Introdução: Análise de Complexidade de Algoritmos (2)

Prof. Renê R. Veloso

Slides dos autores: Cormen e Ziviani

Exemplo - Maior e Menor Elemento (1)

- Encontrar o maior e o menor elemento de $A[1..n], n \geq 1$.
- Um algoritmo simples pode ser derivado do algoritmo para achar o maior elemento.

```
void MaxMin1(TipoVetor A, int *Max, int *Min)
{ int i; *Max = A[0]; *Min = A[0];
  for (i = 1; i < N; i++)
    { if (A[i] > *Max) *Max = A[i];
      if (A[i] < *Min) *Min = A[i];
    }
}
```

Custo	Vezes
C1	1
C2	N
C3	N-1
C4	N-1

f(n) = c2*N+c3*(N-1)+c4*(N-1) = (c2+c3+c4)N-c3-c4 = aN+b

Neste caso, MaxMin1 é da **ordem de *3N-2***, para qualquer caso.

Exemplo - Maior e Menor Elemento (2)

```
void MaxMin2(TipoVetor A, int *Max, int *Min)
{ int i; *Max = A[0]; *Min = A[0];
  for (i = 1; i < N; i++)
    { if (A[i] > *Max) *Max = A[i];
      else if (A[i] < *Min) *Min = A[i];
    }
}
```

- Já MaxMin2 evita que as duas comparações sejam feitas sempre.
- Assim temos que:
  - Melhor caso: ***f(n) = 2n – 1*** (elementos em ordem crescente);
  - Pior caso: ***f(n) = 3n – 2*** (elementos em ordem decrescente);
  - Caso médio: ***f(n) = n + (n-1)/2 + n-1 = 5n/2 - 3/2***
    - *A[i] é maior do que Max a metade das vezes.*

Exemplo - Maior e Menor Elemento (2)

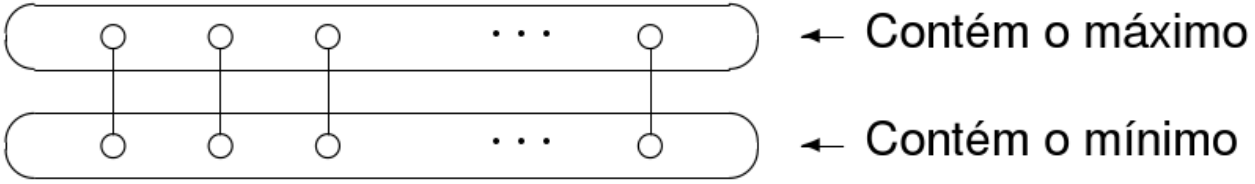
```
void MaxMin2(TipoVetor A, int *Max, int *Min)
{ int i; *Max = A[0]; *Min = A[0];
  for (i = 1; i < N; i++)
    { if (A[i] > *Max) *Max = A[i];
      else if (A[i] < *Min) *Min = A[i];
    }
}
```

- Se considerarmos somente o **número de comparações** realizadas sobre o vetor de entrada:
  - Melhor caso: $f(n) = n - 1$ (elementos em ordem crescente);
  - Pior caso: $f(n) = 2(n - 1)$ (elementos em ordem decrescente);
  - Caso médio: $f(n) = (n-1)/2 + n-1 = 3n/2 - 3/2$

Exemplo - Maior e Menor Elemento (3)

- Considerando o número de comparações realizadas, existe a possibilidade de obter um algoritmo mais eficiente:

  1) Compare os elementos de A aos pares, separando-os em dois subconjuntos (maiores em um e menores em outro), a um custo de $\lceil n/2 \rceil$ comparações.
  2) O máximo é obtido do subconjunto que contém os maiores elementos, a um custo de $\lceil n/2 \rceil - 1$ comparações.
  3) O mínimo é obtido do subconjunto que contém os menores elementos, a um custo de $\lceil n/2 \rceil - 1$ comparações.



Exemplo - Maior e Menor Elemento (4)

```
void MaxMin3(TipoVetor A, int *Max, int *Min)
{ int i, FimDoAnel;
  if ((N & 1) > 0) { A[N] = A[N - 1]; FimDoAnel = N;}
  else FimDoAnel = N - 1;
  if (A[0] > A[1])                                   ---> 1ª comparação
  { *Max = A[0]; *Min = A[1]; }
  else { *Max = A[1]; *Min = A[0]; }
  i = 3;
  while (i <= FimDoAnel)
    { if (A[i - 1] > A[i])                           ---> 2ª comparação
      { if (A[i - 1] > *Max) *Max = A[i - 1];
        if (A[i] < *Min) *Min = A[i]; }
      else { if (A[i - 1] < *Min) *Min = A[i - 1];
             if (A[i] > *Max) *Max = A[i]; }
      i += 2;
    }
}
```

Comparação entre MaxMin1, MaxMin2 e MaxMin3

- A tabela apresenta o número de comparações dos programas MaxMin1, MaxMin2 e MaxMin3.
- Os algoritmos MaxMin2 e MaxMin3 são superiores ao algoritmo MaxMin1 de forma geral.
- O algoritmo MaxMin3 é superior ao algoritmo MaxMin2 com relação ao pior caso e bastante próximo quanto ao caso médio.

Os três algoritmos	$f(n)$		
	Melhor caso	Pior caso	Caso médio
MaxMin1	$2(n-1)$	$2(n-1)$	$2(n-1)$
MaxMin2	$n-1$	$2(n-1)$	$3n/2-3/2$
MaxMin3	$3n/2-2$	$3n/2-2$	$3n/2-2$