Fila de prioridade

Estruturas de Dados

Bruno Prado

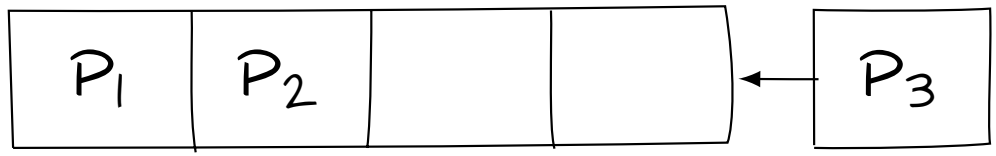
Departamento de Computação / UFS

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

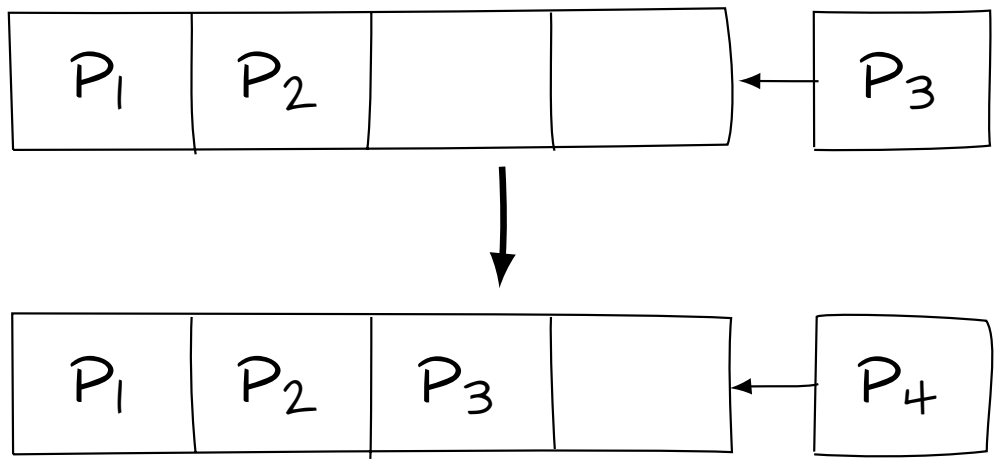
Introdução

- Pensando em pessoas
  - Enfileirar (*push_back*)



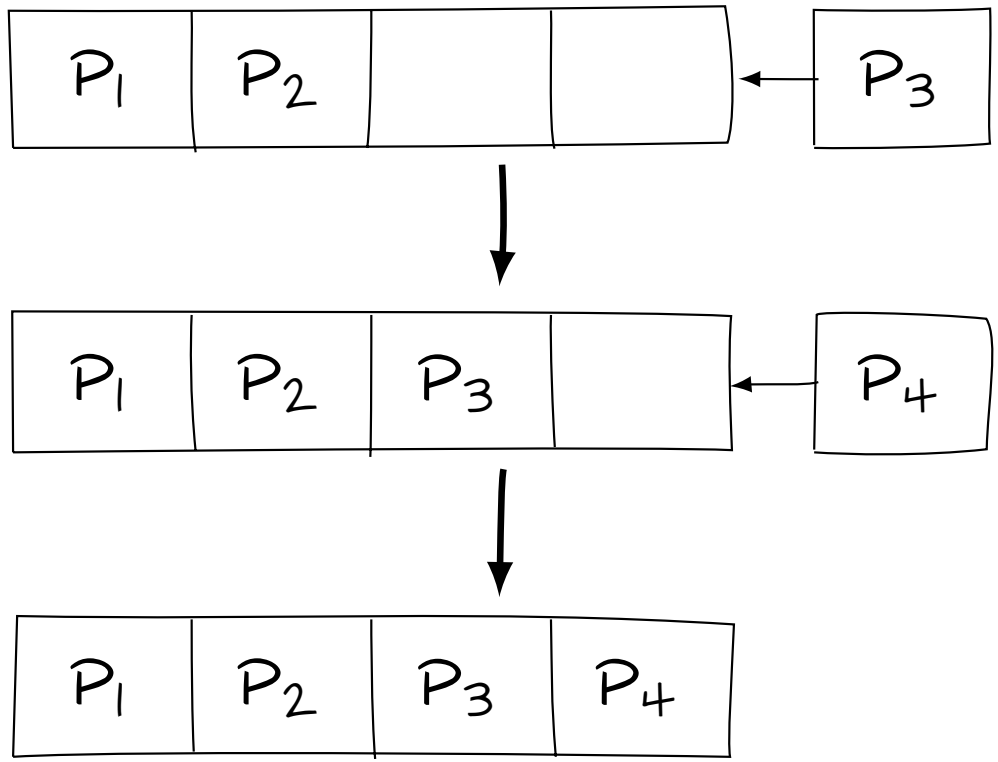
Introdução

- Pensando em pessoas
  - Enfileirar (*push_back*)



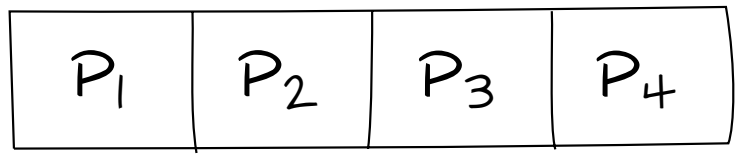
Introdução

- Pensando em pessoas
  - Enfileirar (*push_back*)



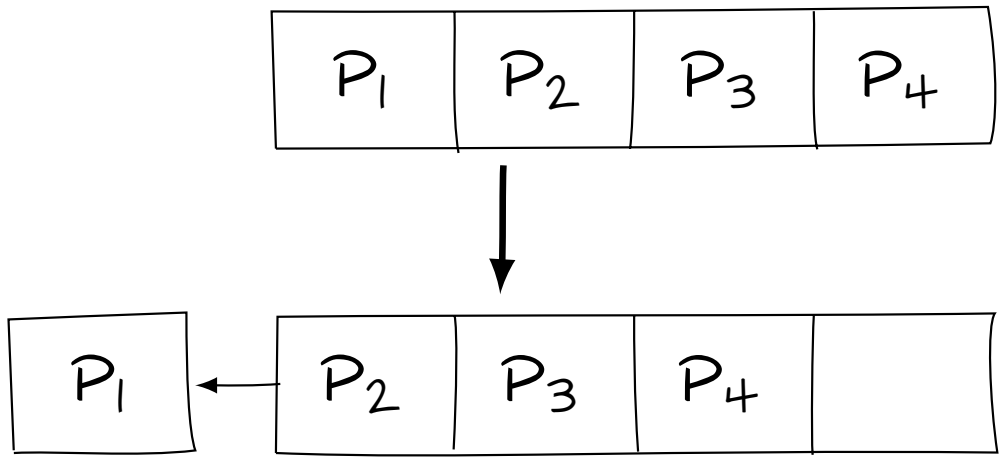
Introdução

- Pensando em pessoas
  - Desenfileirando (*pop_front*)



