SCC0202 – Algoritmos e Estruturas de Dados I

Filas e Deques

Prof.: Dr. Rudinei Goularte

(rudinei@icmc.usp.br)

Instituto de Ciências Matemáticas e de Computação - ICMC

Sala 4-229

Conteúdo

- Filas
- Deques

Fila

- O que é?
- Para que serve?

Fila (Queue)

- O que é?
  - *É uma estrutura para armazenar um conjunto de elementos, que funciona da seguinte forma*
    - Novos elementos sempre entram no fim da fila
    - O único elemento que se pode retirar da fila em um dado momento é seu primeiro elemento
- Para que serve?
  - Modelar situações em que é preciso armazenar um conjunto ordenado de elementos, no qual o primeiro elemento a entrar no conjunto será também o primeiro elemento a sair do conjunto, e assim por diante
    - Política FIFO: first in, first out

Aplicações

- Exemplos de aplicações de filas
  - Filas de espera e algoritmos de simulação
  - Controle por parte do sistema operacional de recursos compartilhados, tais como impressoras
  - Buffers de Entrada/Saída
  - Estrutura de dados auxiliar em alguns algoritmos como a busca em largura

TAD Fila

- Operações principais
  - fila_criar(): cria uma fila F vazia
  - fila_inserir(F,x): insere o elemento x no final da fila F. Retorna **true** se foi possível inserir, **false** caso contrário
  - fila_remover(F): remove o elemento no início de F, e retorna esse elemento. Retorna NULL se não foi possível remover

TAD Fila

- Operações auxiliares
  - fila_frente(F): retorna o elemento no início de F, sem remover
  - fila_tamanho(F): retorna o número de elementos em F
  - fila_vazia(F): indica se a fila F está vazia
  - fila_cheia(F): indica se a fila F está cheia (útil para implementações estáticas)

TAD Fila - exemplo

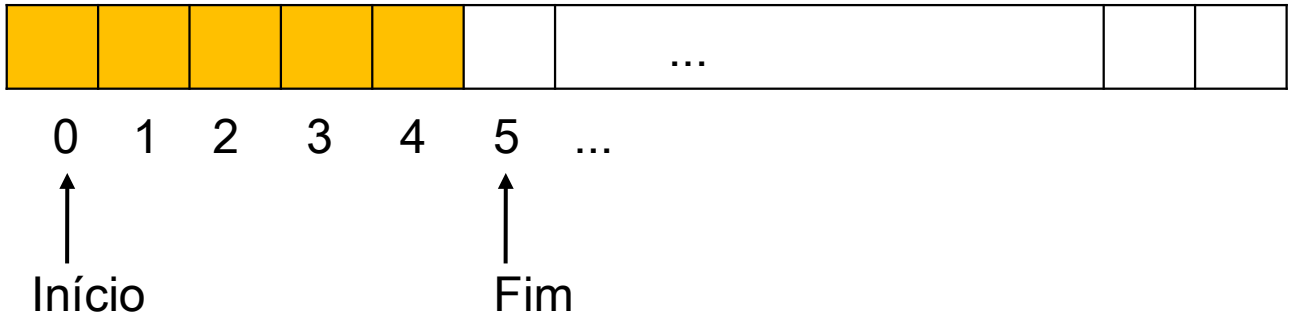
Operação	Fila	resultado
fila_criar()	1º da fila →	
fila_inserir(F,a)	1º da fila → a	
fila_inserir(F,b)	1º da fila → a, b	
fila_inserir(F,c)	1º da fila → a, b, c	
fila_remover(F)	1º da fila → b, c	return (a)
fila_inserir(F,d)	1º da fila → b, c, d	
fila_remover(F)	1º da fila → c, d	return (b)

Implementação

- Alocação sequencial
  - Os elementos da fila ficam, necessariamente, em sequência (um ao lado do outro) na memória
  - Todo o espaço reservado permanece reservado durante todo o tempo de execução do programa, independentemente de estar sendo efetivamente usado ou não

Implementação de Fila

- **Início:** aponta para/indica o primeiro da fila, ou seja, o primeiro elemento a sair
- **Fim:** aponta para/indica o fim da fila, ou seja, onde o próximo elemento entrará



Implementação de Fila

- Qual a condição inicial, quando a fila é criada?
- Qual a condição para fila vazia?
- Qual a condição para fila cheia?

