Complexidade de Algoritmos

Mariana Kolberg

Projeto de Algoritmos
Divisão e conquista

Aula 2

Complexidade de Algoritmos

Divisão e conquista

- Divide um problema em subproblemas independentes, resolve-os e combina as soluções obtidas em uma solução para o problema original.
  - resulta em um **processo recursivo** de decomposições e recombinações.

- Pode ser aplicado em problemas de:
  - buscas em tabelas, como buscas seqüencial e binária;
  - classificação, como classificação por seleção (selectionsort), por intercalação (mergesort) e por particionamento (quicksort);
  - multiplicação (de matrizes e de números binários, por exemplo);
  - seleção (para determinar máximo e mínimo, etc.).

Divisão e Conquista

- Métodos para resolver recorrências:
  - Método da árvore de recursão
  - Método da substituição
  - Método mestre

Recorrências - Método da árvore de recursão

- Bem intuitiva para a análise de complexidade
  - Numa árvore de recursão cada nó representa o custo de um único subproblema da respectiva chamada recursiva
  - Somam-se os custos de todos os nós de um mesmo nível, para obter o custo daquele nível
  - Somam-se os custos de todos os níveis para obter o custo da árvore

Recorrências - Método da árvore de recursão

Dada a recorrência $T(n) = 3T(n/2) + cn$

- Em que nível da árvore o tamanho do problema é 1?
- Quantos níveis tem a árvore?
- Quantos nós têm cada nível?
- Qual o tamanho do problema em cada nível?
- Qual o custo de cada nível i da árvore?
- Quantos nós tem o último nível?
- Qual o custo da árvore?

Recorrências - Método da árvore de recursão

Outro exemplo é a recorrência $T(n) = 3T(n/2) + cn$ da multiplicação de números binários. Temos $\log_2 n$ níveis na árvore, o nível i com 3^i nós, tamanho do problema $n/2^i$, trabalho $cn/2^i$ por nó e portanto $(3/2)^i n$ trabalho total por nível. O número de folhas é $3^{\log_2 n}$ e portanto temos

$$\begin{aligned} T(n) &= \sum_{0 \le i < \log_2 n} (3/2)^i n + \Theta(3^{\log_2 n}) \\ &= n\left(\frac{(3/2)^{\log_2 n} - 1}{3/2 - 1}\right) + \Theta(3^{\log_2 n}) \\ &= 2(n^{\log_2 3} - 1) + \Theta(n^{\log_2 3}) \\ &= \Theta(n^{\log_2 3}) \end{aligned}$$

Observe que a recorrência $T(n) = 3T(n/2) + c$ tem a mesma solução. $\Diamond$

Recorrências - Método mestre

Para aplicar o método mestre deve ter a recorrência na seguinte forma:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

onde $a \geq 1$, $b > 1$ e $f(n)$ é uma função assintoticamente positiva. Se a recorrência estiver no formato acima, então $T(n)$ é limitada assintoticamente como:

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para algum $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$
2. Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \log n)$
3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para algum $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para $c < 1$ e para todo n suficientemente grande, então $T(n) = \Theta(f(n))$

Considerações

- Nos casos 1 e 3 $f(n)$ deve ser polinomialmente menor, resp. maior que $n^{\log_b a}$, ou seja, $f(n)$ difere assintoticamente por um fator n^{ϵ} para um $\epsilon > 0$.
- Os três casos não abrangem todas as possibilidades

Recorrência – Método mestre simplificado

- Algoritmos de divisão e conquista seguem um padrão genérico:
  - Resolvem um problema de tamanho n solucionando recursivamente a subproblemas de tamanho n/b e então combinando essas resposta em tempo $O(n^d)$, para certos a,b,d>0.
  - Seus tempos podem ser capturados pela equação $T(n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + O(n^d)$
- Teorema mestre:
  - Se $T(n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + O(n^d)$ para constantes a>0, b>1 e d≥0, então

$$T(n) = \begin{cases} O(n^d) & se\ d > log_b a \\ O(n^d log n) & se\ d = log_b a \\ O(n^{log_b a}) & se\ d < log_b a \end{cases}$$

  - Esse teorema nos dá o tempo de execução da maioria dos problemas de divisão e conquista

Recorrências - Método da Substituição

- O método da substituição envolve duas etapas:
  1. pressupõe-se um limite hipotético.
  2. usa-se indução matemática para provar que a suposição está correta.
- Aplica-se este método em casos que é fácil pressupor a forma de resposta.
- Pode ser usado para estabelecer limites superiores ou inferiores.

