MC-102 — Aula 15
Ordenação – Selection Sort e Bubble Sort

Eduardo C. Xavier

Instituto de Computação – Unicamp

24 de Abril de 2018

Roteiro

Ordenação

- Vamos estudar alguns algoritmos para o seguinte problema:

Dado uma coleção de elementos com uma relação de ordem entre si, devemos gerar uma saída com os elementos ordenados.

- Nos nossos exemplos usaremos um vetor de inteiros para representar tal coleção.
  - É claro que quaisquer inteiros possuem uma relação de ordem entre si.
- Apesar de usarmos inteiros, os algoritmos servem para ordenar qualquer coleção de elementos que possam ser comparados.

Ordenação

- O problema de ordenação é um dos mais básicos em computação.
  - Mas muito provavelmente é um dos problemas com o maior número de aplicações diretas ou indiretas (como parte da solução para um problema maior).
- Exemplos de aplicações diretas:
  - Criação de *rankings*, Definir preferências em atendimentos por prioridade, Criação de Listas etc.
- Exemplos de aplicações indiretas:
  - Otimizar sistemas de busca, manutenção de estruturas de bancos de dados etc.

Selection Sort

- Seja **vet** um vetor contendo números inteiros.
- Devemos deixar **vet** em ordem crescente.
- A idéia do algoritmo é a seguinte:
  - Ache o menor elemento a partir da posição 0. Troque então este elemento com o elemento da posição 0.
  - Ache o menor elemento a partir da posição 1. Troque então este elemento com o elemento da posição 1.
  - Ache o menor elemento a partir da posição 2. Troque então este elemento com o elemento da posição 2.
  - E assim sucessivamente...

Selection-Sort

Exemplo: (5,3,2,1,90,6).

Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).

Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).

Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).

Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).

Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

Exemplo: (5,3,2,1,90,6).
Iteração 0. Acha menor: (5,3,2,1,90,6). Faz troca: (1,3,2,5,90,6).
Iteração 1. Acha menor: (1,3,2,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 2. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 3. Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,90,6).
Iteração 5: Acha menor: (1,2,3,5,90,6). Faz troca: (1,2,3,5,6,90).

Selection-Sort

- Como achar o menor elemento a partir de uma posição inicial?
- Vamos achar **o índice** do menor elemento em um vetor, a partir de uma posição inicial **ini**:

```
menor = ini
for j in range(ini+1, len(v)):
    if v[menor] > v[j]:
        menor = j
```

Selection-Sort

- Como achar o menor elemento a partir de uma posição inicial?
- Vamos achar **o índice** do menor elemento em um vetor, a partir de uma posição inicial **ini**:

```
menor = ini
for j in range(ini+1, len(v)):
    if v[menor] > v[j]:
        menor = j
```

Selection-Sort

- Criamos então uma função que retorna o índice do elemento mínimo de um vetor, a partir de uma posição **ini** passada por parâmetro:

```
def ind_menor(v, ini):
    menor = ini
    for j in range(ini+1, len(v)):
        if v[menor] > v[j]:
            menor = j
    return menor
```

Selection-Sort

- Dada a função anterior para achar o índice do menor elemento, como implementar o algoritmo de ordenação?
- Ache o menor elemento a partir da posição 0, e troque com o elemento da posição 0.
- Ache o menor elemento a partir da posição 1, e troque com o elemento da posição 1.
- Ache o menor elemento a partir da posição 2, e troque com o elemento da posição 2.
- E assim sucessivamente...

Selection-Sort

```
def selectionSort(v):
    for i in range(len(v)):
        menor = ind_menor(v, i) #Acha pos. menor ele. a partir de i
        aux = v[i]
        v[i] = v[menor] #Troca v[menor] com v[i]
        v[menor] = aux
```

Bubble-Sort

- Trocar o conteúdo de 2 variáveis é muito comum.

```
a = 4
b = 10

#troca a com b
aux = a
a = b
b = aux
```

- Em python existe uma forma sucinta de escrever esta troca:

```
a = 4
b = 10

#troca a com b
a, b = b, a
```

- Podemos usar esta forma sucinta nos nossos algoritmos de ordenação.

