INSTITUTO FEDERAL DE ALAGOAS – IFAL / CAMPUS MACEIÓ
Curso Bacharelado em Sistemas de Informação
Disciplina de Estruturas de Dados

Prof. MSc. Ricardo Nunes
Ricardo (arroba) ifal.edu.br

Listas Encadeadas (linked lists)

Objetivos

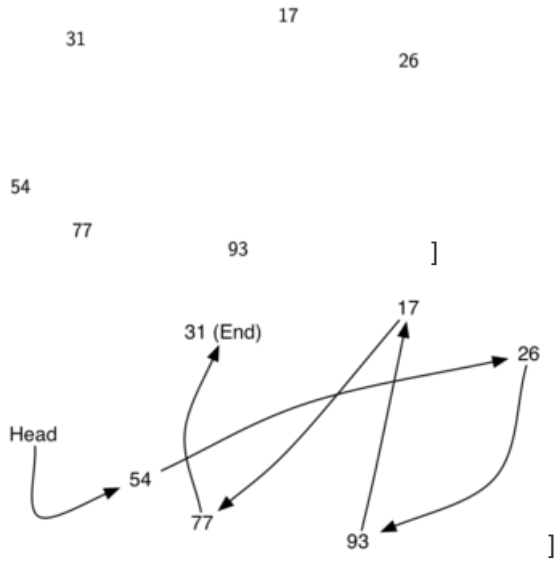
1. Entender o conceitos de listas encadeadas
2. Ser capaz de implementar listas encadeadas
3. Ser capaz de avaliar o desempenho das listas encadeadas

Conteúdo

1. Conceito
2. Implementação da Classe Noh
3. Implementação da Classe Lista Encadeada e seus métodos
4. Análise de desempenho

Listas Encadeadas Não ordenadas

- Mantém a posição de relativa entre os itens
- Não estão em posições contíguas na memória



Simulação ?]

A classe Nó

- O bloco básico para construir um lista encadeada é um Nó

- Cada nó armazena duas informações:
- o item da lista em si.

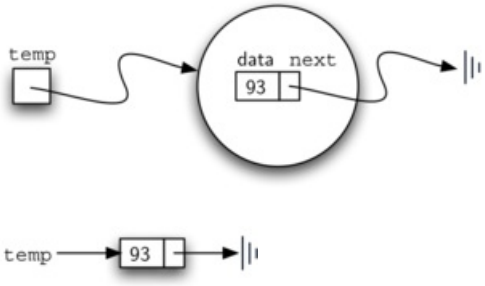
- referência ao próximo nó da lista

Implementação da classe Noh

```
class Noh:
    def __init__(self,valor_inicial):
        self._dados = valor_inicial
        self._proximo = None

    def getData(self):
        return self._dados

    def getNext(self):
        return self._proximo

    def setData(self, novo_valor):
        self._dados =  novo_valor

    def setNext(self, novo_proximo):
        self._proximo = novo_proximo
```

Exemplo de uso da classe Noh

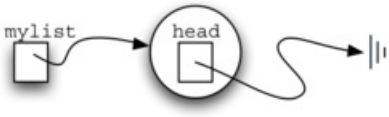
```
from Noh import *
noh1 = Noh(93)
print(noh1.getData())
print(noh1) #?
```



Lista não ordenada (construtor)

- uma lista não ordenada será construída a partir de uma coleção de nós (noh), cada um ligado (encadeado) com o próximo nó através de referência explícita

```
class ListaNaoOrdenada:
    def __init__(self): #construtor
        self.head = None

minha_lista =ListaNaoOrdenada() #exemplo de uso
```

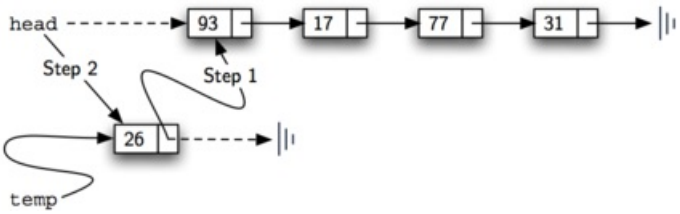


- a classe lista em si não contém nó algum, somente apontador (head) para o primeiro nó da lista.

