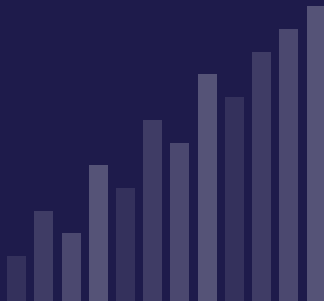

AULA 3 · BLOCO 1: FUNDAMENTOS, ORDENAÇÃO E CUSTO

Análise de algoritmos: o modelo RAM e os casos

Seção 2.2

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Descrever o modelo RAM e o que ele ignora
- ▶ Calcular $T(n)$ da inserção somando custo $\times$ vezes de cada linha
- ▶ Distinguir melhor caso, pior caso e caso médio
- ▶ Prever e medir o caso médio exato:
$$n(n-1)/4 + n - H_n$$



Teoria

Modelo RAM, custo por linha, melhor e pior caso, ordem de crescimento



Prática em C

Instrumentar cada linha da inserção e conferir a tabela do livro



Experimento

Média de 300 permutações e o histograma das comparações



Conexões

Tempo real, segurança, compiladores e serviços web

O modelo RAM

- ▶ Um processador, instruções executadas **uma após a outra**, sem concorrência.
- ▶ Operações elementares custam tempo **constante**: aritmética, comparação, cópia, desvio, acesso a uma posição da memória.
- ▶ Palavras de tamanho limitado ($c \lg n$ bits): não se esconde trabalho em inteiros gigantes.
- ▶ Ignora a hierarquia de memória (cache, memória virtual). Quando ela importa, medimos.

Tamanho da entrada depende do problema: número de itens (ordenação), número de bits (multiplicação), vértices e arestas (grafos).

Custo linha a linha da inserção

Linha	Custo	Vezes
1 para $j \leftarrow 2$ até n	c_1	n
2 $chave \leftarrow A[j]$	c_2	$n - 1$
4 $i \leftarrow j - 1$	c_4	$n - 1$
5 enquanto $i > 0$ e $A[i] > chave$	c_5	$\sum_{j=2}^n t_j$
6 $A[i + 1] \leftarrow A[i]$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $i \leftarrow i - 1$	c_7	$\sum_{j=2}^n (t_j - 1)$
8 $A[i + 1] \leftarrow chave$	c_8	$n - 1$

t_j = número de testes do **enquanto** para aquele j . $T(n)$ = soma de custo $\times$ vezes.

Melhor caso e pior caso

Melhor caso: vetor ordenado

$t_j = 1$ para todo j :

$$T(n) = (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) =$$

Função **linear** de n .

Pior caso: vetor invertido

$t_j = j$, e $\sum_{j=2}^n j = \frac{n(n+1)}{2} - 1$:

$$T(n) = an^2 + bn + c$$

Função **quadrática** de n .

Por que olhar o **pior caso**? É uma garantia; ocorre com frequência (buscar algo ausente); e o caso médio costuma ser tão ruim quanto ele.

Caso médio e ordem de crescimento

- ▶ Numa permutação aleatória, cada elemento é menor que metade dos anteriores, em média: $t_j \approx j/2$. Ainda **quadrático**.
- ▶ Mais precisamente: comparações = inversões + $(n-1) - (\text{mínimos à esquerda} - 1)$.
Como $E[\text{inversões}] = n(n-1)/4$ e $E[\text{mínimos}] = H_n$:

$$E[\text{comparações}] = \frac{n(n-1)}{4} + n - H_n, \quad H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n} \approx \ln n$$

- ▶ **Ordem de crescimento:** só o termo dominante importa, sem a constante: a inserção é $\Theta(n^2)$ no pior caso.

Um algoritmo é mais eficiente que outro se seu pior caso tem ordem de crescimento menor. Para n pequeno, as constantes podem inverter isso.



Prática em C

- ▶ Um contador para cada linha do pseudocódigo
- ▶ Conferir a coluna “vezes da tabela do livro
- ▶ Testes: ordenado, invertido e a entrada da Figura 2.2

Contando cada linha (1/3)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* vezes[k] = quantas vezes a linha k do pseudocódigo do livro executou (k = 1..8). */
5  long vezes[9];
6
7  void ordena_insercao(int A[], int n) {
8      int j;
9      for (j = 1; vezes[1]++, j < n; j++) {    // linha 1: o teste roda n vezes
10         int chave = A[j];          vezes[2]++;
11         int i = j - 1;              vezes[4]++;
12         while (vezes[5]++, i >= 0 && A[i] > chave) {    // linha 5: t_j testes
```

Contando cada linha (2/3)

```
13         A[i + 1] = A[i];           vezes[6]++;
14         i--;                       vezes[7]++;
15     }
16     A[i + 1] = chave;               vezes[8]++;
17 }
18 }
```

Contando cada linha (3/3)