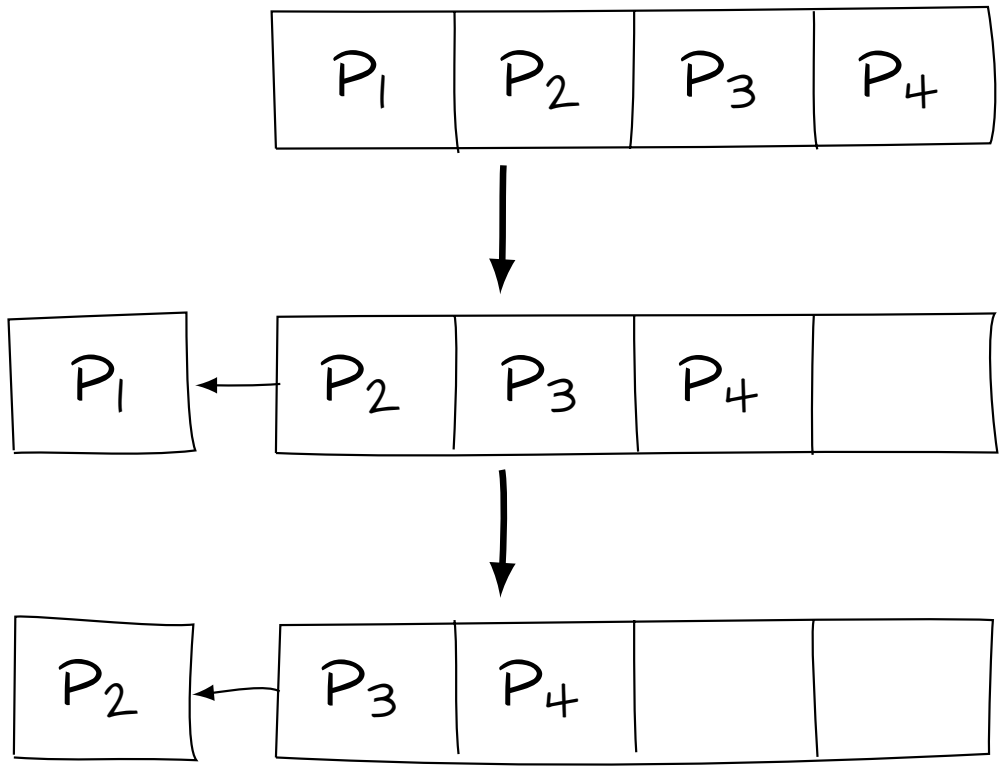
Introdução

- Pensando em pessoas
  - Desenfileirando (*pop_front*)



Introdução

- Pensando em pessoas
  - Desenfileirando (*pop_front*)