Exemplo de uso

- Array com 6 posições
- fila_criar()
  - Início = 0, Fim = 0
- fila_inserir(F, a), fila_inserir(F, b), fila_inserir(F,c)
  - Qual o estado de F, Início e Fim?
- fila_inserir(F,z), fila_inserir(F,r), fila_inserir(F,s)
  - fila_cheia(F) == TRUE
- fila_remover(F), fila_remover(F)
  - Qual o problema?

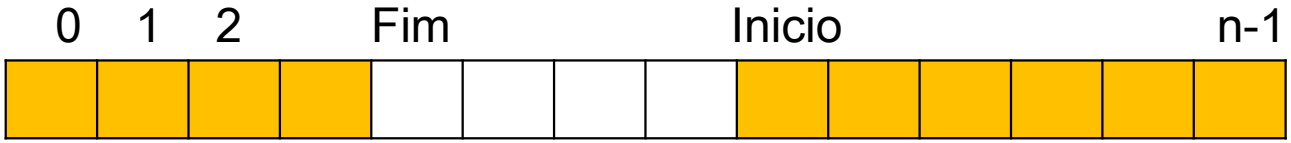
Implementação Sequencial

- As inserções fazem com que o contador Fim seja incrementado (O(1))

- Mas as remoções requerem deslocar todos os elementos (O(n))
- É possível melhorar isso?

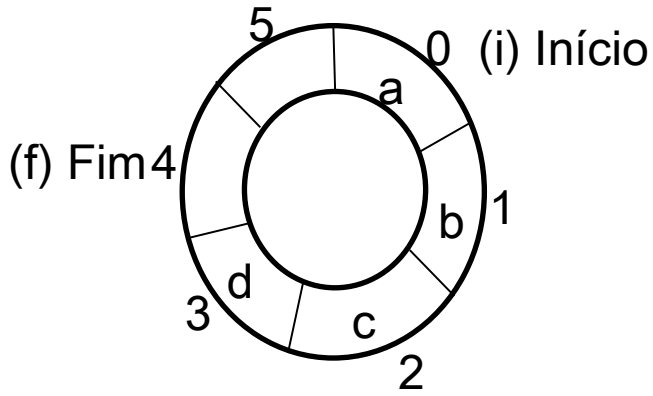
Implementação Sequencial Circular

- A solução é fazer com que o Início não seja fixo na primeira posição do vetor
  - Pode-se permitir que Fim volte para o início do vetor quando esse contador atingir o final do vetor
- Essa implementação é conhecida como fila circular
- Na figura abaixo a fila circular possui 4 posições vagas e Fim < Inicio



Implementação Sequencial Circular

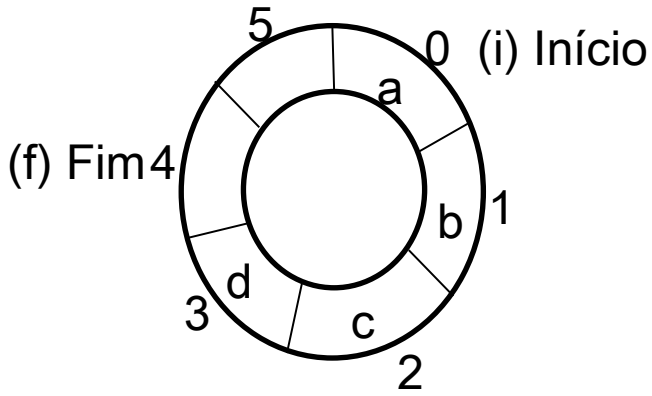
- Portanto, deve-se “ver” a fila como um “anel” – Fila Circular



  - Qual a condição inicial (quando a fila é criada)?
  - Qual a condição para fila vazia?
  - Qual a condição para fila cheia?

Implementação Sequencial Circular

- **Solução**: campo extra para guardar número de elementos



Total = 4

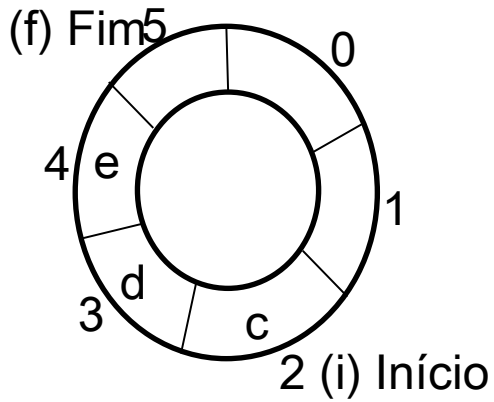
  - Qual a condição inicial (quando a fila é criada)?
  - Qual a condição para fila vazia?
  - Qual a condição para fila cheia?

