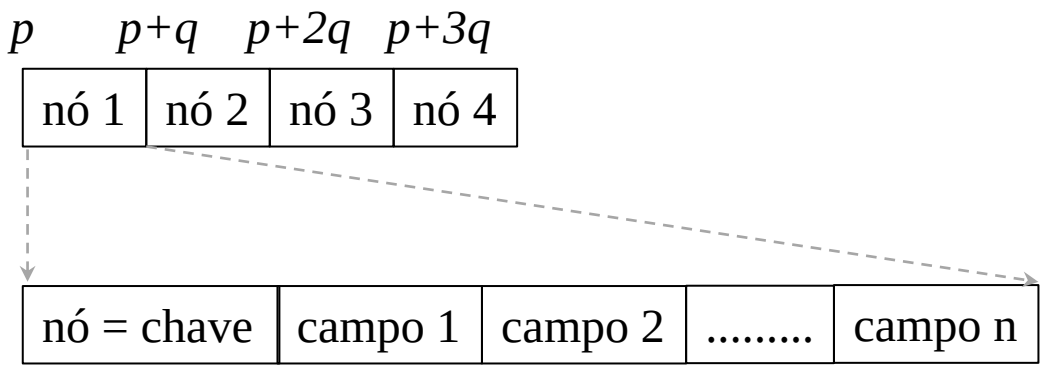
Complexidade dos algoritmos de busca

profº Mauricio Conceição Mario

Listas lineares em alocação sequencial

→ Uma ***lista linear*** é caracterizada pelo fato de seus nós estarem em posições contíguas, ou seja, o endereço do (*j + 1*)-*ésimo* nó da lista se encontra ***q*** unidades adiante do nó correspondente ao *j-ésimo*. A constante ***q*** é o número de palavras de memória que cada nó ocupa. Exemplo:

p = posição de memória



(Szwarcfiter & Markenzon, 2020)

***Busca em listas lineares em alocação sequencial → manipulação para retornar os campos das chaves*:**

```
L = [chave 1, campo 1, campo 2, ...,campo m, chave 2, ...]
n = dimensão (L)
        função busca_chave(nó)
            i := 0
            enquanto (i < n)  fazer:
              se (L[i] == nó ) fazer:
                retorna L[i], L[i + 1], L[i + 2], L[i + m]
              senão
                i := i + 1
busca_chave(x)
```

(Szwarcfiter & Markenzon, 2020)

***Exercício*:**

a) Adaptar o código a seguir para que possam ser retornados, além dos nós, os campos relativos a cada um deles;
b) Considerando na chamada à função uma única busca de nó por vez, alterar a ordem das buscas, assim como a dificuldade de encontrar os nós (ausência do nó na lista por exemplo), fazendo a estimativa de tempo para estes casos.

Lista linear em alocação sequencial → algoritmo de busca executado 1 vez

Lista_linear_sequencial_I.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 13 19:42:53 2022

@author: cmari
"""
import time

L = [1, "Goiânia", "Goiás", 2, "Bonito", "Mato Grosso", 3, "Carangola", "Minas Gerais", 4, "Peruíbe", "São Paulo", 20, "Serra", "Espírito Santo"]
n = len(L)
print ("dimensão da lista L = ", n)
print ("L[0]", L[0])
print ("L[3]", L[3])
print ("L[6]", L[6])
print ("L[9]", L[9])
print ("L[4]", L[4])
print ("L[4][2]", L[4][2])
print ("L[2][1]", L[2][1])
print("\n")
def busca_linear(x):
  for p in range(1):
       i = 0
       while i < n:
         if L[i] == x:
           return i
         else:
            i = i + 1
tempo_inicial = time.time()
print("índice da chave (5) = ", busca_linear(5))
print("índice da chave (1) = ", busca_linear(1))
print("índice da chave (4) = ", busca_linear(4))
print("índice da chave (3) = ", busca_linear(3))
print("índice da chave (2) = ", busca_linear(2))
print("índice da chave (0) = ", busca_linear(0))
print("índice da chave (20) = ", busca_linear(20))
tempo_final = time.time()
print("tempo de execução = ", tempo_final - tempo_inicial)
```

Console 1/A

```
In [23]: runfile('C:/Users/cmari/.spyder-py3/
Lista_linear_sequencial_I.py', wdir='C:/
dimensão da lista L =  15
L[0] 1
L[3] 2
L[6] 3
L[9] 4
L[4] Bonito
L[4][2] n
L[2][1] o


índice da chave (5) =  None
índice da chave (1) =  0
índice da chave (4) =  9
índice da chave (3) =  6
índice da chave (2) =  3
índice da chave (0) =  None
índice da chave (20) =  12
tempo de execução =  0.0
```

n = 15
x = 5
i = 0
enquanto (i < 15)
se L[i] = = x (L[0] = 1)
retorna i
senão
i = 1 + 1 .....
i = 5
L[5] = “Mato Grosso” ⇒ nó não achado

x = 1
i = 0
enquanto (i < 15)
se L[i] = = x (L[0] = 1)
retorna i = 0 ⇒ índice do nó = 1

Lista linear → algoritmo de busca retornando os campos das chaves

C:\Users\cmari\.spyder-py3\Lista_linear_sequencial_IV.py

inverte_sequência.py | datetime.py | soma_matricial.py | produto_matricial_3x3.py | Lista_linear_sequencial_II.py | busca_binária_II.py | busca_binária_random.py | Lista_linear_sequencial_IV.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 13 19:42:53 2022

@author: cmari
"""
import time

L = [1, "Goiânia", "Goiás", 2, "Bonito", "Mato Grosso", 3, "Carangola", "Minas Gerais", 4, "Peruíbe", "São Paulo", 20, "Serra", "Espírito Santo"]
n = len(L)
print ("dimensão da lista L = ", n)
print("\n")
def busca_linear(x):
  for p in range(1):
        i = 0
        while i < n:
          if L[i] == x:
            return L[i], L[i+1], L[i+2]
            #i = n + 1
          else:
             i = i + 1
tempo_inicial = time.time()
print("chave (1) = ", busca_linear(1))
print("chave (2) = ", busca_linear(2))
print("chave (3) = ", busca_linear(3))
print("chave (4) = ", busca_linear(4))
print("chave (20) = ", busca_linear(20))
print("chave (0) = ", busca_linear(0))
print("chave (6) = ", busca_linear(6))
tempo_final = time.time()
print("tempo de execução = ", tempo_final - tempo_inicial)
```

Console 1/A

```
In [4]: runfile('C:/Users/cmari/.spyder-py3/Lista_linear_sequencial_IV.py', wdir='C:/Users/cmari/.spyder-py3')
dimensão da lista L =  15


chave (1) =  (1, 'Goiânia', 'Goiás')
chave (2) =  (2, 'Bonito', 'Mato Grosso')
chave (3) =  (3, 'Carangola', 'Minas Gerais')
chave (4) =  (4, 'Peruíbe', 'São Paulo')
chave (20) =  (20, 'Serra', 'Espírito Santo')
chave (0) =  None
chave (6) =  None
tempo de execução =  0.0009980201721191406
```

Referências Bibliográficas

- Estruturas de Dados e Seus Algoritmos
  Jayme L. Szwarcfiter & Lilian Markenzon
  3ª edição – editora *gen* LTC – 2010 - 2020

- Matemática Avançada para Engenharia
  Dennis G. Zill & Michael R. Cullen
  Álgebra Linear e Cálculo Vetorial (2) – 3ª edição, editora Bookman – 2009

- Algoritmos – Teoria e Prática
  Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.
  3ª edição – editora Elsevier - *gen* LTC – 2012