SME0827 - Estruturas de Dados

Listas Encadeadas
Aula 11

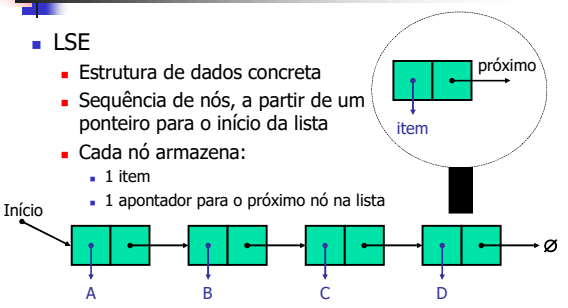
Professor: André C. P. L. F. de Carvalho, ICMC-USP
PAE: Moisés Rocha dos Santos
Monitor: Marília Costa Rosendo Silva

Tópicos

- Introdução
- Listas simplesmente encadeadas
  - Pilha (stack)
  - Fila (queue)
- Docstring
- Listas duplamente encadeadas
  - Pilha (stack)
  - Fila (queue)

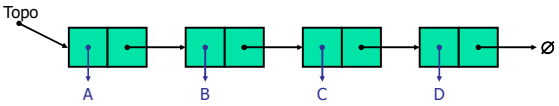
Lista simplesmente encadeada

- LSE
  - Estrutura de dados concreta
  - Sequência de nós, a partir de um ponteiro para o início da lista
  - Cada nó armazena:
    - 1 item
    - 1 apontador para o próximo nó na lista



Pilha usando LSE

- Implementação
  - O item no topo da pilha é armazenado no primeiro nó da lista
    - Espaço utilizado é O(n)
    - Cada operação tem complexidade O(1)



LSE para pilha em Python

```
#Classe no para LSE
class No_LSE:

  # Método para inicializar objeto nó em LSE
  def __init__ (si, dados):
    # Atribui dados
    si.dados = dados
    # Inicializa proximo como nó null
    si.proximo = None

# classe Pilha para LSE
class Pilha:
  # Método para inicializar inicio
  def __init__(si):
    si.inicio  = None

  # Método para verificar se pilha está vazia
  def  isEmpty (si):
    return si.inicio == None
```

```
#Método para incluir item na pilha
def push(si, dados):
  if si.inicio is None:
    si.inicio =  No_LSE(dados)
  else:
    temp = No_LSE(dados)
    temp.proximo = si.inicio
    si.inicio = temp
    print(dados)

# Método para remover itens da pilha
def pop(si):
  if si.isEmpty():
    return
  temp = si.inicio
  si.inicio = si.inicio.proximo
  temp.proximo = None
```


LSE para pilha em Python

```
#Método primeiro item da pilha
  def top(si):
    return si.inicio.dados
```

```
# Método para retornar o tamanho da pilha
def size(si):
  temp = si.inicio
  cont = 0
  while temp is not None:
      cont = cont + 1
      temp = temp.proximo
  return cont
```

```
# Método para imprimir itens da pilha
  def printpilha(si):
    print("Os itens da pilha são:")
    temp=si.inicio
    while (temp != None):
      print(temp.dados,end="<-")
      temp=temp.proximo
```

```
if __name__=='__main__':
  pilha1 = Pilha()
  print("Operacoes de pilha para LSE")
  pilha1.push(4)
  pilha1.push(3)
  pilha1.push(2)
  pilha1.pop()
  pilha1.pop()
  pilha1.push(1)
  print("Topo é: ", pilha1.top())
  print("Tamanho é: ",pilha1.size())
  printpilha(pilha1)
```

LSE para pilha em Python

```
if __name__=='__main__':
  pilha1 = Pilha()
  print("Operacoes de pilha para LSE")
  pilha1.push(4)
  pilha1.push(3)
  pilha1.push(2)
  pilha1.pop()
  pilha1.pop()
  pilha1.push(1)
  print("Topo é: ", pilha1.top())
  print("Tamanho é: ",pilha1.size())
  printpilha(pilha1)
```

```
Operacoes de pilha para lista simplesmente encadeada
4
3
2
1
Topo é:  1
Tamanho é:  2
Os itens da pilha são:
1<-4<-
```