Lista não ordenada (is_empty)

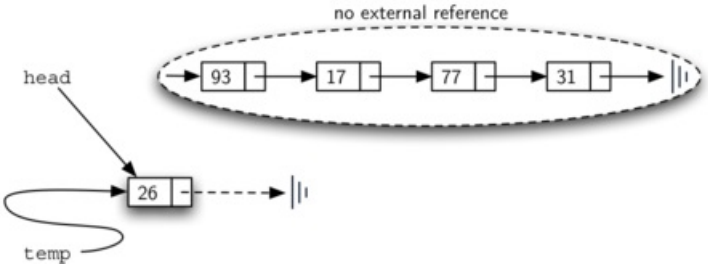
```
class ListaNaoOrdenada:
    def __init__(self): #construtor
        self.head = None

    def is_empty(self):  #<<<<
        return self.head == None

minha_lista = ListaNaoOrdenada() #exemplo de uso
minha_lista.is_empty()
```

Lista não ordenada (add)




```
def add(self,item):
    temp = Noh(item)
    temp.setNext(self.head)
    self.head = temp
```

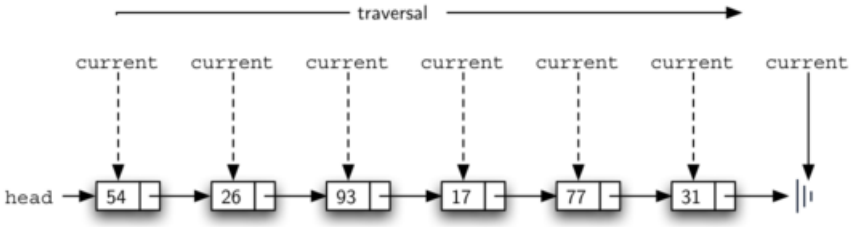
Se a linha 3 e a linha 4 forem invertidas, o que acontece?



Lista não ordenada (percorrer a lista)

- size(), search e remove() são baseado na técnica de percorrer a lista
- visita cada nó da lista

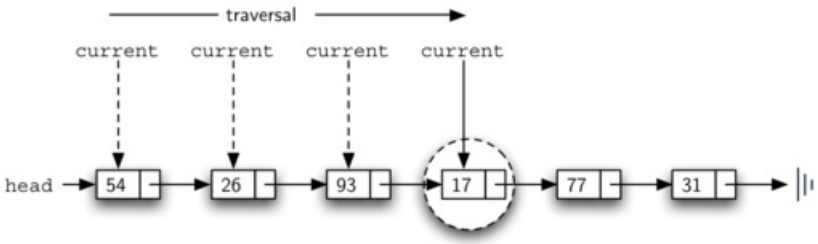
```
def size(self):
    atual    = self.head
    contador = 0
    while atual != None:
        contador = contador + 1
        atual = atual.getNext()
    return contador
```



Lista não ordenada (search)

```
def search(self,item):
    atual    = self.head #atual == temp
    encontrou = False
    while atual != None and not encontrou:
        if atual.getData() == item:
            encontrou = True
        else:
            atual = atual.getNext()

    return encontrou
```

O que muda para size()?



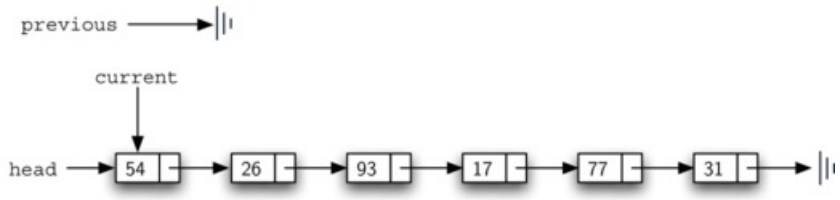
Lista não ordenada (remove)

Passos: * percorrer a lista para encontrar o elemento a ser removido * removê-lo!

