Estruturas de Dados

Filas de Prioridade

Prof. André Luiz Moura
<andreluiz@inf.ufg.br>

INSTITUTO DE
INFORMÁTICA
UFG

Filas de Prioridade: Definição

São um tipo especial de fila que generaliza a ideia de "ordenação"

Os elementos inseridos na fila possuem um dado extra associados a eles: a sua "prioridade"

É o valor associado a essa "prioridade" que determina a posição de uma elemento na fila, assim como quem deve ser o primeiro a ser removido da fila, quando necessário.

Filas de Prioridade: Aplicações

Basicamente, qualquer problema em que seja preciso estabelecer uma prioridade de acesso aos elementos pode ser representado com uma fila de prioridade.

Exemplos:

- Processos em um processador
- Uma fila de pacientes esperando transplante de fígado
- Busca em grafos (algoritmo de Dijkstra)
- Compressão de dados (código de Huffman)
- Sistemas operacionais (manipulação de interrupções)
- Fila de pouso de aviões em aeroporto (prioridade por combustível disponível)

Filas de Prioridade: Operações

Em uma "Fila de Prioridade", pode-se realizar as seguintes operações básicas:

- Criação de fila
- Inserção de um elemento na fila com prioridade
- Remoção do elemento da fila com maior prioridade
- Acesso ao elemento do início da fila (maior prioridade)
- Destruição da fila
- Informações sobre tamanho, se a fila estiver cheia ou vazia

O modo de implementar essas operações depende da implementação: *lista dinâmica encadeada*, *array desordenado*, *array ordenado*, *heap binária*

Filas de Prioridade: Implementação

Qual implementação escolher?

- A implementação escolhida para uma "fila de prioridade" depende da sua aplicação
- Algumas implementações são eficientes na operação de "inserção"; outras na de "remoção"
- Há também implementações que são eficientes nas duas operações

Implementação	Inserção	Remoção
Lista dinâmica encadeada	O(N)	O(1)
Array desordenado	O(1)	O(N)
Array ordenado	O(N)	O(1)
Heap binária	O(log N)	O(log N)

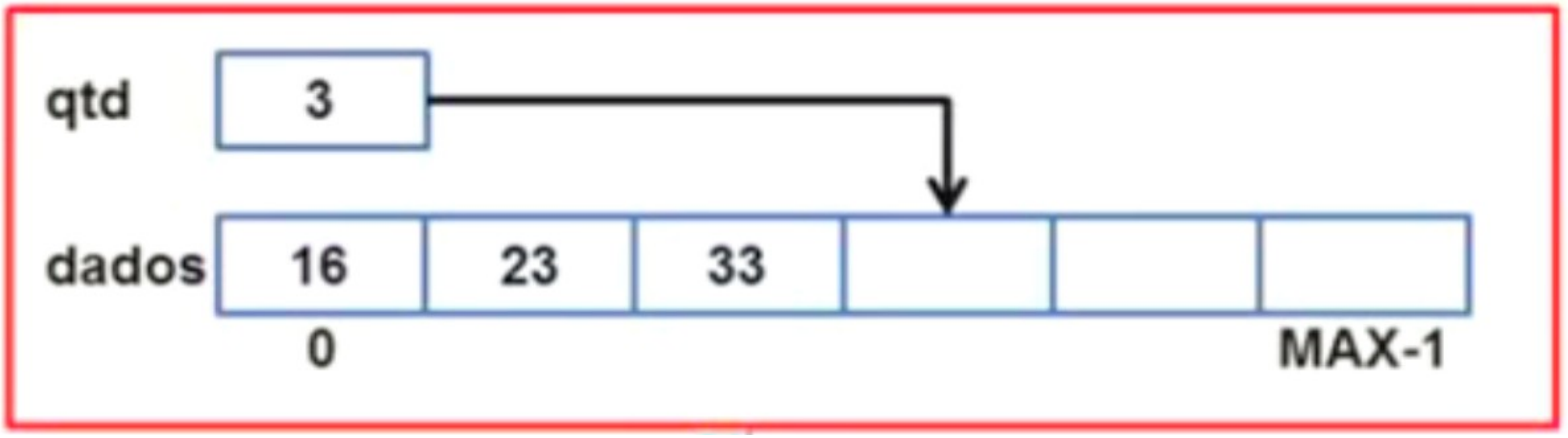
Implementando uma Fila de Prioridade

"Fila de Prioridades Estática"

