CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho, Bruno Müller Jr, Carmem Hara

Departamento de Informática/UFPR

18 de agosto de 2020

Resumo

Ordenação em vetores

Objetivos da aula

- Apresentar o método de ordenação por seleção
- Analisar brevemente o método

Motivação

- Ordenar vetores é extremamente importante em computação
- É muito comum que uma saída de um programa seja ordenada
- Este é um assunto com muitos resultados teóricos interessantes
- Tema de boa parte das disciplinas:
  - Algoritmos II
  - Análise de Algoritmos
- Veremos somente um dos métodos

Ordenação por seleção

- É o método mais simples de ser compreendido, pois já há uma base estudada
- A implementação dele é bastante simples também
- Princípio:
  - *selecionar* os elementos corretos para cada posição do vetor
  - iniciando da primeira e terminando na última

Método de $N - 1$ etapas

Seja N o número de elementos do vetor.

- O método consiste de $N - 1$ etapas
- A cada etapa i:
  - seleciona-se o i-ésimo menor elemento do vetor
  - coloca-se este elemento na posição i do vetor

Etapa 1

Seja o vetor inicial:

1	2	3	4	5	6	7	8	9	10
15	12	27	23	7	2	0	18	19	21

A *posição* do menor de todos é selecionada

1	2	3	4	5	6	7	8	9	10
15	12	27	23	7	2	0	18	19	21

O *conteúdo* desta posição é trocado com o conteúdo da primeira posição, por isso precisamos dos índices

1	2	3	4	5	6	7	8	9	10
0	12	27	23	7	2	15	18	19	21

Observação sobre a etapa 1

1	2	3	4	5	6	7	8	9	10
0	12	27	23	7	2	15	18	19	21

- A ordenação do primeiro elemento é *total*
- Este elemento não sairá mais desta posição
- Vamos anotar isso em cor verde

1	2	3	4	5	6	7	8	9	10
0	12	27	23	7	2	15	18	19	21

Etapa 2

O vetor ao final da etapa 1 está assim:

1	2	3	4	5	6	7	8	9	10
0	12	27	23	7	2	15	18	19	21

A *posição* do segundo menor de todos é selecionada, mas *a partir* da posição 2, pois o primeiro já está em seu lugar definitivo

1	2	3	4	5	6	7	8	9	10
0	12	27	23	7	2	15	18	19	21

O *conteúdo* desta posição é trocado com o conteúdo da segunda posição

1	2	3	4	5	6	7	8	9	10
0	2	27	23	7	12	15	18	19	21

Observação sobre a etapa 2

- A ordenação dos dois primeiros elementos é *total*
- Eles não sairão mais destas posições

1	2	3	4	5	6	7	8	9	10
0	2	27	23	7	12	15	18	19	21

Etapa 3

O vetor ao final da etapa 2 está assim:

1	2	3	4	5	6	7	8	9	10
0	2	27	23	7	12	15	18	19	21

A *posição* do terceiro menor de todos é selecionada, mas *a partir* da posição 3, pois os dois primeiros já estão em seus lugares definitivos

1	2	3	4	5	6	7	8	9	10
0	2	27	23	7	12	15	18	19	21

O *conteúdo* desta posição é trocado com o conteúdo da terceira posição

1	2	3	4	5	6	7	8	9	10
0	2	7	23	27	12	15	18	19	21

Observação sobre a etapa 3

- A ordenação dos três primeiros elementos é *total*
- Eles não sairão mais destas posições

1	2	3	4	5	6	7	8	9	10
0	2	7	23	27	12	15	18	19	21

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	27	23	15	18	19	21

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	23	27	18	19	21

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	27	23	19	21

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	23	27	21

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	27	23

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27

Continuando...

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27

Etapa 9

O vetor ao final da etapa 9 está assim:

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27

Como os 9 primeiros elementos estão em suas posições definitivas, por consequência lógica, o décimo e último também está!
Logo, o vetor foi ordenado em 9 etapas ($N - 1$).

1	2	3	4	5	6	7	8	9	10
0	2	7	12	15	18	19	21	23	27

Algoritmo

```
1  procedure selecao (var v: vetor_r; n: integer);
2  var i, j, pos_menor: integer; aux: real;
3
4  begin
5      for i:= 1 to n-1 do
6      begin
7          (* acha a posicao do menor a partir de i *)
8          pos_menor:= i;
9          for j:= i+1 to n do (* inicia a partir de i+1 *)
10             if v[j] < v[pos_menor] then
11                 pos_menor:= j;
12
13         aux:= v[pos_menor]; (* troca os elementos *)
14         v[pos_menor]:= v[i];
15         v[i]:= aux;
16     end;
17 end;
```

Primeira observação

- Em cada etapa i, a ordenação dos primeiros $i - 1$ elementos é definitiva (ordenação total)

Segunda observação

- A busca pelo menor elemento em cada etapa i exige percorrer um vetor de $N - i$ elementos
- Como isto é feito N vezes, então o número de comparações feitas na parte mais interna do algoritmo é sempre $\frac{N(N-1)}{2}$
- Comportamento quadrático para as comparações, *em qualquer caso*, não importando a forma com que os dados estejam organizados na entrada!!!

De fato

1. 9 comparações
2. 8 comparações
3. 7 comparações
4. 6 comparações
5. 5 comparações
6. 4 comparações
7. 3 comparações
8. 2 comparações
9. 1 comparação

Ou seja: $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$

$$\sum_{i=1}^{N-1} i = \frac{N(N-1)}{2} = \frac{N^2 - N}{2}$$

Terceira observação

- As trocas de posições no vetor ocorrem no laço mais externo, por isto o número total de trocas feitas pelo algoritmo é sempre $N - 1$
- Isto ocorre *em qualquer caso*, não importando a forma como os dados estão organizados na entrada

Em resumo...

- Este método exigiu algo próximo de n^2 comparaçoes e n trocas
- Em Algoritmos II veremos outros métodos interessantes
- Será que algum deles é melhor?

Fim do tópico

- este material está no livro no capítulo 9, seção 9.6.1

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`