Estruturas de Dados - Fila

Tipo abstrato de dados: fila

- Conjunto que permite inclusão e exclusão de elementos com as seguintes propriedades:
  - inclusão de um elemento: o elemento é adicionado ao final da fila
  - exclusão de um elemento: o elemento excluído é o que está no começo da fila (o que está na fila a mais tempo)
- *first in, first out*
- Exemplos de aplicações:
  - fila de impressão
  - digitação no teclado
  - SGBD (sistema de gerenciamento de banco de dados)

Fila sequencial

- Estrutura de dados utilizada: vetor
  - saber a quantidade de elementos na fila não é o bastante para trabalharmos eficientemente
  - temos de saber onde começa e onde acaba a fila
  - obs: para sermos eficientes, a fila pode "dar uma volta" no vetor, ou seja, começar no fim do vetor e acabar no começo
- Informações utilizadas:
  - *tam*: número máximo de elementos
  - *card*: número de elementos na fila
  - *inicio*: índice para o início da fila
  - *fim*: índice para o fim da fila
- Operações a serem analisadas:
  - CriaFila criação de uma fila sequencial
  - Front: consulta o elemento no começo da fila sequencial
  - Enfileira: inclusão de um elemento em uma fila sequencial
  - Desenfileira: exclusão de um elemento em uma fila sequencial

Fila sequencial: criar

Algoritmo: CriarFilaSequencial(n)

Entrada: tamanho n da fila

Saída: fila sequencial F

1 criar nova fila sequencial F com um vetor de n posições $F.V[]$
2 $F.tam = n$
3 $F.card = 0$
4 $F.inicio = 0$
5 $F.fim = -1$
6 **retorne** F

Complexidade: $O(1)$

Fila sequencial: checar a frente da fila

Algoritmo: FrontSequencial(F)

Entrada: fila sequencial F

Saída: elemento na frente da fila

```
1 se F.card == 0 então
2 |   retorne "Erro: fila vazia!"
3 retorne F.V[F.inicio]
```

Complexidade: $O(1)$

Fila sequencial: enfileirar

Algoritmo: EnfileiraSequencial(F, x)

Entrada: fila sequencial F, valor x

```
1 se F.card == F.tam então
2 |   retorne "Erro: fila cheia!"
3 F.fim = (F.fim + 1) mod n
4 F.V[F.fim] = x
5 F.card = F.card + 1
```

Complexidade: $O(1)$

Fila sequencial: desenfileirar

Algoritmo: DesenfileiraSequencial(F)

Entrada: fila sequencial F

Saída: valor excluído, ou um erro caso a fila esteja vazia

```
1 se F.card == 0 então
2 |   retorne "Erro: fila vazia!"
3 x = F.V[F.inicio]
4 F.inicio = (F.inicio + 1) mod n
5 F.card = F.card − 1
6 retorne x
```

Complexidade: $O(1)$

Fila encadeada

- Estrutura de dados utilizada: nós encadeados
- Para acessar a fila, basta conhecermos o primeiro nó da fila
  - apesar disso, saber qual é o último nó pode ser útil
- Informações de um nó:
  - *chave*: guarda o elemento
  - *prox*: indica a localização do nó que o sucede na fila
- Operações a serem analisadas:
  - CriaFila: criação de uma fila encadeada
  - Front: consulta o elemento no começo da fila encadeada
  - Enfileira: inclusão de um elemento em uma fila encadeada
  - Desenfileira: exclusão de um elemento em uma fila encadeada

Fila encadeada: criar

Algoritmo: CriarFilaEncadeada()

Saída: nó inicial v da fila encadeada

1 criar novo nó v
2 $v \rightarrow prox = \lambda$
3 **retorne** v

Complexidade: $O(1)$

Fila encadeada: checar a frente da fila

Algoritmo: FrontEncadeada(v)

Saída: nó na frente da fila, ou λ se a fila estiver vazia

1 **retorne** $v \rightarrow prox$

Complexidade: $O(1)$

Fila encadeada: enfileirar (sem último)

Algoritmo: EnfileiraEncadeada(v, x)

Entrada: nó inicial v, valor x

1 criar novo nó u
2 $u \rightarrow chave = x$
3 $u \rightarrow prox = \lambda$
4 **enquanto** $v \rightarrow prox \neq \lambda$ **faça**
5 $\quad v = v \rightarrow prox$
6 $v \rightarrow prox = u$

Complexidade: $O(n)$

Fila encadeada: enfileirar (com último)

Algoritmo: EnfileiraEncadeada(w, x)

Entrada: nó final w (último nó da fila), valor x

1 criar novo nó u

2 $u \rightarrow chave = x$

3 $u \rightarrow prox = \lambda$

4 $w \rightarrow prox = u$

Complexidade: $O(1)$

Fila encadeada: excluir

Algoritmo: DesenfileiraEncadeada(v)

Entrada: nó inicial v

Saída: nó removido, ou λ se a fila estiver vazia

```
1 se v→prox == λ então
2 |   retorne λ
3 r = v→prox
4 v→prox = r→prox
5 retorne r
```

Complexidade: $O(1)$

Fila: aplicações

Algoritmo: QuickFila(V, p, q)

Entrada: vetor V, índices p e r

Saída: vetor V ordenado

```
1 criar nova fila F vazia      //considere que a fila tem campos a e b
    (representando os índices inicial e final)
2 Enfileira(F, p, r)      //a = p e b = r
3 enquanto F não estiver vazia faça
4 |   x = Desenfileira(F)
5 |   se x.a < x.b então
6 |   |   q = Partition(V, a, b)      //se x.a < x.b, temos mais de um
  |   |     elemento, então devemos ordenar!
7 |   |   Enfileira(F, a, q − 1)
8 |   |   Enfileira(F, q + 1, b)
```