INF 02779 Análise de Algoritmos - 2002/2	12/12/02

Aula 5: Tipos de Análise

Instrutor: Berilhes Borges Garcia — Escriba: Anselmo B. de Aguiar

DRAFT

1 Pesquisa Seqüencial e Pesquisa Binária

Considere o problema de encontrar um determinado tópico em um livro texto. Por exemplo, depois dos algoritmos 4 e 5 você resolveu saber o que um livro texto diz a respeito da multiplicação de matriz.

- Você procuraria no índice, normalmente este listará uma série de páginas, algumas das quais são apenas marginalmente relevantes.
- Uma segunda alternativa poderia ser ler a tabela de conteúdo até encontrar um título que parecesse relevante, por exemplo 16.1 e 31.2.

1.1 Pesquisa Seqüencial

O segundo método (tabela de conteúdo) é um exemplo de Pesquisa Seqüencial. Tipos similares de problemas são freqüentes em programação. Porexemplo, o Unix utiliza busca seqüencial para procurar nomes de arquivos em um diretório.
De uma forma bastante abstrata:

Algoritmo 1. *Busca-Seqüencial(L,x)*
  for *cada item y em L*
    if*(y encaixa com x)*
      return *y;*
  return *nenhum encaixe;*

Existem muitas variantes de busca seqüencial:

- Pare uma vez que um encaixe tenha sido encontrado?
- Ou continue a busca até que todos tenham sido encontrados?
- Represente a lista por meio de ponteiros, vetores ou alguma outra alternativa?
- Como indicar que nenhum encaixe foi encontrado?

Mesmo assim, desejamos analisar o algoritmo.

1.2 Análise da Pesquisa Seqüencial

De modo a analisar o tempo de execução do algoritmo, nós devemos de- terminar quão rápido o passo "y encaixa com x" pode ser executado, todo o restante é direto.
Note que o algoritmo foi escrito em um pseudo-código de nível bem alto, de mode que esta parte necessita tomar uma forma mais precisa, para que o algoritmo possa ser implementado. Mas mesmo neste nível nós podemos analisar o algoritmo.
Para fazer isto nós mediremos o tempo em termos do número de com- parações.
Note que o tempo parece depender de L e de x. É muito tentador pre- ver o tempo de execução dizendo algo como:

$$comparacoes(x, L) = posicao\ de\ x\ em\ L. \tag{1}$$

Esta afirmação é verdadeira, porém não muito informativa. Então como nós podemos prever o tempo de execução do algoritmo? Vamos olhar os métodos de análise que nós podemos utilizar.

2 Métodos de Análise

Nosso objetivo é prever o tempo de execução sem entrar nos detalhes da entrada fornecida ao algoritmo. Portanto, nós medimos o tempo como uma função do tamanho da entrada.
Neste caso específico, x é basicamente de tamanho constante e o tamanho de L é apenas o números de itens nesta lista.
Desta forma, dado uma lista L com n itens, quantas comparações o algoritmo demanda?
A resposta depende ...
Nós queremos uma resposta que não dependa ...
Existem várias maneiras de obter uma, dependendo da forma pela qual nós combinamos os tempos de execução para entradas diferentes com o mesmo tamanho.

Definição 1. ***Pior Caso***: $T_{wc} = max_{I,|I|=n}\{T(I)\}$

Número máximo de comparações que não importando quão perversa é a entrada.

Definição 2. ***Melhor Caso***: $T_{bc}(n) = min_{I,|I|=n}\{T(I)\}$

Número mínimo de comparações que o algoritmo poderia demandar se a entrada fosse bem comportada.

Definição 3. ***Caso Médio***: $T_{avg}(n) = \sum_{I,|I|=n} P[I]T[I]$

Tempo que o algoritmo demandaria sob uma entrada "típica". A instancia I de tamanho n tem probabilidade P[I] de ser entrada real.

2.1 Análise do Caso Médio

Provavelmente a mais importante de uma forma geral, contudo é difícil caracterizar o que é uma entrada típica para um determinado problema.
Dessa forma nós necessitamos fazer suposições acerca da distribuição de probabilidades das entradas.
A análise dependerá da precisão destas suposições.

Existe também o problema que pode ser que o algoritmo nunca encontre a entrada com o tempo médio, por exemplo, um determinado algoritmo pode demandar 1 passo ou 100 passos, na média este demanda aproximadamente 50, contudo não existe nenhuma entrada que realmente demande 50 passos.