```
20 int main(void) {
21     int n;
22     if (scanf("%d", &n) != 1 || n <= 0) return 1;
23     int *A = malloc(n * sizeof(int));
24     for (int k = 0; k < n; k++)
25         if (scanf("%d", &A[k]) != 1) return 1;
26     ordena_insercao(A, n);
27     const char *nome[9] = {"", "para", "chave = A[j]", "comentário",
28                             "i = j - 1",
29                             "teste do enquanto", "A[i+1] = A[i]", "i = i - 1",
30                             "A[i+1] = chave"};
31     for (int k = 1; k <= 8; k++)
32         if (k != 3) printf("linha %d (%s): %ld vezes\n", k, nome[k], vezes[k]);
33     free(A);
34     return 0;
35 }
```



Experimento

- ▶ Hipótese: média exata $n(n-1)/4 + n - H_n$
- ▶ 300 permutações uniformes (Fisher-Yates) por n
- ▶ Histograma de 100.000 execuções com $n = 100$

O caso médio, medido

Hipótese. A média de comparações da inserção numa permutação uniforme é

$$\frac{n(n-1)}{4} + n - H_n,$$

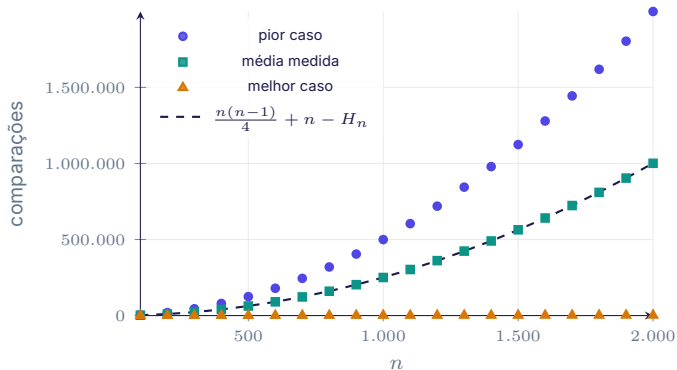
entre o melhor caso $(n-1)$ e o pior $(n(n-1)/2)$, praticamente na metade do pior. Além disso, as execuções se **concentram** em torno da média.

Método

1. Permutações uniformes com o embaralhamento de Fisher–Yates
2. n de 100 a 2.000; média de 300 permutações por n
3. Histograma: $n = 100, 100.000$ permutações, classes de largura 50
4. Comparação direta com a fórmula (sem ajuste de constante)

EXPERIMENTO

Média medida e fórmula exata



n	medido	razão
100,0	2.565	0,998
500,0	62.881	1,000
900,0	203.238	1,000
1.300	423.848	1,001
1.700	723.404	0,999
2.000	1.001.022	1,000

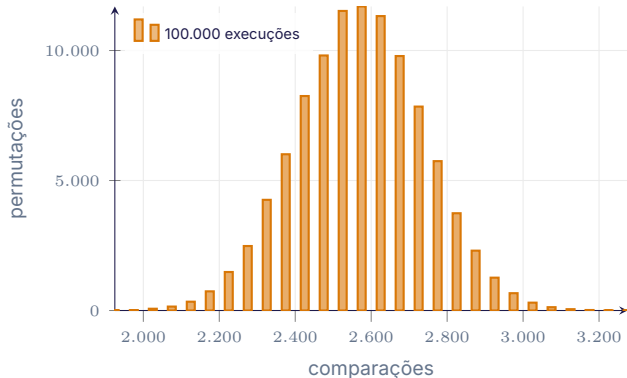
$$\text{razão} = \text{medido} \div \frac{n(n-1)}{4} + n - H_n$$

✓ desvio máx. 0,2%

A média fica a menos de 1% da fórmula: com 300 amostras, a média amostral já está muito perto da esperança.

🧪 EXPERIMENTO

Como as execuções se distribuem ($n = 100$)



Forma de sino em torno de $100 \cdot 99/4 + 100 - H_{100} \approx 2570$. O desvio padrão cresce como $n^{3/2}$, mais devagar que a média (n^2): em proporção, as execuções ficam cada vez mais parecidas.

🧪 EXPERIMENTO

O que o experimento mostrou

A fórmula do caso médio é confirmada: a média medida fica a menos de 1% de $n(n-1)/4 + n - H_n$, praticamente a metade do pior caso. O histograma mostra que execuções muito diferentes da média são raras. **Caso médio quadrático:** para dados aleatórios, a inserção não escapa de $\Theta(n^2)$.

➤ Reproduza: `gcc -O2 aula03_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Sistemas de tempo real
- ▶ Segurança
- ▶ Compiladores
- ▶ Serviços web

Onde isso aparece na computação

Sistemas de tempo real

Freio ABS e marca-passo não podem atrasar: a análise de **pior caso** de tempo de execução (WCET) garante que todo prazo seja cumprido.

Segurança

Ataques de complexidade enviam a entrada do pior caso de propósito: expressões regulares catastróficas (ReDoS) e colisões em tabelas hash derrubam servidores.

Compiladores

Otimizadores usam modelos de custo por instrução, como a tabela desta aula, para decidir desenrolar laços, embutir funções ou vetorizar.

Serviços web

Média engana: plataformas monitoram a **cauda** (latência p99, p99,9), porque com milhões de requisições o pior caso acontece a cada segundo.

Para casa

Exercícios

1. (2.2-1) Escreva $n^3/1000 - 100n^2 - 100n + 3$ em notação Θ .
2. (2.2-2) Ordenação por seleção: escreva o pseudocódigo, dê o invariante e o tempo de melhor e pior caso.
3. (2.2-3) Busca linear: quantos elementos são examinados, em média e no pior caso?
4. Use `aula03_custo_por_linha.c` para conferir: linha 5 executa $\sum t_j$ vezes; linhas 6 e 7, $\sum (t_j - 1)$.
5. Altere o experimento para $n = 400$ e compare a largura relativa do histograma com a de $n = 100$.

Leitura

Cormen et al., seção 2.2; para H_n , apêndice A.1.

Código

josedihego.net/estruturas