Notas sobre crescimento de função

Anjolina Grisi de Oliveira

Centro de Informática
Universidade Federal de Pernambuco

CIn-UFPE

Motivação

- Estimar custo de algoritmo em relação ao tempo e espaço.
- Noção de problemas computáveis
- Noção problemas tratáveis

Definição

Definição

Sejam f e g funções do conjunto dos inteiros (ou conjunto dos números reais) no conjunto dos números reais. Dizemos que $f(x)$ é $O(g(x))$ se existem constantes c e n_0 de forma que:

$$| f(x) | \leq C | g(x) |$$

quando $x > n_0$. Dizemos que $f(x)$ é "O grande de $g(x)$".

Observações

- Usamos a notação $f(x) = O(g(x))$ quando temos que $f(x)$ é "O grande de $g(x)$". Isso significa que $f(x)$ é *limitada superiormente por* $g(x)$. Aqui há uma abuso de notação, pois na realidade $f(x) \in O(g(x))$, pois $O(g(x))$ é o conjunto das funções que g limita superiormente.
- A notação "*big-Oh*" ("O grande ") é utilizada para expressar formalmente um limite superior para a função. Quando dizemos que $f(x)$ é $O(x^2)$, queremos dizer que f *nunca cresce mais que* $c.x^2$ a partir de um determinado valor de x e para uma constante c.

Observações

- Usamos a notação $f(x) = O(g(x))$ quando temos que $f(x)$ é "O grande de $g(x)$". Isso significa que $f(x)$ é *limitada superiormente por* $g(x)$. Aqui há uma abuso de notação, pois na realidade $f(x) \in O(g(x))$, pois $O(g(x))$ é o conjunto das funções que g limita superiormente.
- A notação "*big-Oh*" ("O grande ") é utilizada para expressar formalmente um limite superior para a função. Quando dizemos que $f(x)$ é $O(x^2)$, queremos dizer que f *nunca cresce mais que* $c.x^2$ a partir de um determinado valor de x e para uma constante c.

Observações

- Em termos de algoritmos, quando dizemos que dentre as soluções algoritmicas para um determinado problema, a melhor é $O(x^2)$ em função da entrada, isso significa que se alguém achar um outro algoritmo cuja função para estimar o tempo é $O(logn)$ significa que a **cota superior** ("*upper bound*") para aquele problema diminuiu.

Exemplos

Exemplo

Prove que $f(x) = x^2 + 2x + 1$ *é* $O(x^2)$.
$x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 \rightarrow x^2 + 2x + 1 \leq 4x^2$.
Logo, $c = 4$ *e* $x > 1$ *(*$n_0 = 1$*).*
Outra abordagem:
$2x \leq x^2$ *para* $x > 2$. *Logo* $x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$.
Logo $c = 3$ *e* $x > 2$.

Mais observações

- Observe que qualquer função com valores maiores que x^2 é cota superior para $f(x)$. Dessa forma, se $f(x)$ é $O(x^2)$, $f(x)$ é também $O(x^3), O(x^4)$, etc.
- Também é verdade que x^2 é $O(x^2 + 2x + 1)$, pois $x^2 \leq x^2 + 2x + 1$, para $x > 1$. Nesse caso, as duas funções são da mesma ordem.
- O monômio de maior grau domina o crescimento de funções polinomiais. Dessa forma podemos ignorar todos os monômios de menor grau. Isso é estabelecido no teorema dado a seguir.

Teorema

Seja $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, *onde* $a_0, a_1, \ldots a_{n-1}, a_n$ *são números reais. Então* $f(n)$ *é* $O(x^n)$.

Mais observações

- Observe que qualquer função com valores maiores que x^2 é cota superior para $f(x)$. Dessa forma, se $f(x)$ é $O(x^2)$, $f(x)$ é também $O(x^3), O(x^4)$, etc.
- Também é verdade que x^2 é $O(x^2 + 2x + 1)$, pois $x^2 \leq x^2 + 2x + 1$, para $x > 1$. Nesse caso, as duas funções são da mesma ordem.
- O monômio de maior grau domina o crescimento de funções polinomiais. Dessa forma podemos ignorar todos os monômios de menor grau. Isso é estabelecido no teorema dado a seguir.

Teorema

Seja $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, onde $a_0, a_1, \ldots a_{n-1}, a_n$ são números reais. Então $f(n)$ é $O(x^n)$.

Mais observações

- Observe que qualquer função com valores maiores que x^2 é cota superior para $f(x)$. Dessa forma, se $f(x)$ é $O(x^2)$, $f(x)$ é também $O(x^3), O(x^4)$, etc.
- Também é verdade que x^2 é $O(x^2 + 2x + 1)$, pois $x^2 \leq x^2 + 2x + 1$, para $x > 1$. Nesse caso, as duas funções são da mesma ordem.
- O monômio de maior grau domina o crescimento de funções polinomiais. Dessa forma podemos ignorar todos os monômios de menor grau. Isso é estabelecido no teorema dado a seguir.

Teorema

Seja $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, *onde* $a_0, a_1, \ldots a_{n-1}, a_n$ *são números reais. Então* $f(n)$ *é* $O(x^n)$.

Mais exemplos

- Como estimar a soma dos primeiros n inteiros positivos.
- $1 + 2 + 3 + \ldots + n \leq n + n + \ldots + n = n.n = n^2$.
- Logo, a soma dos primeiros n inteiros positivos é $O(n^2)$.

Mais exemplos

- Como estimar a soma dos primeiros n inteiros positivos.
- $1 + 2 + 3 + \ldots + n \leq n + n + \ldots + n = n.n = n^2$.
- Logo, a soma dos primeiros n inteiros positivos é $O(n^2)$.