LSE para pilha em Python

```
#Classe no para LSE
class No_LSE:

  # Método para inicializar objeto nó em LSE
  def __init__(si, dados):
    # Atribui dados
    si.dados = dados
    # Inicializa proximo como nó null
    si.proximo = None

# classe Pilha para LSE
class Pilha:
  # Método para inicializar inicio
  def __init__(si):
    si.inicio = None

  # Método para verificar se pilha está vazia
  def isEmpty(si):
    return si.inicio == None
```

```
#Método para incluir item na pilha
def push(si, dados):
  if si.inicio is None:
    si.inicio = No_LSE(dados)
  else:
    temp = No_LSE(dados)
    temp.proximo = si.inicio
    si.inicio = temp
    print(dados)

# Método para remover itens da pilha
def pop(si):
  if si.isEmpty():
    return
  temp = si.inicio
  si.inicio = si.inicio.proximo
  temp.proximo = None
```


LSE para pilha em Python

```
#Método primeiro item da pilha
def top(si):
  return si.inicio.dados
```

```
# Método para retornar o tamanho da pilha
def size(si):
  temp = si.inicio
  cont = 0
  while temp is not None:
    cont = cont + 1
    temp = temp.proximo
  return cont
```

```
# Método para imprimir itens da pilha
def printpilha(si):
  print("Os itens da pilha são:")
  temp=si.inicio
  while (temp != None):
    print(temp.dados,end="<-")
    temp=temp.proximo
```

```
if __name__=='__main__':
  pilha1 = Pilha()
  print("Operacoes de pilha para LSE")
  pilha1.push(4)
  pilha1.push(3)
  pilha1.push(2)
  pilha1.pop()
  pilha1.pop()
  pilha1.push(1)
  print("Topo é: ", pilha1.top())
  print("Tamanho é: ",pilha1.size())
  printpilha(pilha1)
```

Docstrings

Docstring

- *Documentation string (docstring)*
  - *String* de documentação
  - Constante (literal) string que pode ser usada na primeira linha do código de uma função
- Várias ferramentas usam *docstring* para automaticamente:
  - Criar documentação online ou impressa
  - Deixar usuário navegar interativamente pelo código
- Seu uso é uma boa prática de programação

Docstring

- Para que é usado?
  - Para entender o que faz um trecho grande de um código
  - Para que propósito foi criada uma classe, ou módulo, um método ou uma função
    - Ex.: Descreve o que a função faz
- Diferente de comentários
  - Usados para códigos curtos, funções, comandos ou expressões
    - Ex. Descreve como a função faz
  - Em geral pequenos

Comentários

- Em geral describem seu código para outros desenvolvedores
- Audiência pretendida:
  - Desenvolvedores e as pessoas que farão manutenção do código
- Ajudam a melhor entender o código e para que ele foi escrito

Docstring

- Pode explicar funcionamento de um código em diferentes níveis:
  - Módulo
    - Arquivo com as definições e comandos de um programa em Python
  - Classe
  - Método
  - Função

Método x Função

- Método é similar a uma função, mas com as seguintes diferenças
  - Método é associado a uma classe
  - Método é usado pelo objeto da classe onde é declarado
  - Método só pode acessar dados declarados na classe

Pausa

Docstrings

- Declaração:
  - Usar três aspas duplas "''" na linha seguinte a declaração da classe, método ou função
  - Usa mesma identação das primeiras três aspas na linha seguinte
  - Todos os métodos deveriam ter um docstring
- Acesso:
  - Usar a função print com o método __doc__ do objeto ou a função pré-existente help()

Docstrings para métodos

- O docstring de cada método de uma classe devem conter as seguintes informações:
  - Descrição curta do que ele é e o que ele faz
  - Argumentos (obrigatórios e opcionais) passados para ele
  - Informação sobre que argumentos são opcionais ou têm um valor default
  - Possíveis efeitos colaterais, exceções e restrições
- Sugere-se que cada método de uma classe deve ser documentado com um docstring individual

Exemplo

```
def faz():
    """So mostra exemplo de docstring."""
    return None

print "--- Usando __doc__:"
print faz.__doc__
print "--- Usando help:"
help(faz)
```

```
--- Usando __doc__:
So mostra exemplo de docstring.
--- Usando help:
Help on function faz in module __main__:

faz()
    So mostra exemplo de docstring.
```

