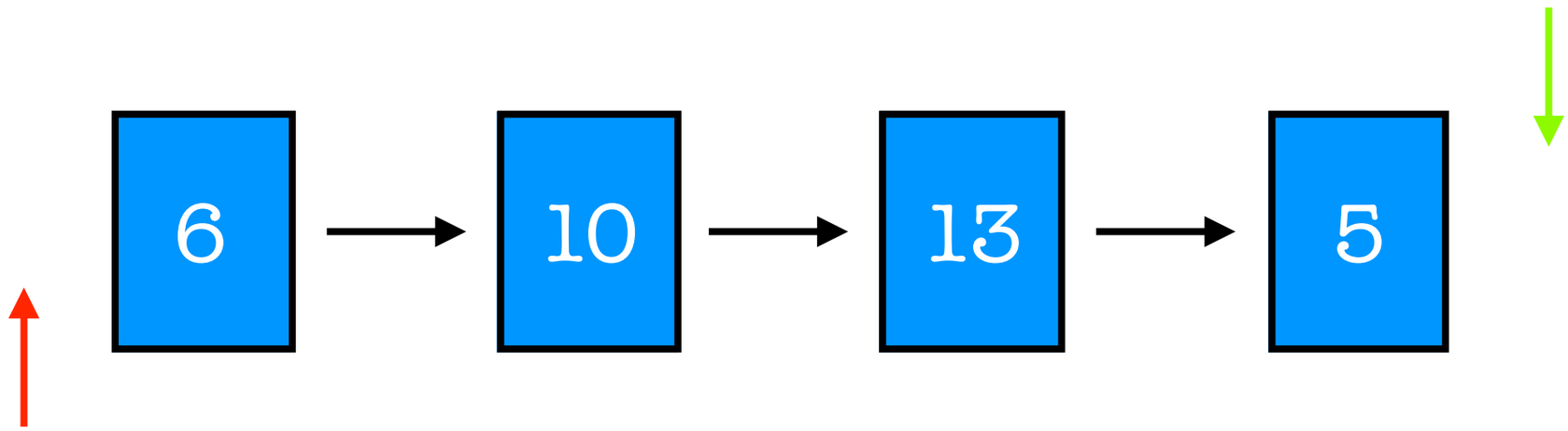
- Ordem de **inserção**
- Ordem de **remoção**



Filas

TADs

(lineares)



filas

```
template <typename T>
class Queue {
    private:
        vector<T> dados;
    public:
        unsigned size (void) {
            return dados.size();
        }
        const T & peek (void) {
            return dados.front();
        }
        void push (T chave) {
            dados.push_back(chave);
        }
        void pop (void) {
            dados.erase(dados.begin());
        }
};
```

filas

```
template <typename T>
class Queue {
    private:
        list<T> dados;
    public:
        unsigned size (void) {
            return dados.size();
        }
        const T & peek (void) {
            return dados.front();
        }
        void push (T chave) {
            dados.push_back(chave);
        }
        void pop (void) {
            dados.pop_front();
        }
};
```

filas

```
template <typename T, typename C>
class Queue : private C {
    private:
        using C::front;
        using C::push_back;
        using C::erase;
        using C::begin;

    public:
        using C::size;
        using C::operator=;
        const T & peek (void) {
            return front();
        }
        void push (T chave) {
            push_back(chave);
        }
        void pop (void) {
            erase(begin());
        }
};
```


ANY
QUESTIONS?
imgflip.com

Operação	Pilha		Fila		
	Vetor	Lista Encadeada	Vetor	Vetor Circular	Lista Encadeada
peek	O(1)	O(1)	O(1)	O(1)	O(1)
push	O(1)	O(1)	O(n)	O(1)	O(1)
pop	O(1)	O(1)	O(1)	O(1)	O(1)

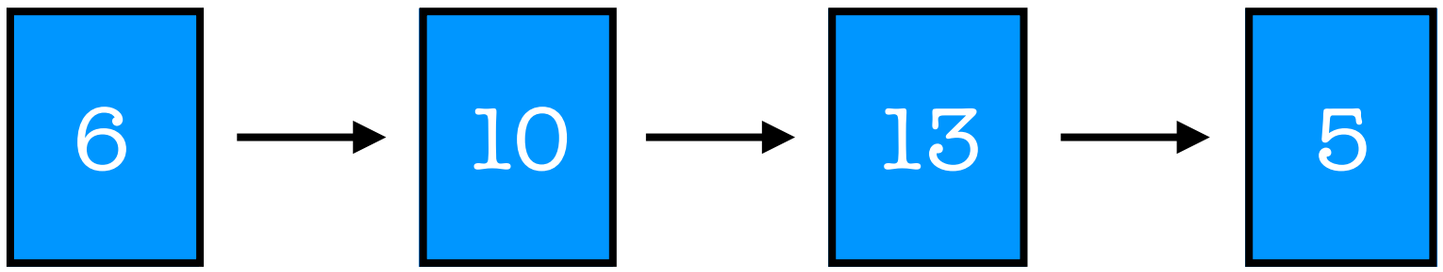
TADs
(lineares)

Políticos entram aqui

Deques

TADs
(lineares)

