



# Estrutura de Dados I

## Fila de prioridade

Bruno Prado

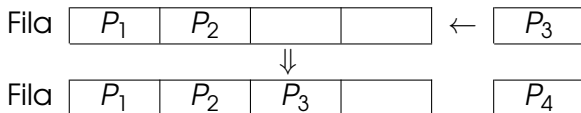
Departamento de Computação / UFS

# Introdução

- ▶ O que é uma fila?
  - ▶ É uma estrutura de dados First-In First-Out (FIFO)
  - ▶ Duas operações principais: enfileirar e desenfileirar
  - ▶ A restrição imposta é que o primeiro elemento inserido é o primeiro a ser removido

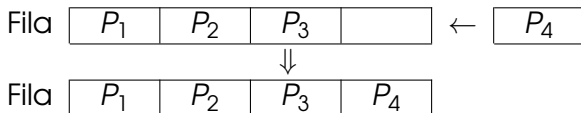
# Introdução

- ▶ Pensando em pessoas
  - ▶ Enfileirar (push\_back)



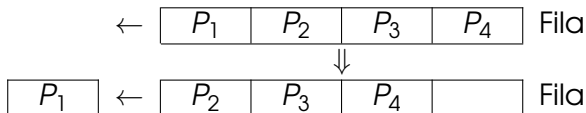
# Introdução

- ▶ Pensando em pessoas
  - ▶ Enfileirar (push\_back)



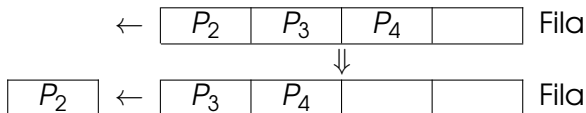
# Introdução

- ▶ Pensando em pessoas
  - ▶ Desenfileirando (pop\_front)



# Introdução

- ▶ Pensando em pessoas
  - ▶ Desenfileirando (pop\_front)

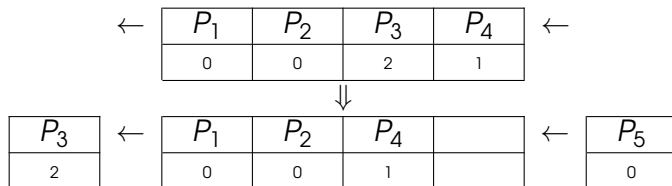


# Introdução

- ▶ O que é uma fila de prioridade?
  - ▶ É uma estrutura de dados First-In First-Out (FIFO) com níveis de priorização para os elementos
  - ▶ As operações de enfileiramento e desenfileiramento consideram a ordem de inserção e o nível de prioridade de cada elemento
  - ▶ Na situação em que mais de um elemento possuir a mesma prioridade e exista o requisito de estabilidade, será considerada a ordem de inserção do elemento

# Introdução

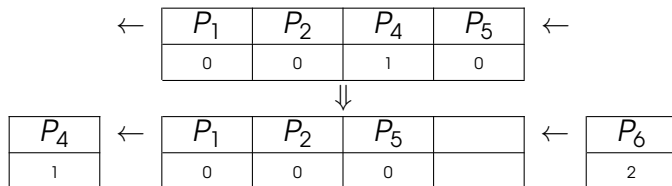
- ▶ Níveis de prioridade para pessoas
  - ▶ Prioridade 2: especiais
  - ▶ Prioridade 1: deficientes, idosos e gestantes
  - ▶ Prioridade 0: regulares





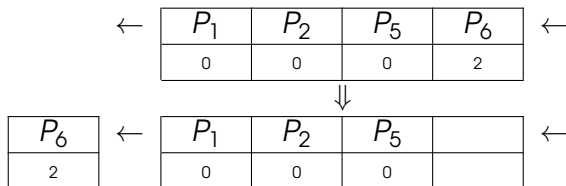
# Introdução

- ▶ Níveis de prioridade para pessoas
  - ▶ Prioridade 2: especiais
  - ▶ Prioridade 1: deficientes, idosos e gestantes
  - ▶ Prioridade 0: regulares