Implementação Sequencial Circular

- Qual a condição inicial (quando a fila é criada)?
  - Total=0, Inicio=0, Fim=0
- Qual a condição para fila vazia?
  - Total=0
- Qual a condição para fila cheia?
  - Total=tamanho da fila

Implementação Sequencial Circular

- Exemplo
  - Inicio = 2, Fim = 5, Total = 3
  - fila_inserir(f), fila_inserir(g), fila_inserir(h)



Implementação Sequencial Circular na Heap

- A fila é sequencial, como no caso de alocação estática. Contudo, está alocada na Heap.
  - Facilidade de modelagem e implementação como TAD.
  - Chamaremos daqui em diante apenas de Fila Sequencial Circular

```
0 (fila.h)
1 ...
10
11 #include "item.h"
12 #include <stdbool.h>
13 #define TAM_MAX 100
14
15 typedef struct fila_ FILA;
16
17 FILA *fila_criar(void);
18 bool fila_inserir(FILA *fila, ITEM *item);
19 ITEM *fila_remover(FILA *fila);
20 int fila_tamanho(FILA *fila);
...
```

```
0 (fila.c)
1...
10 #include "fila.h"
11
12
13 struct fila_{
14        ITEM *fila[TAM_MAX];
15        int inicio;  /*posicao do 1o elemento da fila*/
16        int fim;     /*posicao do ultimo elemento da fila*/
17        int tamanho;
18 };
19 ...
```

Implementação Sequencial Circular

```
1 /*Cria logicamente uma fila, inicialmente vazia*/
2 FILA *fila_criar(void){
3     /*pré-condição: existir espaço na memória.
4     Na implementação estática não há o que verificar*/
5     FILA *fila = (FILA *) malloc(sizeof(FILA));
6     fila->inicio = 0;
7     fila->fim = 0;
8     fila->tamanho = 0; /* fila vazia*/
9
10     return (fila);
11}
12
13 bool fila_cheia(FILA *fila) {
14   return (fila->tamanho == TAM);
15 }
16
17 bool fila_vazia(FILA *fila) {
18   return (fila->tamanho == 0);
19 }
```

Implementação Sequencial Circular

```
1 /*Insere um elemento no fim da fila.
2  Fila Circular.
3 */
4 bool fila_inserir(FILA *fila, ITEM *item){
5
6    if (fila != NULL && (!fila_cheia(fila)) ){
7       fila->fila[fila->fim] = item;
8       fila->fim = (fila->fim+1) % TAM_MAX;
9       fila->tamanho ++;
10
11        return(true);
12    }
13
14    return(false);
15}
```

Implementação Sequencial Circular

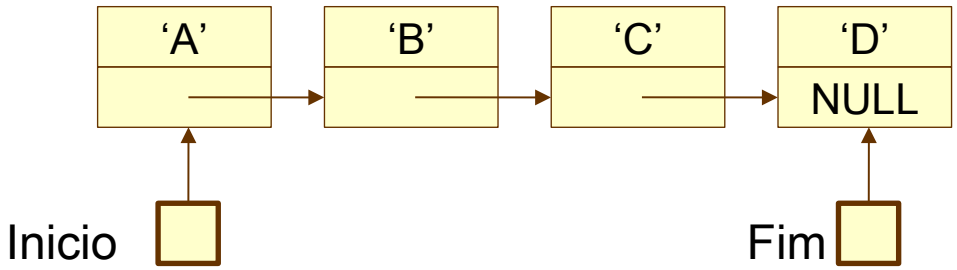
```
1 ITEM *fila_remover(FILA *fila){
2     if (fila != NULL && (!fila_vazia(fila)) ) {
3        ITEM *i = fila->fila[fila->inicio];
4        fila->fila[fila->inicio] = NULL;
5        fila->inicio = (fila->inicio + 1) % TAM_MAX;
6        fila->tamanho --;
7        return (i);
8    }
9    return (NULL);
10}
```

Implementação Sequencial Circular

```
1 int fila_tamanho(FILA *fila) {
2   if (fila != NULL)
3      return (fila->tamanho);
4   return (ERRO);
5 }
```

Implementação Encadeada

- A implementação encadeada* pode ser realizada mantendo-se dois ponteiros, um para o início e outro para o final da fila
- Com isso pode-se ter acesso direto às posições de inserção e remoção
  - * Supõe-se aqui encadeamento com alocação dinâmica



