Algoritmos e Estrutura de Dados

Fabrício Olivetti de França

02 de Fevereiro de 2019

UFABC

CMCC
Centro de Matemática, Computação e Cognição
Center for Mathematics, Computation and Cognition

Topics

Análise de Algoritmos

Custo de um algoritmo

Frequentemente parte da solução de um problema admite:

- Múltiplos algoritmos
- Múltiplas estruturas de dados com as mesmas funções básicas

Custo de um algoritmo

Nessas situações, como fazer uma escolha acertada?

A Wikipedia lista cerca de 70 algoritmos de ordenação! Qual deles é o melhor e em qual situação?

Custo de um algoritmo

Para comparar precisamos quantificar a qualidade de um algoritmo, isso pode ser feito medindo

- O custo computacional do algoritmo
- O uso de memória
- Combinação de ambos

Custo de um algoritmo

Considere o seguinte algoritmo para encontrar o maior valor de uma lista:

```
int maximo (int * x, int n) {
.L1     int j = n-1;
.L2     for (int k=n-2; k>=0; k--)
        {
.L3         if (x[k] > x[j])
            {
.L4             j = k;
            }
        }
.L5     return X[j];
}
```

Custo de um algoritmo

Esse algoritmo percorre a lista do final para o começo atualizando o índice `j` sempre que encontra uma alternativa de maior valor.

Custo de um algoritmo

Linha	operações
L1	1
L2	n
L3	n-1
L4	A
L5	1

Custo de um algoritmo

O custo dess algoritmo em número de operações é:

$1 + n + n - 1 + A + 1 = 2n + A + 1.$

Em uma análise mais detalhada, traduziríamos o código para a linguagem Assembly e, em seguida, consideraríamos o custo de ciclos de CPU para cada instrução.

Custo de um algoritmo

Instrução	ciclos
ADD	1
MOV	1
JMP	1
XOR	2

Retornando a nossa análise, uma vez que n é dado, precisamos estimar o valor de A. Podemos fazer uma análise:

- **Otimista:** menor valor possível para A.
- **Pessimista:** maior valor possível para A.
- **Probabilística:** calcular a média e desvio-padrão (quão perto esperamos que o valor esteja da média).

Custo de um algoritmo

O menor valor para A é **zero**, ocorre quando X_{n-1} contém o maior valor.

O maior valor para A é **n-1**, ocorre quando X_0 contém o maior valor.

Custo de um algoritmo

A média está no intervalo $[0, n-1]$. Mas seu valor é $\frac{n}{2}$? $\sqrt{n}$?

Vamos considerar o índice começando do 1 e terminando em n e assumir que $X_1 \neq X_2 \neq \ldots \neq X_n$.

Custo de um algoritmo

Considerando que as probabilidades de quaisquer permutações de X são iguais.

Obs.: se conhecermos algum detalhe da natureza dos dados podemos fazer outra suposição.

Permutações e o valor de A

Para $n = 3$:

π	A
$X_1 < X_2 < X_3$	0
$X_1 < X_3 < X_2$	1
$X_2 < X_1 < X_3$	0
$X_2 < X_3 < X_1$	1
$X_3 < X_1 < X_2$	1
$X_3 < X_2 < X_1$	2

Permutações e o valor de A

Qual a média dos valores de A?

π	A
$X_1 < X_2 < X_3$	0
$X_1 < X_3 < X_2$	1
$X_2 < X_1 < X_3$	0
$X_2 < X_3 < X_1$	1
$X_3 < X_1 < X_2$	1
$X_3 < X_2 < X_1$	2

Permutações e o valor de A

$$\frac{0+1+0+1+1+2}{6} = \frac{5}{6}$$

Permutações e o valor de A

p_{nk} = probabilidade que $A = k$ para n objetos.

