Construção e Análise de Algoritmos

Discussão: Análise do tempo médio de execução do algoritmo *Quicksort*

Alguns podem ter ficado insatisfeitos com a análise que nós fizemos do algoritmo Quicksort.

Quer dizer, nós dissemos que o seu desempenho de pior caso é mesmo muito ruim, mas que na prática ele funciona bem.

Até aqui tudo certo, isso é verdade.

Mas, as justificativas que nós apresentamos para essa observação não foram muito rigorosas.

A seguir, nós vamos apresentar uma análise matemática do número médio de comparações realizadas pelo algoritmo.

Para isso, suponha que o algoritmo ordena uma lista com os números $1, 2, \ldots, 100$.

Essa suposição não é importante, mas ajuda a tornar as coisas mais concretas.

Além disso, suponha que o algoritmo será executado 1000 vezes.

Essa é uma análise probabilística, daí as repetições.

Finalmente, assuma que o procedimento `Partição` sempre utiliza um elemento escolhido aleatoriamente como pivot.

Dessa maneira, as execuções serão todas diferentes — e daí faz sentido estimar o tempo médio das execuções.

A primeira pergunta que nós fazemos nesse contexto é a seguinte:

- *Quantas vezes os números 17 e 18 são comparados nas 1000 execuções do algoritmo?*

A observação chave é que 17 e 18 são números consecutivos (não existe ninguém entre eles).

Isso significa que, no momento de uma partição, os números 17 e 18 sempre vão juntos para o mesmo lado — e não são comparados um com o outro.

A menos que um deles seja escolhido como pivot!

Nesse caso, aquele que foi escolhido como pivot ficará no meio, entre as duas partições.

E, o mais imporante, nesse momento ocorre a comparação entre 17 e 18.

Ora, mas isso sempre acontece.

Quer dizer, mais cedo ou mais tarde o 17 ou o 18 terá que ser escolhido como pivot.

(porque?)

Portanto, os números 17 e 18 serão comparados em todas as 1000 repetições do algoritmo.

(E o mesmo vale para qualquer par de elementos consecutivos.)

A próxima pergunta é a seguinte:

- *Quantas vezes os números 17 e 19 são comparados nas 1000 execuções do algoritmo?*

Bom, dessa vez a situação é um pouco diferente.

Os números 17 e 19 são próximos um do outro, e quase sempre que ocorre uma partição eles vão para o mesmo lado.

A menos que a partição esteja sendo realizada com o 18 como pivot!

Nesse caso, o 17 vai para um lado e o 19 vai para o outro.

E eles não são comparados entre si — nem agora, nem mais tarde.

Por outro lado, se o 17 ou o 19 é escolhido como pivot para uma partição (isto é, antes do 18 ter sido escolhido), então a comparação entre eles acontece.

Isto é, o destino é selado no momento em que um dos elementos do conjunto $\{17, 18, 19\}$ é escolhido como pivot — o que tem que acontecer, necessariamente.

Em dois casos a comparação acontece — i.e., quando 17 ou 19 é o escolhido.

E em um caso a comparação não acontece — i.e., quando 18 é o escolhido.

Então, como o pivot é escolhido de maneira completamente aleatória, a chance de que a comparação aconteça é de 2/3.

E isso significa que, nas 1000 repetições do algoritmo, os números 17 e 19 serão comparados em média $2/3 \times 1000 \;\simeq\; 667$ vezes.

(E o mesmo vale para qualquer par que possua exatamente um elemento entre eles.)

Agora as coisas estão começando a ficar mais claras.

Suponha que perguntamos em seguida

- *Quantas vezes os números 17 e 20 são comparados nas 1000 execuções do algoritmo?*

De acordo com o raciocínio acima, o destino dessa comparação é selado no momento em que um dos elementos do conjunto $\{17, 18, 19, 20\}$ é escolhido como pivot.

Em dois casos a comparação acontece — i.e., quando 17 ou 20 é o escolhido.

E em dois casos a comparação não acontece — i.e., quando 18 ou 19 é o escolhido.

Logo, a chance de que a comparação aconteça é de 1/2.

E isso significa que, nas 1000 repetições do algoritmo, os números 17 e 20 serão comparados em média $1/2 \times 1000 \;\simeq\; 500$ vezes.

(E o mesmo vale para qualquer par que possua exatamente dois elementos entre eles.)