Selection-Sort

Antes:

```
def selectionSort(v):
    for i in range(len(v)):
        menor = ind_menor(v, i) #Acha pos. menor ele. a partir de i
        aux = v[i]
        v[i] = v[menor] #Troca v[menor] com v[i]
        v[menor] = aux
```

Depois:

```
def selectionSort(v):
    for i in range(len(v)):
        menor = ind_menor(v, i) #Acha pos. menor ele. a partir de i
        v[i], v[menor]  = v[menor], v[i] #Troca v[menor] com v[i]
```

Selection-Sort

Com as funções anteriores implementadas podemos executar o exemplo:

```
import random

def main():
    v = [ random.randint(0, 100) for i in range(10)]
    print(v)
    selectionSort(v) #lista é mutável por isso v será ordenada
    print(v)

def ind_menor(v, ini):
    menor = ini
    for j in range(ini+1, len(v)):
        if v[menor] > v[j]:
            menor = j
    return menor

def selectionSort(v):
    for i in range(len(v)):
        menor = ind_menor(v, i) #Acha pos. menor ele. a partir de i
        v[i], v[menor]  = v[menor], v[i] #Troca v[menor] com v[i]

main()
```

Selection-Sort

- O uso da função para achar o índice do menor elemento não é estritamente necessária.
- Podemos refazer a função selectionSort como segue:

```
def selectionSort(v):
    for i in range(len(v)):
        #Acha pos. menor ele. a partir de i
        menor = i
        for j in range(i, len(v)):
            if v[menor] > v[j]:
                menor = j
        v[i], v[menor]  = v[menor], v[i] #Troca v[menor] com v[i]
```

Selection-Sort

- O uso da função para achar o índice do menor elemento não é estritamente necessária.
- Podemos refazer a função selectionSort como segue:

```
def selectionSort(v):
    for i in range(len(v)):
        #Acha pos. menor ele. a partir de i
        menor = i
        for j in range(i, len(v)):
            if v[menor] > v[j]:
                menor = j
        v[i], v[menor]  = v[menor], v[i] #Troca v[menor] com v[i]
```

Selection-Sort

Antes:

```
def selectionSort(v):
    for i in range(len(v)):
        menor = ind_menor(v, i) #Acha pos. menor ele. a partir de i
        v[i], v[menor]  = v[menor], v[i] #Troca v[menor] com v[i]
```

Depois:

```
def selectionSort(v):
    for i in range(len(v)):
        #Acha pos. menor ele. a partir de i
        menor = i
        for j in range(i, len(v)):
            if v[menor] > v[j]:
                menor = j
        v[i], v[menor]  = v[menor], v[i] #Troca v[menor] com v[i]
```

Bubble-Sort

- Seja **vet** um vetor contendo n números inteiros.
- Devemos deixar **vet** em ordem crescente.
- O algoritmo faz algumas iterações repetindo o seguinte:
  - Compare $vet[0]$ com $vet[1]$ e troque-os se $vet[0] > vet[1]$.
  - Compare $vet[1]$ com $vet[2]$ e troque-os se $vet[1] > vet[2]$.
  - .....
  - Compare $vet[n-2]$ com $vet[n-1]$ e troque-os se $vet[n-2] > vet[n-1]$.

Após uma iteração repetindo estes passos o que podemos garantir???

- O maior elemento estará na posição correta!!!

Bubble-Sort

- Seja **vet** um vetor contendo n números inteiros.
- Devemos deixar **vet** em ordem crescente.
- O algoritmo faz algumas iterações repetindo o seguinte:
  - Compare $vet[0]$ com $vet[1]$ e troque-os se $vet[0] > vet[1]$.
  - Compare $vet[1]$ com $vet[2]$ e troque-os se $vet[1] > vet[2]$.
  - .....
  - Compare $vet[n-2]$ com $vet[n-1]$ e troque-os se $vet[n-2] > vet[n-1]$.

  Após uma iteração repetindo estes passos o que podemos garantir???
  - O maior elemento estará na posição correta!!!

