Projeto e Análise de Algoritmos

Hamilton José Brumatto

Bacharelado em Ciência da Computação - UESC

12 de maio de 2016

Projeto de Algoritmos por Indução

Uso da Indução em Algoritmos

- Indução tem se mostrado importante para provar a corretude de algoritmos.
- Em *loops* usa-se indução para mostrar um invariante correto.
- Em recorrências usa-se indução para provar a classe de assintocidade.

Projetando Algoritmos

- A indução pode ser utilizada como mecanismo de prova em vários teoremas ou propriedades envolvendo elementos da matemática discreta.
- A prova por indução, em seu PASSO e BASE já representa instruções que descrevem um algoritmo.
- Um mesmo problema pode apresentar provas diferentes da indução e portanto, algoritmos diferentes.
- Vamos trabalhar este tema a partir de exemplos.

Problema 1

Dada uma seqüência de números reais a_n, a_{n-1}, . . ., a_1, a_0, e um valor real x, é possível calcular o valor do polinômio:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

- Este problema é interessante, pois a abordagem da indução pode levar a várias soluções, e assintoticamente melhores.
- A idéia do uso da indução é tentar reduzir a um problema menor, e portanto, recursivamente, aplicar a Hipótese da Indução e construir o passo.
- A base representa o ponto final da recursão.

Solução 1 para o problema do polinômio

- Dado o problema original, vamos "retirar" do problema o termo com a_n, ficamos com:

$$P_{n-1}(x) = a_{n-1}x^{n-1} + \ldots + a_2x^2 + a_1x + a_0$$

- Basicamente é o mesmo problema, no entanto "menor", logo podemos aplicar a Hipótese da Indução:

Hipótese da Indução:

Nós sabemos como calcular o polinômio representado pelas entradas: $a_{n-1}, \ldots, a_2, a_1, a_0$ no ponto x.

Solução 1 para o problema do polinômio

- Precisamos definir a *Base*, o que é trivial: $P_0(x) = a_0$, na base $(n = 0)$.
- Falta construir o passo. Que é simplesmente calcular x^n, multiplicar por a_n e adicionar a $P_{n-1}(x)$.

Passo da Indução

$$P_n(x) = a_n x^n + P_{n-1}(x)$$

- Este resultado parece muito intuitivo, mas ele leva à construção do algoritmo por indução.
- Aparentemente estamos complicando algo simples, com a vantagem de que não precisamos provar o invariante de um *loop* para mostrar sua corretude, a construção por indução já prova a corretude.

Solução 1 para o problema do polinômio

Algoritmo POLINOMIO(A,n,x)
  se $n = 0$ **então**
    Retorna A[0]
  senão
    $X \leftarrow 1$
    para $i \leftarrow 1$ **até** n **faça**
      $X \leftarrow X \cdot x$
    $P \leftarrow A[n] \cdot X + POLINOMIO(A, n-1, x)$
    Retorna P

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $(n) + (n-1) + (n-2) + ... + 2 + 1 + 0$
- Ou seja, $T(n) \in O(n^2)$

Solução 1 para o problema do polinômio

Algoritmo POLINOMIO(A,n,x)
  se $n = 0$ **então**
    Retorna A[0]
  senão
    $X \leftarrow 1$
    para $i \leftarrow 1$ **até** n **faça**
      $X \leftarrow X \cdot x$
    $P \leftarrow A[n] \cdot X + POLINOMIO(A, n-1, x)$
    Retorna P

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $(n) + (n-1) + (n-2) + ... + 2 + 1 + 0$
- Ou seja, $T(n) \in O(n^2)$

Solução 1 para o problema do polinômio

Algoritmo POLINOMIO(A,n,x)
  se $n = 0$ **então**
    Retorna A[0]
  senão
    $X \leftarrow 1$
    para $i \leftarrow 1$ **até** n **faça**
      $X \leftarrow X \cdot x$
    $P \leftarrow A[n] \cdot X + POLINOMIO(A, n-1, x)$
    Retorna P

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $(n) + (n-1) + (n-2) + ... + 2 + 1 + 0$
- Ou seja, $T(n) \in O(n^2)$

Solução 1 para o problema do polinômio

- O algoritmo não precisa ser recursivo:

Algoritmo POLINOMIO(A,n,x)
    $P \leftarrow 0$
    para $i \leftarrow 0$ **até** n **faça**
        $X \leftarrow 1$
        para $j \leftarrow 1$ **até** i **faça**
            $X \leftarrow X \cdot x$
        $P \leftarrow A[i] \cdot X + P$

- São $(n^2 + n)/2 + n$ multiplicações e n somas.
- Ou seja, continuamos com $T(n) \in O(n^2)$
- O algoritmo é correto, mas não muito eficiente. Dá para fazer melhor?

