

# Filas de prioridade

## Conceitos e implementação

Prof. Marcelo de Souza

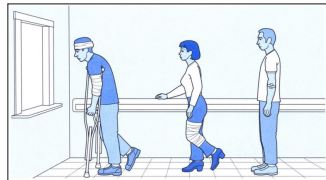
45RPE – Resolução de Problemas com Estruturas de Dados  
Universidade do Estado de Santa Catarina

# Filas de prioridade

## Ideia geral

**Fila de prioridade:** cada elemento tem uma prioridade, que determina a ordem de remoção.

- ▶ O elemento prioritário é o próximo a ser removido.
- ▶ **Atendimento médico:** a gravidade do paciente define sua prioridade de atendimento.



# Filas de prioridade

## Ideia geral

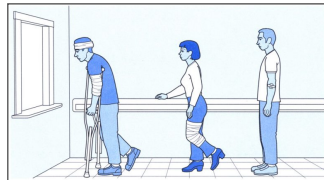


**Fila de prioridade:** cada elemento tem uma prioridade, que determina a ordem de remoção.

- ▶ O elemento prioritário é o próximo a ser removido.
- ▶ **Atendimento médico:** a gravidade do paciente define sua prioridade de atendimento.

### Operações:

- ▶ `put`: insere um elemento na fila.
- ▶ `min`: retorna o elemento prioritário da fila.
- ▶ `get`: remove (e retorna) o elemento prioritário da fila.



# Filas de prioridade

## Ideia geral



**Fila de prioridade:** cada elemento tem uma prioridade, que determina a ordem de remoção.

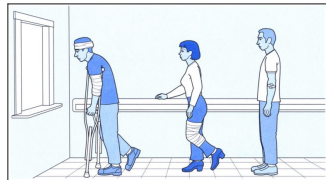
- ▶ O elemento prioritário é o próximo a ser removido.
- ▶ **Atendimento médico:** a gravidade do paciente define sua prioridade de atendimento.

### Operações:

- ▶ `put`: insere um elemento na fila.
- ▶ `min`: retorna o elemento prioritário da fila.
- ▶ `get`: remove (e retorna) o elemento prioritário da fila.

### Aplicações:

- ▶ Busca em grafos (e.g. algoritmo de Dijkstra).
- ▶ Otimização combinatória (e.g. bin packing).
- ▶ Inteligência artificial (e.g. algoritmo A\*).



# Filas de prioridade

## Ideia geral



### Funcionamento (em Python)

- ▶ A prioridade é definida do **menor** valor para o maior (i.e. o menor tem prioridade).
- ▶ Para armazenar tipos naturalmente comparáveis, basta inserí-los na estrutura.
  - ▶ e.g. números (inteiros, reais), string, ...



# Filas de prioridade

## Ideia geral

### Funcionamento (em Python)

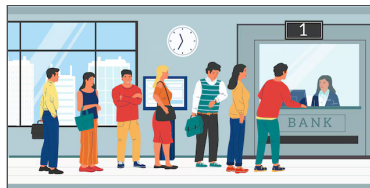
- ▶ A prioridade é definida do **menor** valor para o maior (i.e. o menor tem prioridade).
- ▶ Para armazenar tipos naturalmente comparáveis, basta inserí-los na estrutura.
  - ▶ e.g. números (inteiros, reais), string, ...
- ▶ Para armazenar objetos de outros tipos:
  1. Armazenar uma tupla contento o valor de prioridade e o objeto.
  2. e.g. (7, Livro("Holly", "Stephen King", 2023)).

# Filas de prioridade

## Ideia geral

### Funcionamento (em Python)

- ▶ A prioridade é definida do **menor** valor para o maior (i.e. o menor tem prioridade).
- ▶ Para armazenar tipos naturalmente comparáveis, basta inserí-los na estrutura.
  - ▶ e.g. números (inteiros, reais), string, ...
- ▶ Para armazenar objetos de outros tipos:
  1. Armazenar uma tupla contento o valor de prioridade e o objeto.
  2. e.g. (7, Livro("Holly", "Stephen King", 2023)).
- ▶ Exemplo: **fila de um banco**.
  - ▶ Prioridade definida por vários atributos (idoso, gestante, cliente especial, tempo de chegada, ...).





