Aula 3: Complexidade de Algoritmos

- O conceito de complexidade
- Complexidade de pior caso, melhor caso e caso médio
- Exemplos: algoritmos para soma e produto de matrizes

Avaliação de Tempo de Algoritmos

- A importância da avaliação de tempo, na história:
  - algoritmos primitivos;
  - o interesse matemático no tempo.
    Exemplo: velocidade de convergência de séries;
  - o aparecimento do computador:
    o tempo como fator primordial.
- A avaliação de espaço somente se tornou importante, com o surgimento do computador:
  - nos primórdios da computação: espaço era fundamental;
  - atualmente: importância decresceu.

Avaliando o Tempo

- Como avaliar o tempo?
  - métodos empíricos
  - métodos analíticos
- Empíricos:
  - Baseados na medição do tempo de execução do algoritmo.
  - Dependem do computador, linguagem de programação, compilador, dados, etc., utilizados na experiência.
  - Admitem um tratamento estatístico.
- Analíticos:
  - Determinação de expressões matemáticas que possam estimar o comportamento de tempo de um algoritmo.
  - Independem do computador, linguagem de programação e compilador.
  - Objetivo ambicioso.

Estudo de Métodos Analíticos

- Simplificações
  - Grande quantidade de dados
  - Considerar apenas valores assintóticos
  - Desconsiderar constantes aditivas e multiplicativas.
  - Exemplo: $3.n^2+5.n-7$ e $12.n^2+11.n+32$ seriam expressões equivalentes.
- Variável independente da expressão aritmética: Entrada do Algoritmo

- Exprimir tempo de execução em função da entrada.

Em busca de uma expressão

- O processo de execução de um algoritmo é dividido em"passos".
- Cada passo consiste na execução de um número fixo de operações básicas, cujos tempos de execução são considerados constantes.
- A operação básica de maior freqüência de execução é a "operação dominante".
- O número de passos do algoritmo corresponde à freqüência da operação dominante.
- Exemplo: em algoritmos de ordenação, geralmente, a operação dominante é a comparação.
- Objetivo: avaliar a ordem de grandeza do número de passos de um algoritmo, a partir de uma certa entrada.

Avaliação do Nº de Passos

- Exemplo: algoritmo de inversão de uma seqüência.
- Algoritmo: Inversão de uma seqüência.

```
para i=1,...,⌊n/2⌋ faça
        temp := S[ i ]
        S[ i ] := S[ n-i+1 ]
        S[ n-i+1 ] := temp
```

- Cada passo corresponde a uma troca de posições entre dois elementos.
- Variável independente = tamanho da seqüência = n
- Número de passos= número de execuções do bloco

$$\text{para } i = 1, \ldots, \lfloor n/2 \rfloor$$

Exemplo: Soma de Matrizes

$$\begin{vmatrix} 2 & 3 & 5 \\ 0 & 1 & 2 \\ 3 & 0 & 1 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 2 \\ 3 & -1 & 4 \\ -4 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 2 & 3 & 7 \\ 3 & 0 & 6 \\ -1 & 1 & 1 \end{vmatrix}$$

- $A = (a_{ij})$, $B = (b_{ij})$, $1 <= i, j <= n$
  $C = A + B$
  $c_{ij} = a_{ij} + b_{ij}$, $1 <= i,j <= n$

- Algoritmo: soma de matrizes

```
para i := 1, ..., n faça
     para j := 1, ..., n faça

          cij := aij + bij
```

- Variável independente= número de linhas da matriz = n
  Número de passos = n^2

Exemplo: Produto de Matrizes

$$\begin{vmatrix} 2 & 3 & 5 \\ 0 & 1 & 2 \\ 3 & 0 & 1 \end{vmatrix} . \begin{vmatrix} 0 & 0 & 2 \\ 3 & -1 & 4 \\ -4 & 1 & 0 \end{vmatrix} = \begin{vmatrix} -11 & 2 & 16 \\ -5 & 1 & 4 \\ -4 & 1 & 6 \end{vmatrix}$$

- $A = (a_{ij})$, $B = (b_{ij})$, $1 <= i, j <= n$
  $D = A . B$

  $$d_{ij} = \sum_{1 \leq k \leq n} a_{ik} . b_{kj}$$

- Algoritmo: produto de matrizes

```
para i := 1, ..., n faça
        para j := 1, ..., n faça

                d_ij := 0
                para k := 1, ..., n faça

                    d_ij := d_ij + a_ik . b_kj
```

- Variável independente= número de linhas da matriz = n
  Número de passos = n^3

Exemplos

- Nos exemplos de soma e produto das matrizes, o n^o de passos depende apenas do tamanho da entrada, mas não depende do seu valor.
  Em geral, o n^o de passos depende do valor da entrada.