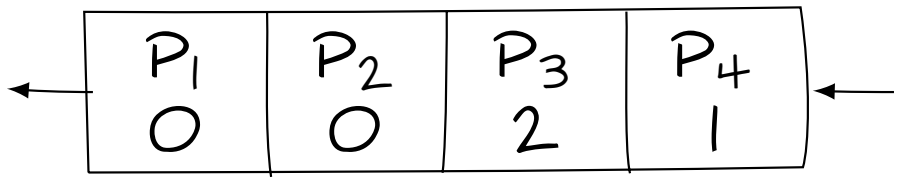


Introdução

- 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

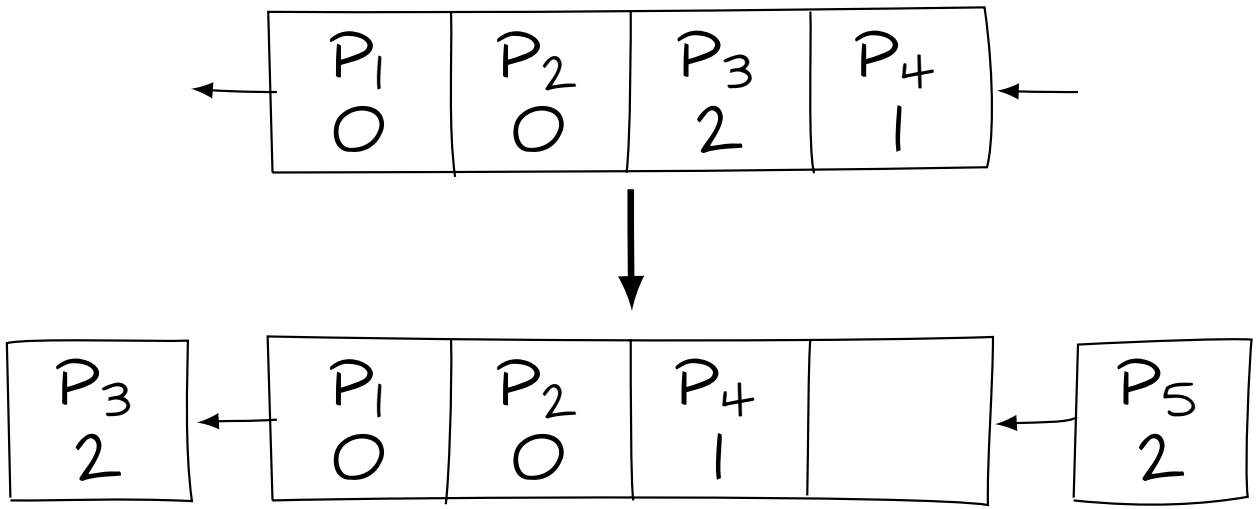
Introdução

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



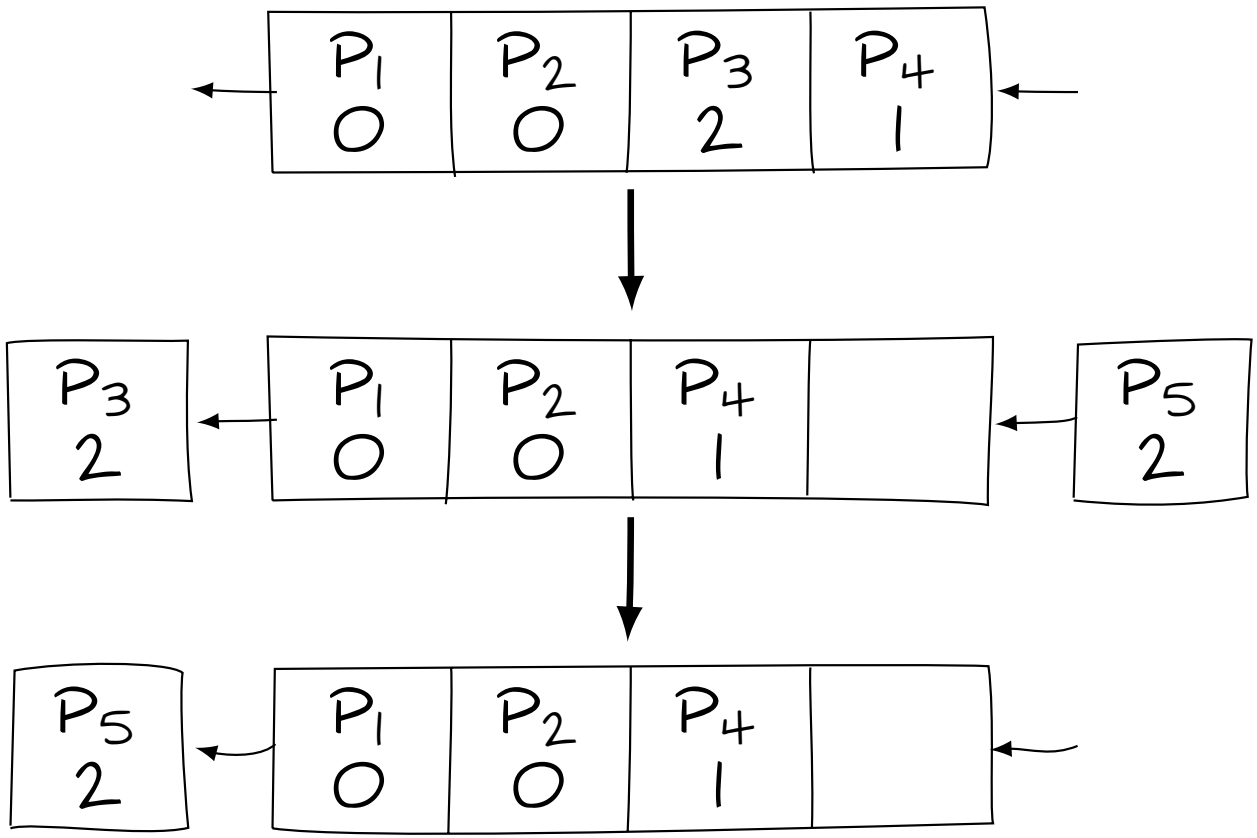
Introdução

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



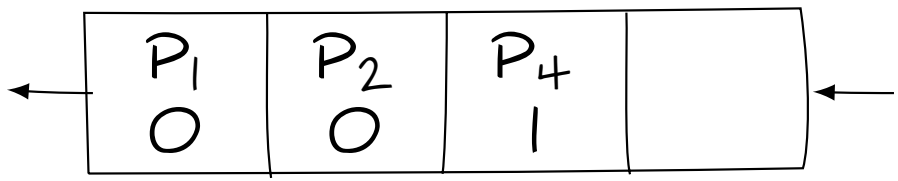
Introdução

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



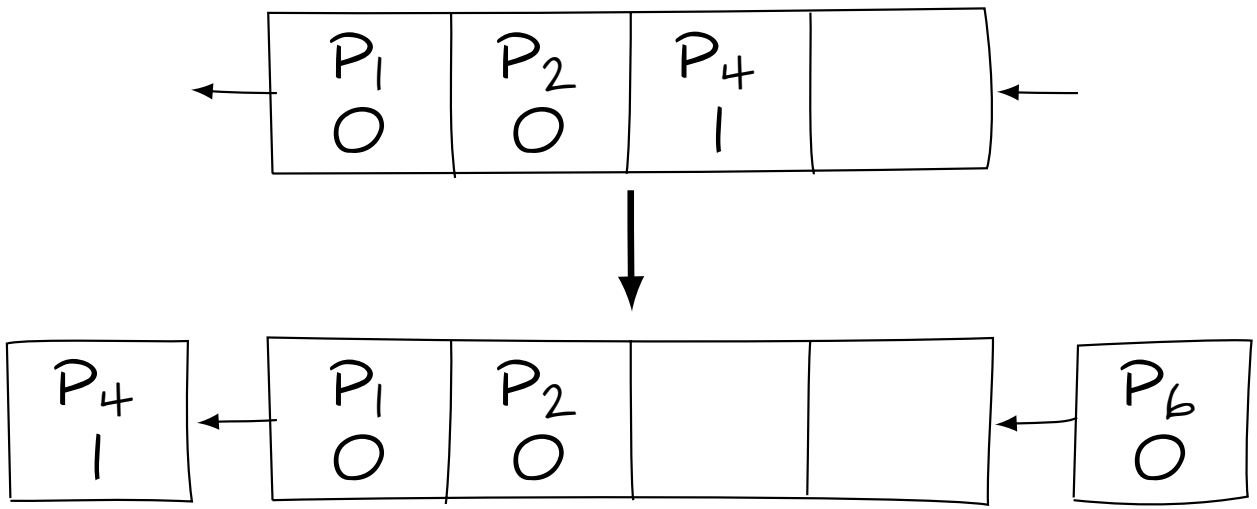
Introdução

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



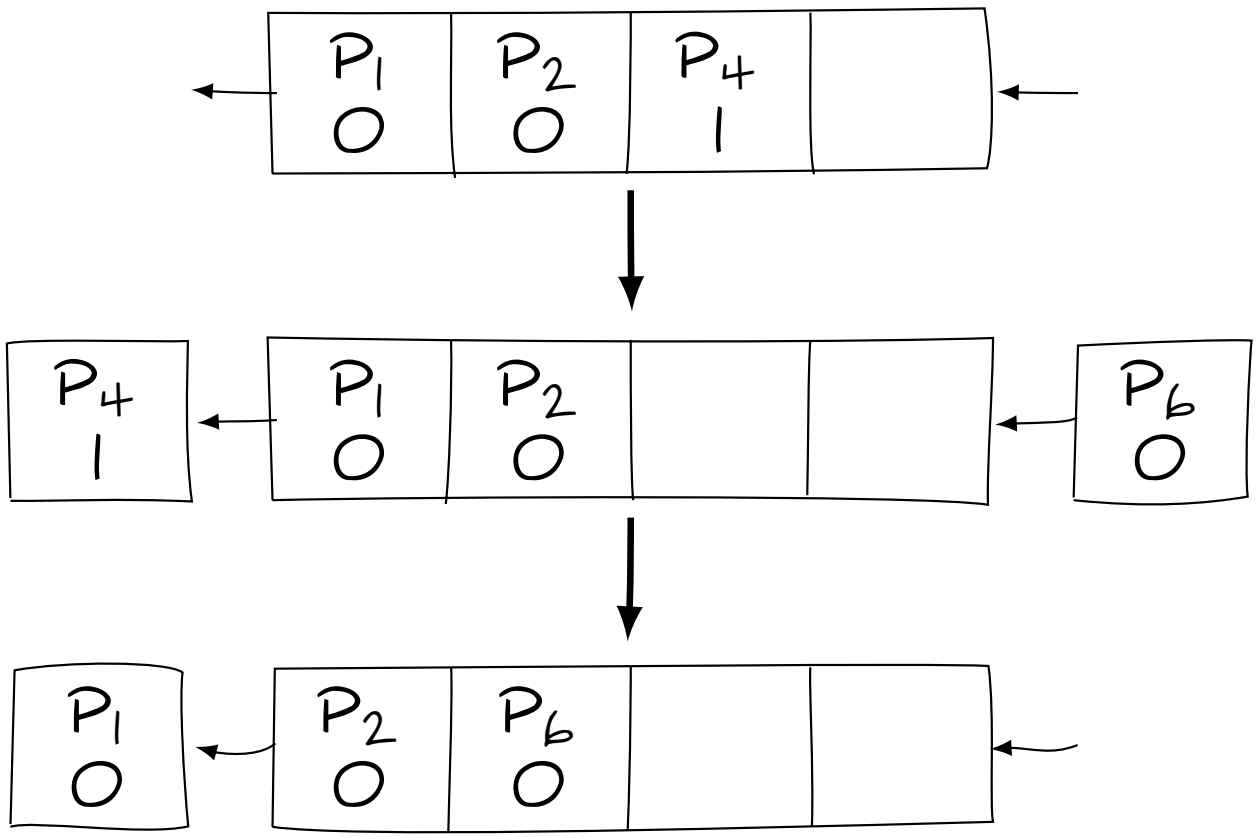
Introdução

- 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



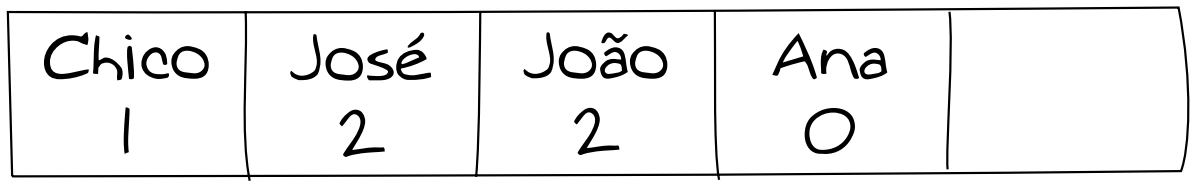
Fila de prioridade

- Implementação em C
  - Cada pessoa possui um nome e um nível de prioridade

```
1 // Padrão de tipos por tamanho
2 #include <stdint .h>
3 // Estrutura de pessoa
4 typedef struct pessoa {
5     // Nome
6     char* nome;
7     // Prioridade
8     uint32_t prioridade;
9 } pessoa;
```

