Projeto e Análise de Algoritmos

Indução

Flávio L. C. de Moura*

1 Introdução

O objetivo deste curso é estudar técnicas para análise e desenvolvimento (ou projeto) de algoritmos. As técnicas de análise permitem estimar o tempo e o espaço exigidos por um algoritmo ao processar uma determinada entrada, conceitos conhecidos como *complexidade de tempo* e *de espaço*. Já as técnicas de projeto de algoritmos se referem ao processo de construção de algoritmos a partir de um problema proposto. Neste contexto, a técnica de *divisão e conquista* resolve um problema dividindo-o em subproblemas menores, e em seguida, combinando adequadamente das soluções desses subproblemas para resolver o problema original. Além desta técnica, estudaremos outras, como a *programação dinâmica* e os *algoritmos gulosos*.

Uma vez que um algoritmo é desenvolvido, a primeira questão que devemos considerar é se ele funciona corretamente, ou seja, se ele resolve o problema de forma precisa para qualquer entrada possível. Só então devemos nos preocupar com sua análise de complexidade. Intuitivamente, dizemos que um algoritmo A é correto quando ele sempre retorna a resposta correta para qualquer entrada possível. Por exemplo, se A é um algoritmo de ordenação de inteiros, espera-se que, para qualquer lista (ou vetor) de inteiros l, A retorne uma permutação de l que esteja ordenada. Isto significa que o algoritmo produz a resposta certa para todas as entradas e faz isso em tempo finito.

Uma ferramenta fundamental que será utilizada frequentemente para provar a correção de algoritmos é a indução matemática. Além disso, na análise da eficiência dos algoritmos, usaremos várias ferramentas matemáticas, como somatórios, conjuntos, funções e matrizes. O apêndice VIII do livro [2, 3] pode ser consultado para revisar esses tópicos.

2 Indução Matemática

Indução matemática é uma técnica de prova muito poderosa que desempenha um papel fundamental tanto em Matemática quanto em Computação. Se $P(n)$ denota uma propriedade dos números naturais $\mathbb{N} = \{0, 1, 2, \ldots\}$ então o princípio da indução matemática (PIM) é dado por:

$$\frac{P\ 0 \qquad \forall k, P\ k \Longrightarrow P\ (k+1)}{\forall n, P\ n}\ (PIM)$$

Na descrição acima, chamamos $P\ 0$ de *base da indução* e $\forall k, P\ k \Longrightarrow P\ (k+1)$ de *passo indutivo*. No passo indutivo, $P\ k$ é chamado de *hipótese de indução*. Vejamos um exemplo:

*flaviomoura@unb.br

Exemplo

Considere a seguinte propriedade sobre os números naturais:

$$\text{A soma dos } n \text{ primeiros números naturais ímpares é igual a } n^2. \tag{1}$$

Esta propriedade vale trivialmente para o 0 (a soma dos 0 primeiros números ímpares é igual a 0^2), o que corresponde à base da indução. Agora seja k um natural arbitrário, e suponha que a soma dos k primeiros números ímpares seja igual a k^2 (hipótese de indução). Precisamos provar que a soma dos $k+1$ primeiros números ímpares é igual a $(k+1)^2$. De fato, o $(k+1)$-ésimo número ímpar é igual a $2.k+1$ (por que?), e portanto $k^2+2.k+1=(k+1)^2$ como queríamos provar.
Uma prova mais detalhada de (1) pode ser feita da seguinte forma: a soma dos n primeiros números ímpares pode ser escrita por meio do somatório $\sum_{i=1}^{n}(2.i-1)$, que por definição é igual a 0, se $n=0$. Queremos provar que

$$\sum_{i=1}^{n}(2.i-1)=n^2, \forall n \tag{2}$$

Aplicando o princípio da indução matemática (PIM), temos 2 casos para analisar:

- (**Base da indução**): Para $n=0$, a igualdade (2) é trivial porque o lado esquerdo da igualdade é igual a 0 por definição.
- (**Passo indutivo**): No passo indutivo assumimos que (2) vale para um número natural arbitrário, digamos k, e provamos que esta propriedade continua valendo para o natural $k+1$. Ou seja, assumimos que $\sum_{i=1}^{k}(2.i-1)=k^2$, e vamos provar que $\sum_{i=1}^{k+1}(2.i-1)=(k+1)^2$. Partindo do lado esquerdo desta última igualdade, podemos decompor o somatório da seguinte forma $\sum_{i=1}^{k+1}(2.i-1)=\sum_{i=1}^{k}(2.i-1)+(2.k+1)$, e agora podemos utilizar a hipótese de indução (h.i.) para assim chegarmos ao lado direito da mesma: $\sum_{i=1}^{k+1}(2.i-1)=\sum_{i=1}^{k}(2.i-1)+(2.k+1) \overset{h.i.}{=} k^2+(2.k+1)=(k+1)^2$.