# Introdução

- ▶ Níveis de prioridade para pessoas
  - ▶ Prioridade 2: especiais
  - ▶ Prioridade 1: deficientes, idosos e gestantes
  - ▶ Prioridade 0: regulares



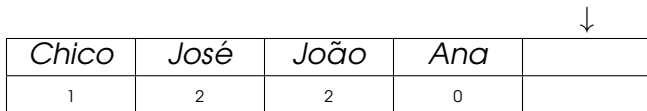
# Fila de Prioridade

- ▶ Implementação em C
  - ▶ Cada pessoa possui um nível de prioridade
  - ▶ É utilizada uma cadeia de caracteres para o nome

```
// Elemento da fila de prioridade
typedef struct pessoa {
    // Prioridade
    int prioridade;
    // Nome
    char* nome;
} pessoa;
```

# Fila de Prioridade

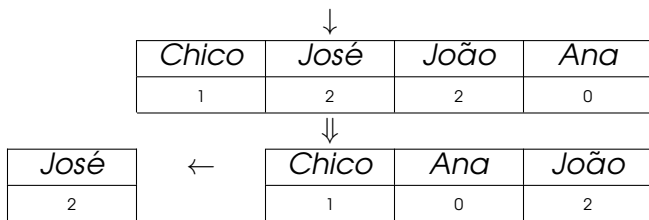
- ▶ Implementação sem ordenação
  - ▶ Os elementos são inseridos no final da estrutura
  - ▶ A operação de enfileiramento custa  $O(1)$



<i>Chico</i>	<i>José</i>	<i>João</i>	<i>Ana</i>	
1	2	2	0	

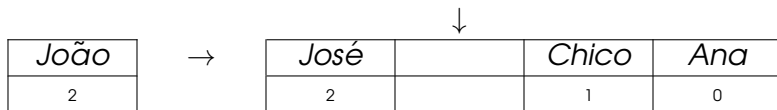
# Fila de Prioridade

- Implementação sem ordenação
  - Para desenfileirar é preciso realizar uma busca sequencial pelo elemento de maior prioridade
  - O elemento removido é substituído pelo último
  - A operação de desenfileiramento custa  $O(n)$



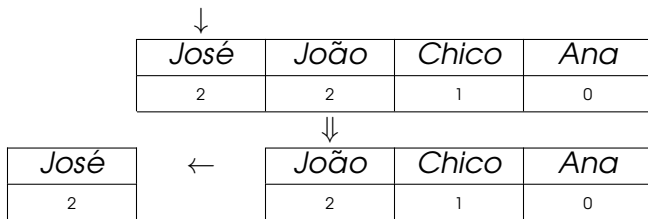
# Fila de Prioridade

- ▶ Implementação com ordenação
  - ▶ Os elementos são inseridos de forma ordenada
  - ▶ A operação de enfileiramento custa  $O(n)$



# Fila de Prioridade

- Implementação com ordenação
  - Para desenfileirar é preciso acessar o primeiro elemento da fila que possui maior prioridade
  - A operação de desenfileiramento custa  $O(1)$



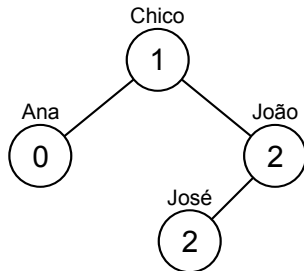
# Fila de Prioridade

- ▶ Análise de complexidade
  - ▶ Considerando  $m$  operações de desenfileamento em uma fila de prioridade com  $n$  elementos
  - ▶ Espaço  $\Theta(n)$
  - ▶ Tempo  $O(m \times n)$