- Exemplo: soma ou produto de matrizes
  Dadas as matrizes A, B e a variável x,
  calcular C = A + B, se x = 0;
  ou D = A . B, se x $\neq$ 0.

- Algoritmo: soma ou produto de matrizes.

```
se x = 0 então C := A + B senão D := A . B
```

- Número de passos = n^2 ou n^3

Exemplos

- Ideal: conhecer o valor do número de passos, conforme a entrada.

- O ideal pode ser difícil de ser atingido.

- Alternativa: Determinar o número de passos, para entradas específicas e "representativas".

- Conceito de complexidade.

Complexidade de Tempo

- A = algoritmo
  $E = \{ E_1, \ldots, E_n \}$ = conjunto de todas entradas possíveis de A
  t_i = número de passos efetuados por A, quando a entrada for E_i
- Complexidade pior caso = $\max_{E_i \in E} \{ t_i \mid E_i \in E \}$
- Complexidade melhor caso = $\min_{E_i \in E} \{ t_i \mid E_i \in E \}$
- Complexidade caso médio = $\sum_{1 \leq i \leq n} p_i t_i$, onde

  p_i = probabilidade de ocorrência de E_i
- Complexidade de espaço: similar

Comentários sobre as Complexidades

Complexidade de pior caso:

- n° de passos da entrada mais desfavorável
- quase sempre relevante (ex.: aplicações de segurança)
- a mais utilizada
- termo complexidade equivale ao pior caso

Complexidade de melhor caso:

- n° de passos da entrada mais favorável
- em geral, menor relevância
- utilizada em situações específicas

Complexidade de caso médio:

- n° de passos da "entrada média"
- em geral, relevante
- o tratamento é matemático, e pode não ser simples
- depende das probabilidades de ocorrência das entradas

Exemplo de Cálculo das Probabilidades

algoritmo	complexidade de pior caso	complexidade de melhor caso	complexidade de caso médio
inversão de sequências	n	n	n
soma de matrizes	n^2	n^2	n^2
produto de matrizes	n^3	n^3	n^3
soma ou produto de matrizes	n^3	n^2	$q.n^2+(1-q).n^3$

- constantes aditivas ou multiplicativas são irrelevantes
- algoritmo de soma ou produto:
  seja q, $0 \leq q \leq 1$, a probabilidade de que o valor da entrada x seja igual a zero.

Exercícios

- Dado um inteiro $\texttt{n} \geq 0$, determinar a complexidade do algoritmo de cálculo de $\texttt{n}!$.

- Sejam $\texttt{A = (a}_{\texttt{ik}}\texttt{)}$, $\texttt{B = (b}_{\texttt{kj}}\texttt{)}$, duas matrizes, tais que $1 \leq \texttt{i} \leq \texttt{n}$, $1 \leq \texttt{k} \leq \texttt{m}$ e $1 \leq \texttt{j} \leq \texttt{p}$.
  Escrever os algoritmos para calcular $\texttt{C = A + B}$ e $\texttt{D = A . B}$.
  Determinar as suas complexidades.
  Os valores de n, m e p podem ser quaisquer?

Tempo: 9 minutos

Solução

Complexidade = n

```
algoritmo: soma de matrizes
para i := 1, ..., n faça
        para j := 1, ..., m faça

                c_ij := a_ij + b_ij
```

Restrição: n=m=p

Complexidade: n.m

```
algoritmo: produto de matrizes
para i := 1, ..., n faça
        para j := 1, ..., p faça

                d_ij := 0
                para k := 1, ..., m faça

                    d_ij := d_ij + a_ik . b_kj
```

Complexidade: n.m.p