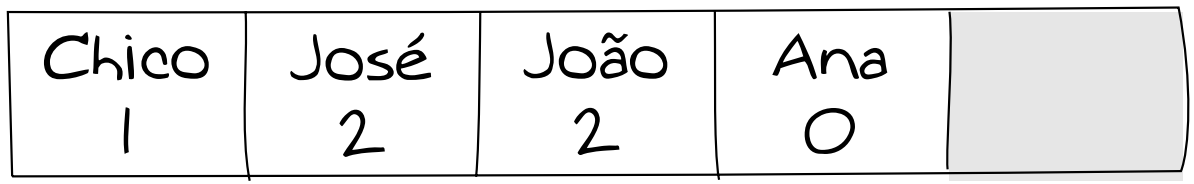
Fila de prioridade

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



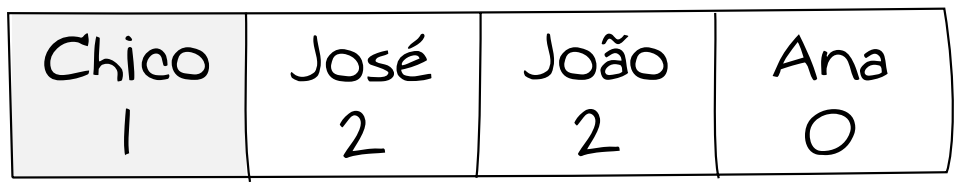
Fila de prioridade

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



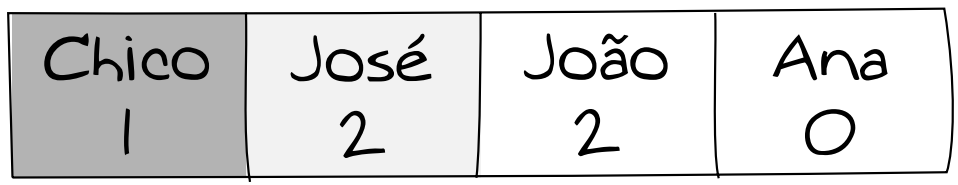
Fila de prioridade

- 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 $O(n)$



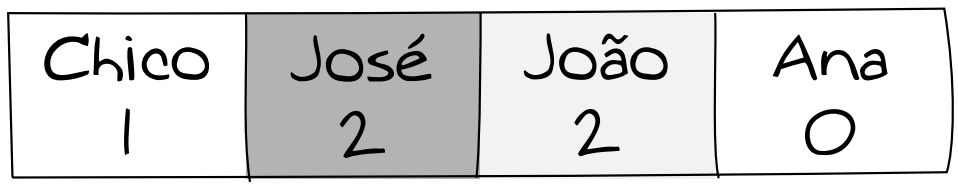
Fila de prioridade

- Implementação 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 $O(n)$



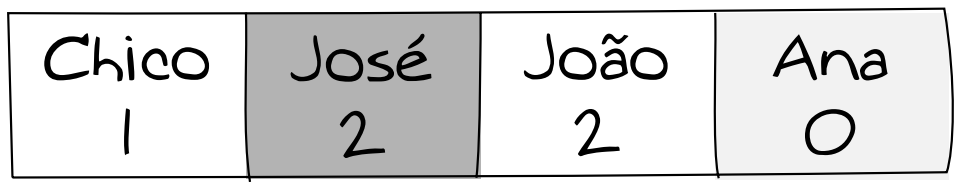
Fila de prioridade

- Implementação 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 $O(n)$



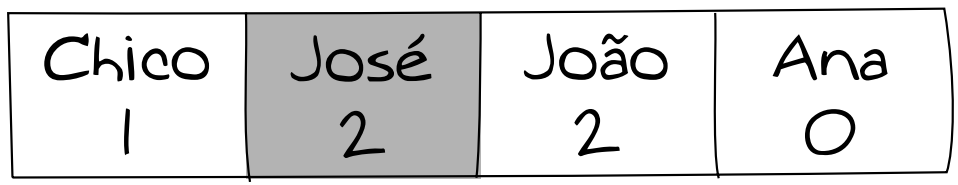
Fila de prioridade

- Implementação 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 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 $O(n)$