Nesse ponto, já é possível generalizar.

Considere o par de números i e $i + k$.

O destino da comparação entre i e $i + k$ é selado no momento em que algum elemento do conjunto $\{i, i + 1, \ldots, i + k\}$ é escolhido como pivot.

Em dois casos a comparação acontece — i.e., quando i ou $i + k$ é o escolhido.

E em $k - 1$ casos a comparação não acontece — i.e., algum dois outros elementos é o escolhido.

Logo, a chance de que a comparação aconteça é de 1/2.

E isso significa que, nas 1000 repetições do algoritmo, os números i e $i + k$ serão comparados em média $2/(k + 1) \times 1000 \;\simeq\; 2000/(k + 1)$ vezes.

Agora nós já estamos prontos para finalizar o argumento.

Basta observar que

- Existem $n - 1$ pares de elementos consecutivos, que são comparados com probabilidade 1

Nas 1000 repetições do algoritmo, esses pares vão contribuir com

$$(n - 1) \cdot 1 \cdot 1000 \qquad \text{comparações}$$

- Existem $n - 2$ pares de elementos com diferença igual a 2, que são comparados com probabilidade 2/3

Nas 1000 repetições do algoritmo, esses pares vão contribuir com

$$(n - 2) \cdot \frac{2}{3} \cdot 1000 \qquad \text{comparações, em média}$$

- Existem $n - 3$ pares de elementos com diferença igual a 2, que são comparados com probabilidade 2/4

Nas 1000 repetições do algoritmo, esses pares vão contribuir com

$$(n - 3) \cdot \frac{2}{4} \cdot 1000 \qquad \text{comparações, em média}$$

E assim por diante ...

Portanto, o número total de comparações nas 1000 repetições do algoritmo é dado pela soma

$$\frac{2(n-1)}{2} \cdot 1000 \;\;+\;\; \frac{2(n-2)}{3} \cdot 1000 \;\;+\;\; \frac{2(n-3)}{4} \cdot 1000 \;\;+\;\; \ldots \;\;+\;\; \frac{2(1)}{n} \cdot 1000$$

Arredondando para cima, para simplificar as coisas, nós obtemos:

$$\frac{2n}{2} \cdot 1000 \quad + \quad \frac{2n}{3} \cdot 1000 \quad + \quad \frac{2n}{4} \cdot 1000 \quad + \quad \ldots \quad + \quad \frac{2n}{n} \cdot 1000$$

E, a seguir, uma pequena reorganização nos dá

$$2000n \cdot \Big[\frac{1}{2} \; + \; \frac{1}{3} \; + \; \frac{1}{4} \; + \; \ldots \; + \; \frac{1}{n} \Big]$$

Os termos entre colchetes não formam nem uma PA e nem uma PG, e aqui é preciso um truque novo para estimar a sua soma.

A ideia, mais uma vez, é arredondar para cima.

Em particular, nós observamos que

- $\frac{1}{2} + \frac{1}{3} \;\; \leq \;\; \frac{1}{2} + \frac{1}{2} \;\; = \;\; 1$
- $\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \;\; \leq \;\; \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} \;\; = \;\; 1$
- $\frac{1}{8} + \frac{1}{9} + \ldots + \frac{1}{15} \;\; \leq \;\; \frac{1}{8} + \frac{1}{8} + \ldots + \frac{1}{8} \;\; = \;\; 1$

Ou seja, arredondando os termos para a potência de 2 mais próxima, e formando grupos de tamanho 2, 4, 8, . . . , nós observamos que todos os grupos tem soma 1.

A seguir, nós observamos que os $n-1$ termos da soma podem formar no máximo $\log_2 n$ grupos.

E isso já nos permite concluir que

$$\frac{1}{2} \; + \; \frac{1}{3} \; + \; \frac{1}{4} \; + \; \ldots \; + \; \frac{1}{n} \quad \leq \quad \log_2 n$$

Substituindo esse resultado na expressão acima, nós chegamos à conclusão de que as 1000 repetições do algoritmo realizam não mais que

$$2000n \cdot \log_2 n \qquad \text{comparações, em média}$$

Mas, isso significa que o número médio de comparações em uma única execução do algoritmo é

$$2n \log_2 n \quad = \quad O(n \log n)$$

que é o resultado que nós queríamos demonstrar.