Docstrings

- Convenção
  - Geralmente tem um texto explicativo
    - Começa com letra maiúscula termina com .
    - Ex.: *"""Esta classe faz isso e aquilo."""*
  - Texto pode ter mais de uma linha
    - Se tiver:
      - Segunda linha é deixada em branco
      - Detalhes são colocados a partir de terceira linha
- Como definir um docstring para um objeto?
  - Incluindo a constante string como primeiro comando da definição de sua classe

Docstrings para classes

- O docstring de uma classe deve ter as seguintes informações:
  - Resumo do propósito e comportamento da classe
  - Métodos públicos da classe e sua descrição
  - Atributos (variáveis) da classe e suas propriedades
  - Caso a classe tenha subclasses, informações sobre a interface com elas
  - Um construtor para a classe
    - Neste caso, têm papel importante os métodos *__init__* e *__main__*

Docstrings para classes

- Construtores
  - Métodos usados para instanciar objetos da classe
  - Inicializam os objetos da classe, atribuindo valor a cada variável da classe
- Parâmetros para o construtor de uma classe devem ser documentados no docstring
  - Python usa o método construtor *__init__*
    - Parecido com construtores usados em outras linguagens de programação orientadas a objeto, como C++ e java

Docstrings para classes

- Diferente da linguagem C, python não possui função *main* (programa principal em C)
  - Código do interpretador Python padrão é escrito em C
    - Ao rodar um programa escrito Python no interpretador Python, é executado o código no nível 0 de indentação
      - É considerado como o programa principal (main)
      - Antes disso, interpretador define algumas variáveis especiais, dentre elas, a variável __name__
      - Se a execução começar pelo programa principal, o interpretador faz: *__name__ = "__main__"*
        - *Se arquivo do Código é importado de outro modulo, o interpretador faz: __name__ = "__nomedomodulo__"*

LSE para pilha em Python

```
#Classe no para LSE
class No_LSE:

  #Método para inicializar objeto nó em LSE
  def __init__(si, dados):
     # Atribui dados
     si.dados = dados
     # Inicializa proximo como nó null
     si.proximo = None

# classe Pilha para LSE
class Pilha:
   #Método para inicializar inicio
   def __init__(si):
      si.inicio = None

   # Método para verificar se pilha está vazia
   def isEmpty (si):
      return si.inicio == None
```

```
#Método para incluir item na pilha
def push(si, dados):
  if si.inicio is None:
     si.inicio = No_LSE(dados)
  else:
     temp = No_LSE(dados)
     temp.proximo = si.inicio
     si.inicio = temp
     print(dados)

# Método para remover itens da pilha
def pop(si):
  if si.isEmpty():
    return
  temp = si.inicio
  si.inicio = si.inicio.proximo
  temp.proximo = None
```


LSE para pilha em Python

```
#Método primeiro item da pilha
  def top(si):
     return si.inicio.dados
```

```
# Método para retornar o tamanho da pilha
def size(si):
   temp = si.inicio
   cont = 0
   while temp is not None:
      cont = cont + 1
      temp = temp.proximo
   return cont
```

```
# Método para imprimir itens da pilha
   def printpilha(si):
     print("Os itens da pilha são:")
     temp=si.inicio
     while (temp != None):
       print(temp.dados,end="<-")
       temp=temp.proximo
```

```
if __name__=='__main__':
   pilha1 = Pilha()
   print("Operacoes de pilha para LSE")
   pilha1.push(4)
   pilha1.push(3)
   pilha1.push(2)
   pilha1.pop()
   pilha1.pop()
   pilha1.push(1)
   print("Topo é: ", pilha1.top())
   print("Tamanho é: ",pilha1.size())
   printpilha(pilha1)
```