Mais exemplos

- Como estimar a soma dos primeiros n inteiros positivos.
- $1 + 2 + 3 + \ldots + n \leq n + n + \ldots + n = n.n = n^2$.
- Logo, a soma dos primeiros n inteiros positivos é $O(n^2)$.

Mais exemplos

- Estimando $n!$: $n! = 1.2.3.\ldots.n \leq n.n.n.\ldots.n = n^n$.
- Logo $n!$ é $O(n^n)$.
- Podemos concluir também que $\log n! \leq \log n^n \rightarrow \log n! \leq n.\log n$. Logo, $\log n!$ é $O(n.\log n)$.
- Para $n > 1$ temos que $n < 2^n$. Logo, $\log n < \log 2^n$ (*usando a base 2* $\rightarrow \log n < n$. Logo $\log n$ é $O(n)$.

Mais exemplos

- Estimando $n!$: $n! = 1.2.3.\ldots.n \leq n.n.n.\ldots.n = n^n$.
- Logo $n!$ é $O(n^n)$.
- Podemos concluir também que $\log n! \leq \log n^n \rightarrow \log n! \leq n.\log n$. Logo, $\log n!$ é $O(n.\log n)$.
- Para $n > 1$ temos que $n < 2^n$. Logo, $\log n < \log 2^n$ (*usando a base 2* $\rightarrow \log n < n$. Logo $\log n$ é $O(n)$.

Mais exemplos

- Estimando $n!$: $n! = 1.2.3.\ldots.n \leq n.n.n.\ldots.n = n^n$.
- Logo $n!$ é $O(n^n)$.
- Podemos concluir também que $\log n! \leq \log n^n \rightarrow \log n! \leq n.\log n$. Logo, $\log n!$ é $O(n.\log n)$.
- Para $n > 1$ temos que $n < 2^n$. Logo, $\log n < \log 2^n$ (*usando a base 2* $\rightarrow \log n < n$. Logo $\log n$ é $O(n)$.

Mais exemplos

- Estimando $n!$: $n! = 1.2.3.\ldots.n \leq n.n.n.\ldots.n = n^n$.
- Logo $n!$ é $O(n^n)$.
- Podemos concluir também que $\log n! \leq \log n^n \rightarrow \log n! \leq n.\log n$. Logo, $\log n!$ é $O(n.\log n)$.
- Para $n > 1$ temos que $n < 2^n$. Logo, $\log n < \log 2^n$ (*usando a base 2* $\rightarrow \log n < n$. Logo $\log n$ é $O(n)$.

O crescimento de combinação de funções

Teorema

Suponha que $f1(x)$ *é* $O(g1(x)$ *e* $f2(x)$ *é* $O(g2(x))$. *Então* $(f1 + f2)(x)$ *é* $O(max(|\ g1(x)\ |, |\ g2(x)\ |)$.

- Se $f1(x)$ e $f2(x)$ são ambas $O(g(x))$. Então $(f1 + f2)(x)$ é O de que função?

O crescimento de combinação de funções

Teorema

Suponha que $f1(x)$ *é* $O(g1(x))$ *e* $f2(x)$ *é* $O(g2(x))$*. Então* $(f1f2)(x)$ *é* $O(g1(x).g2(x))$*.*

Exemplos

Determine o menor inteiro n de forma que $f(x)$ é $O(x^n)$ para cada uma das seguintes funções. Justifique sua resposta.

a) $f(x) = (x^5 + x^2 + 1)/(x^4 + 1)$

b) $f(x) = 5x^2 + x^3 log x + x$

Exemplos

Dê a melhor estimativa *O grande* possível para as seguintes funções:

a) $(n^3 + n^2 logn)(logn + 1) + (15logn + 30)(n^3 + 2)$

b) $(nlogn + 1)^2 + (logn + 1)(n^2 + 1)$

Definição

- Quando duas funções possuem a mesma ordem de grandeza usamos a notação "*big-theta*" (Θ) para expressar essa relação.

Definição

Sejam f e g funções do conjunto dos inteiros ou do conjunto dos números reais no conjunto dos números reais. Dizemos que $f(x)$ é $\Theta(g(x))$ se existem constantes n_0, c_1, c_2, de forma que se $x > n_0$ então $c_1 g(x) \leq f(x) \leq c_2 g(x)$.

- É como se as funções $c_1 g(x)$ e $c_2 g(x)$ formasem um envelope em torno dos valores de $f(x)$. Uma mudança nos valores das constantes muda a largura do envelope.

Definição

- Usamos a notação "big-omega" (Ω) para denotar um limite inferior ou cota inferior ("*lower bound*").
- Em termos de algoritmos, significa que às vezes para um determinado problema é possível mostrar que qualquer algoritmo usado requer um mínimo de operações.

Definição

Sejam f e g funções do conjunto dos inteiros (ou conjunto dos números reais) no conjunto dos números reais. Dizemos que $f(x)$ é $\Omega(g(x))$ se existem constantes c e n_0 de forma que:

$$| f(x) | \leq C | g(x) |$$

quando $x > n_0$. Dizemos que $f(x)$ é "big-Omega de $g(x)$".

Definição

- Usamos a notação "big-omega" (Ω) para denotar um limite inferior ou cota inferior ("*lower bound*").
- Em termos de algoritmos, significa que às vezes para um determinado problema é possível mostrar que qualquer algoritmo usado requer um mínimo de operações.

Definição

Sejam f e g funções do conjunto dos inteiros (ou conjunto dos números reais) no conjunto dos números reais. Dizemos que $f(x)$ é $\Omega(g(x))$ se existem constantes c e n_0 de forma que:

$$| f(x) | \leq C | g(x) |$$

quando $x > n_0$. Dizemos que $f(x)$ é "big-Omega de $g(x)$".