Bubble-Sort

- Após uma iteração de trocas, o maior elemento estará na última posição.
- Após outra iteração de trocas, o segundo maior elemento estará na posição correta.
- E assim sucessivamente.
- Quantas iterações repetindo estas trocas precisamos para deixar o vetor ordenado?

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:

$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

Exemplo: (5,3,2,1,90,6).
Valores sublinhados estão sendo comparados:
$(\underline{5}, \underline{3}, 2, 1, 90, 6)$
$(3, \underline{5}, \underline{2}, 1, 90, 6)$
$(3, 2, \underline{5}, \underline{1}, 90, 6)$
$(3, 2, 1, \underline{5}, \underline{90}, 6)$
$(3, 2, 1, 5, \underline{90}, \underline{6})$
$(3, 2, 1, 5, 6, 90)$

- Isto termina a primeira iteração de trocas. Temos que repetir todo o processo mais 4 vezes!!!
- Mas notem que não precisamos mais avaliar a última posição!

Bubble-Sort

- O código abaixo realiza as trocas de uma iteração.
- São comparados e trocados, os elementos das posições: 0 e 1; 1 e 2; ...; $i-1$ e i.
- Assumimos que de $(i+1)$ até $(n-1)$, o vetor já tem os maiores elementos ordenados.

```
for j in range(0, i): #faz trocas até pos. i.
    if v[j] > v[j+1]:
        v[j], v[j+1] = v[j+1], v[j]
```

Bubble-Sort

- O código abaixo realiza as trocas de uma iteração.
- São comparados e trocados, os elementos das posições: 0 e 1; 1 e 2; ...; $i-1$ e i.
- Assumimos que de $(i+1)$ até $(n-1)$, o vetor já tem os maiores elementos ordenados.

```
for j in range(0, i): #faz trocas até pos. i.
    if v[j] > v[j+1]:
        v[j], v[j+1] = v[j+1], v[j]
```

Bubble-Sort

```
def bubbleSort(v):
    for i in range(len(v)-1, 0, -1):  #i = n-1,n-2,...,1
        #faz trocas até pos. i
        for j in range(0, i): #j = 0, 1,...,i-1.
            if v[j] > v[j+1]:
                v[j], v[j+1] = v[j+1], v[j]
```

Bubble-Sort

- Note que as trocas na primeira iteração ocorrem até a última posição.
- Na segunda iteração ocorrem até a penúltima posição.
- E assim sucessivamente.
- Por que?

Bubble-Sort

```
import random

def main():
    v = [ random.randint(0, 100) for i in range(10)]
    print(v)
    bubbleSort(v)
    print(v)

def bubbleSort(v):
    for i in range(len(v)-1, 0, -1):  #i = n-1,n-2,...,1
        #faz trocas até pos. i
        for j in range(0, i): #j = 0, 1,...,i-1.
            if v[j] > v[j+1]:
                v[j], v[j+1] = v[j+1], v[j]
main()
```

Bubble-Sort

- Python já possui um algoritmo de ordenação implementado no método **sort** de listas:

```
>>> l = [86, 31, 23, 71, 58, 64, 78, 51, 25, 38]
>>> l.sort()
>>> l
[23, 25, 31, 38, 51, 58, 64, 71, 78, 86]
```

- Então por que estamos aprendendo estes algoritmos de ordenação?
- O foco do curso é a criação de algoritmos, e não aprender tudo sobre uma linguagem de programação (Python neste caso).
- Entender como funcionam algoritmos básicos de ordenação é fundamental para exercício da lógica de programação e criação de algoritmos.

Exercício

Altere os algoritmos vistos nesta aula para que estes ordenem um vetor de inteiros em ordem decrescente ao invés de ordem crescente.

Exercício

No algoritmo SelectionSort, o laço principal pode ser executado de i=0 até $i = \text{len}(v) - 2$ e não $i = \text{len}(v) - 1$. Por que?

```
def selectionSort(v):
    for i in range(len(v)-1): #não precisa ser range(len(v)). Por que?
        #Acha pos. menor ele. a partir de i
        menor = i
        for j in range(i+1, len(v)):
            if v[menor] > v[j]:
                menor = j
        v[i], v[menor] = v[menor], v[i]
```