Sua implementação utiliza a mesma estrutura da "Lista Sequencial Estática"

Utiliza um "array" para armazenar os elementos

```
Fila *fp;
```



Implementação com Array

Desvantagens:

- Necessitar definir o tamanho do array previamente
- Isso limita o número de elementos a serem armazenados

Vantagens:

- O uso de um array permite que se utilize a mesma estrutura de fila para duas implementações distintas:
  - Array ordenado
  - Heap binária
- Para tanto, basta modificar as funções de
  - Inserção
  - Remoção
  - Consulta

Implementação em C

```
//Arquivo FilaPrioridade.h
#define MAX 100
typedef struct fila_prioridade FilaPrio;
FilaPrio* cria_FilaPrio();
void libera_FilaPrio(FilaPrio * fp);
int consulta_FilaPrio(FilaPrio * fp, char * nome);
int insere_FilaPrio(FilaPrio * fp, char * nome, int prio);
int remove_FilaPrio(FilaPrio * fp);
int tamanho_FilaPrio(FilaPrio * fp);
int estaCheia_FilaPrio(FilaPrio * fp);
int estaVazia_FilaPrio(FilaPrio * fp);

//Programa principal
FilaPrio* fp;
```

Implementação em C

```
//Arquivo FilaPrioridade.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "FilaPrioridade.h"   //Inclui os protótipos
struct paciente {
   char nome[30];
   int prio;
} ;

//Definição do tipo fila de prioridade
struct fila_prioridade {
   int qtd;
   struct paciente dados[MAX];
}
```

Funções básicas

Importante:

- Essa Fila de Prioridades utiliza a mesma implementação da Lista Sequencial Estática

Assim, as funções abaixo tem implementação exatamente igual a das funções da Lista Sequencial Estática

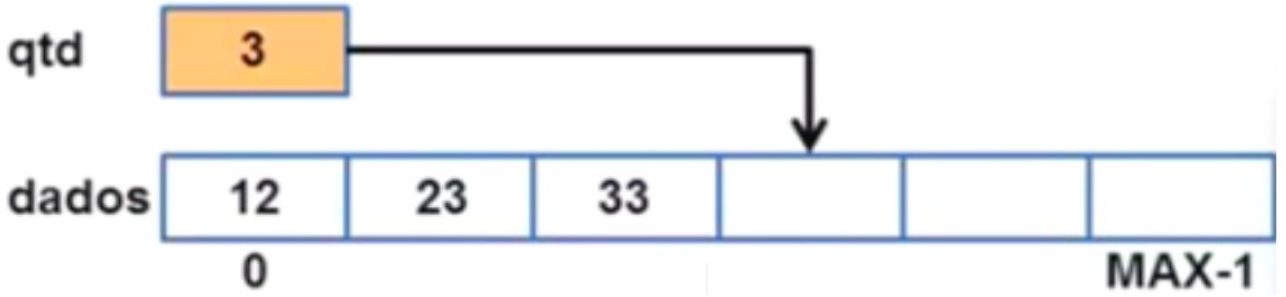
```
FilaPrio* cria_FilaPrio();
void libera_FilaPrio(FilaPrio * fp);
int tamanho_FilaPrio(FilaPrio * fp);
int estaCheia_FilaPrio(FilaPrio * fp);
int estaVazia_FilaPrio(FilaPrio * fp);
```

Fila de Prioridade usando array ordenado

- Os elementos na fila de prioridade são ordenados de forma crescente dentro do array
- O elemento de maior prioridade estará no final do array (início da fila)
- O elemento de menor prioridade estará no final do array (final da fila)

Custo:

- Inserção O(N) (precisa-se procurar o ponto de inserção no array)
- Remoção O(1) (último elemento do array)