P_{nk} = em quantas permutações de n objetos $A = k$.

$$p_{nk} = \frac{P_{nk}}{n!}$$

Permutações e o valor de A

$p_{30} = \frac{2}{6} = \frac{1}{3}$

$p_{31} = \frac{3}{6} = \frac{1}{2}$

$p_{32} = \frac{1}{6}$

Média e Desvio-Padrão

A média pode ser calculada como:

$A_n = \sum_k k \cdot p_{nk}$

A variância V_n é definida como a média de $(A - A_n)^2$:

$$V_n = \sum_k (k - A_n)^2 p_{nk} = \sum_k k^2 p_{nk} - 2A_n \sum_k k \cdot p_{nk} + A_n^2 \sum_k p_{nk}$$

$$V_n = \sum_k k^2 p_{nk} - 2A_n A_n + A_n^2 = \sum_k k^2 p_{nk} - A_n^2$$

Média e Desvio-Padrão

O Desvio-Padrão é simplesmente $\sigma_n = \sqrt{V_n}$.

Permutações e o valor de A

Lembrando que:

$$p_{nk} = \frac{P_{nk}}{n!}$$

temos que:

$$P_{nk} = n! p_{nk}$$

Permutações e o valor de A

Assumindo que $X_1, X_2, \ldots, X_n$ pode ser qualquer permutação de valores.

Fixando $X_1 = n$ e fixando todo o resto de forma arbitrária, podemos dizer que:

$A_n^{X_1 \ldots X_n} = A_{n-1}^{X_2 \ldots X_n} + 1$

Pois precisará fazer uma troca a mais!

Permutações e o valor de A

Da mesma forma, se fizermos $X_1 \neq n$, temos:

$$A_n^{X_1 \dots X_n} = A_{n-1}^{X_2 \dots X_n}$$

Pois não precisaremos trocar o último elemento.

Permutações e o valor de A

$$P_{nk} = \underbrace{P_{(n-1)(k-1)}}_{X_1=n} + \underbrace{(n-1)}_{X_1=1,2,\ldots n-1} \cdot \underbrace{P_{(n-1)k}}_{X_1\neq n}$$

Permutações e o valor de A

$$
\begin{aligned}
n! \cdot p_{nk} &= (n-1)! \cdot p_{(n-1)(k-1)} + (n-1)!(n-1) \cdot p_{(n-1)k} \\
p_{nk} &= \tfrac{1}{n} \cdot p_{(n-1)(k-1)} + \tfrac{(n-1)}{n} \cdot p_{(n-1)k}
\end{aligned}
$$

Permutações e o valor de A

Basta definirmos as condições inicias da recursão:

$$p_{1k} = \begin{cases} 1, & \text{se } k = 0 \\ 0, & \text{c.c.} \end{cases}$$

$$p_{nk} = 0, k < 0$$

Permutações e o valor de A

$$
\begin{aligned}
p_{30} &= \tfrac{1}{3} \cdot p_{2(-1)} + \tfrac{2}{3} \cdot p_{20} \\
&= \tfrac{2}{3} \cdot \left(\tfrac{1}{2} \cdot p_{1(-1)} + \tfrac{1}{2} \cdot p_{10}\right) \\
&= \tfrac{2}{3} \cdot \tfrac{1}{2} \cdot 1 \\
&= \tfrac{1}{3}
\end{aligned}
$$

Permutações e o valor de A

$$
\begin{aligned}
p_{31} &= \frac{1}{3} \cdot p_{20} + \frac{2}{3} \cdot p_{21} \\
&= \frac{1}{3}\frac{1}{2} + \frac{2}{3} \cdot \left(\frac{1}{2} \cdot p_{10} + +frac12 \cdot p_{11}\right) \\
&= \frac{1}{6} + \frac{2}{3}\frac{1}{2} \\
&= \frac{1}{2}
\end{aligned}
$$

Permutações e o valor de A

$$
\begin{aligned}
p_{32} &= \tfrac{1}{3} \cdot p_{21} + \tfrac{2}{3} \cdot p_{22} \\
&= \tfrac{1}{3}\tfrac{1}{2} + \tfrac{2}{3} \cdot \left(\tfrac{1}{2}p_{11} + \tfrac{1}{2}p_{12}\right) \\
&= \tfrac{1}{3}\tfrac{1}{2} \\
&= \tfrac{1}{6}
\end{aligned}
$$

Tendo o valor de p_{nk} é possível calcular A_n, σ_n.