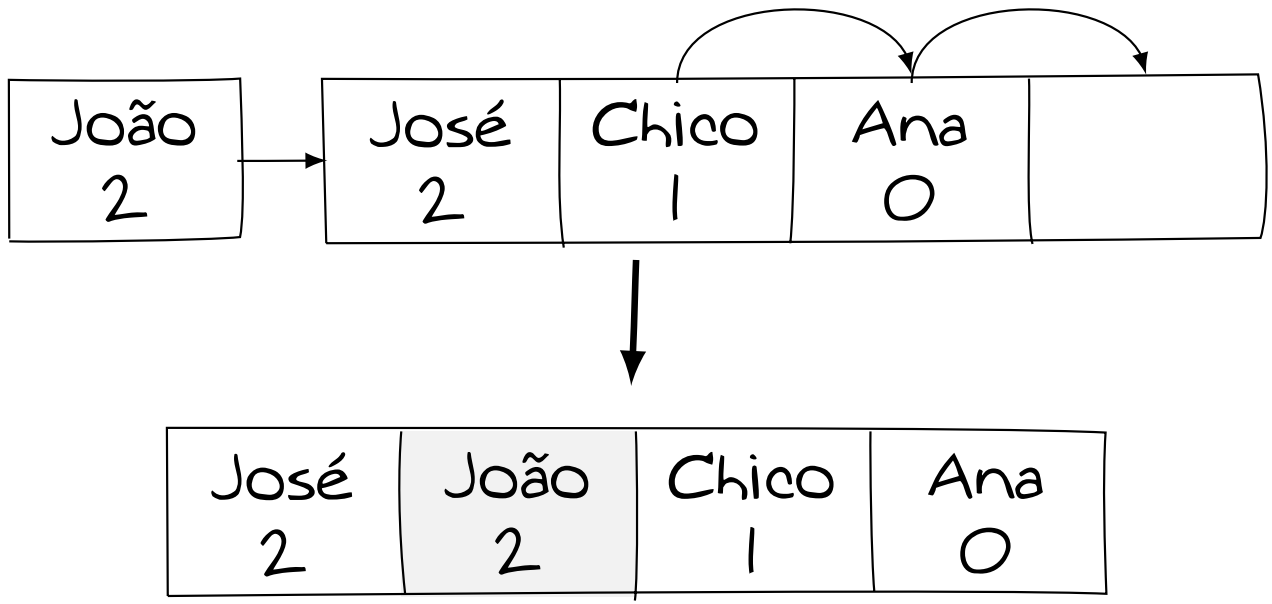


Fila de prioridade

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

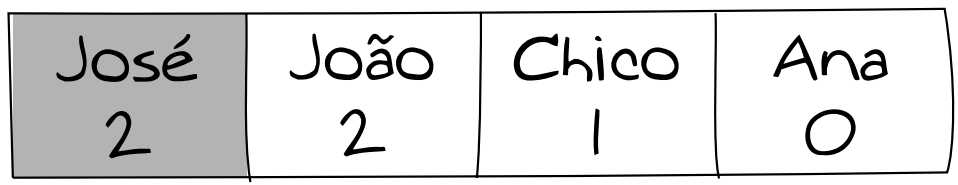
Fila de prioridade

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



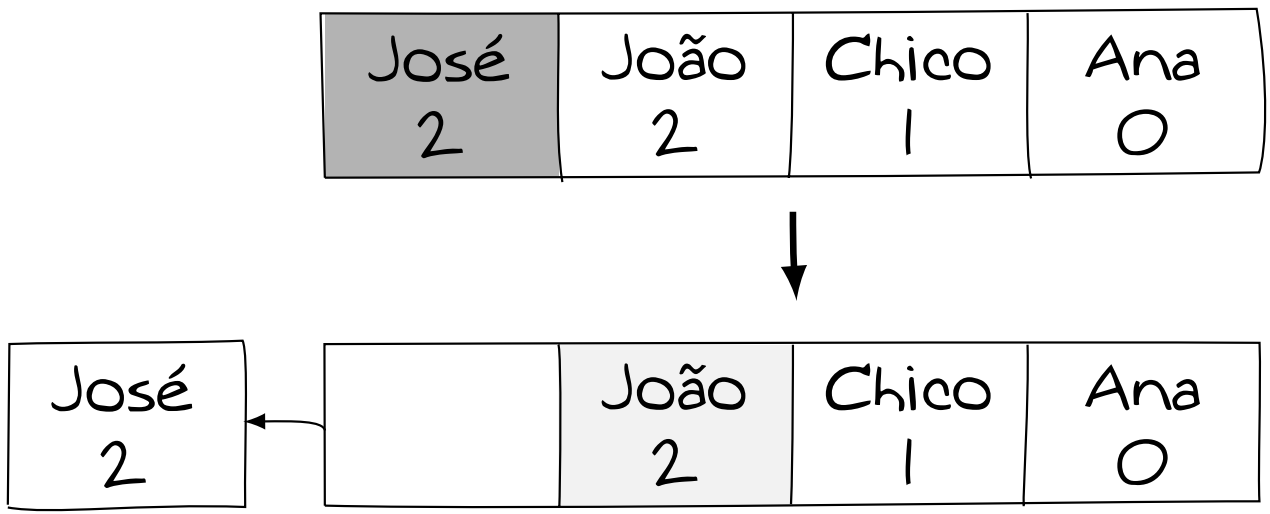
Fila de prioridade

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



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 $\Theta(1)$

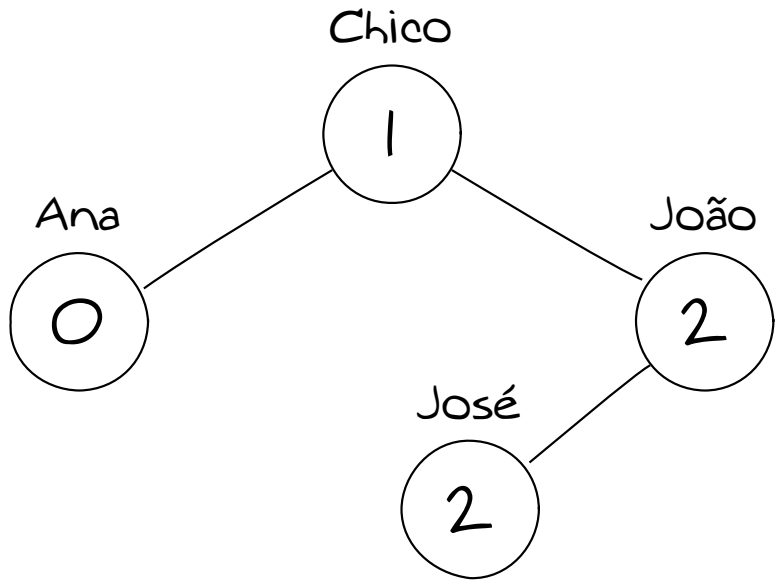


Fila de prioridade

- Análise de complexidade
  - Considerando filas de prioridade com n elementos que foram enfileirados e desenfileirados
  - Espaço: $\Theta(n)$
  - Tempo: $O(n^2)$