Fila de Prioridade usando array ordenado

Inserção:

- É necessário procurar o ponto de inserção do elemento na fila de acordo com a sua prioridade
- Esse local pode ser no início, no meio ou no final do array

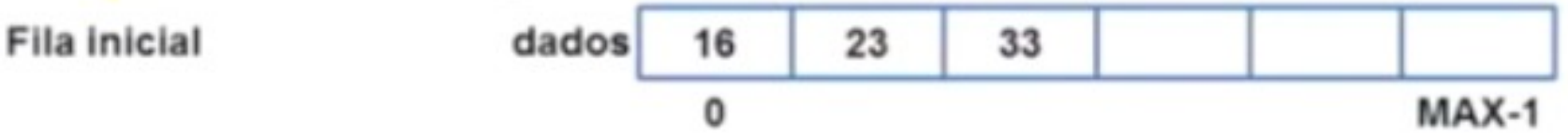
Nos dois primeiros casos (início ou meio), é preciso mudar o lugar dos demais elementos do array

Fila de Prioridade usando array ordenado

```
1  //Programa principal
2  int x = insere_FilaPrio(fp, "Bianca", 10);
3  //Arquivo FilaPrioridade.h
4  int insere_FilaPrio(FilaPrio* fp, char *nome, int prio);
5  //Arquivo FilaPrioridade.c
6  int insere_FilaPrio(FilaPrio* fp, char *nome, int prio){
7      if(fp == NULL) return 0;
8      if(fp->qtd == MAX) return 0;//fila cheia
9
10     int i = fp->qtd-1;
11     while(i >= 0 && fp->dados[i].prio >= prio){
12         fp->dados[i+1] = fp->dados[i];
13         i--;
14     }
15     strcpy(fp->dados[i+1].nome,nome);
16     fp->dados[i+1].prio = prio;
17     fp->qtd++;
18     return 1;
19 }
```

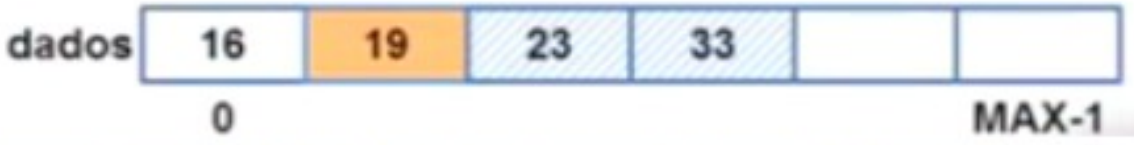
Fila de Prioridade usando array ordenado

```
1  //Programa principal
2  int x = insere_FilaPrio(fp, "Bianca", 10);
```



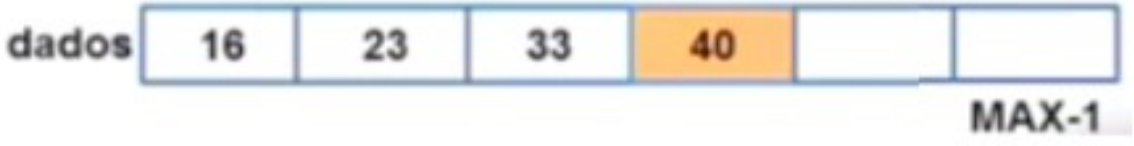
Busca posição na fila deslocando os elementos (se necessário)

```
i = fp->qtd-1;
while(i >= 0 && fp->dados[i].prio >= prioridade){
   fp->dados[i+1] = fp->dados[i];
   i--;
}
```



Inserir elemento

```
strcpy(fp->dados[i+1].nome,nome);
fp->dados[i+1].prio = prioridade;
fp->qtd++;
```