Solução 2 para o problema do polinômio

- A falta de eficiência no algoritmo anterior é decorrente da quantidade de recálculos realizados.
- Apesar de termos calculado x^{n-1} em uma recursão, para calcular x^n partimos do nada e fazemos toda a multiplicação novamente.
- Podemos tornar a "Hipótese de Indução" mais forte (a verdade apresentada é maior ou tem mais significado).

Hipótese da Indução:

Nós sabemos como calcular o polinômio representado pelas entradas: $a_{n-1}, ..., a_2, a_1, a_0$ no ponto x, também sabemos calcular $X_{n-1} = x^{n-1}$.

Solução 2 para o problema do polinômio

- Precisamos definir a *Base*, o que é trivial: $P_0(x) = a_0$, e $X_0 = x^0 = 1$ na base $(n = 0)$.
- Falta construir o passo. Que é simplesmente calcular X_n, e multiplicar por a_n e adicionar a $P_{n-1}(x)$.

Passo da Indução

$$X_n = x \cdot X_{n-1}$$

$$P_n(x) = a_n X_n + P_{n-1}(x)$$

- O algoritmo agora não é mais tão intuitivo. Algoritmos que não são tão claros podem ser difíceis de serem provados corretos.
- A indução mostra que o algoritmo é correto, ele foi construído como sendo correto.

Solução 2 para o problema do polinômio

Algoritmo POLINOMIO(A,n,x)
  se $n = 0$ **então**
    Retorna (A[0],1)
  senão
    $(P, X) = POLINOMIO(A, n-1, x)$
    $X \leftarrow x \cdot X$
    $P \leftarrow A[n] \cdot X + P$
    Retorna (P, X)

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $2 + 2 + 2 + ... + 2 + 2 + 2$, ou $2n$ multiplicações.
- Ou seja, $T(n) \in O(n)$
- O algoritmo é correto, sua eficiência é melhor

Solução 2 para o problema do polinômio

Algoritmo POLINOMIO(A,n,x)
  se $n = 0$ **então**
    Retorna (A[0],1)
  senão
    $(P, X) = POLINOMIO(A, n-1, x)$
    $X \leftarrow x \cdot X$
    $P \leftarrow A[n] \cdot X + P$
    Retorna (P, X)

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $2 + 2 + 2 + ... + 2 + 2 + 2$, ou $2n$ multiplicações.
- Ou seja, $T(n) \in O(n)$
- O algoritmo é correto, sua eficiência é melhor

Solução 2 para o problema do polinômio

Algoritmo POLINOMIO(A,n,x)
  se $n = 0$ **então**
    Retorna (A[0],1)
  senão
    $(P, X) = POLINOMIO(A, n-1, x)$
    $X \leftarrow x \cdot X$
    $P \leftarrow A[n] \cdot X + P$
    Retorna (P, X)

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $2 + 2 + 2 + ... + 2 + 2 + 2$, ou $2n$ multiplicações.
- Ou seja, $T(n) \in O(n)$
- O algoritmo é correto, sua eficiência é melhor

Solução 2 para o problema do polinômio

- Novamente, o algoritmo não precisa ser recursivo:

Algoritmo POLINOMIO(A,n,x)
  $P \leftarrow 0$
  $X \leftarrow 1$
  para $i \leftarrow 0$ **até** n **faça**
    $P \leftarrow A[i] \cdot X + P$
    $X \leftarrow X \cdot x$

- Temos portanto: $2n$ multiplicações e n somas.
- Ou seja, $T(n) \in O(n)$
- O algoritmo é correto, sua eficiência é melhor, o algoritmo é simples. Dá para fazer melhor?

Solução 3 para o problema do polinômio - Regra de Horner

- Nós fizemos a indução removendo o item a_n, o que parecia mais intuitivo.
- Podemos aplicar a indução de outras maneiras no mesmo problema.
- Podemos aplicar a indução removendo o item a_0 e diminuindo o grau de x.

Hipótese da Indução:

Nós sabemos como calcular o polinômio representado pelas entradas: $a_n, ..., a_3, a_2, a_1$ no ponto x, ou seja, sabemos calcular o polinômio $P'_{n-1} = a_n x^{n-1} + a_{n-1} x^{n-2} + ... + a_2 x + a_1$.