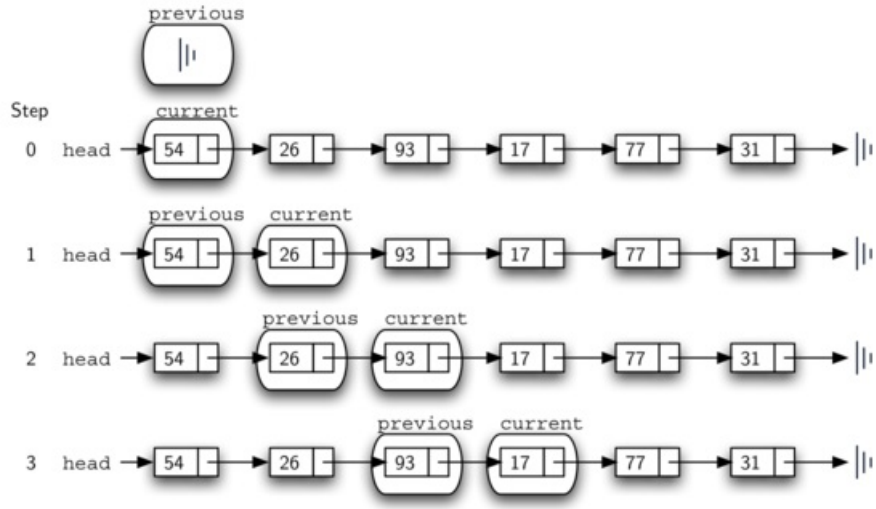
```
def remove(self,item):
    atual = self.head
    anterior = None
    encontrou = False
    while not encontrou: #percorre a lista
        if atual.getData() == item:
            encontrou = True
        else:
            anterior = atual
            atual = atual.getNext()

    if anterior == None:
        self.head = atual.getNext()
    else:
        anterior.setNext(atual.getNext())
```

Lista não ordenada (remove) [2]

estado inicial

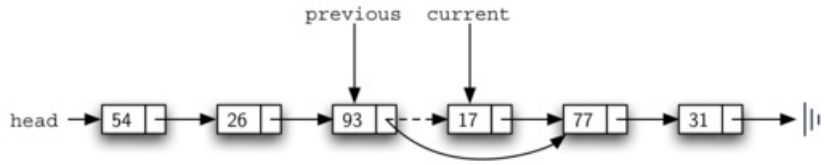


procurando o elemento a ser removido

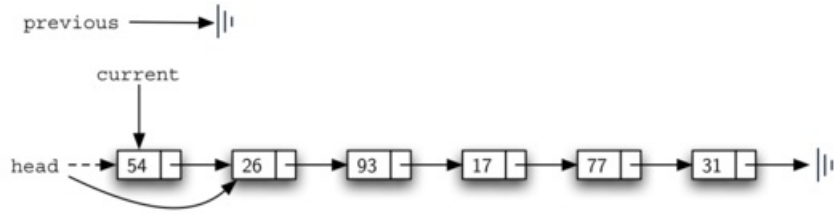


Lista não ordenada (remove) [3]

situação 1: *não* é o primeiro elemento da lista



situação 2: *é* o primeiro elemento da lista



Análise do desempenho da implementação da lista encadeada

Operação	Tempo de Execução
add	O(1)
remove	O(n)
search	O(n)
is_empty()	O(1)
size()	O(n)

Ferramentas indicadas

- Visualize em https://visualgo.net/pt/list
- PythonTutor.com

Para estudar

- Filas
  - Seções 3.19 a 3.21 do livro [5] https://panda.ime.usp.br/pythonds/static/pythonds_pt/index.html
  - Seção 2.1 do livro [4]
  - Capítulo 17 (Litas encadeadas) do livro [2] (texto bastante resumido!) https://chevitarese.files.wordpress.com/2009/09/aprendacomputaocompython3k.pdf

Referências

1. Tradução do livro *Howto Think Like a Computer Scientist: Interactive Version*, de Brad Miller e David Ranum. link: https://panda.ime.usp.br/pensepy/static/pensepy/index.html
2. Allen Downey, Jeff Elkner and Chris Meyers. *Aprenda Computação com Python 3.0.* link: https://chevitarese.files.wordpress.com/2009/09/aprendacomputaocompython3k.pdf
3. SANTOS, A. C. *Algoritmo e Estrutura de Dados I*. 2014. Disponível em http://educapes.capes.gov.br/handle/capes/176522
4. SANTOS, A. C. *Algoritmo e Estrutura de Dados II*. 2014. Disponível em 2014. Disponível em https://educapes.capes.gov.br/handle/capes/176557
5. Tradução do livro [5] *Problem Solving with Algorithms and Data Structures using Python* de Brad Miller and David Ranum. link: https://panda.ime.usp.br/pythonds/static/pythonds_pt/index.html
6. Brad Miller and David Ranum. *Problem Solving with Algorithms and Data Structures using Python* link: https://panda.ime.usp.br/pythonds/static/pythonds_pt/index.html
7. Caelum. Algoritmos e Estruturas Dados em Java. Disponível em https://www.caelum.com.br/download/caelum-algoritmos-estruturas-dados-java-cs14.pdf

That’s all Folks