Docstrings

LSE para pilha em Python

```
class No_LSE:

    """
    Esta é uma classe para pilhas usando LSE

    Attributos:
      dados (int): valor armazenado em um nó.
      proximo (int): Apontador para o próximo no.
    """

    # Função para inicializar objeto nó
    def __init__(si, dados):
      # Atribui dados
      si.dados = dados
      # Inicializa proximo como nó null
      si.proximo = None

# classe Pilha para LSE
class Pilha:
    # Função para inicializar inicio
    def __init__(si):
      si.inicio  = None
```

```
# Função para verificar se lista está vazia
    def isEmpty(si):
      return si.inicio == None

    # Função para incluir item na pilha
    def push(si, dados):
      if si.inicio  is None:
        si.inicio  =  No_LSE(dados)
        print(dados)
      else:
        temp = No_LSE(dados)
        temp.proximo = si.inicio
        si.inicio  = temp
        print(dados)

    # Função para remover itens da pilha
    def pop(si):
      if si.isEmpty():
        return
      temp = si.inicio
      si.inicio  = si.inicio.proximo
      temp.proximo = None
```


LSE para pilha em Python

```
if __name__=='__main__':
  pilha1 = Pilha()
  print("Operacoes de pilha para LSE")
  pilha1.push(4)
  pilha1.push(3)
  pilha1.push(2)
  pilha1.pop()
  pilha1.pop()
  pilha1.push(1)
  print("Topo é: ", pilha1.top())
  print("Tamanho é: ",pilha1.size())
  printpilha(pilha1)
  print ("\n")
  help(No_LSE)
  print (No_LSE.__doc__)
```

```
Help on class No_LSE in module __main__:

class No_LSE(builtins.object)
 |  Esta é uma classe para pilhas usando
 |  listas simplesmente encadeadas.
 |
 |  Attributos:
 |    dados (int): valor armazenado em um nó.
 |    proximo (int): Apontador para o próximo no.
 |
 |  Methods defined here:
 |
 |  __init__(si, dados)
 |      Initialize self.  See help(type(self))  for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables  (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

  Esta é uma classe para pilhas usando
  listas simplesmente encadeadas.

  Attributos:
    dados (int): valor armazenado em um nó.
    proximo (int): Apontador para o próximo no.
```


Exemplo

```
>>> help(str)
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors are specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
# Truncated for readability
```


LSE para pilha em Python 2

```
#Classe no para LSE
class No_LSE:

  # Método para inicializar objeto nó em LSE
  def inicia-no (si, dados):
    # Atribui dados
    si.dados = dados
    # Inicializa proximo como nó null
    si.proximo = None

# classe Pilha para LSE
class Pilha:
    # Método para inicializar inicio
    def inicia-ini (si):
      si.inicio  = None

    # Método para verificar se pilha está vazia
    def  isEmpty (si):
      return si.inicio == None
```

```
#Método para incluir item na pilha
def push(si, dados):
  if si.inicio is None:
    si.inicio =  No_LSE(dados)
  else:
    temp = No_LSE(dados)
    temp.proximo = si.inicio
    si.inicio = temp
    print(dados)

# Método para remover itens da pilha
def pop(si):
  if si.isEmpty():
    return
  temp = si.inicio
  si.inicio = si.inicio.proximo
  temp.proximo = None
```


LSE para pilha em Python 2

```
#Método primeiro item da pilha
  def top(si):
    return si.inicio.dados
```

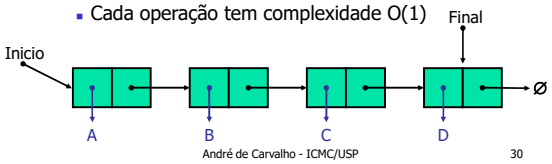
```
 # Método para retornar o tamanho da pilha
def size(si):
  temp = si.inicio
  cont = 0
  while temp is not None:
      cont = cont + 1
      temp = temp.proximo
  return cont
```

```
# Método para imprimir itens da pilha
  def printpilha(si):
    print("Os itens da pilha são:")
    temp=si.inicio
    while (temp != None):
      print(temp.dados,end="<-")
      temp=temp.proximo
```

```
pilha1 = Pilha()
print("Operacoes de pilha para LSE")
pilha1.push(4)
pilha1.push(3)
pilha1.push(2)
pilha1.pop()
pilha1.pop()
pilha1.push(1)
print("Topo é: ", pilha1.top())
print("Tamanho é: ",pilha1.size())
printpilha(pilha1)
```