Recorrências - Método da Substituição

Mergesort usando o método da substituição

Supõe-se que a recorrência

$$T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$$

tem limite superior igual a $n \log n$, ou seja, $T(n) = O(n \log n)$. Devemos provar que $T(n) \leq cn \log n$ para uma escolha apropriada da constante $c > 0$.

$$\begin{aligned} T(n) &\leq 2(c\lfloor \frac{n}{2} \rfloor \log(\lfloor \frac{n}{2} \rfloor) + n) \\ &\leq cn \log n/2 + n = cn \log n - cn + n \\ &\leq cn \log n \end{aligned}$$

para $c \geq 1$.

A expressão na equação

$$c \cdot n \log n \underbrace{-c \cdot n + n}_{\text{residuo}}$$

se chama *resíduo*. O objetivo na prova é mostrar, que o resíduo é negativo.

Recorrências - Método da Substituição

- Como fazer um bom palpite?
  - Usar o resultado de recorrências semelhantes. Por exemplo, considerando a recorrência $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$, tem-se que $T(n) \in O(n \log n)$.
  - Provar limites superiores (ou inferiores) e reduzir o intervalo de incerteza. Por exemplo, para equação do Mergesort podemos provar que $T(n) = \Omega(n)$ e $T(n) = O(n^2)$. Podemos gradualmente diminuir o limite superior e elevar o inferior, até obter $T(n) = \Theta(n \log n)$.
  - Usa-se o resultado do método de árvore de recursão como limite hipotético para o método da substituição.

Recorrências - Método da Substituição

Dada a recorrência $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- Prove por indução que $T(n) = \Theta(n^2)$

Recorrências - Método da Substituição

Dada a recorrência $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- Prove por indução que $T(n) = \Theta(n^2)$

$$\begin{aligned} T(n) &\leq 3T(\lfloor n/4 \rfloor) + cn^2 \\ &\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\ &\leq 3d(n/4)^2 + cn^2 \\ &= \frac{3}{16}dn^2 + cn^2 \leq dn^2 \end{aligned}$$

para $(\frac{3d}{16} + c) \leq d$, ou seja, para valores de d tais que $d \geq \frac{16}{13}c$

Recorrências - Método da Substituição

1. Procurar o máximo em uma sequência

MÁXIMO

Entrada Uma seqüência a e dois índices l, r tal que $a_l, \ldots, a_{r-1}$ é definido.

Saída $\max_{l \le i < r} a_i$

```
1  m1 := M(a, l, ⌊(l + r)/2⌋)
2  m2 := M(a, ⌊(l + r)/2, r⌋)
```

   - Qual a recorrência?
   - Resolva a recorrência utilizando os 3 métodos

2. Resolva a recorrência $T(n) = 3T\left(\frac{n}{2}\right) + n^2$ utilizando os 3 métodos

Recorrências - Método da Substituição

- Procurar o máximo em uma sequência

MÁXIMO

Entrada Uma seqüência a e dois índices l, r tal que $a_l, \ldots, a_{r-1}$ é definido.

Saída $\max_{l \le i < r} a_i$

```
1  m1 := M(a, l, ⌊(l + r)/2⌋)
2  m2 := M(a, ⌊(l + r)/2, r⌋
```

- Qual a recorrência? $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
- Resolva a recorrência utilizando os 3 métodos.

Exercícios

POTENCIAÇÃO-TRIVIAL (PT)

Entrada Uma base $a \in \mathbb{R}$ e um exponente $n \in \mathbb{N}$.
Saída A potência a^n.

```
1  if n = 0
2    return 1
3  else
4    return PT(a, n − 1) × a
5  end if
```

Algoritmo Potenciação para $n = 2^i$

POTENCIAÇÃO-NPOTÊNCIA2 (P2)

Entrada Uma base $a \in \mathbb{R}$ e um exponente $n \in \mathbb{N}$.

Saída A potência a^n.

```
1  if n = 1 then
2    return a
3  else
4     x := P2(a , n ÷ 2)
5     return x × x
6  end if
```

Exercícios

BUSCA-BINÁRIA(I,F,X,S)

Entrada Um inteiro x, índices i e f e uma seqüência $S = a_1, a_2, \ldots a_n$ de números ordenados.

Saída Posição i em que x se encontra na seqüência S ou ∞ caso $x \notin S$.

```
1  if i = f then
2    if a_i = x return i
3    else return ∞
4  end if
5  m := ⌊(f−i)/2⌋ + i
6  if x < a_m then
7    return Busca Binária(i, m−1)
8  else
9    return Busca Binária(m+1, f)
10 end if
```

QUICKSORT

Entrada Índices l, r e um vetor a com elementos $a_l, \ldots, a_r$.

Saída a com os elementos em ordem não-decrescente, i.e. para $i < j$ temos $a_i \leq a_j$.

```
1  if l < r then
2    m := Partition(l,r,a);
3    Quicksort(l,m−1,a);

4    Quicksort(m+1,r,a);
5  end if
```

Exercícios

- Resolva as recorrencias abaixo:
  1. $T(n) = 2T(n/2) + n \log n$
  2. $T(n) = 9T(n/3) + n$