Solução 3 para o problema do polinômio - Regra de Horner

- Precisamos definir a *Base*: $P'_0(x) = a_n$.
- Falta construir o passo. Que é definido pela operação: $P'_k(x) = xP'_{k-1}(x) + a_{n-k}$.
- De uma forma geral, o polinômio pode ser escrito como:

$$a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0 = ((\ldots((a_nx + a_{n-1})x + a_{n-2})\ldots)x + a_1)x + a_0$$

- Este algoritmo é conhecido como *Regra de Horner* devido a W. G. Horner.

Solução 3 para o problema do polinômio - Regra de Horner

Algoritmo POLINOMIO(A,k,x)
  se $k = 0$ **então**
    Retorna $A[n]$
  senão
    Retorna $POLINOMIO(A, k-1, x) \cdot x + A[n-k]$

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $1 + 1 + 1 + ... + 1 + 1 + 1$, ou n multiplicações.
- Ou seja, $T(n) \in O(n)$
- Em termos de assintocidade não houve mudança, no entanto as constantes associadas são menores.
- O algoritmo em si é muito mais simples.

Solução 3 para o problema do polinômio - Regra de Horner

Algoritmo POLINOMIO(A,k,x)
  se $k = 0$ **então**
    Retorna $A[n]$
  senão
    Retorna $POLINOMIO(A, k-1, x) \cdot x + A[n-k]$

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $1 + 1 + 1 + ... + 1 + 1 + 1$, ou n multiplicações.
- Ou seja, $T(n) \in O(n)$
- Em termos de assintocidade não houve mudança, no entanto as constantes associadas são menores.
- O algoritmo em si é muito mais simples.

Solução 3 para o problema do polinômio - Regra de Horner

Algoritmo POLINOMIO(A,k,x)
 se $k = 0$ **então**
  Retorna $A[n]$
 senão
  Retorna $POLINOMIO(A, k-1, x) \cdot x + A[n-k]$

- Qual a assintocidade deste algoritmo? (Vamos considerar as multiplicações)
- Calculando temos: $1 + 1 + 1 + ... + 1 + 1 + 1$, ou n multiplicações.
- Ou seja, $T(n) \in O(n)$
- Em termos de assintocidade não houve mudança, no entanto as constantes associadas são menores.
- O algoritmo em si é muito mais simples.

Solução 3 para o problema do polinômio - Regra de Horner

- Novamente, o algoritmo não precisa ser recursivo:

Algoritmo POLINOMIO(A,n,x)

$P \leftarrow A[n]$

para $k \leftarrow 1$ **até** n **faça**

$P \leftarrow P \cdot x + A[n-k]$

- Temos portanto: n multiplicações e n somas.
- Ou seja, $T(n) \in O(n)$

O Problema da Celebridade

Em uma reunião de pessoas, existe uma celebridade quando: Todas as pessoas conhecem a celebridade, a celebridade não conhece ninguém. O problema a ser resolvido é: Encontrar a celebridade ou determinar que não existe celebridade na reunião.

- O ataque da força bruta levará a um algoritmo muito lento, pois teremos de perguntar a cada pessoa se ela conhece as $(n-1)$ pessoas restantes, logo $T(n) = n(n-1) \in O(n^2)$.
- Podemos tentar resolver este problema por indução:
  - Retira uma pessoa da reunião.
  - Pela Hipótese da Indução, sabemos na reunião de $(n-1)$ pessoas determinar se existe e quem é a celebridade.
  - A base é simples, somente uma única pessoa, então ela é a celebridade
  - O Passo é mais complexo.

Passo no Problema da Celebridade

- Ao aplicar a Hipótese em uma reunião de $(n-1)$ pessoas, podemos encontrar ou não uma celebridade, nesta reunião.
- Para o passo temos três situações:
  1. A celebridade está no conjunto de $(n-1)$ pessoas.
  2. A celebridade é a n-ésima pessoa.
  3. Não existe celebridade.
- A primeira situação é simples, basta perguntar se a celebridade encontrada conhece a n-ésima pessoa, e vice-versa. Com duas perguntas, decidimos se existe ou não celebridade.
- A segunda situação e terceira situação são mais complexas, pois teríamos de perguntar para as $(n-1)$ pessoas se conhecem a n-ésima pessoa e vice-versa, ou seja, caímos na mesma eficiência da força bruta.
- Neste caso, precisamos fazer uma escolha mais cuidadosa com relação à pessoa que devemos retirar.