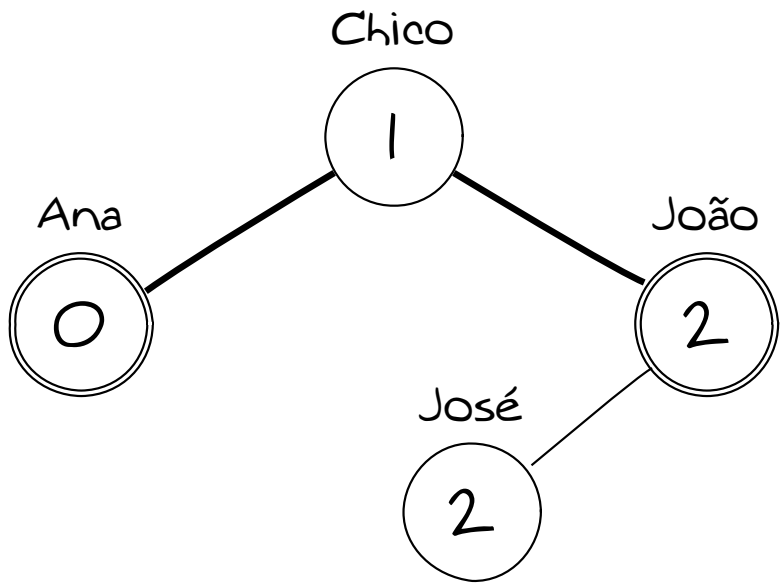
Fila de prioridade

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



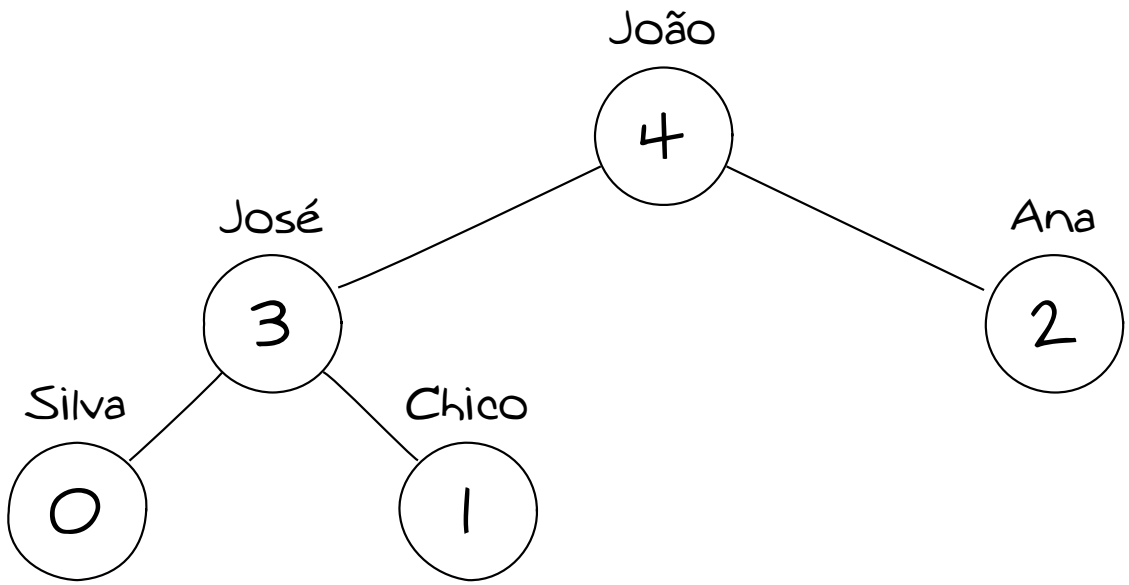
Fila de prioridade

- 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 $O(\log_2 n)$



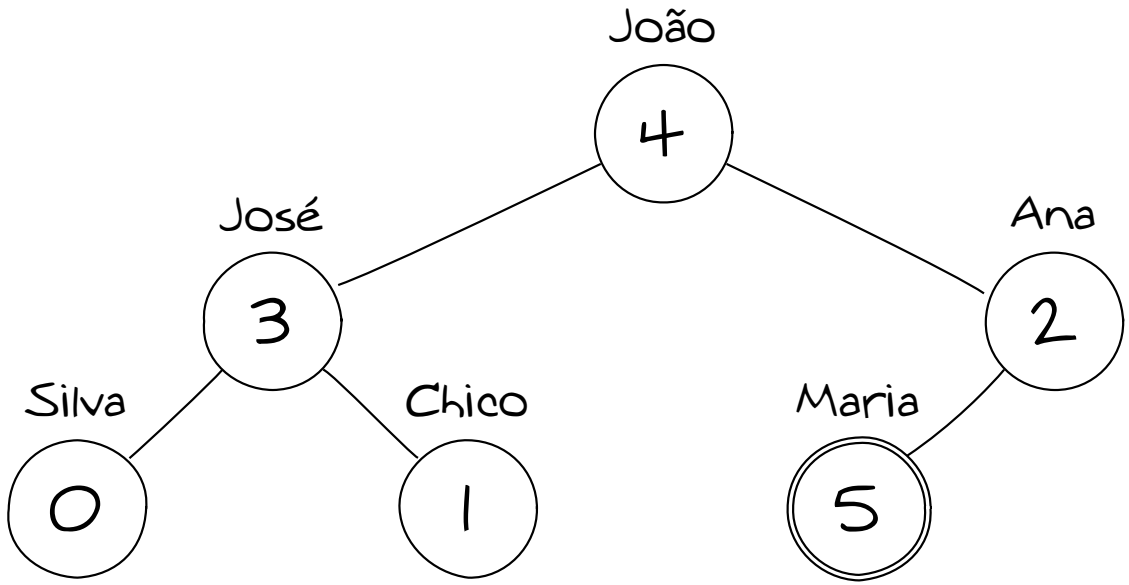
Fila de prioridade

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



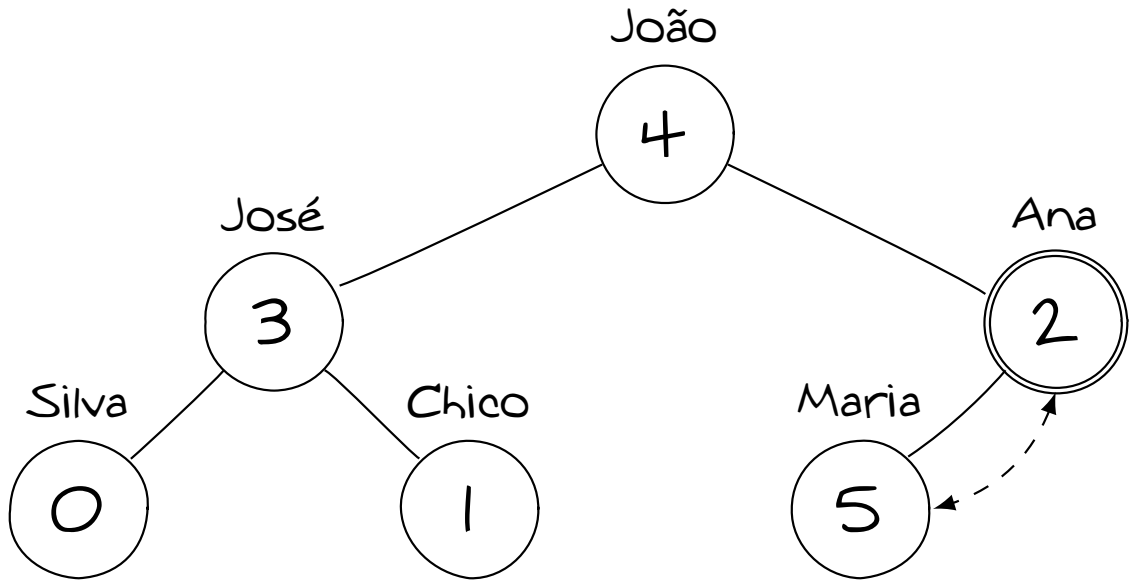
Fila de prioridade

- 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



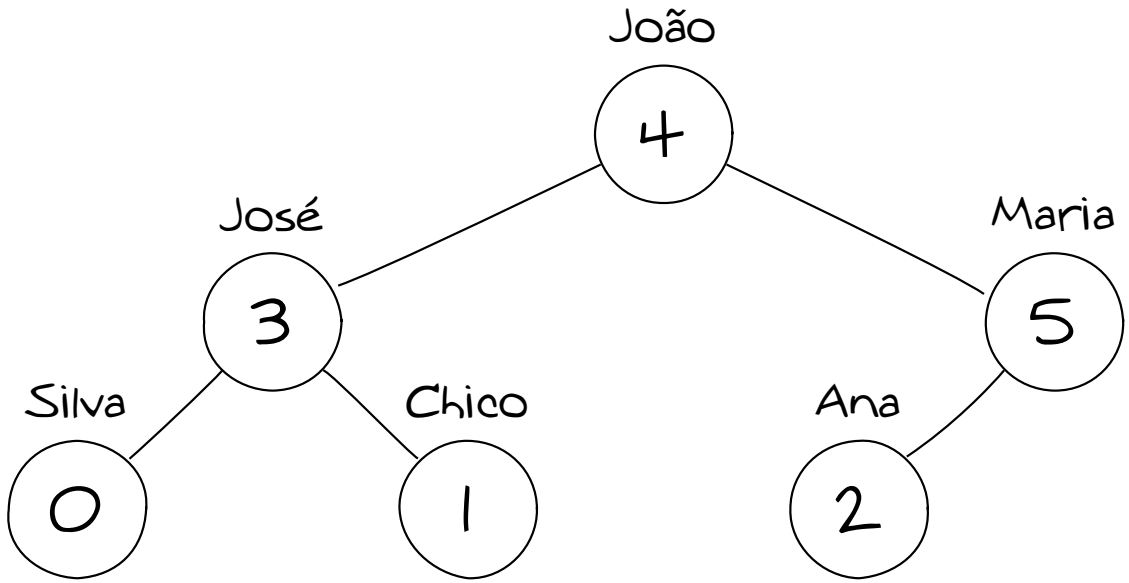
Fila de prioridade

- 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



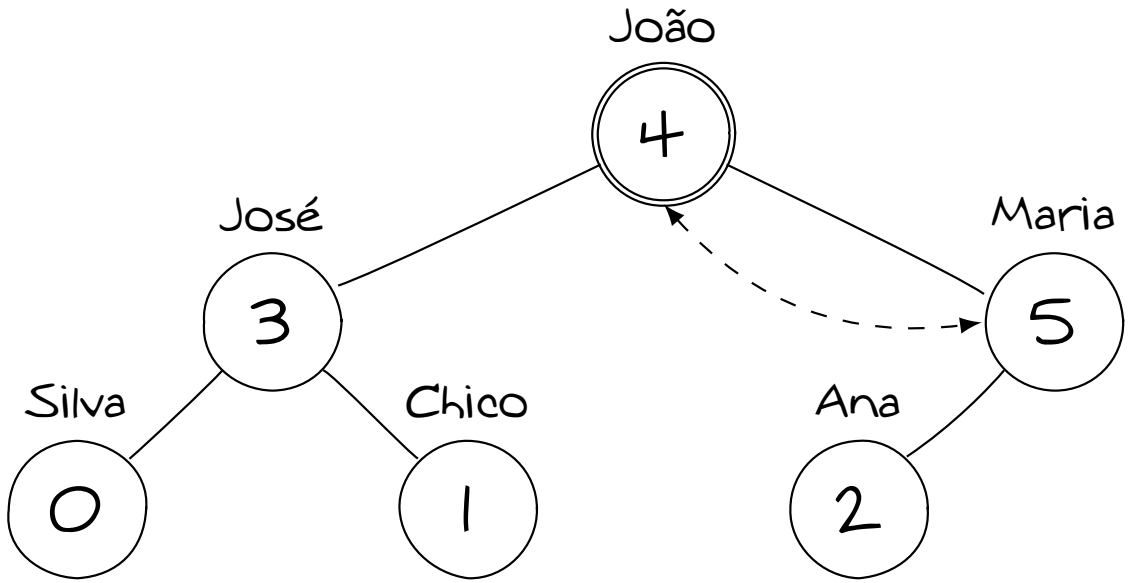
Fila de prioridade

- 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



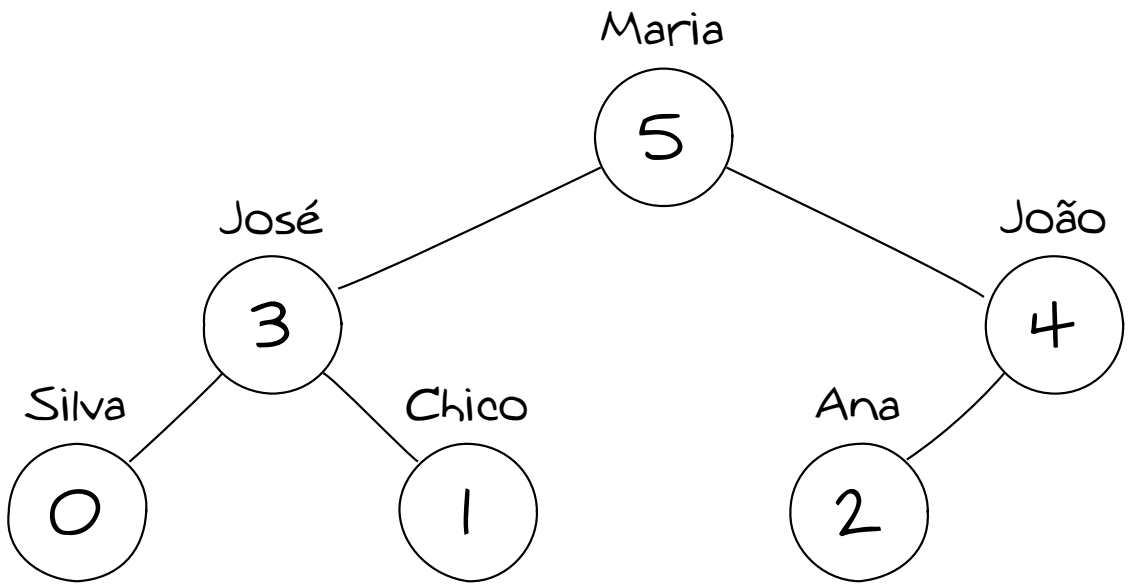
Fila de prioridade

- Implementação com *heap*
  - 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