Implementação Encadeada

- As operações **inserir** e **remover** implementadas dinamicamente são bastante eficientes
- Deve-se tomar cuidado apenas para que os ponteiros para início e final tenham valor NULL quando a fila estiver vazia

Sequencial versus Encadeada

Operação	Sequencial	Encadeada
Criar	O(1)	O(1)
Apagar	O(1)*	O(n)
Inserir	O(1)	O(1)
Remover	O(1) (circular)	O(1)
Frente	O(1)	O(1)
Vazia	O(1)	O(1)
Cheia	O(1)	O(1)
Tamanho	O(1)	O(1) (c/ contador)

- * Se a Fila for implementada como um array na *stack.* Caso seja implementada como um array na *heap* (como temos feito), será O(n).

Sequencial versus Encadeada

- Sequencial
  - Implementação simples
  - Tamanho da fila definido a priori
- Encadeada
  - Alocação dinâmica permite gerenciar melhor estruturas cujo tamanho não é conhecido a priori ou que variam muito de tamanho

Deques

- *Double Ended QUEue*
- Deques são estruturas que permitem inserir e remover de ambos os extremos

Aplicações

- Alguns Sistemas de escalonamento de processos com multiplas CPU's
- Verificadores de palindromos (sequências de itens que são iguais quando lidos da esquerda para direita ou da direita para esquerda)
- Opção desfazer/refazer em programas de edição de imagem, texto etc.

TAD Deques

- Operações principais
  - inserir_inicio(D,x): insere o elemento x no início da deque D. Retorna **true** se foi possível inserir e **false** caso contrário
  - inserir_fim(D,x): insere o elemento x no final da deque D. Retorna **true** se foi possível inserir e **false** caso contrário
  - remover_inicio(D): remove o elemento no inicio de D, e retorna esse elemento. Retorna NULL se não foi possível remover
  - remover_fim(D): remove o elemento no final de D, e retorna esse elemento. Retorna NULL se não foi possível remover

TAD Deques

- Operações auxiliares
  - primeiro(D): retorna o elemento no início de D. Retorna NULL se o elemento não existe
  - ultimo(D): retorna o elemento no final de D. Retorna NULL se o elemento não existe
  - contar(D): retorna o número de elementos em D
  - vazia(D): indica se a deque D está vazia
  - cheia(D): indica se a deque D está cheia (útil para implementações estáticas)

TAD Deques

- Como deques requerem inserir e remover elementos em ambos os extremos
  - Implementação sequencial circular
  - Implementação dinâmica duplamente encadeada
- Nesses casos, as operações do TAD são O(1)

Implementação Sequencial Circular

- Exercício: Implementar uma DEQUE
  - Aproveite a implementação de uma fila circular estática e acrescente as duas operações que faltam:
    - remover do fim; e
    - inserir no início
  - As outras operações são as mesmas

Implementação - Inserir no Início

```
1 boolean deque_inserir_inicio(DEQUE *deque, ITEM *item) {
2    if (deque != NULL && !deque_cheia(deque)) {
3       deque->inicio = (deque->inicio - 1 + TAM) % TAM;
4       deque->itens[deque->inicio] = item;
5       deque->tamanho ++;
6       return (TRUE);
7    }
8    return (FALSE);
9}
```

- Se x ≥ 0, então (x + N) % N = x
- Se x = -1, então (x + N) % N = N – 1
  - x é a posição que se deseja inserir: deque->inicio - 1

Implementação - Remover do Fim

```
1 ITEM *deque_remover_fim(DEQUE *deque) {
2    if (deque != NULL && !deque_vazia(deque)) {
3       deque->fim = (deque->fim - 1 + TAM) % TAM;
4       ITEM* i = deque->itens[deque->fim];
5       deque->itens[deque->fim] = NULL;
6       deque->tamanho --;
7       return (i);
8    }
9    return (NULL);
10}
```

- Se x ≥ 0, então (x + N) % N = x
- Se x = -1, então (x + N) % N = N - 1

Exercícios

- Implemente uma fila dinâmica
- Implemente uma deque dinâmica
- Implemente um procedimento recursivo capaz de esvaziar uma fila
- Implemente um procedimento para inverter uma fila (o primeiro elemento se tornará o último e vice-versa)

Referências

- Material baseado nos originais produzidos pelos professores:
  - Gustavo E. de A. P. A. Batista
  - Fernando V. Paulovich
  - Maria das Graças Volpe Nunes