- Ordem de **inserção**
- Ordem de **remoção**

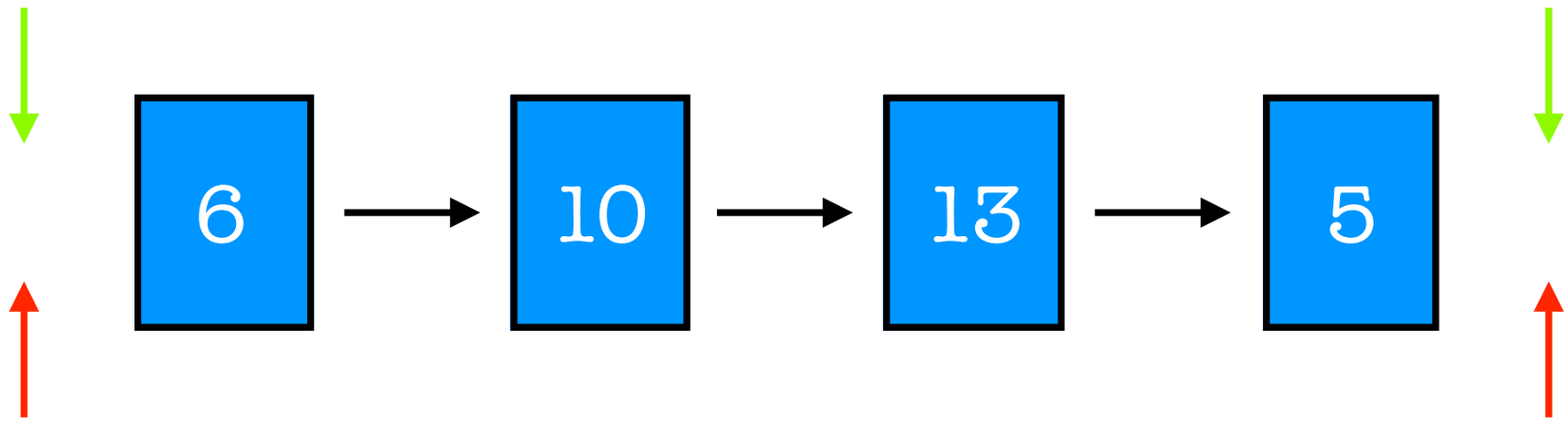


Políticos entram aqui

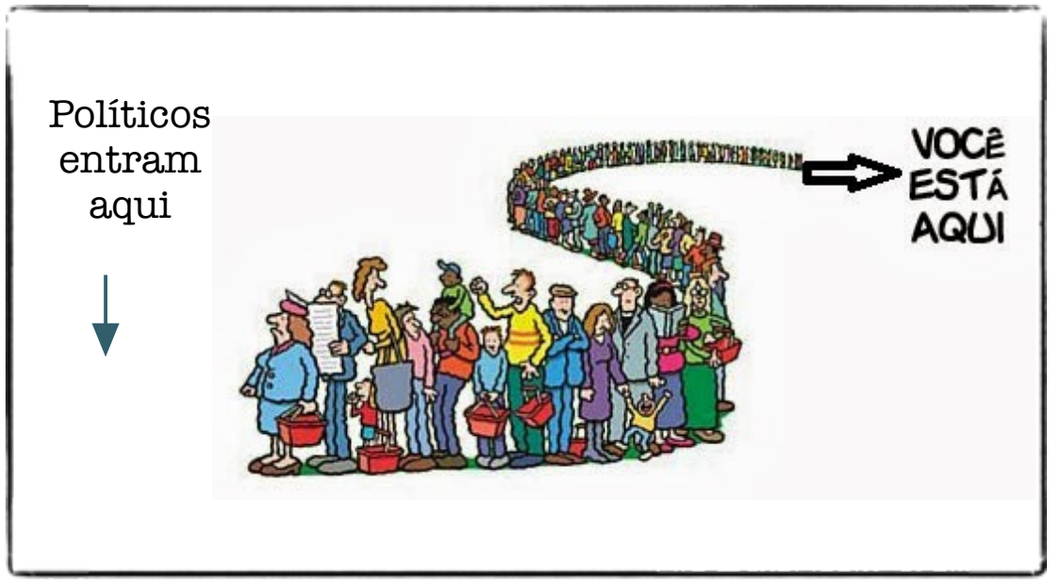
TADs

(lineares)

Deques



deques




```
template <typename T, typename C>
class Deque : private C {
  private:
    using C::insert;
    using C::erase;
  public:
    using C::front;
    using C::back;
    using C::push_back;
    using C::pop_back;
    using C::size;
    using C::operator=;
    void push_front (T chave) {
      insert(begin(), chave);
    }
    void pop_front () {
      erase(begin());
    }
};
```

Operação	Deque		
	Vetor	Vetor circular	Lista Encadeada
front	O(1)	O(1)	O(1)
back	O(1)	O(1)	O(1)
push_front	O(n)	O(1)	O(1)
pop_front	O(n)	O(1)	O(1)
push_back	O(1)	O(1)	O(1)
pop_back	O(1)	O(1)	O(1)


ANY
QUESTIONS?
imgflip.com