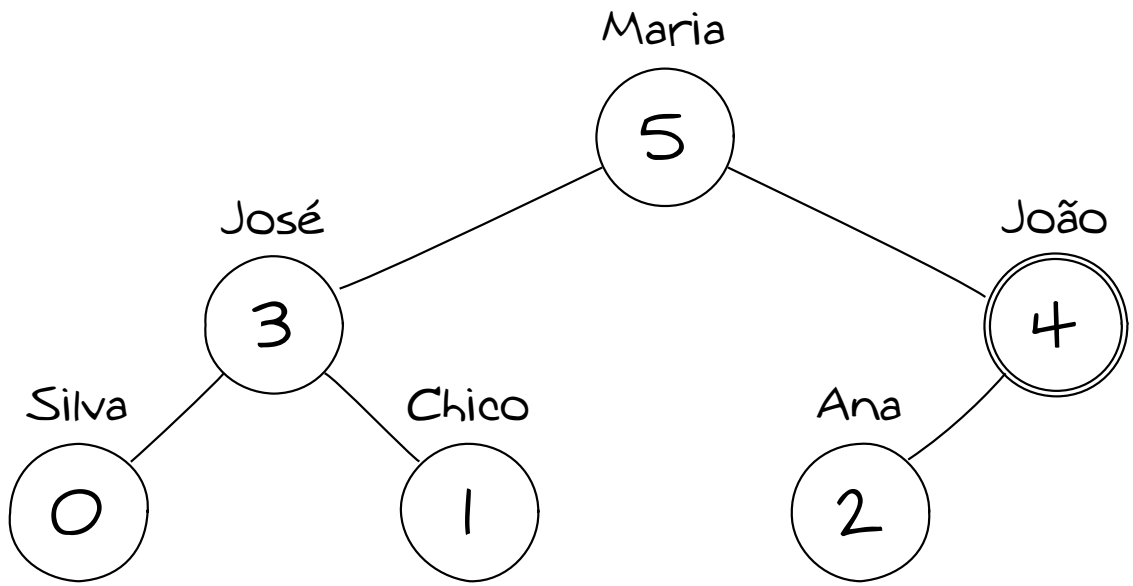
Fila de prioridade

- 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



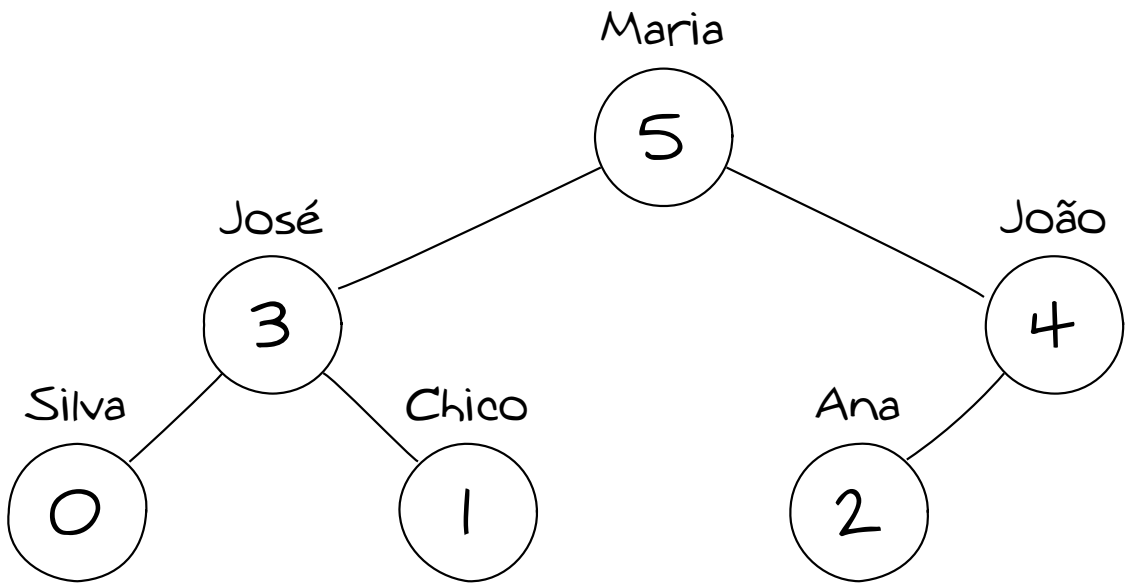
Fila de prioridade

- 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



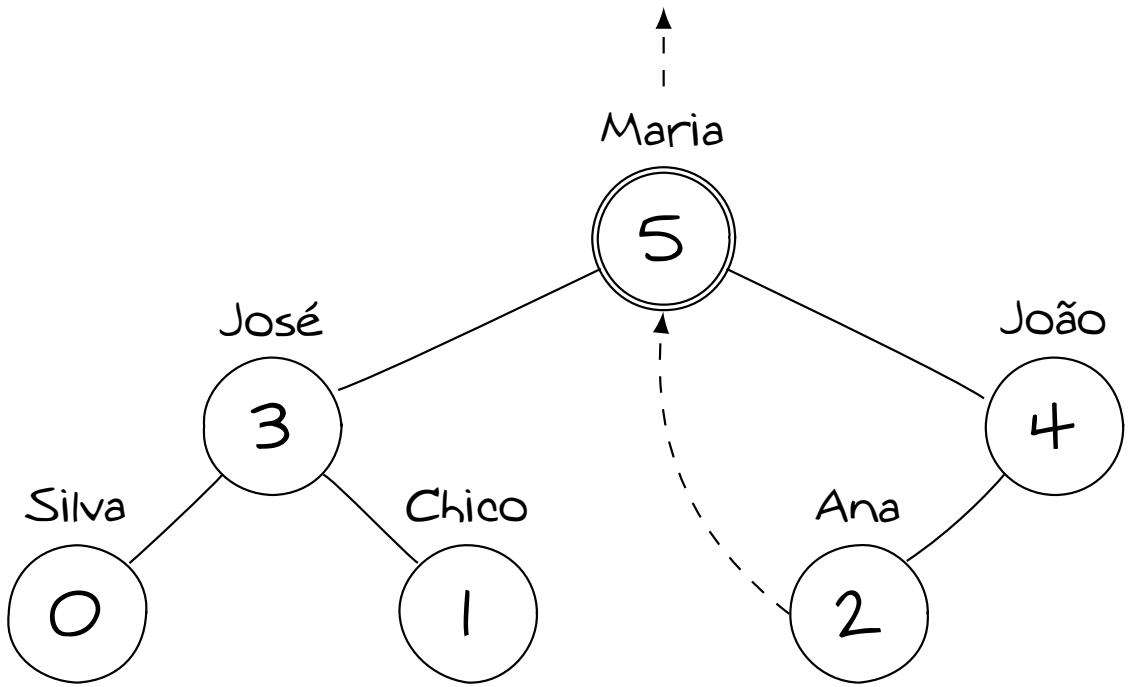
Fila de prioridade

- Implementação com *heap*
  - 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