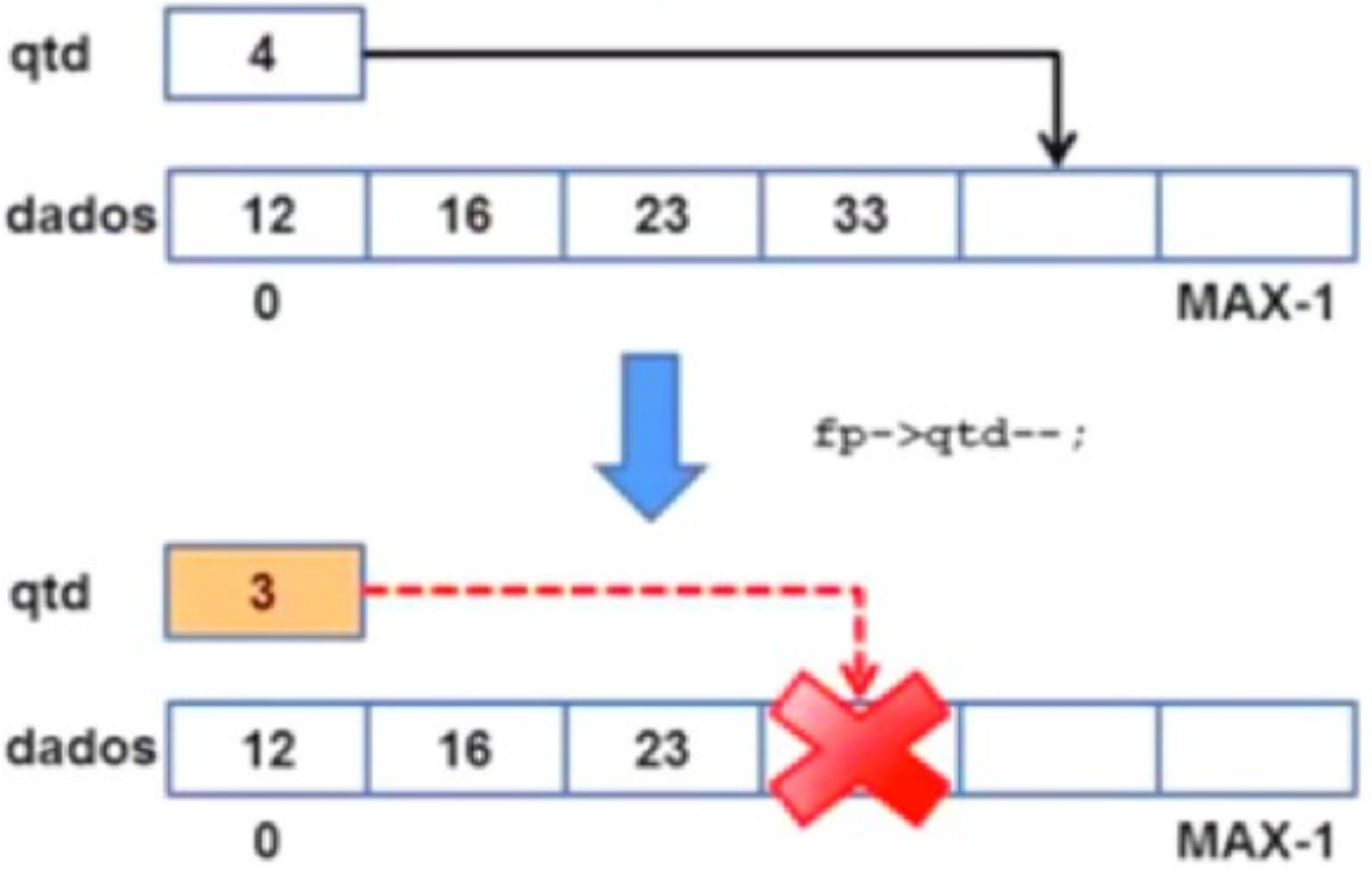


Fila de Prioridade usando array ordenado

```
1  //Programa principal
2  int x = remove_FilaPrio(fp);
3  //Arquivo FilaPrioridade.h
4  int remove_FilaPrio(FilaPrio* fp);
5  //Arquivo FilaPrioridade.c
6  int remove_FilaPrio(FilaPrio* fp){
7      if(fp == NULL)
8          return 0;
9      if(fp->qtd == 0)
10         return 0;
11     fp->qtd--;
12     return 1;
13 }
```

