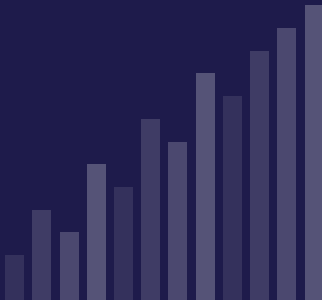

AULA 12 · BLOCO 2: DICIONÁRIOS E ESTRUTURAS LINEARES

Filas

Seção 10.1



Roteiro da aula

Ao final, você será capaz de

- ▶ Implementar filas em vetor circular e em lista ligada
- ▶ Distinguir fila cheia de fila vazia
- ▶ Conhecer deque (filas de duas pontas)
- ▶ Simular uma fila de atendimento e conferir com a teoria das filas



Teoria

FIFO, cabeça e cauda, o vetor circular, a fila ligada e o deque



Prática em C

Fila circular e fila ligada em C



Experimento

Simulação M/M/1: espera média contra $\rho/(1 - \rho)$



Conexões

Sistemas operacionais, redes, simulação e computação em nuvem

Primeiro a entrar, primeiro a sair

- ▶ Sai sempre o elemento há mais tempo na fila: política FIFO.
- ▶ $Q.cabeça$: o próximo a sair.
 $Q.cauda$: a próxima posição livre.
- ▶ Os índices andam "em círculo": depois de n vem 1 (em C: $(i + 1) \% n$).
- ▶ Vazia: $cabeça = cauda$. Cheia: $cabeça = cauda + 1$. Com n posições, cabem $n - 1$ elementos (ou guarde a contagem).

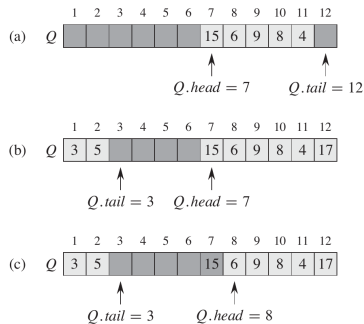


Figura 10.2 de Cormen et al. (2009) — (a) fila com 5 elementos em $Q[7..11]$; (b) após Enqueue de 17, 3 e 5: a cauda dá a volta; (c) após Dequeue.

As operações

- ▶ Ambas $O(1)$; o livro omite os testes de cheia e vazia (exercício 10.1-4).
- ▶ **Fila ligada:** entra na cauda, sai da cabeça; com ponteiros para as duas pontas, $O(1)$ sem limite de tamanho.
- ▶ **Deque** (exercício 10.1-5): inserir e remover nas duas pontas, também $O(1)$ num vetor circular.

Enqueue(Q, x) e Dequeue(Q)

```

1                                     // Enqueue( $Q, x$ )
2  $Q[Q.cauda] \leftarrow x$ 
3 se  $Q.cauda = Q.comprimento$  então
4    $Q.cauda \leftarrow 1$ 
5 senão
6    $Q.cauda \leftarrow Q.cauda + 1$ 
7                                     // Dequeue( $Q$ )
8  $x \leftarrow Q[Q.cabeça]$ 
9 se  $Q.cabeça = Q.comprimento$  então
10   $Q.cabeça \leftarrow 1$ 
11 senão
12   $Q.cabeça \leftarrow Q.cabeça + 1$ 
13 retorne  $x$ 

```

Filas no mundo: teoria das filas

- ▶ Clientes chegam ao acaso (taxa λ) e são atendidos um por vez (taxa μ), em ordem de chegada: a fila **M/M/1**.
- ▶ Utilização $\rho = \lambda/\mu < 1$; se $\rho \geq 1$, a fila cresce sem limite.
- ▶ Resultado clássico: o tempo médio de espera **na fila** é

$$W_q = \frac{\rho}{\mu - \lambda} = \frac{\rho}{1 - \rho} \quad (\text{com } \mu = 1).$$

- ▶ Não linear: com $\rho = 0,5$, espera-se 1 atendimento; com $\rho = 0,9$, 9.

Um simulador de eventos discretos é exatamente uma fila FIFO de clientes mais um relógio: é o experimento desta aula.



Prática em C

- ▶ Fila circular de 4 posições com contagem
- ▶ Fila ligada com ponteiros para cabeça e cauda
- ▶ Comandos e x (enfileira) e d (desenfileira)

Filas em C (1/4)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* Fila circular (Figura 10.2): cabeca é o próximo a sair, cauda a próxima posição
5     livre; os índices dão a volta com %. Guardamos n para separar cheia de vazia. */
6  #define TAM 4
7  typedef struct { int v[TAM], cabeca, cauda, n; } FilaCircular;
8
9  int fc_enfileira(FilaCircular *Q, int x) {           // ENQUEUE
10     // cheia: na lista ligada isso não ocorre
11     if (Q->n == TAM) return 0;
12     Q->v[Q->cauda] = x;
13     Q->cauda = (Q->cauda + 1) % TAM;
14     Q->n++;
15     return 1;
16 }
```

Filas em C (2/4)

```
17 int fc_desenfileira(FilaCircular *Q) {                // DEQUEUE (supõe não vazia)
18     int x = Q->v[Q->cabeca];
19     Q->cabeca = (Q->cabeca + 1) % TAM;
20     Q->n--;
21     return x;
22 }
23
24 /* Fila em lista simplesmente ligada: entra na cauda, sai da cabeça, ambos O(1). */
25 typedef struct No { int x; struct No *prox; } No;
26 typedef struct { No *cabeca, *cauda; } FilaLista;
27
28 void fl_enfileira(FilaLista *Q, int x) {
29     No *n = malloc(sizeof(No));
30     n->x = x;
31     n->prox = NULL;
```

Filas em C (3/4)