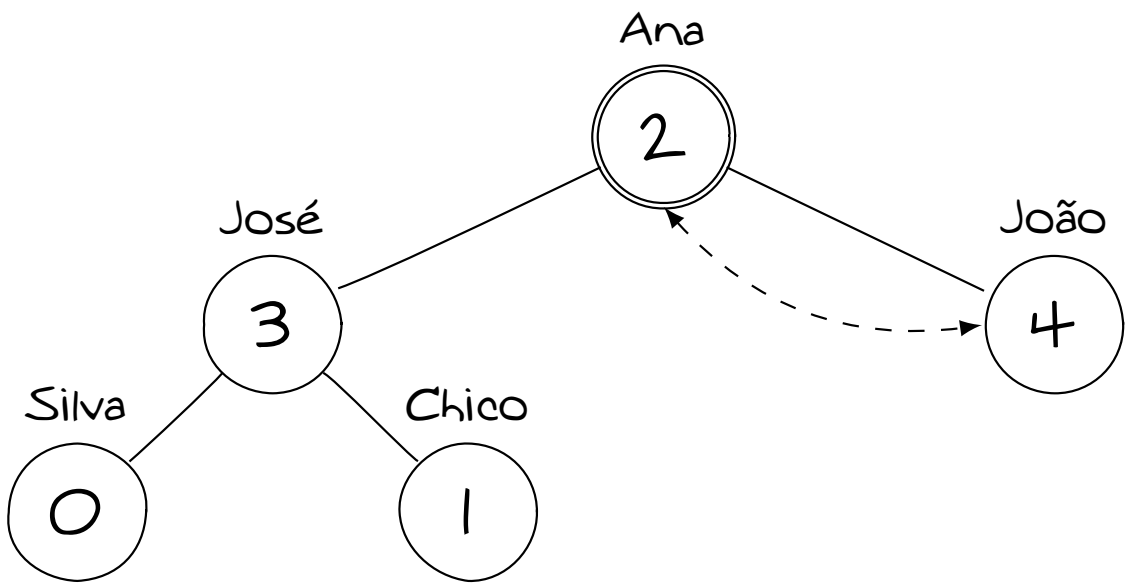
Fila de prioridade

- 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 $O(\log_2 n)$



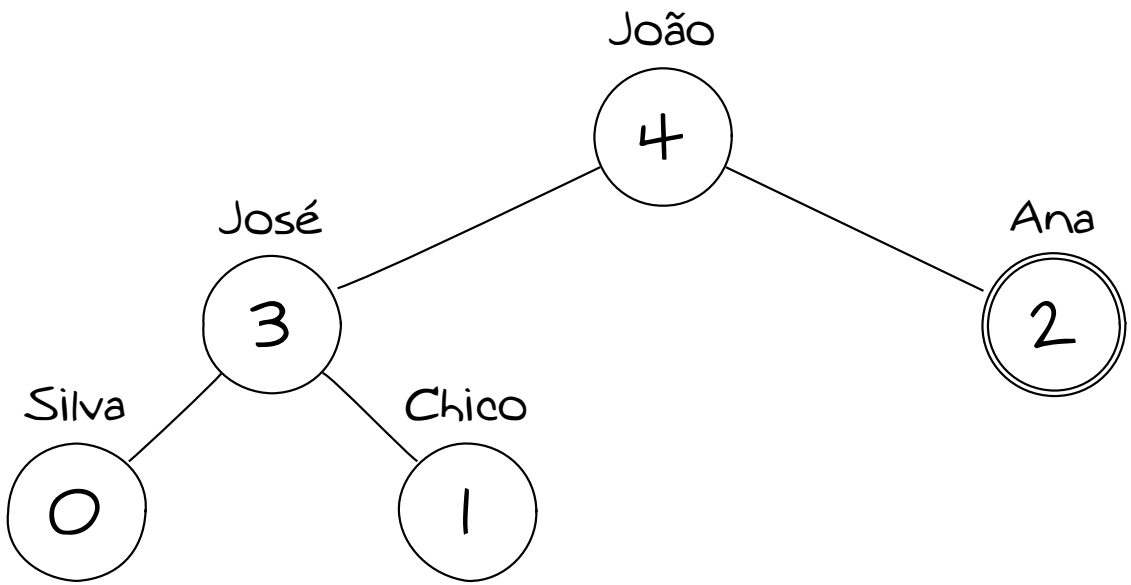
Fila de prioridade

- Implementação com *heap*
  - 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 $O(\log_2 n)$



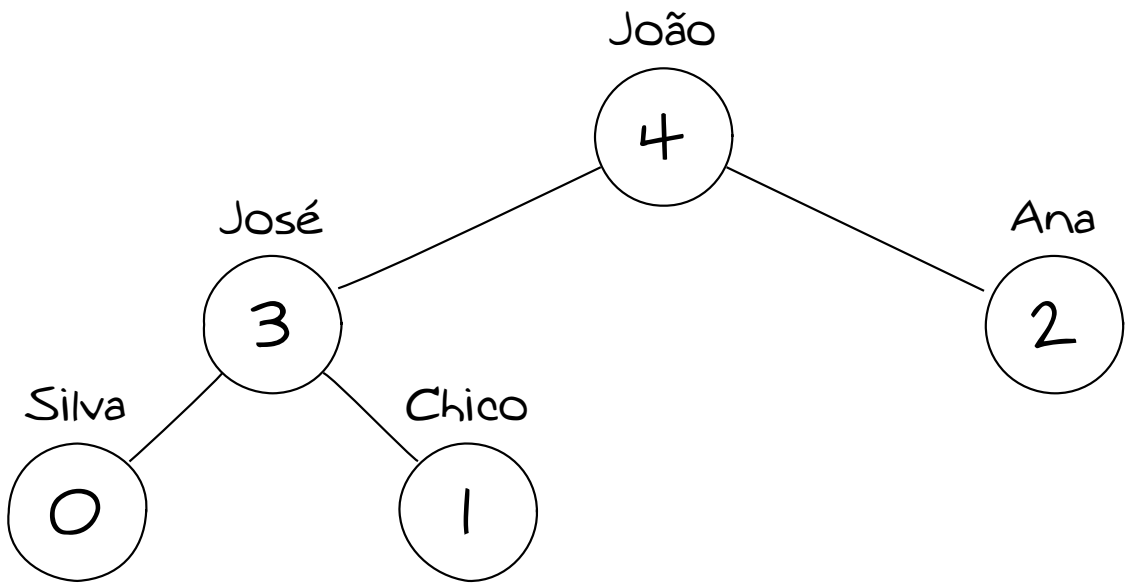
Fila de prioridade

- Implementação com *heap*
  - 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*
  - 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

- Análise de complexidade
  - Considerando filas de prioridade com n elementos que foram enfileirados e desenfileirados
  - Espaço $\Theta(n)$
  - Tempo $O(n \log_2 n)$

Fila de prioridade

- Quadro comparativo das implementações

Implementação	Enfileirar	Desenfileirar	Total
Sem ordenação	$O(1)$	$O(n)$	$O(n^2)$
Com ordenação	$O(n)$	$O(1)$	$O(n^2)$
Árvore balanceada	$O(log_2 n)$	$O(log_2 n)$	$O(nlog_2 n)$
Heap	$O(log_2 n)$	$O(log_2 n)$	$O(nlog_2 n)$

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

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 (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 registrada a ordem de chegada
  - Existem dois tipos de atendimento: convencional (idade < 60 anos) e preferencial (idade ≥ 60 anos), com menor e maior prioridade, respectivamente
  - Os nomes utilizados para os órgãos e pessoas possuem até 50 caracteres compostos por letras

Exercício

- Formato de arquivo de entrada
  - [*Quantidade de órgãos*(n)]
  - [*Órgão*$_1$] [#*Atendentes*]
  - $\vdots$
  - [*Órgão*$_n$] [#*Atendentes*]
  - [*Quantidade de pessoas*(m)]
  - [*Órgão*$_1$] | [*Nome*$_1$] | [*Idade*$_1$]
  - $\vdots$
  - [*Órgão*$_m$] | [*Nome*$_m$] | [*Idade*$_m$]

Exercício

- Formato de arquivo de entrada

```
2
DETRAN␣2
SSP␣1
7
SSP|Joao␣da␣Silva|33
SSP|Jose␣dos␣Santos|22
DETRAN|Josefa␣Souza|35
DETRAN|Ana␣Maria|22
DETRAN|Chico␣Jose|77
DETRAN|Jair␣Luiz|61
DETRAN|Inacio␣Messias|64
```

Exercício

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

```
1 DETRAN:Chico␣Jose,Jair␣Luiz
2 SSP:Joao␣da␣Silva
3 DETRAN:Inacio␣Messias,Josefa␣Souza
4 SSP:Jose␣dos␣Santos
5 DETRAN:Ana␣Maria
```