Fila de Prioridade usando array ordenado

```
1  //Programa principal
2  int x = remove_FilaPrio(fp);
```



Fila de Prioridade usando array ordenado

```
1  //Programa principal
2  int x = consulta_FilaPrio(fp, nome);
3  //Arquivo FilaPrioridade.h
4  int consulta_FilaPrio(FilaPrio* fp, char* nome);
5  //Arquivo FilaPrioridade.c
6  int consulta_FilaPrio(FilaPrio* fp, char* nome){
7      if(fp == NULL || fp->qtd == 0)
8          return 0;
9      strcpy(nome,fp->dados[fp->qtd-1].nome);
10     return 1;
11 }
```

Fila de Prioridade usando heap binária

- Uma heap permite simular uma árvore binária completa ou quase completa
- Cada posição do array passa a ser considerado o pai de duas outras posições, chamadas filhos
- Posição i passa a ser o pai das posições
  - 2*i+1 (filho da esquerda)
  - 2*i+2 (filho da direita)
- Os elementos são dispostos na heap de forma que um pai tem sempre uma prioridade maior ou igual à prioridade de seus filhos

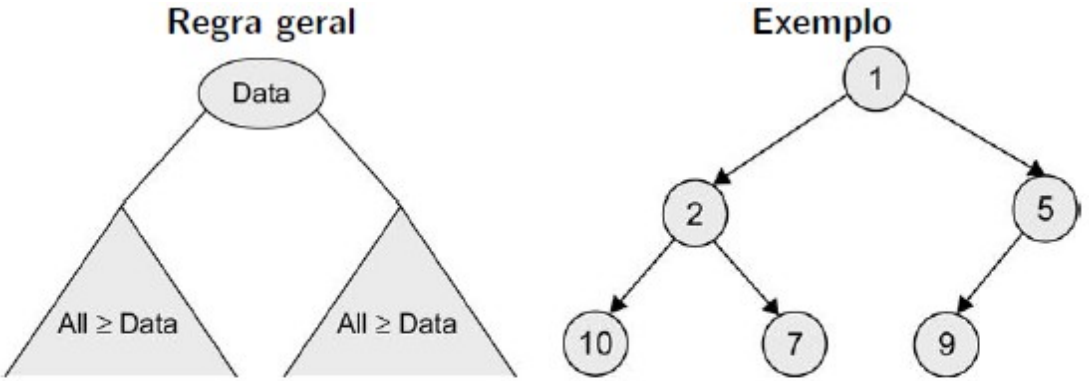
Custo:

- Inserção: O(logN)
- Remoção: O(logN)

Em ambas, necessita-se verificar e corrigir violações de propriedades da heap

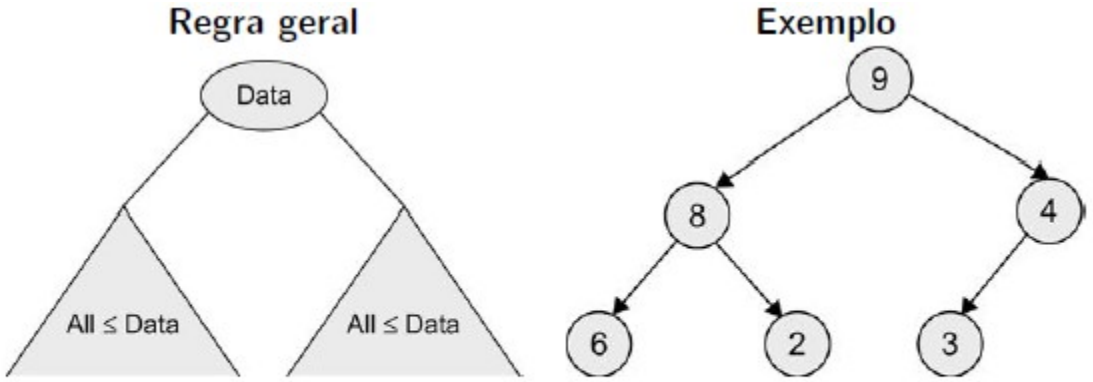
Heap Binária

- Heap Mínimo



Todo caminho da raiz para nós folha são ordenados em ordem crescente.