Permutações e o valor de A

A sequência de passos para obter esses valores fogem do escopo desse curso (e requer conhecimento de funções geradoras e séries), porém o resultado será:

$A_n = H_n - 1 = \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \approx \ln n$

para n grande.

Representação Assintótica

Representação Assintótica

A **Análise Assintótica** é utilizada para determinar o comportamento aproximado de um algoritmo para valores grandes de n.

Notação-O

A **notação-O** é utilizada na matemática para definir termos de erro de aproximação.

Nessa notação os termos mais significativos são escritos explicitamente e o restante são *compactados* com essa notação.

Notação-O

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \\ &= 1 + x + \frac{x^2}{2!} + O\big(x^3\big) \\ &= 1 + x + O\big(x^2\big) \end{aligned}$$

para $x \to 0$. Lemos $O\big(f(n)\big)$ como uma quantidade desconhecida de baixa magnitude.

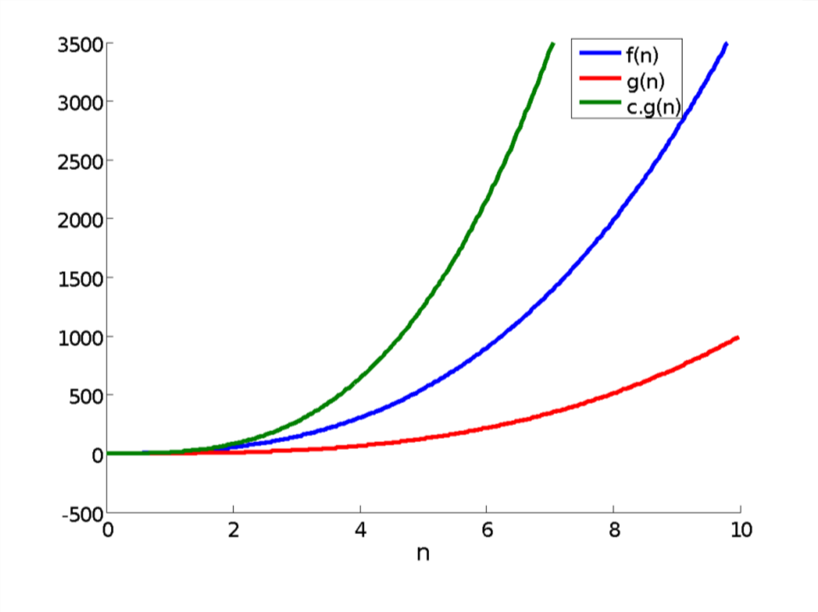
A definição da função $O(.)$ para análise de algoritmos é a de mostrar o termo dominante quando $n \longrightarrow \infty$.

Notação-O

$O(f(n)) : \exists M, n_0, |g(n)| \leq M|f(n)| \forall n \geq n_0$

Para todo $n \geq n_0$, $g(n)$ se mantém menor que $M \cdot f(n)$.

Notação-O



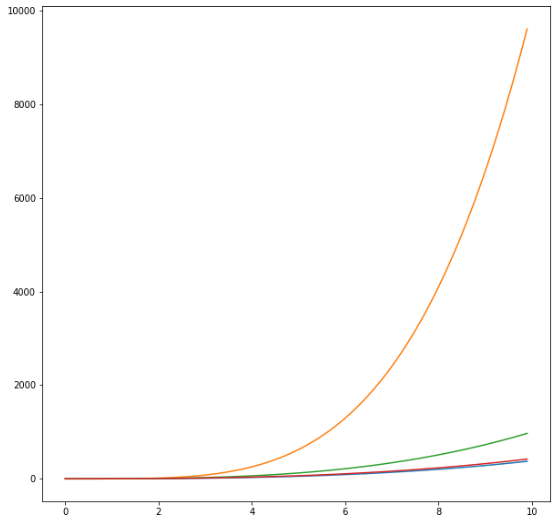
Notação-O

Notação: dizemos que $g(n) = O(f(n))$ se $f(n)$ aproxima assintoticamente o comportamento de $g(n)$. Não podemos dizer que $O(f(n)) = g(n)$.

