Análise de Complexidade de Algoritmos

Estrutura de Dados

Profa. Carla Koike - CIC

Principais Classes de Problemas

- $f(n) = O(1)$: complexidade constante
- $f(n) = O(\log n)$: complexidade logarítmica
- $f(n) = O(n)$: complexidade linear
- $f(n) = O(n \log n)$: ene log de ene
- $f(n) = O(n^2)$: complexidade quadrática
- $f(n) = O(n^3)$: complexidade cúbica.
- $f(n) = O(2^n)$: complexidade exponencial
- $f(n) = O(n!)$: complexidade fatorial

Se 1 instrução = 1μs ...

n	*n log n*	*n^2*	*n^3*
10	33.2	100	1000
100	664	10000	1seg
1000	9966	1seg	16min
100000	1.7s	2.8 hours	31.7 years

Exemplo: Caixeiro Viajante

- Um caixeiro viajante precisa visitar *n* cidades de tal forma que sua viagem inicie e termine em uma mesma cidade, e cada cidade deve ser visitada uma única vez. Supondo que sempre há uma rota entre duas cidades quaisquer, encontre a menor rota que o caixeiro viajante pode utilizar na sua viagem.
- Algoritmo simples: verificar todas as rotas e escolher a menor delas. Existe *(n-1)!* rotas possíveis, e cada rota envolve n adições para determinar a distância total: *n!* total de adições!
- Computador realiza 10^9 adições por segundo, resolver esse problema para 50 cidades levaria 10^{45} séculos de cálculo!

Algoritmos

- Algoritmo exponencial
  - tempo de execução tem função de complexidade $O(c^n)$; $c > 1$.
- Algoritmo polinomial:
  - tempo de execução tem função de complexidade $O(p(n))$, onde $p(n)$ é um polinômio.
- Algoritmos exponenciais: simples variações de pesquisa exaustiva.
- Algoritmos polinomiais: obtidos mediante entendimento aprofundado da estrutura do problema.

Mas, tudo depende de n...

- Existem valores de n onde um algoritmo exponencial é mais rápido que um polinomial:
  - $f(n) = 2^n$ e $g(n) = n^5$, para $n < 20$, o algoritmo exponencial é mais rápido.

Técnicas de Análise de Algoritmos

- Considerar memória infinita
- Não considerar o sistema operacional nem o compilador
- Analisar de preferência o algoritmo e não o programa, e levar em conta o tamanho das entradas
- Somente alguns comandos são considerados: atribuição, adição, multiplicação e comparação, e eles executam em um único passo de tempo

Algumas dicas...

- A complexidade de um laço é igual ao número de comandos internos vezes o número de vezes que ele é executado
- Exemplo:
  - para i de 1 até n faça soma <- soma+1
  - 1 atribuição externa ao laço
  - comandos internos do laço: 1 atribuição, 1 soma, 1 incremento
  - laço é executado n vezes
  - Complexidade g(n) = 3n + 1, O(n)

Algumas dicas ...

- Laços Aninhados: a complexidade de laços aninhados é o produto dos tamanhos dos laços
- Exemplo:
  - para i de 1 até n faça
  - para j de 1 até m faça
  - soma <- soma + i + j
  - Fora dos laços: 1 atribuição
  - Dentro somente do laço i: 1 atribuição
  - Dentro dos dois laços: 1 atribuição e 2 somas
  - g(n) = 1 + n + 3(nm), O(nm)

Algumas dicas ...

- Para uma seqüência de laços do algoritmo:
  - para i de 1 até n faça soma <- soma+1
  - para i de 1 até n faça
  - para j de 1 até n faça
  - soma <- soma + i + j
  - A primeira parte é O(n) e a segunda parte é $O(n^2)$
  - Portanto esse trecho de algoritmo é $O(n^2)$

Algumas dicas ...

- No caso de testes condicionais, a complexidade é a maior das duas partes do teste
- Exemplo:
  - Se teste = 1 então
  - para i de 1 até n faça soma <- soma +i
  - senão
  - para i de 1 até n faça
  - para i de 1 até n faça
  - soma <- soma + i + j
- “Se” é O(n) e “Então” é $O(n^2)$, portanto complexidade é $O(n^2)$

Algumas dicas ...

- Funções não recursivas
  - O tempo de execução de cada procedimento deve ser computador separadamente, um a um, iniciando com os procedimentos que não chamam outros procedimentos.
  - A seguir, avalia-se os procedimentos que chamam os procedimentos que não chamam outros procedimentos, usando os tempos já avaliados
  - Continua sucessivamente até chegar ao programa principal.

Algumas dicas ...

- Funções Recursivas
  - Associamos a complexidade da função recursiva uma equação de recorrência.
  - Uma equação de recorrência é uma equação ou inequação que descreve uma função em termos do seu valor para entradas menores.
  - Resolvendo a equação de recorrência é possível avaliar a classe do algoritmo
  - Existem técnicas para resolver equações de recorrência...

Exercícios 1

- Encontre a complexidade computacional para o seguinte laço:

```
for (cnt1 =0,i=1;i<=n;i++)
   for (j=1;j<=n;j++)
      cnt++;
```

Exercícios 2

- Encontre a complexidade computacional para o seguinte algoritmo de ordenação:

```
inteiro i,j,min,x
inicio
 para i de 1 ate n-1 faca
     inicio
    min = 1
     para j de i+1 ate n faca
        se A[j] < A[min] entao min = j
    x = A[min]
    A[min] = A[i]
    A[i] = x
    fim
fim
```