```
32     if (Q->cauda) Q->cauda->prox = n; else Q->cabeca = n;    // fila vazia: caso especial
33     Q->cauda = n;
34 }
35 int fl_desenfileira(FilaLista *Q) {
36     No *n = Q->cabeca;
37     int x = n->x;
38     Q->cabeca = n->prox;
39     if (!Q->cabeca) Q->cauda = NULL;
40     free(n);
41     return x;
42 }
43
44 int main(void) {
45     FilaCircular C = {{0}, 0, 0, 0};
46     FilaLista L = {NULL, NULL};
```

Filas em C (4/4)

```
47     char op;
48     int x;
49     while (scanf(" %c", &op) == 1) {
50         if (op == 'e' && scanf("%d", &x) == 1) {
51             if (!fc_enfileira(&C, x)) printf("fila circular cheia\n");
52             fl_enfileira(&L, x);
53         } else if (op == 'd') {
54             if (!L.cabeca) { printf("fila vazia\n"); continue; }
55             int a = fl_desenfileira(&L);
56             if (C.n) fc_desenfileira(&C);
57             printf("saiu %d\n", a);
58         }
59     }
60     while (L.cabeca) fl_desenfileira(&L);
61     return 0;
62 }
```



Experimento

- ▶ Chegadas e atendimentos exponenciais; atendimento médio = 1
- ▶ Fila circular dos instantes de chegada
- ▶ 2 milhões de clientes para cada ρ de 0,05 a 0,85

Uma fila de banco, simulada

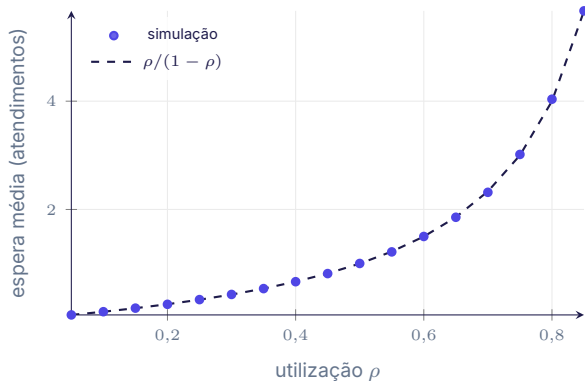
Hipótese. Com chegadas de taxa ρ e atendimentos de taxa 1, a teoria das filas prevê espera média na fila de $\rho/(1 - \rho)$. A espera deve explodir perto de $\rho = 1$: dobrar a ocupação de 40% para 80% multiplica a espera por 6.

Método

1. Simulação de eventos: próximo evento é uma chegada ou um fim de atendimento
2. Chegadas guardadas numa fila FIFO circular
3. Espera = início do atendimento – instante de chegada
4. $\rho = 0,05, 0,10, \dots, 0,85$

🧪 EXPERIMENTO

Espera média na fila



ρ	medido	razão
0,050	0,052	0,996
0,200	0,248	0,993
0,350	0,538	1,000
0,500	1,003	1,003
0,650	1,856	0,999
0,850	5,668	1,000

razão = medido $\div$ $\rho/(1 - \rho)$

✓ desvio máx. 0,9%

Com $\rho = 0,85$, a espera média é de 5,7 atendimentos. A curva é a mesma de filas de banco, de CPUs e de roteadores.

EXPERIMENTO

O que o experimento mostrou

A simulação reproduz $\rho/(1 - \rho)$ com erro de poucos por cento. A fila FIFO, uma estrutura simples, é o coração de toda simulação de atendimento. A lição prática: um sistema com 85% de ocupação já tem esperas longas; perto de 100%, elas explodem.

>_ Reproduza: `gcc -O2 aula12_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Sistemas operacionais
- ▶ Redes
- ▶ Computação em nuvem
- ▶ Algoritmos de grafos

Onde isso aparece na computação

 Sistemas operacionais

Processos prontos esperam numa fila; o escalonamento *round robin* é uma fila circular. Buffers de teclado, disco e impressão são filas.

 Redes

Roteadores guardam pacotes em filas de saída; a teoria das filas desta aula explica o atraso e a perda quando o enlace fica congestionado.

 Computação em nuvem

Filas de mensagens (RabbitMQ, Kafka, SQS) desacoplam serviços: quem produz enfileira, quem consome desenfileira no seu ritmo.

 Algoritmos de grafos

A busca em largura (aula 28) explora um grafo camada por camada usando exatamente uma fila FIFO.

Para casa

Exercícios

1. (10.1-3) Ilustre Enqueue(4), Enqueue(1), Enqueue(3), Dequeue, Enqueue(8), Dequeue em $Q[1..6]$.
2. (10.1-4) Reescreva Enqueue e Dequeue detectando esvaziamento e transbordamento.
3. (10.1-5) Implemente um deque com as quatro operações em $O(1)$ num vetor circular.
4. (10.1-6) Implemente uma fila com duas pilhas. Qual o custo amortizado por operação?
5. Altere a simulação para dois atendentes (M/M/2) e compare a espera com a de um atendente duas vezes mais rápido.

Leitura

Cormen et al., seção 10.1.

Código

josedihego.net/estruturas