Técnicas de Análise

Algoritmos e Estruturas de Dados 2
2017-1
Flavio Figueiredo (http://flaviovdf.github.io)

Técnicas de Análise de Algorimos

- Determinar o tempo de execução de um programa pode ser um problema complexo
  - Ex: $f(n) = 3n^2 + n \log n + 42$

- Determinar a ordem do tempo de execução, sem preocupação com o valor da constante envolvida, pode ser uma tarefa mais simples
  - Ex: $f(n) = O(n^2)$

Sequência de Comandos

- Comandos de atribuição, leitura ou escrita:
  - O(1)

- Comando de decisão:
  - Tempo dos comandos dentro do condicional
    \+ tempo para avaliar a condição, que é O(1)

- Sequência de comandos:
  - Determinado pelo maior tempo de execução de qualquer comando da sequência (<u>regra da soma</u>)

For/While

- Geralmente efeito multiplicativo

- Um for de tamanho n
  - Aninhado com constance
  - n * c
  - O(n)

- Um for de tamanho n
  - Aninhado com outro for t tamanho m
  - n * m
  - O(n * m)

Exemplo 1

```
-  int exemplo1(int n) {
1      int i;
1      int acumulador = 0;
?      for(i = 0; i < n; i++) {
?          acumulador += i;
-      }
1      return acumulador;
-  }
```

Exemplo 1: O(n)

```
-  int exemplo1(int n) {
1      int i;
1      int acumulador = 0;
n      for(i = 0; i < n; i++) {
n          acumulador += i;
-      }
1      return acumulador;
-  }
```

Exemplo 2

```
-  void exemplo2(int n)
-  {
-      int i, j;
1      int a = 0;
n      for(i = 0; i < n+1; i++)
?         for(j = 0; j < i; j++)
?            a += i + j;
?      exemplo1(n);
-  }
```

Exemplo 2

```
                  -  void exemplo2(int n)
                  -  {
                  -      int i, j;
                  -      int a = 0;
                  -      for(i = 0; i < n+1; i++)
  1 + 2 + 3 + … n            for(j = 0; j < i; j++)
  1 + 2 + 3 + … n                a += i + j;
1 (chamada), n ex1       exemplo1(n);
                  -  }
```

Exemplo 2: O(n * n)

- Progressão aritmética ali no meio
  - Depende do primeiro for

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2},$$

- Parte que importa
  - Comandos “mais internos”

- **1 + 2 + 3 + … n** = n(n + 1) / 2

  = (n*n + n) / 2

  O(n * n)

Ordenação

```
   -  void ordena(int *V, int n) {
   -      int i, j, min, x;
 n-1      for(i = 0; i < n - 1; i++) {
 n-1          min = i;
   ?          for(j = i + 1; j < n; j++)
   ?              if(V[j] < V[min])
   ?                min = j;
   -          /* troca A[min] e A[i]: */
 n-1          x = V[min];
 n-1          V[min] = V[i];
 n-1          V[i] = x;
   -      }
   -  }
```

Ordenação

Similar ao exemplo anterior O(n*n)

```
   -  void ordena(int *V, int n) {
   -      int i, j, min, x;
 n-1      for(i = 0; i < n - 1; i++) {
 n-1          min = i;
   ?          for(j = i + 1; j < n; j++)
   ?              if(V[j] < V[min])
   ?                min = j;
   -          /* troca A[min] e A[i]: */
 n-1          x = V[min];
 n-1          V[min] = V[i];
 n-1          V[i] = x;
   -      }
   -  }
```

Exemplo 4

```
// A, B e C são matrizes
void p1(int **A, int **B, int **C, int n) {
    int i, j, k;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++) {
            C[i][j] = 0;
            for (k = n-1; k >= 0; k--)
                C[i][j] += A[i][k] * B[k][j];
        }
}
```

Sobre Matrizes

- Strassen em 1969 achou um algoritmo que resolve este problema em $O(n^{2.8074})$
- Coppersmith-Winograd: $O(n^{2.375477})$ em 1990
- Andrew Stothers: $O(n^{2.374})$ em 2010
- Virginia Williams: $O(n^{2.3728642})$ em 2011
- François Le Gall: $O(n^{2.3728639})$ em 2014
  - Baseado no método de Williams

Exemplo 5

```
void p2(int n) {
    int i, j, x, y;
    x = y = 0;
    for (i = 1; i <= n; i++) {
        for (j = i; j <= n; j++)
            x = x + 1;
        for (j = 1; j < i; j++)
            y = y + 1;
    }
}
```