Observe que o passo indutivo é a parte interessante de qualquer prova por indução. A base da indução consiste apenas na verificação de que a propriedade vale para uma situação particular. Agora resolva os exercícios a seguir:

Exercício 2.1. *Mostre que* $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

Exercício 2.2. *Prove que* $\sum_{i=0}^{n} i(i+1) = \frac{n.(n+1).(n+2)}{3}$.

Exercício 2.3. *Prove que* $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$.

Exercício 2.4. *Prove que* $3 \mid (2^{2n}-1)$ *para todo* $n \geq 0$.

Existem propriedades que valem apenas para um subconjunto próprio dos números naturais:

Por exemplo, $2^n < n!$ só vale para $n \geq 4$. Para este tipo de problema utilizamos uma generalização do PIM onde a base de indução não precisa ser o 0. Chamaremos esta variação de *Princípio da Indução Generalizado (PIG)*:

$$\frac{P\ m \qquad \forall k, P\ k \Longrightarrow P\ (k+1)}{\forall n \geq m, P\ n}\ (PIG)$$

Exemplo

Prove que $2^n < n!, \forall n \geq 4$.

1. (Base de indução) A propriedade vale para $n = 4$, o que é trivial, e;
2. (Passo indutivo) Mostraremos que $2^{k+1} < (S\ k)!$ assumindo que $2^k < k!, \forall k \geq 4$. De fato, temos que $2^{k+1} = 2.2^k \overset{(h.i)}{<} 2.k! \overset{(*)}{<} (k+1).k! = (k+1)!$, onde a desigualdade (*) se justifica pelo fato de k ser maior ou igual a 4.

Exercício 2.5. *Prove que a soma dos n primeiros números naturais é igual a* $\frac{n(n+1)}{2}$.

Exercício 2.6. *Prove que a soma dos n primeiros quadrados é igual a* $\frac{n(n+1)(2n+1)}{6}$. *Ou seja, mostre que* $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

Exercício 2.7. *Prove que a soma dos n primeiros cubos é igual ao quadrado da soma de 1 até n, ou seja, que* $1^3 + 2^3 + \ldots + n^3 = (1 + 2 + \ldots + n)^2$.

Exercício 2.8. *Prove que* $2^n - 1$ *é múltiplo de 3, para todo número natural n par.*

Exercício 2.9. *Prove que* $3^n \geq n^2 + 3$ *para todo* $n \geq 2$.

Exercício 2.10. *Prove que* $n^2 < 4^{n-1}$ *para todo* $n \geq 3$.

Exercício 2.11. *Prove que* $n! > 3^n$ *para todo* $n \geq 7$.

Exercício 2.12. *Prove que* $n! \leq n^n$ *para todo* $n \geq 1$.

Uma variação do PIM bastante útil é conhecida como *Princípio da Indução Forte (PIF)*:

$$\frac{\forall k, (\forall m, m < k \Longrightarrow P\ m) \Longrightarrow P\ k}{\forall n, P\ n}\ (PIF)$$

Exercício 2.13. *Prove que qualquer inteiro $n \geq 2$ é um número primo ou pode ser escrito como um produto de primos (não necessariamente distintos), i.e. na forma $n = p_1.p_2.\cdots.p_r$, onde os fatores p_i $(1 \leq i \leq r)$ são primos.*

Exercício 2.14. *Mostre que PIM e PIF são princípios equivalentes.*

3 Indução Estrutural

Nesta seção veremos que o princípio de indução matemática (PIM) visto anteriormente é um caso particular de um princípio geral que está associado a qualquer conjunto definido indutivamente. Vimos dois tipos de regras utilizadas na construção de um conjunto definido indutivamente:

1. As regras não recursivas, ou seja, aquelas que definem diretamente um elemento do conjunto definido indutivamente;
2. As regras recursivas, ou seja, aquelas que constroem novos elementos a partir de elementos já construídos.

Como veremos no próximo exemplo, estas regras podem fazer uso de elementos de outros conjuntos previamente definidos. Formalmente, se $A_1, A_2, \ldots$ são conjuntos então a estrutura geral das regras de um conjunto definido indutivamente B é como a seguir:

1. Inicialmente temos as regras não recursivas que definem diretamente os elementos $b_1, \ldots, b_m$ de B:

$$\frac{a_1 \in A_1 \qquad a_2 \in A_2 \ldots\ a_{j_1} \in A_{j_1}}{b_1[a_1, \ldots, a_{j_1}] \in B} \qquad \cdots \qquad \frac{a_1 \in A_1 \qquad a_2 \in A_2 \ldots\ a_{j_m} \in A_{j_m}}{b_m[x_1, \ldots, x_{j_m}] \in B}$$

