INF 02779 Análise de Algoritmos - 2002/2 22/11/02

Aula 2: Fibonacci - Quando coelhos crescem

Instrutor: Berilhes Borges Garcia Escriba: Ieda Manfre Santana e Maria Cecilia Simmer

DRAFT

1 Números de Fibonacci

Nós começaremos nosso curso de análise de algoritmo com um exemplo de *brinquedo*:computação de números de Fibonacci.Note que provavelmente você não necessitará realmente resolver este problema, contudo brincar é uma forma lúdica de aprender, talvez a forma mais eficiente. Adicionalmente nós temos que este problema é simples e fácil de entender, mas surpreendentemente existem muitas soluções diferentes para ele.

2 A História de Fibonacci

Leonardo de Pisa (também conhecido como Fibonacci) estave interessado em muitas coisas, inclusive um assunto que nós hoje chamamos de dinâmica populacional. "Quão rapidamente uma população de coelhos cresce sob condições apropriadas ?". Como é típico em matemática (e análise de algoritmo é uma forma de matemática), nós tornaremos o problema mais abstrato de modo a obter uma idéia das características gerais do problema sem nos perdermos em detalhes irrelavantes.

3 Suposição sob os Coelhos

- Um par de coelhos tem um par de filhos(coelhinhos) todo ano.
- Estes filhos são muito jovens para ter as suas próprias crias até os dois anos de idade.
- Coelhos nunca morrem (meu filho de 2 anos acredita nisso!).

Esta suposição não parece muito sensata,contudo ela torna o problema bem mais simples. Além disso, você pode analizar este modelo mais simples, e retornar ao problema original adicionando uma suposição qualquer, por exemplo, coelhos morrem aos dez anos - a seguir você verfica se o comportamento global do problema mudou muito.

4 Quão rapidamente a população dos coelhos cresce?

Expresse o número de pares de coelhos como uma função do tempo (anos transcorridos desde o começo do experimento):

```
F(1) = 1 (comece com um par)
F(2) = 1 (muito jovem para ter cria, apenas 1 ano).
F(3) = 2 (no 2 ano, um par de crias).
F(4) = 3 (no 3 ano, eles tem outro par).
F(5) = 5 (obt\'{e}m o primeiro par de "netos").
```

Generalizando:

```
F(n) = F(n-1) + F(n-2)
```

Todos os coelhos que existem no instante de tempo anterior (F(n-1)) mais um par de crias para todo par de coelhos que nós tinhamos dois anos atrás (F(n-2)).
Considere o problema algoritmico de: como computar F(n)?

5 Um algoritmo recursivo

A fórmula acima parece clamar por um algoritmo recursivo:
Algoritmo 1 Fib(n)
if (n ¡= 2) then
return 1
else
return (Fib(n-1) + Fib(n-2))
Exemplo do tipo de pergunta básica que nós estudaremos nesta disciplina: Qual é a demandade tempo deste algoritmo?

6 Como nós deveríamos medir o tempo ?

A medida natural seria em segundos, mas seria desejável ter respostas que não mudassem toda vez que um processador mais rápido é lançado no mercado.
Tempo medido em termos de instruções de máquinas, dividido pela velocidade da máquina (em instruções / segundo) nos forneceria o tempo real que nós desejamos. Contudo, é difícil do pseudo - código processar de forma exata o número de instruções que o compilador gerará.
De modo a obter uma aproximação grosseira disto, nós mediremos o número de linhas de código que são executadas.
Medindo o número de linhas de código executadas:
Fib(uit n)
if (n¡=2) then return 1
else return (Fib(n-1) + Fib(n-2))
Cada chamada à Fib retorna em uma ou duas linhas. Se n¡=2, nós executamos somente uma linha (o if / return). Se n=3, executa 2 para Fib(3), mais uma para Fib(2) e mais uma para Fib(1), total 4.
Este padrão é semelhante aos coelhos! Exceto que há duas linhas em cada chamada, o tempo para n é a soma dos tempos para duas chamadas recursivas menores: T(n) = 2 + T(n-1) + T(n-2).

7 Relações de Recorrência

Em geral, qualquer algoritmo recursivo tal como o anterior gera uma relação de recorrência, de modo que o tempo para qualquer rotina é o tempo denro da própria rotina, mais o tempo para as chamadas recursivas.
Isto conduz naturalmente a uma equação semelhante a equação mostrada anteriormente, a qual pode então ser resolvida de modo a determinar o tempo de execução da rotina.
No exemplo anterior, a relação de recorrência é muito similar a própria definição de números de Fibonacci.

8 Resolvendo a Relação de Recorrência

Pense na recursão como formando uma árvore. Desenhe um nó, a raiz da árvore, para a primeira chamada. Então desenhe um outro nó filho na árvore cada vez que a rotina chama a si própria .
F(5) / F(4) F(3) / / F(3) F(2) F(2) F(1) / F(2) F(1)
Cada um dos quatro nós internos desta árvore, para Fib(5), demandam 2 linhas de código, enquanto cada um dos cinco nós folhas demandam uma linha de código. Dessa forma o número total de linhas executada em todas as chamadas recursivas é: 4 * 2 + 5 * 1 = 13.

9 Passos para determinar F(n)

Generalize para qualquer chamada de Fib(n). Note que folhas contam como uma linha de código, nós internos contam como 2. Conte o número de folhas e nós internos na árvore de recursão.

1. Contando o número de folhas. O número de Fibonacci F(i) em cada nó interno é o número de folhas abaixo daquele nó. Desta forma, o número de folhas na árvore é igual a F(n).

2. Contando o número de nós internos. Utilize o seguinte fato básico acerca de árvores binários (árvores onde cada nó tem 2 filhos): o número de nós internos é sempre igual ao número de folhas menos 1. (Prove este fato por meio de indução: isto é verdade se há apenas uma folha e nenhum nó interno, e este é sempre verdade se nós adicionamos 2 filhos a uma folha)

3. Coloque tudo isto junto. Desta forma há F(n) linhas de código executadas nas folhas e 2F(n) - 2 nos nós internos, para um total de 3F(n) - 2. Isto é incrivelmente lento, por exemplo, para n= 45 este algoritmo demanda quase um bilhão de passos! Talvez nós possamos criar um algoritmo mais rápido.