# Filas de prioridade

## Exemplo de funcionamento

Seja uma fila de prioridade implementada usando uma lista sequencial, inicialmente vazia.

- Armazenaremos veículos, cuja prioridade é definida pelo ano de fabricação.

Método	Retorno	Conteúdo (não ordenado)
<code>put((Fusca, 67))</code>	—	{ (Fusca, 67) }
<code>put((Uno, 95))</code>	—	{ (Fusca, 67), (Uno, 95) }
<code>put((Kombi, 60))</code>	—	{ (Fusca, 67), (Uno, 95), (Kombi, 60) }
<code>min()</code>	(Kombi, 60)	{ (Fusca, 67), (Uno, 95), (Kombi, 60) }
<code>get()</code>	(Kombi, 60)	{ (Fusca, 67), (Uno, 95) }
<code>put((Corcel, 74))</code>	—	{ (Fusca, 67), (Uno, 95), (Corcel, 74) }
<code>get()</code>	(Fusca, 67)	{ (Uno, 95), (Corcel, 74) }
<code>get()</code>	(Corcel, 74)	{ (Uno, 95) }
<code>get()</code>	(Uno, 95)	{ }
<code>get()</code>	None	{ }
<code>is_empty()</code>	True	{ }





# Filas de prioridade

## Exemplo de funcionamento

Seja uma fila de prioridade implementada usando uma lista sequencial, inicialmente vazia.

- Armazenaremos veículos, cuja prioridade é definida pelo ano de fabricação.

Método	Retorno	Conteúdo (ordenado)
<code>put((Fusca, 67))</code>	—	{ (Fusca, 67) }
<code>put((Uno, 95))</code>	—	{ (Uno, 95), (Fusca, 67) }
<code>put((Kombi, 60))</code>	—	{ (Uno, 95), (Fusca, 67), (Kombi, 60) }
<code>min()</code>	(Kombi, 60)	{ (Uno, 95), (Fusca, 67), (Kombi, 60) }
<code>get()</code>	(Kombi, 60)	{ (Uno, 95), (Fusca, 67) }
<code>put((Corcel, 74))</code>	—	{ (Uno, 95), (Corcel, 74), (Fusca, 67) }
<code>get()</code>	(Fusca, 67)	{ (Uno, 95), (Corcel, 74) }
<code>get()</code>	(Corcel, 74)	{ (Uno, 95) }
<code>get()</code>	(Uno, 95)	{ }
<code>get()</code>	None	{ }
<code>is_empty()</code>	True	{ }

# Filas de prioridade

## Análise de complexidade



Podemos implementar uma fila de prioridade usando **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.



# Filas de prioridade

## Análise de complexidade

Podemos implementar uma fila de prioridade usando **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.

Operação	Não ordenado		Ordenado	
	Arranjo	Lista encadeada	Arranjo	Lista encadeada
put	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
min	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
get	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

— não estamos considerando o tempo gasto com *resize* no arranjo.

# Filas de prioridade

## Análise de complexidade

Podemos implementar uma fila de prioridade usando **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.

Operação	Não ordenado		Ordenado		Heap
	Arranjo	Lista encadeada	Arranjo	Lista encadeada	
put	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
min	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
get	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$

— não estamos considerando o tempo gasto com *resize* no arranjo.

Uma **heap** é um tipo de **árvore binária** usado para implementar filas de prioridade.

- ▶ É a forma como uma fila de prioridade é geralmente implementada.
- ▶ Estudaremos esse tipo de estrutura mais adiante!

45RPE – Resolução de Problemas com Estruturas de Dados  
Prof. Marcelo de Souza