2. Em seguida, temos as regras recursivas que constroem novos elementos a partir de elementos já construídos:

$$\frac{a_1 \in A_1 \ldots\ a_{j_1'} \in A_{j_1'}\ d_1, \ldots, d_{k_1} \in B}{c_1[x_1, \ldots, x_{j_1'}, d_1, \ldots, d_{k_1}] \in B} \qquad \cdots \qquad \frac{a_1 \in A_1 \ldots\ a_{j_n} \in A_{j_n}\ d_1, \ldots, d_{k_n} \in B}{c_n[a_1, \ldots, a_{j_n}, d_1, \ldots, d_{k_n}] \in B}$$

Qualquer elemento de um conjunto definido indutivamente pode ser construído após um número finito de aplicações das regras que o definem (e somente com estas regras). Os elementos $d_1, d_2, \ldots, d_{k_i}$ são ditos *estruturalmente menores* do que o elemento $c_i[a_1, \ldots, a_{j_i}, d_1, \ldots, d_{k_i}]$. Isto significa que os elementos $d_1, d_2, \ldots, d_{k_i}$ são subtermos próprios de $c_i[a_1, \ldots, a_{j_i}, d_1, \ldots, d_{k_i}]$.

Podemos associar um princípio de indução a qualquer conjunto definido indutivamente. No contexto genérico acima, teremos um caso base (base da indução) para cada regra não recursiva, e um passo indutivo para cada regra recursiva. O esquema simplificado (omitindo os parâmetros por falta de espaço) tem a seguinte forma:

$$\frac{\overbrace{P(b_1)\dots P(b_m)}^{\text{casos base}} \quad \overbrace{(\forall d_1 \dots d_{k_1}, P(d_1), \dots, P(d_{k_1}) \Rightarrow P(c_1)) \dots (\forall d_1 \dots d_{k_1}, P(d_1), \dots, P(d_{k_n}) \Rightarrow P(c_n))}^{\text{casos indutivos}}}{\forall x \in B, P\ x}$$

Retornando ao caso do conjunto dos números naturais, temos um princípio indutivo com apenas um caso base e um caso indutivo:

$$\frac{P\ 0 \qquad \forall k, P\ k \Longrightarrow P\ (S\ k)}{\forall n, P\ n}$$

O conjunto dos booleanos possui um princípio indutivo com dois casos base, e nenhum caso indutivo:

$$\frac{P\ \text{true} \qquad P\ \text{false}}{\forall b, P\ b}$$

Futuramente estudaremos diversos algoritmos que utilizam a estrutura de lista encadeada, definida pela seguinte gramática $l ::= nil \mid a :: l$, onde nil representa a lista vazia, e $a :: l$ representa a lista com primeiro elemento a e cauda l. Como esta gramática possui um construtor não recursivo, e um construtor recursivo, teremos um princípio de indução com um caso base, e um passo indutivo:

$$\frac{P\ nil \qquad \forall l\ h, P\ l \Longrightarrow P\ (h :: l)}{\forall l, P\ l}$$

O comprimento de uma lista, isto é, o número de elementos que a lista possui, é definido recursivamente por:

$$|l| = \begin{cases} 0, & \text{se } l = nil \\ 1 + |l'|, & \text{se } l = a :: l' \end{cases}$$

Uma operação importante que nos permite construir uma nova lista a partir de duas listas já construídas é a concatenação. Podemos definir a concatenação de duas listas por meio da seguinte função recursiva:

$$l_1 \circ l_2 = \begin{cases} l_2, & \text{se } l_1 = nil \\ a :: (l' \circ l_2), & \text{se } l_1 = a :: l' \end{cases}$$

Por fim, o reverso de uma lista é definido recursivamente por:

$$rev(l) = \begin{cases} l, & \text{se } l = nil \\ (rev(l')) \circ (a :: nil), & \text{se } l = a :: l' \end{cases}$$

Os exercícios a seguir expressam diversas propriedades envolvendo estas operações. Resolva cada um deles utilizando indução.

Exercício 3.1. *Prove que $|l_1 \circ l_2| = |l_1| + |l_2|$, quaisquer que sejam as listas l_1, l_2.*

Exercício 3.2. *Prove que $l \circ nil = l$, qualquer que seja a lista l.*

Exercício 3.3. *Prove que a concatenação de listas é associativa, isto é, $(l_1 \circ l_2) \circ l_3 = l_1 \circ (l_2 \circ l_3)$ quaisquer que sejam as listas l_1, l_2 e l_3.*

Exercício 3.4. *Prove que* $|rev(l)| = |l|$*, qualquer que seja a lista* l*.*

Exercício 3.5. *Prove que* $rev(l_1 \circ l_2) = (rev(l_2)) \circ (rev(l_1))$*, quaisquer que sejam as listas* l_1, l_2*.*

Exercício 3.6. *Prove que* $rev(rev(l)) = l$*, qualquer que seja a lista* l*.*

4 Leitura complementar:

- [4] (Capítulo 2)
- [1] (Capítulo 1)

Referências

[1] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics.* Prentice-Hall, Inc., USA, 1996.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, Cambridge, MA, USA, 4 edition, April 2022.

[4] Udi Manber. *Introduction to Algorithms: A Creative Approach.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.