

Algoritmos e Estruturas de Dados I

PILHAS

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

www.tiagodemelo.info

Observações

- O conteúdo dessa aula é parcialmente proveniente do Capítulo 6 do livro “*Data Structures and Algorithms in Python*”.
- As palavras com a fonte `Courier` indicam uma palavra-reservada da linguagem de programação.

Introdução

Introdução

- Coleção de objetos que são inseridos e removidos de acordo com o seguinte princípio: **LIFO** (*last in, first out*).

Introdução

- Coleção de objetos que são inseridos e removidos de acordo com o seguinte princípio: **LIFO** (*last in, first out*).
- O usuário insere objetos em uma pilha a qualquer momento, mas somente pode acessar ou remover o último (topo) objeto inserido.

Introdução

Introdução



Introdução



Introdução



Introdução



É possível tirar um livro do meio diretamente?



Introdução

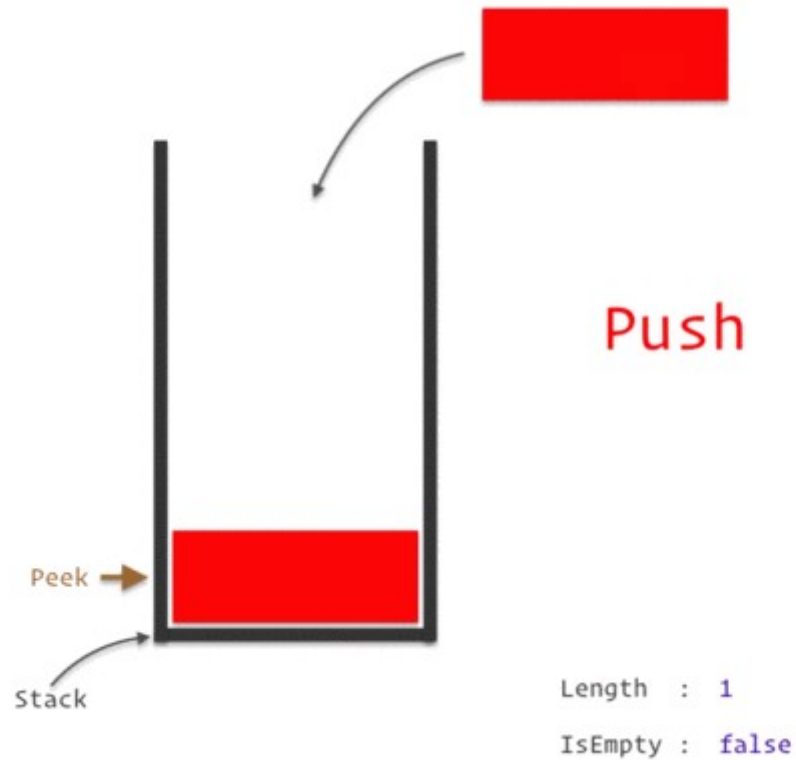


É possível tirar um livro do meio diretamente?



Introdução

- Funcionamento:



Aplicações (1/2)

Aplicações (1/2)

- Navegadores web armazenam os endereços (URLs) que os usuários visitam em uma pilha.

Aplicações (1/2)

- Navegadores web armazenam os endereços (URLs) que os usuários visitam em uma pilha.
- Cada vez que o usuário visita um novo site, este endereço é empilhado (“*pushed*”) na pilha de endereços.

Aplicações (1/2)

- Navegadores web armazenam os endereços (URLs) que os usuários visitam em uma pilha.
- Cada vez que o usuário visita um novo site, este endereço é empilhado (“*pushed*”) na pilha de endereços.
- O navegador então permite que o usuário desempilhe (“*pop*”) os sites visitados usando o botão de Voltar.

Aplicações (2/2)

Aplicações (2/2)

- Editores de texto geralmente fornecem um mecanismo de “voltar” (*undo*) que cancela as operações recentes de edição e reverte o estado anterior do documento.

Aplicações (2/2)

- Editores de texto geralmente fornecem um mecanismo de “voltar” (*undo*) que cancela as operações recentes de edição e reverte o estado anterior do documento.
- Essa operação de *undo* pode ser manipulada por manter as mudanças do texto em uma pilha.

TAD Pilha

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.
- Uma pilha **S** deverá ter os seguintes métodos:

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.
- Uma pilha **S** deverá ter os seguintes métodos:
 - **S.push(e)**

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.
- Uma pilha **S** deverá ter os seguintes métodos:
 - **S.push(e)**
 - Adiciona um elemento **e** no topo da pilha **S**.