Solução por indução no Problema da Celebridade

- A primeira situação anterior foi mais simples, vamos tentar forçar que isto aconteça sempre, na primeira situação retiramos do conjunto uma pessoa que não é celebridade.
- A idéia será escolher no conjunto uma pessoa que não é celebridade e retirá-la, restam somente as situações (1) e (3) anteriores.
- Na reunião de $(n-1)$ pessoas podemos encontrar ou não uma celebridade, se encontrarmos, bastam 2 perguntas para determinar se ela é celebridade do conjunto de n pessoas, se não econtrarmos, então não existe celebridade no conjunto de n pessoas.

Solução por Indução no Problema da Celebridade

- A nossa sorte é que é fácil encontrar uma pessoa que não é celebridade com apenas uma pergunta:
    - Perguntamos para a pessoa A, se conhece a pessoa B. Se a resposta for SIM, então A não é celebridade, se a resposta for NÃO, então B não é celebridade. Sabemos portanto quem retirar.
- Vamos definir nossa Hipótese de Indução, Base e Passo para o problema.

Hipótese de Indução

Em uma reunião com $(n-1)$ pessoas, sabemos encontrar uma celebridade, ou determinar que não existe.

Base

Em uma reunião com 1 pessoa, ela é celebridade.

Solução por Indução no Problema da Celebridade

Passo

Seja **C** a celebridade encontrada na reunião com $(n-1)$ pessoas. Se **C** conhece a n-ésima pessoa ou se a n-ésima pessoa não conhece **C** então não existe celebridade, caso contrário **C** é a celebridade. Se não foi encontrada a celebridade na reunião com $(n-1)$ pessoas então não existe celebridade.

Algoritmo para o Problema da Celebridade

Algoritmo CELEBRIDADE(R)
    se $tamanho[R] = 1$ **então**
        Retorna $R[1]$
    senão
        se $R[1]$ *conhece* $R[2]$ **então**
            $A \leftarrow R[1]$
        senão
            $A \leftarrow R[2]$
        $R \setminus A$ ▷ Retiro A de R
        $C \leftarrow CELEBRIDADE(R)$

        se $C =$ NINGUÉM **ou** C *conhece* A **ou** $\neg(A$ *conhece* $C)$ **então**
            Retorna NINGUÉM
        senão
            Retorna C

Insert Sort

Hipótese de Indução

Consigo ordenar um conjunto com $n-1$ elementos.

Base da Indução

Em um conjunto com 1 elemento, já está ordenado.

Passo da Indução

Dado um conjunto com n elementos, retiro o n-ésimo elemento. Aplico a Hipótese de Indução no conjunto com $(n-1)$ elementos, insiro o elemento retirado em sua posição correta no conjunto de $(n-1)$ elementos ordenados.

Select Sort

Hipótese de Indução

Consigo ordenar um conjunto com $n-1$ elementos.

Base da Indução

Em um conjunto com 1 elemento, já está ordenado.

Passo da Indução

Dado um conjunto com n elementos, retiro o menor elemento. Aplico a Hipótese de Indução no conjunto com $(n-1)$ elementos, insiro o elemento retirado na primeira posição.

Merge Sort

Hipótese de Indução

Consigo ordenar um conjunto com menos de n elementos.

Usamos aqui a Indução Forte

Base da Indução

Em um conjunto com 1 elemento, já está ordenado.

Passo da Indução

Dado um conjunto com n elementos, divido-o em dois. Aplico a Hipótese de Indução em cada conjunto com metade dos elementos. Intercalo os elementos em uma ordem final.

Considerações sobre o uso da Indução para Projetar Algoritmos

- A indução gera naturalmente um algoritmo recursivo. Um algoritmo iterativo pode ser construído a partir do algoritmo recursivo. Ou seja, um laço representa uma indução.
  - A Base da Indução representa a base da recursão (término das chamadas recursivas).
  - A Hipótese da Indução representa uma chamada recursiva em um conjunto menor que o conjunto de entrada original.
  - O Passo da Indução representa as ações realizadas com o retorno das chamadas recursivas para resolver o problema de forma geral.
- Quando aplicamos a indução forte no projeto do algoritmo, resolvemos um problema dividindo-o em partes menores, esta técnica também é conhecida como DIVISÃO E CONQUISTA

Atividades

- Resolver a Lista 7.