Fila usando LSE

- Implementação
  - O item no início da fila é armazenado no primeiro nó da lista
  - O item no final da fila é armazenado no último nó da lista
    - Espaço utilizado é O(n)
    - Cada operação tem complexidade O(1)



LSE para fila em Python

```
# classe no para LSE
class No_LSE:

  # Função para inicializar objeto nó
  def __init__ (si, dados):
    # Atribui dados
    si.dados = dados
    # Inicializa proximo como nó null
    si.proximo = None

# classe Fila para LSE
class Fila:
  # Função para inicializar inicio
  def __init__(si):
    si.inicio = si.final  = None

  # Função para verificar se lista está vazia
  def isEmpty(si):
    return si.inicio == None
```

```
# Função para incluir item na fila
def enqueue(si, dados):
  if si.final is None:
    si.inicio = si.final = No_LSE(dados)
  else:
    si.final.proximo  = No_LSE(dados)
    si.final = si.final.proximo

# Função para remover itens da fila
def dequeue(si):
  if si.isEmpty():
    return
  temp = si.inicio
  si.inicio = temp.proximo
  if(si.inicio  == None):
    si.final = None

# Função para imprimir itens da fila
def printqueue(si):
  print("Os itens da fila são:")
  temp=si.inicio
  while temp is not None:
    print(temp.dados,end="->")
    temp=temp.proximo
```

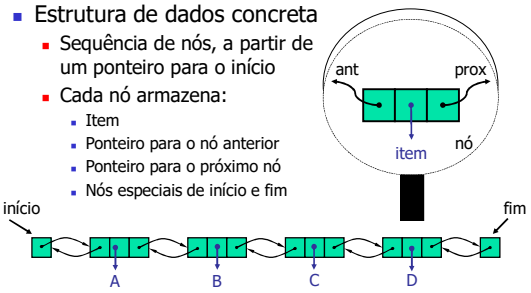

LDE para pilha em Python

```
# Classe nó para LDE
class No_LDE:

  # Função para inicializar o objeto nó
  def __init__(self, dados):
    self.dados = dados # Atribuir dados
    self.proximo = None # Inicializar proximo como null
    self.anterior = None # Inicializar anterior como null

# Classe pilha contém um objeto nó
class Stack:

  # Função para Inicializar inicio
  def __init__(si):
    si.inicio = None

  # Função para verificar se a pilha esta vazia
  def isEmpty(si):
    if si.inicio is None:
      return True
    else:
      return False
```

```
# Função para por item no topo da pilha e retorna-lo
def pop(si):
  if si.inicio is None:
    return None
  elif si.inicio.proximo is None:
    temp = si.inicio.dados
    si.inicio = None
    return temp
  else:
    temp = si.inicio.dados
    si.inicio = si.inicio.proximo
    si.inicio.anterior = None
    return temp

# Função para incluir itens na pilha
def push(si, dados):
  if si.inicio is None:
    si.inicio = Node(dados)
  else:
    new_node = Node(dados)
    si.inicio.anterior = new_node
    new_node.proximo = si.inicio
    new_node.anterior = None
    si.inicio = new_node
```


Lista duplamente encadeada

- Estrutura de dados concreta
  - Sequência de nós, a partir de um ponteiro para o início
  - Cada nó armazena:
    - Item
    - Ponteiro para o nó anterior
    - Ponteiro para o próximo nó
    - Nós especiais de início e fim



LDE para pilha em Python

```
# Função para retornar item do topo da pilha
def top(si):
  return si.inicio.dados

# Função para retornar o tamanho da pilha
def size(si):
  temp = si.inicio
  count = 0
  while temp is not None:
    count = count + 1
    temp = temp.proximo
  return count

# Função para imprimir a pilha
def printstack(si):
  print("os itens na pilha são:")
  temp = si.inicio
  while temp is not None:
    print(temp.dados, end ="->")
    temp = temp.proximo
```


Exercício

- Refazer com os exercícios feitos com matrizes para pilhas e filas implementadas usando LSE
- Adaptar códigos em Python para pilhas e filas usando LSE para pilhas e filas usando LDE
  - Testar e complementar código do próximo slide para filas
- Comparar custo computacional das filas e pilhas usando:
  - LSE
  - LDE

Questions