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.
- Uma pilha **S** deverá ter os seguintes métodos:
 - **S.push(e)**
 - Adiciona um elemento **e** no topo da pilha **S**.
 - **S.pop()**

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.
- Uma pilha **S** deverá ter os seguintes métodos:
 - **S.push(e)**
 - Adiciona um elemento **e** no topo da pilha **S**.
 - **S.pop()**
 - Remove **e** retorna o elemento do topo da pilha **S**.

TAD Pilha

- Apesar de ser uma estrutura simples, ainda é considerada como uma das mais importantes.
- Uma pilha **S** deverá ter os seguintes métodos:
 - **S.push(e)**
 - Adiciona um elemento **e** no topo da pilha **S**.
 - **S.pop()**
 - Remove **e** retorna o elemento do topo da pilha **S**.
 - Deve ser apresentada uma mensagem especial se a pilha estiver vazia.

TAD Pilha

TAD Pilha

- Outros métodos adicionais:

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**
 - Retorna uma referência ao elemento do topo da pilha **S**, mas sem removê-lo da pilha.

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**
 - Retorna uma referência ao elemento do topo da pilha **S**, mas sem removê-lo da pilha.
 - Deve ser apresentada uma mensagem especial se a pilha estiver vazia.

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**
 - Retorna uma referência ao elemento do topo da pilha **S**, mas sem removê-lo da pilha.
 - Deve ser apresentada uma mensagem especial se a pilha estiver vazia.
 - **S.is_empty ()**

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**
 - Retorna uma referência ao elemento do topo da pilha **S**, mas sem removê-lo da pilha.
 - Deve ser apresentada uma mensagem especial se a pilha estiver vazia.
 - **S.is_empty ()**
 - Retorna `True` se a pilha **S** está vazia.

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**
 - Retorna uma referência ao elemento do topo da pilha **S**, mas sem removê-lo da pilha.
 - Deve ser apresentada uma mensagem especial se a pilha estiver vazia.
 - **S.is_empty ()**
 - Retorna `True` se a pilha **S** está vazia.
 - **len (S)**

TAD Pilha

- Outros métodos adicionais:
 - **S.top ()**
 - Retorna uma referência ao elemento do topo da pilha **S**, mas sem removê-lo da pilha.
 - Deve ser apresentada uma mensagem especial se a pilha estiver vazia.
 - **S.is_empty ()**
 - Retorna `True` se a pilha **S** está vazia.
 - **len (S)**
 - Retorna o número de elementos na pilha **S**.

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]
S.push(4)	–	[7, 9, 4]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]
S.push(4)	–	[7, 9, 4]
len(S)	3	[7, 9, 4]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]
S.push(4)	–	[7, 9, 4]
len(S)	3	[7, 9, 4]
S.pop()	4	[7, 9]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]
S.push(4)	–	[7, 9, 4]
len(S)	3	[7, 9, 4]
S.pop()	4	[7, 9]
S.push(6)	–	[7, 9, 6]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]
S.push(4)	–	[7, 9, 4]
len(S)	3	[7, 9, 4]
S.pop()	4	[7, 9]
S.push(6)	–	[7, 9, 6]
S.push(8)	–	[7, 9, 6, 8]

TA Pilha (exemplo)

Operation	Return Value	Stack Contents
S.push(5)	–	[5]
S.push(3)	–	[5, 3]
len(S)	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.is_empty()	True	[]
S.pop()	“error”	[]
S.push(7)	–	[7]
S.push(9)	–	[7, 9]
S.top()	9	[7, 9]
S.push(4)	–	[7, 9, 4]
len(S)	3	[7, 9, 4]
S.pop()	4	[7, 9]
S.push(6)	–	[7, 9, 6]
S.push(8)	–	[7, 9, 6, 8]
S.pop()	8	[7, 9, 6]

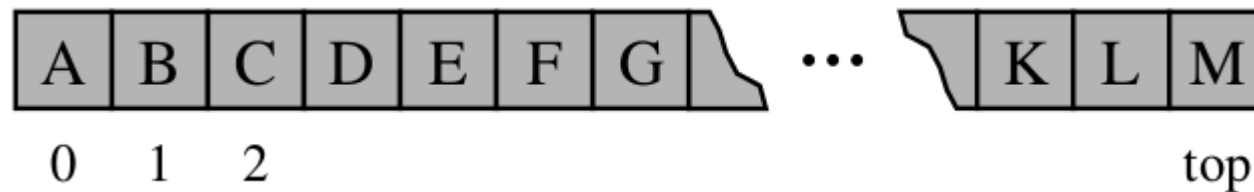
Pilha baseada em Lista

Pilha baseada em Lista

- Nós podemos implementar uma pilha facilmente por armazenar os seus elementos em uma lista em Python.