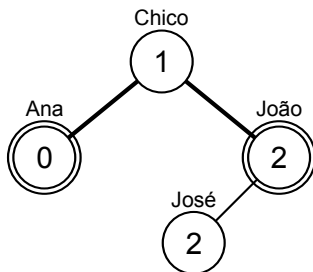
# Fila de Prioridade

- Implementação com árvore binária balanceada
  - É utilizada uma representação explícita de árvore
  - A operação de enfileiramento custa  $O(\log_2 n)$



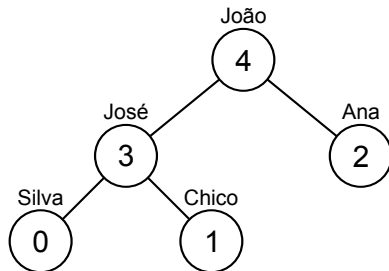
# Fila de Prioridade

- Implementação com árvore binária balanceada
  - Para desenfileirar é necessário preciso acessar o elemento de maior prioridade, realizando o percurso para o nó mínimo ou máximo da árvore
  - A operação de desenfileiramento custa  $O(\log_2 n)$



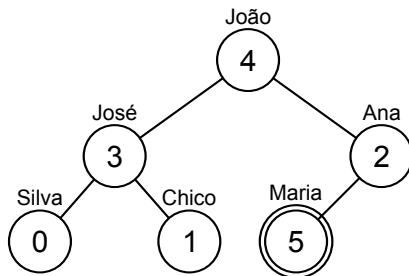
# Fila de Prioridade

- Implementação com heap
  - É uma representação implícita de árvore
  - A operação de enfileiramento custa  $O(\log_2 n)$



# Fila de Prioridade

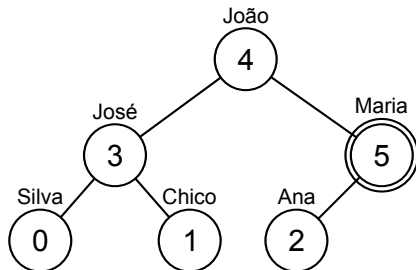
- Implementação com heap
  - A inserção do elemento é feita no final do vetor, sendo aplicada a operação de heapify
  - O procedimento é repetido até a raiz da árvore



# Fila de Prioridade

## ► Implementação com heap

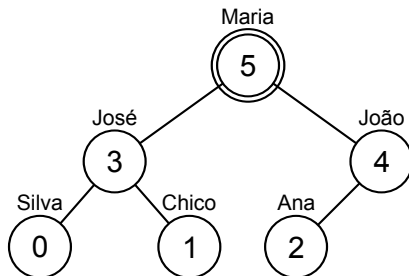
- A inserção do elemento é feita no final do vetor, sendo aplicada a operação de heapify
- O procedimento é repetido até a raiz da árvore



# Fila de Prioridade

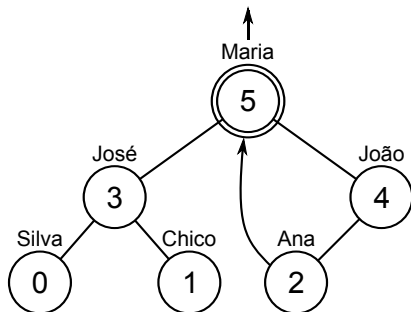
## ► Implementação com heap

- A inserção do elemento é feita no final do vetor, sendo aplicada a operação de heapify
- O procedimento é repetido até a raiz da árvore



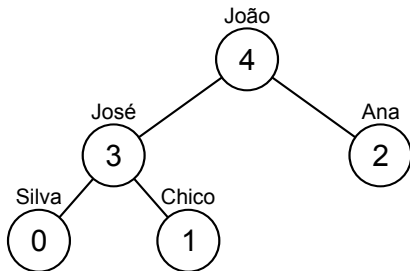
# Fila de Prioridade

- Implementação com heap
  - Para desenfileirar é preciso acessar o primeiro elemento do vetor que possui maior prioridade, sendo feita a substituição da raiz pelo último elemento
  - A operação de desenfileiramento custa  $O(\log_2 n)$