Notação-O

$$
\begin{aligned}
\tfrac{1}{3}n^3 + \tfrac{1}{2}n^2 + \tfrac{1}{6}n &= O\left(n^4\right) \\
\tfrac{1}{3}n^3 + \tfrac{1}{2}n^2 + \tfrac{1}{6}n &= O\left(n^3\right) \\
\tfrac{1}{3}n^3 + \tfrac{1}{2}n^2 + \tfrac{1}{6}n &= \tfrac{1}{3}n^3 + O\left(n^2\right)
\end{aligned}
$$

Notação-O



Notação-O

Quanto mais próxima da equação real, mais forte é a aproximação.

Notação-O para polinômios

$$p(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_m n^m = O(n^m)$$

Notação-O para polinômios

$$\begin{aligned} |p(n)| &\leq |a_0| + |a_1|n + |a_2|n^2 + \ldots + |a_m|n^m \\ &= (|a_0|/n^m + |a_1|/n^{m-1} + |a_2|/n^{m-2} + \ldots + |a_m|)n^m \end{aligned}$$

Notação-O para polinômios

$$(|a_0|/n^m + |a_1|/n^{m-1} + |a_2|/n^{m-2} + \ldots + |a_m|)n^m \leq$$
$$(|a_0| + |a_1| + |a_2| + \ldots + |a_m|)n^m$$

para $n \geq 1$.

$$M = a_0| + |a_1| + |a_2| + \ldots + |a_m|, n_0 = 1$$

Propriedades

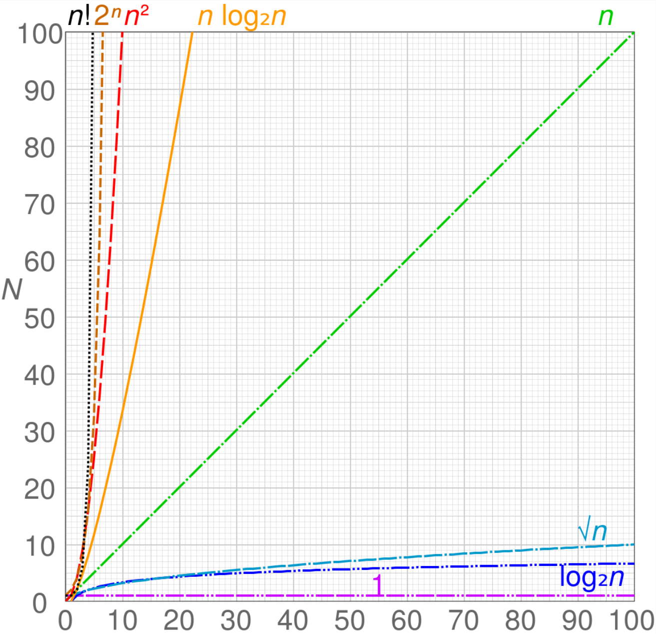
$$
\begin{aligned}
f(n) &= O(f(n)) \\
cO(f(n)) &= O(f(n)) \\
O(f(n)) + O(g(n)) &= O(f(n) + g(n)) \\
O(O(f(n))) &= O(f(n)) \\
O(f(n))O(g(n)) &= O(f(n)g(n)) \\
O(f(n)g(n)) &= f(n)O(g(n))
\end{aligned}
$$

Notações comuns na computação

Lista de notações em ordem crescente de complexidade

notação	nome
$O(1)$	constante
$O(\log \log n)$	duplo logaritmo
$O(\log n)$	logaritmo
$O(n)$	linear
$O(n \log n)$	loglinear
$O(n^2)$	quadrático
$O(n^c)$	polinomial
$O(c^n)$	exponencial
$O(n!)$	fatorial

Notações comuns na computação



Notação-O

A notação-O não pode ser interpretada como um limitante inferior ou superior: $O(n^2)$ não implica que um algoritmo não pode executar em $O(n)$ em alguns casos.

Outras notações

Outras notações possíveis:

- notação-Ω: define um limitante inferior.
- notação-Θ: define um valor exato exceto por uma constante.

Próxima aula

Aprenderemos sobre a **álgebra dos tipos**, **registros** e **estruturas lineares**.