Pilha baseada em Lista

- Nós podemos implementar uma pilha facilmente por armazenar os seus elementos em uma lista em Python.
- Lembre-se que a classe `List` em Python já possui um método para adicionar um elemento ao final da lista (`append`) e outro método para remover o último elemento (`pop`).



Pilha baseada em Lista

Pilha baseada em Lista

- Qual é o problema de usar a representação anterior?

Pilha baseada em Lista

- Qual é o problema de usar a representação anterior?
 - Listas também possuem comportamentos que contradizem a abstração que uma pilha representa.

Pilha baseada em Lista

- Qual é o problema de usar a representação anterior?
 - Listas também possuem comportamentos que contradizem a abstração que uma pilha representa.
 - Exemplo: remover e/ou acessar um elemento do meio da lista.

Pilha baseada em Lista

- Qual é o problema de usar a representação anterior?
 - Listas também possuem comportamentos que contradizem a abstração que uma pilha representa.
 - Exemplo: remover e/ou acessar um elemento do meio da lista.
 - A nomenclatura usada entre os métodos de pilha e lista são distintos.

Pilha baseada em Lista

- Qual é o problema de usar a representação anterior?
 - Listas também possuem comportamentos que contradizem a abstração que uma pilha representa.
 - Exemplo: remover e/ou acessar um elemento do meio da lista.
 - A nomenclatura usada entre os métodos de pilha e lista são distintos.
 - Exemplo: `append` e `push`.

Pilha baseada em Lista (na mão)

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

<i>Stack Method</i>	<i>Realization with Python list</i>

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

<i>Stack Method</i>	<i>Realization with Python list</i>
S.push(e)	L.append(e)

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

<i>Stack Method</i>	<i>Realization with Python list</i>
S.push(e)	L.append(e)
S.pop()	L.pop()

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

<i>Stack Method</i>	<i>Realization with Python list</i>
S.push(e)	L.append(e)
S.pop()	L.pop()
S.top()	L[-1]

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

<i>Stack Method</i>	<i>Realization with Python list</i>
S.push(e)	L.append(e)
S.pop()	L.pop()
S.top()	L[-1]
S.is_empty()	len(L) == 0

Pilha baseada em Lista (na mão)

- Nós precisamos adaptar os métodos já existentes da classe List para o TAD Pilha.
- Pilha S como uma adaptação da uma lista L (Python):

<i>Stack Method</i>	<i>Realization with Python list</i>
S.push(e)	L.append(e)
S.pop()	L.pop()
S.top()	L[-1]
S.is_empty()	len(L) == 0
len(S)	len(L)

`class` ArrayStack:

criar uma pilha vazia

calcular o número de elementos da pilha

checar se a pilha está vazia

empilhar um elemento

descobrir o elemento do topo da pilha
(isso **não** é desempilhar)

desempilhar

```
class ArrayStack:
```

```
    def __init__(self):  
        self._data = []
```

calcular o número de elementos da pilha

checar se a pilha está vazia

empilhar um elemento

descobrir o elemento do topo da pilha
(isso **não** é desempilhar)

desempilhar

```
class ArrayStack:

    def __init__(self):
        self._data = []

    def __len__(self):
        return len(self._data)
```

checar se a pilha está vazia

empilhar um elemento

descobrir o elemento do topo da pilha
(isso **não** é desempilhar)

desempilhar

```
class ArrayStack:

    def __init__(self):
        self._data = []

    def __len__(self):
        return len(self._data)

    def is_empty(self):
        return len(self._data) == 0
```

empilhar um elemento

descobrir o elemento do topo da pilha
(isso **não** é desempilhar)

desempilhar

```
class ArrayStack:

    def __init__(self):
        self._data = []

    def __len__(self):
        return len(self._data)

    def is_empty(self):
        return len(self._data) == 0

    def push(self, e):
        self._data.append(e)
```

descobrir o elemento do topo da pilha
(isso **não** é desempilhar)

desempilhar

```
class ArrayStack:

    def __init__(self):
        self._data = []

    def __len__(self):
        return len(self._data)

    def is_empty(self):
        return len(self._data) == 0

    def push(self, e):
        self._data.append(e)

    def top(self):
        if self.is_empty():
            raise Empty('Pilha vazia')
        return self._data[-1]
```