# Fila de Prioridade

- Implementação com heap
  - Após a remoção é aplicado o procedimento heapify na raiz da árvore
  - A operação de desenfileamento custa  $O(\log_2 n)$





# Fila de Prioridade

- ▶ Análise de complexidade
  - ▶ Considerando  $m$  operações de desenfileamento em uma fila de prioridade com  $n$  elementos
  - ▶ Espaço  $\Theta(n)$
  - ▶ Tempo  $O(m \times \log_2 n)$

# Fila de Prioridade

## ► Quadro comparativo

Implementação	Enfileirar	Desenfileirar	Total
Sem ordenação	$O(1)$	$O(n)$	$O(n^2)$
Com ordenação	$O(n)$	$O(1)$	$O(n^2)$
Árvore balanceada	$O(\log_2 n)$	$O(\log_2 n)$	$O(n \log_2 n)$
Heap	$O(\log_2 n)$	$O(\log_2 n)$	$O(n \log_2 n)$

# Exemplo

- ▶ Para o conjunto de elementos listados abaixo, construa filas de prioridade mínima e máxima
  - ▶ Utilize a árvore binária balanceada ou heap
  - ▶ Verifique como tornar a implementação estável

<i>Chico</i>	<i>José</i>	<i>João</i>	<i>Ana</i>	<i>Maria</i>
2	2	3	0	1

# Fila de Prioridade

## ► Aplicações

- Escalonamento de processos de um SO que suporta diferentes níveis de prioridade
- Gerenciamento de banda de rede, com priorização de protocolos de tempo real (QoS)
- Compressão de dados (Codificação Huffman)
- Busca em grafos (Dijkstra)
- ...

## Exercício

- ▶ A empresa de tecnologia Poxim Tech está desenvolvendo um sistema de controle de senhas para atendimento de serviços públicos, como para emissão de certidões, documentos de identificação e carteira de trabalho
  - ▶ São informados os órgãos disponíveis no centro de atendimento e a quantidade de atendentes
  - ▶ Na obtenção da senha de atendimento, o cidadão deve informar para qual órgão deseja ser atendido, seu nome completo e sua idade para verificação de prioridade, sendo registrado o horário de chegada
  - ▶ Existem dois tipos de atendimento: convencional (idade  $< 60$  anos) e preferencial (idade  $\geq 60$  anos), com menor e maior prioridade, respectivamente
  - ▶ O tempo de atendimento é padronizado, sendo realizados no mesmo intervalo de tempo
  - ▶ Os nomes utilizados para os órgãos e pessoas possuem até 50 caracteres compostos por letras

# Exercício

- ▶ Formato de arquivo de entrada
  - ▶ [*Quantidade de órgãos*]
  - ▶ [*Órgão*<sub>0</sub>] [*#Atendentes*]
  - ▶ ⋮
  - ▶ [*Órgão*<sub>*n*-1</sub>] [*#Atendentes*]
  - ▶ [*Quantidade de pessoas*]
  - ▶ [*Órgão*<sub>0</sub>] : [*Nome*<sub>0</sub>] – [*Idade*<sub>0</sub>] – [*Horário*<sub>0</sub>]
  - ▶ ⋮
  - ▶ [*Órgão*<sub>*m*-1</sub>] : [*Nome*<sub>*m*-1</sub>] – [*Idade*<sub>*m*-1</sub>] – [*Horário*<sub>*m*-1</sub>]

```
2
DETRAN 2
SSP 1
5
SSP:Joao da Silva-33-08:13:55
SSP:Jose dos Santos-22-08:14:43
DETRAN:Josefa Souza-35-07:30:44
DETRAN:Ana Maria-22-07:01:23
DETRAN:Chico Jose-77-13:34:33
```

# Exercício

- ▶ Formato de arquivo de saída
  - ▶ Fluxo de chamada de senhas para cada órgão

```
[DETRAN] Chico Jose, Ana Maria  
[SSP] Joao da Silva  
[DETRAN] Josefa Souza  
[SSP] Jose dos Santos
```