Análise e Projeto de Algoritmos

Loana Tito Nogueira

13 de setembro de 2016

Complexidade de Algoritmos

- Definimos complexidade de um algoritmo como o número de passos, isto é, de operações básicas, necessárias para sua execução.

- Ex.: Inversão de uma sequência

  Dadas a sequência s_1, s_2, ..., s_n, queremos:

  $s_1 \longleftrightarrow s_{n;}$

  $s_2 \longleftrightarrow s_{n-1;}$

  ...

  $s_{n/2} \longleftrightarrow s_{n/2;}$

Complexidade de Algoritmos

- Algoritmo:

```
Para i=1, 2, ..., n/2 faça
        temp:= s[i]
        s[i]:= s[n-i+1]
        s[n-i+1]:=s[i]
```

Complexidade de Algoritmos: Alguns Princípios

- O tempo de execução de um comando de atribuição, de entrada ou de saída é O(1) ou O(k), k constante.

Complexidade de Algoritmos: Alguns Princípios

- O tempo de execução de um comando de atribuição, de entrada ou de saída é O(1) ou O(k), k constante.

- O tempo de execução de uma sequência de comandos é o maior tempo de execução de qualquer comando da sequência.

Complexidade de Algoritmos: Alguns Princípios

- O tempo de execução de um comando de atribuição, de entrada ou de saída é O(1) ou O(k), k constante.

- O tempo de execução de uma sequência de comandos é o maior tempo de execução de qualquer comando da sequência. $O(f(n)+g(n)) = O(\max\{f(n), g(n)\})$

Complexidade de Algoritmos: Alguns Princípios

- O tempo de execução de um comando de atribuição, de entrada ou de saída é O(1) ou O(k), k constante.

- O tempo de execução de uma sequência de comandos é o maior tempo de execução de qualquer comando da sequência. $O(f(n)+g(n)) = O(\max\{f(n), g(n)\})$

  - Ex.: Suponha 3 trechos sequenciais de um programa com tempos de execução O(n), $O(n^2)$, O(nlogn). O tempo de execução de todos os 3 trechos é, então, $O(\max\{n, n\log n, n^2\})$, que é $O(n^2)$.

Complexidade de Algoritmos: Alguns Princípios

- O tempo de execução de um comando de decisão simples é composto pelo tempo para avaliar a condição mais o tempo dos comandos executados dentro da condição.

```
If (condição) then
        sequência de comandos 1
    Senão
        sequência de comandos 2
```

Complexidade de Algoritmos: Alguns Princípios

- Neste caso, será executada a sequência de comandos 1 ou a sequência de comandos 2. Assim, a complexidade será dada por Max(seq 1, seq2).
  - Ex.: Se a seq 1 é O(n) e a seq 2 é O(1), a complexidade de pior caso para a declaração inteira do if-then-else é O(n).

Complexidade de Algoritmos: Alguns Princípios

- O tempo de execução de um comando de repetição é a soma do tempo de execução do corpo do laço mais o tempo para avaliar a condição de parada multiplicado pelo número de iteração do laço.

```
For (i=0; i<n, i++)
        Sequência de comandos
```

n* O(seq de com)

Complexidade de Algoritmos: Alguns Princípios

- Programas com rotinas recursivas
  - Para analisar uma rotina recursiva é necessário primeiro encontrar a relação de recorrência que descreve a rotina. Em seguida, devemos resolver a relação de recorrência.

Complexidade de Caso Médio, Melhor e Pior Caso

- Dado um algoritmo cujo conjunto de entradas possíveis é:
  - $\{E_1, E_2, \ldots E_m\}$
  - Seja t_i= número de passos correspondentes à entrada E_i, $1 \leq i \leq m$

- Definimos: $\max\{t_i\}$= complexidade do pior caso

  $\min\{t_i\}$= complexidade do melhor caso

- Seja p_i a probabilidade de ocorrência da entrada E_i.
- $\sum p_i t_i$= complexidade do caso médio

$1 \leq i \leq m$

Complexidade de Caso Médio, Melhor e Pior Caso

- Exemplo: Problema da Busca
  - Seja S=$\{s_1, s_2,..., s_n\}$ um conjunto com n elementos e seja x um elemento qualquer. O problema consiste em verificar se x $\in$ S. Em caso positivo, localizá-lo.

Complexidade de Algoritmos

- Algoritmo Busca Linear

  Dados $s_1, s_2, \ldots, s_n, x$

  j:= 1

  enquanto (j < n e x ≠ s_j) faça

  j:= j+1

  Se x= s_j então (SIM, posição j)

  caso contrário (NÃO)

Complexidade: Queremos determinar o número de comparações de x com s_j.

Pior caso, Melhor Caso e Caso Médio

- Pior Caso: $x = s_n$ ou $x \notin S$ (n comparações)
- Melhor Caso: $x = s_1$ (1 comparação)
- Caso Médio:

Caso Médio

- Chamamos de E_i uma entrada onde $x = s_i$, $1 \leq i \leq n$
- E_{n+1} é uma entrada tal que $x \notin S$.

Queremos calcular $\sum p_i t_i$, onde o somatório é sobre as possíveis entradas E_i, as quais o algoritmo é sensível, isto é, se comporta de modo diferente.

Portanto, E_1, E_2, ..., E_n, E_{n+1} são as diferentes entradas possíveis.

Caso Médio

- Vamos chamar de q probabilidade de $x \in S$, logo temos que 1-q é a probabilidade de $x \notin S$.

- Obs.: Vamos supor que $x \in S$, então ele tem igual probabilidade de ser qualquer elemento.

- Logo, $\sum_{1 \leq i \leq n} p(E_i) = q$,

onde $p(E_i)$ é a probabilidade da entrada E_i ocorrer.

- $P(E_{n+1}) = 1 - q$

Caso Médio

- Número de passos para cada entrada possível:
  - $t(E_i) = i$
    -
    -
    -