desempilhar

```
class ArrayStack:

    def __init__(self):
        self._data = []

    def __len__(self):
        return len(self._data)

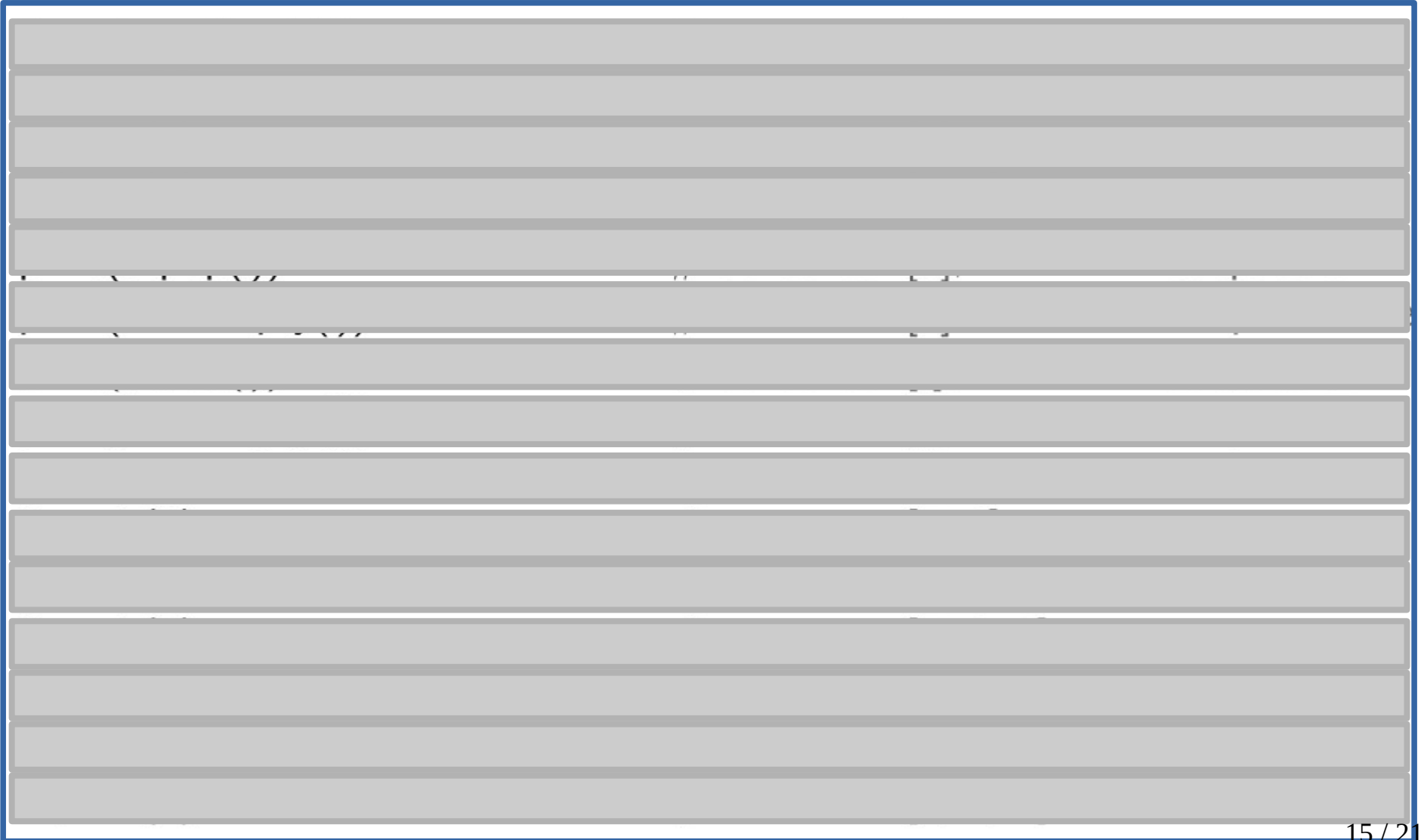
    def is_empty(self):
        return len(self._data) == 0

    def push(self, e):
        self._data.append(e)

    def top(self):
        if self.is_empty():
            raise Empty('Pilha vazia')
        return self._data[-1]

    def pop(self):
        if self.is_empty():
            raise Empty('Pilha vazia')
        return self._data.pop()
```

Exemplo de execução (código anterior)



Exemplo de execução (código anterior)

S = ArrayStack()

contents: []



