Complexidade de Algoritmos

ACH2002 - Introdução à Ciência da Computação II

Delano M. Beder

Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo
dbeder@usp.br

08/2008

Material baseado em slides do professor Marcos Chaim

Projeto de Algoritmos

Algoritmos são projetados para resolver problemas

- Problema: encontrar a melhor rota, em termos de tempo de entrega, dos produtos das Casas Ceará em Ermelino Matarazzo
- Solução: algoritmo para descoberta da melhor rota tendo como entrada os locais de entrega

Projetar algoritmos implica estudar o seu comportamento

- No tempo: quanto tempo vai demorar para encontrar a solução do problema
- No espaço: quanto de memória será necessário para encontrar a solução

Análise de algoritmos

Na área de análise de algoritmos há dois problemas distintos:

1. Análise de um algoritmo particular
   - Qual é o custo de usar um dado algoritmo para resolver um problema específico?
   - Quantas vezes cada parte desse algoritmo vai ser executada?
   - Quanto de memória será necessária?

Análise de algoritmos

2. Análise de uma classe de algoritmos
   - Qual é o algoritmo de menor custo possível para resolver um problema específico?
   - Isto implica investigar toda uma família de algoritmos
   - Para realizar esta investigação limites podem ser impostos:
     - Exemplo: número mínimo de comparações necessárias para ordenar n números por meio de comparações sucessivas.
     - Logo, nenhum algoritmo vai fazer melhor que isto $\Rightarrow$ menor custo possível.
     - Menor custo possível $\Rightarrow$ medida de dificuldade.
   - Se o custo do algoritmo A = menor custo possível $\Rightarrow$ A é ótimo.

Complexidade

- Como medir o custo de um algoritmo?
- Como comparar o custo de vários algoritmos que resolvem um problema?

1. Medição direta do tempo de execução em um computador real. Problemas: depende do compilador; depende do hardware; medidas de tempo podem se influenciadas pela memória disponível.
2. Computador ideal em que cada instrução tem o custo de cada instrução determinado (solução de Donald Knuth).
3. Considerar apenas as operações mais significativas. Mais usual.
   - Exemplo: Ordenação $\Rightarrow$ número de comparações.

Função de complexidade

Para medir o custo de execução de um algoritmo

- Definição de uma função de custo ou complexidade $f(n)$
  - $f(n)$ é a medida do tempo ou espaço necessário para executar um algoritmo para uma entrada de tamanho n.
- Se a medida é de tempo, então $f(n)$ é chamada de função de complexidade de tempo ou temporal do algoritmo.
- Se a medida é a da memória necessária (espaço) para executar o algoritmo, então $f(n)$ é a função de complexidade espacial.

Se nada for dito, entende-se $f(n)$ como complexidade de tempo.

Exemplo

```
int maxArray(int [] A) {
  int i, max;
  max = A[0];

  for(i=1; i < A.length; ++i) {
    if(max < A[i]) {
       max = A[i];
    }
  }

  return max;
}
```


Função de complexidade

Seja $f(n)$ uma função de complexidade

- $f(n)$ é o número de comparações para um vetor A de tamanho n
- Como seria $f(n)$?
  - $f(n) = n - 1$, para $n > 0$.
- Será que o algoritmo apresentado é ótimo ?

Função de complexidade

Teorema

Qualquer algoritmo para encontrar o maior elemento de um conjunto de n elementos, $n \geq 1$, faz ao menos $n-1$ comparações.

Prova:
Cada um dos $n-1$ elementos tem que ser verificado, por meio de comparações, que é menor do que algum outro elemento.
Logo, $n-1$ comparações são necessárias. •

Como `maxArray()` possui complexidade igual ao limite inferior de custo, então seu algoritmo é ótimo.

Função de complexidade

- Normalmente, a medida de custo de execução depende do tamanho da entrada.
  - Mas este não é o único fato que influencia o custo.
- O tipo de entrada pode também influenciar o custo.
  - No caso de `maxArray()` a entrada não influencia.
- Conside um outro método para obter o máximo e o mínimo de um arranjo.

Função de complexidade

```
void  maxArray1(int [] A) {
  int i, max, min;
  max = min = A[0];

  for(i=1; i < A.length; ++i) {
    if(max < A[i]) {
       max = A[i];
    }
    else if(A[i] < min) {
            min = A[i];
    }
  }
  System.out.print("Mínimo = " + min);
  System.out.print(", Máximo = " + max);
}
```


Função de complexidade

Qual a função de complexidade de `maxMin1`?

Depende:

- Se o arranjo já estiver ordenado em ordem crescente
  - $f(n) = n - 1$
- Se o arranjo já estiver ordenado em ordem decrescente
  - $f(n) = 2(n-1)$
- Se o `A[i]` for maior que `max` metade das vezes
  - $f(n) = n - 1 + (n-1)/2 = 3n/2 - 3/2$ para $n > 0$

Função de complexidade

Três situações podem ser observadas:

1. Melhor caso: já ordenado crescentemente
   - Menor tempo de execução
2. Pior caso: já ordenado decrescentemente
   - Maior tempo de execução
3. Caso médio: um elemento `A[i]` tem 50% de chances de ser maior ou menor que `max`
   - Média dos tempos de execução de todas as entradas de tamanho n

Caso médio

- Supõe uma distribuição de probabilidades sobre o conjunto de entradas de tamanho n.
  - O custo médio é obtido com base nessa distribuição.
- Normalmente, o caso médio é muito mais difícil de determinar do que o melhor caso e o pior caso.
  - Usalmente, supõe-se que todas as entradas têm a mesma chance de ocorrer $\Rightarrow$ equiprováveis.
  - Nem sempre isto é verdade, por isto, o caso médio é determinado apenas se fizer sentido.

Resumo

- Problemas requerem algoritmos que os solucione.
- Algoritmo adequado depende do seu comportamento
  - complexidade temporal e espacial.
- Algoritmo ótimo
  - soluciona o problema com o menor custo possível.
- Função de complexidade
  - melhor caso, pior caso e caso médio.

Exercícios

1. Determine a função de complexidade da busca seqüencial de um vetor A de tamanho n para o melhor caso, pior caso e caso médio.
2. Determine a função de complexidade do algoritmo de ordenação por inserção direta no pior caso para um vetor de tamanho n. (Ver [1] Seção 2.2, páginas 16-21)

Referências utilizadas: [2] (Seção 1.3 páginas 3-11).

Referências

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. *Algoritmos - Tradução da 2a. Edição Americana*. Editora Campus, 2002.

[2] Nívio Ziviani. *Projeto de Algoritmos com implementações em C e Pascal*. Editora Thomson, 2a. Edição, 2004.