

The back of the envelope  
Seminario feito a partir da 7ª coluna do livro  
programming pearls

Hugo Bolzan  
Felipe Stein  
Arthur da Matta  
2026

## Sumário

<b>Sumário</b>	<b>1</b>
<b>1 Problema a ser resolvido</b>	<b>2</b>
<b>2 Regras e Técnicas de Estimativa utilizadas</b>	<b>3</b>
2.1 A Regra do 72	3
2.2 $\pi$ segundos é um nanocéculo	3
2.3 Análise Dimensional	3
2.4 Duas respostas são melhores que uma	3
<b>3 Aplicação Prática e Fatores de Segurança</b>	<b>4</b>
3.1 Estimativas de Performance	4
3.2 Fatores de Segurança (O exemplo da Ponte do Brooklyn)	4
3.3 Little Law (Lei de Little)	4
<b>4 Conclusão</b>	<b>5</b>
<b>5 Referências</b>	<b>5</b>

# 1 Problema a ser resolvido

O problema central que a coluna traz é a falha em verificar a viabilidade básica de um sistema antes de construí-lo. Bentley argumenta que muitos programadores e engenheiros mergulham na codificação e em designs complexos sem fazer uma conta de "padaria" para saber se o que estão planejando é fisicamente possível.

Para ilustrar como esse erro de percepção acontece, o autor utiliza o problema da vazão do Rio Mississippi. À primeira vista, estimar quanta água flui por um rio gigantesco parece uma tarefa impossível, que exigiria sensores de alta tecnologia e meses de estudo. No entanto, o "problema" aqui é a nossa tendência de complicar as coisas: Bentley demonstra que, ao ignorar a necessidade dessa estimativa simples, o engenheiro perde a chance de entender a escala do que está enfrentando. Se você não consegue estimar a ordem de magnitude de algo natural como um rio, provavelmente também não perceberá quando seu sistema de software estiver prestes a transbordar e causar um desastre técnico.

## 2 Regras e Técnicas de Estimativa utilizadas

Para fazer essas contas rapidamente, o livro sugere algumas ferramentas:

### 2.1 A Regra do 72

Se você quer saber quanto tempo algo leva para dobrar de tamanho (como o uso de disco ou usuários), divida 72 pela taxa de crescimento. Se cresce 10 % ao ano, dobra em cerca de 7 anos ( $72/10$ ).

### 2.2 $\pi$ segundos é um nanocéculo

Um ano tem aproximadamente segundos (cerca de 31,5 milhões de segundos). Isso ajuda a converter tempos de execução de programas para escalas humanas.

### 2.3 Análise Dimensional

Sempre verifique as unidades (segundos, milhas, bytes). Se você quer um resultado em "milhas por dia", sua conta deve terminar com essas unidades.

### 2.4 Duas respostas são melhores que uma

Tente calcular o mesmo problema de duas formas diferentes (ex: o fluxo do rio pela vazão na foz e pela chuva na bacia hidrográfica). Se os números forem próximos, você provavelmente está certo.

## 3 Aplicação Prática e Fatores de Segurança

Como podemos aplicar isso no dia a dia da programação?

### 3.1 Estimativas de Performance

A coluna demonstra como calcular se 2 milhões de registros cabem na memória de 128MB. Ele descobriu que, devido ao overhead (custo extra) de alocação de memória (malloc), cada registro ocupava muito mais que o esperado.

### 3.2 Fatores de Segurança (O exemplo da Ponte do Brooklyn)

Bentley cita John Roebling, que projetou a ponte seis vezes mais forte do que o necessário porque ele "sabia o que não sabia".

Lição: Em software, se sua estimativa diz que você precisa de 500ms, prometa 1 ou 2 segundos. Use "margens de erro" para compensar sua ignorância sobre detalhes técnicos ocultos.

### 3.3 Little Law (Lei de Little)

"O número médio de objetos em um sistema é o produto da taxa de entrada e o tempo médio de permanência".

(Ex: se entram 20 pessoas por hora num bar e elas ficam 3 horas, o bar terá sempre 60 pessoas).

## 4 Conclusão

Ao usar cálculos aproximados, lembre-se do famoso conselho de Einstein.

Tudo deve ser feito o mais simples possível, mas não mais simples do que isso.

Sabemos que cálculos simples não são tão simples assim, pois incluímos fatores de segurança para compensar nossos erros na estimativa de parâmetros e nossa ignorância sobre o problema em questão.

## 5 Referências

Bentley, Jon (2000). Programming Pearls. 2<sup>a</sup> ed. Addison-Wesley. isbn: 9780201657883.