Exemplo de execução (código anterior)

```
S = ArrayStack( )
```

```
# contents: [ ]
```

```
S.push(5)
```

```
# contents: [5]
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]  
S.push(5)                   # contents: [5]  
S.push(3)                   # contents: [5, 3]
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())        # contents: [5];        outputs False
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())        # contents: [5];        outputs False
print(S.pop())              # contents: [ ];         outputs 5
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())        # contents: [5];        outputs False
print(S.pop())              # contents: [ ];         outputs 5
print(S.is_empty())        # contents: [ ];         outputs True
```


Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())         # contents: [5];        outputs False
print(S.pop())              # contents: [ ];         outputs 5
print(S.is_empty())         # contents: [ ];         outputs True
S.push(7)                   # contents: [7]
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];      outputs 3
print(S.is_empty())        # contents: [5];      outputs False
print(S.pop())              # contents: [ ];      outputs 5
print(S.is_empty())        # contents: [ ];      outputs True
S.push(7)                   # contents: [7]
S.push(9)                   # contents: [7, 9]
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())        # contents: [5];        outputs False
print(S.pop())              # contents: [ ];          outputs 5
print(S.is_empty())        # contents: [ ];          outputs True
S.push(7)                   # contents: [7]
S.push(9)                   # contents: [7, 9]
print(S.top())              # contents: [7, 9];      outputs 9
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];      outputs 3
print(S.is_empty())        # contents: [5];      outputs False
print(S.pop())              # contents: [ ];      outputs 5
print(S.is_empty())        # contents: [ ];      outputs True
S.push(7)                   # contents: [7]
S.push(9)                   # contents: [7, 9]
print(S.top())              # contents: [7, 9];    outputs 9
S.push(4)                   # contents: [7, 9, 4]
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())        # contents: [5];        outputs False
print(S.pop())              # contents: [ ];          outputs 5
print(S.is_empty())        # contents: [ ];          outputs True
S.push(7)                   # contents: [7]
S.push(9)                   # contents: [7, 9]
print(S.top())              # contents: [7, 9];      outputs 9
S.push(4)                   # contents: [7, 9, 4]
print(len(S))               # contents: [7, 9, 4];    outputs 3
```

Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];        outputs 3
print(S.is_empty())        # contents: [5];        outputs False
print(S.pop())              # contents: [ ];          outputs 5
print(S.is_empty())        # contents: [ ];          outputs True
S.push(7)                   # contents: [7]
S.push(9)                   # contents: [7, 9]
print(S.top())              # contents: [7, 9];      outputs 9
S.push(4)                   # contents: [7, 9, 4]
print(len(S))               # contents: [7, 9, 4];    outputs 3
print(S.pop())              # contents: [7, 9];      outputs 4
```


Exemplo de execução (código anterior)

```
S = ArrayStack( )           # contents: [ ]
S.push(5)                   # contents: [5]
S.push(3)                   # contents: [5, 3]
print(len(S))               # contents: [5, 3];      outputs 2
print(S.pop())              # contents: [5];      outputs 3
print(S.is_empty())        # contents: [5];      outputs False
print(S.pop())              # contents: [ ];      outputs 5
print(S.is_empty())        # contents: [ ];      outputs True
S.push(7)                   # contents: [7]
S.push(9)                   # contents: [7, 9]
print(S.top())              # contents: [7, 9];      outputs 9
S.push(4)                   # contents: [7, 9, 4]
print(len(S))               # contents: [7, 9, 4];      outputs 3
print(S.pop())              # contents: [7, 9];      outputs 4
S.push(6)                   # contents: [7, 9, 6]
```

Pilha baseada em lista

- Análise do tempo de execução:

Operation	Running Time
S.push(e)	
S.pop()	
S.top()	
S.is_empty()	
len(S)	

Pilha baseada em lista

- Análise do tempo de execução:

Operation	Running Time
S.push(e)	$O(1)^*$
S.pop()	
S.top()	
S.is_empty()	
len(S)	

Pilha baseada em lista

- Análise do tempo de execução:

Operation	Running Time
S.push(e)	$O(1)^*$
S.pop()	$O(1)^*$
S.top()	
S.is_empty()	
len(S)	

Pilha baseada em lista

- Análise do tempo de execução:

Operation	Running Time
S.push(e)	$O(1)^*$
S.pop()	$O(1)^*$
S.top()	$O(1)$
S.is_empty()	
len(S)	

Pilha baseada em lista

- Análise do tempo de execução:

Operation	Running Time
S.push(e)	$O(1)^*$
S.pop()	$O(1)^*$
S.top()	$O(1)$
S.is_empty()	$O(1)$
len(S)	

Pilha baseada em lista

- Análise do tempo de execução:

Operation	Running Time
S.push(e)	$O(1)^*$
S.pop()	$O(1)^*$
S.top()	$O(1)$
S.is_empty()	$O(1)$
len(S)	$O(1)$

Pilha baseada em Lista (aplicação)

Pilha baseada em Lista (aplicação)

- Um algoritmo para casamento de delimitadores

Pilha baseada em Lista (aplicação)

- Um algoritmo para casamento de delimitadores
 - Uma importante tarefa no processamento de expressões aritméticas é garantir que os símbolos dos delimitadores estão corretamente colocados.

Pilha baseada em Lista (aplicação)

- Um algoritmo para casamento de delimitadores
 - Uma importante tarefa no processamento de expressões aritméticas é garantir que os símbolos dos delimitadores estão corretamente colocados.
 - Exemplo:

Pilha baseada em Lista (aplicação)

- Um algoritmo para casamento de delimitadores
 - Uma importante tarefa no processamento de expressões aritméticas é garantir que os símbolos dos delimitadores estão corretamente colocados.
 - Exemplo:
 - $[(5+x) - (y+z)]$

Pilha baseada em Lista (aplicação)

- Um algoritmo para casamento de delimitadores
 - Uma importante tarefa no processamento de expressões aritméticas é garantir que os símbolos dos delimitadores estão corretamente colocados.
 - Exemplo:
 - $[(5+x) - (y+z)]$

Como seria o algoritmo?



Pilha baseada em Lista (aplicação)

- Algoritmo

Pilha baseada em Lista (aplicação)

- Algoritmo
 - Leitura feita da esquerda para direita.

Pilha baseada em Lista (aplicação)

- Algoritmo
 - Leitura feita da esquerda para direita.
 - Cada vez que encontramos um símbolo de abertura, nós empilhamos na pilha S.

Pilha baseada em Lista (aplicação)

- Algoritmo
 - Leitura feita da esquerda para direita.
 - Cada vez que encontramos um símbolo de abertura, nós empilhamos na pilha S.
 - Cada vez que encontramos um símbolo de fechamento, nós desempilhamos um símbolo de S e verificamos se os dois símbolos formam um par válido.

Pilha baseada em Lista (aplicação)

- Algoritmo

- Leitura feita da esquerda para direita.
- Cada vez que encontramos um símbolo de abertura, nós empilhamos na pilha S.
- Cada vez que encontramos um símbolo de fechamento, nós desempilhamos um símbolo de S e verificamos se os dois símbolos formam um par válido.
- Se chegarmos ao final da expressão e a pilha estiver vazia, então a expressão está bem formada. Caso contrário, há um problema na expressão.

Pilha baseada em Lista (aplicação)

```
def is_matched (expr):
    lefty = '({['
    righty = ')}]}'
    S = ArrayStack()
    for c in expr:
        if c in lefty:
            S.push(c)
        elif c in righty:
            if S.is_empty():
                return False
            if righty.index(c) != lefty.index(S.pop()):
                return False
    return S.is_empty()
```

Pilha baseada em Lista

- Outras aplicações
 - Casamento de TAGs de HTML

```
<body>
<center>
<h1> The Little Boat </h1>
</center>
<p> The storm tossed the little
boat like a cheap sneaker in an
old washing machine. The three
drunken fishermen were used to
such treatment, of course, but
not the tree salesman, who even as
a stowaway now felt that he
had overpaid for the voyage. </p>
<ol>
<li> Will the salesman die? </li>
<li> What color is the boat? </li>
<li> And what about Naomi? </li>
</ol>
</body>
```

(a)

The Little Boat

The storm tossed the little boat like a cheap sneaker in an old washing machine. The three drunken fishermen were used to such treatment, of course, but not the tree salesman, who even as a stowaway now felt that he had overpaid for the voyage.

1. Will the salesman die?
2. What color is the boat?
3. And what about Naomi?

(b)

Exercícios

- Explique dois exemplos de aplicações que usam o TAD Pilha.
- Uma letra representa uma operação de *push* e um asterisco representa uma operação de *pop* na seguinte sequência. Dada a sequência de valores retornados pelas operações de *pop* quando esta sequência de operações é executada sobre uma pilha vazia (Sedgwick, Exercício 4.6).

E A S * Y * Q U E * * * S T * * * I O * N * * *