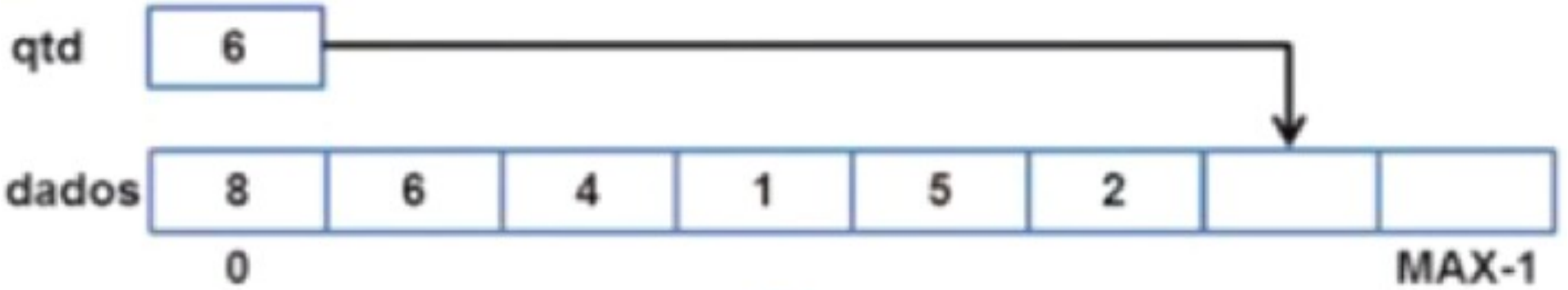
- Heap Máximo

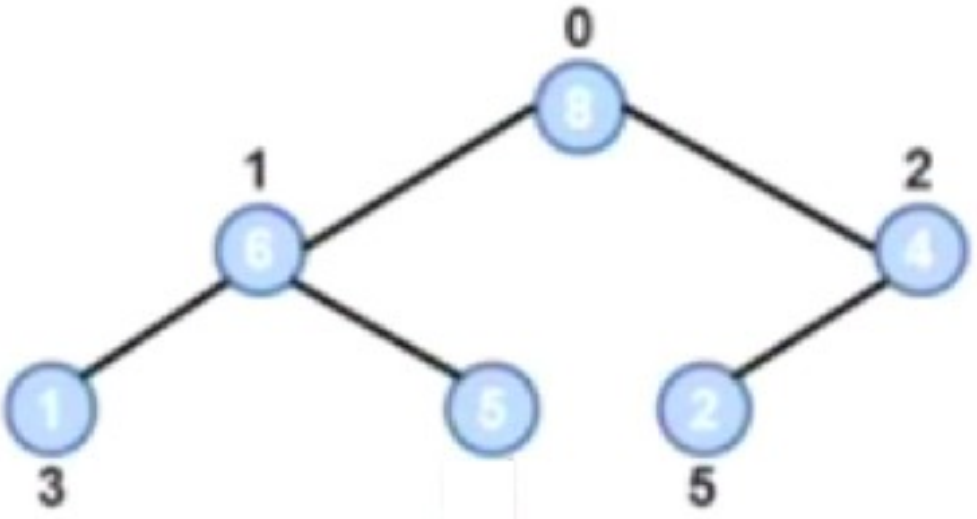


Todo caminho da raiz para nós folha são ordenados em ordem decrescente.

Fila de Prioridade usando heap binária

- Uma heap permite simular uma árvore binária completa ou quase completa





Fila de Prioridade usando heap binária

Inserção:

- Inserir o novo elemento na primeira posição vaga do array, ou seja, no final do array
- Levar o elemento inserido para a sua respectiva posição na heap de acordo com a sua prioridade

Fila de Prioridade usando heap binária

```
//Programa principal
int x = insere_FilaPrio(fp, "Banca, 10);
//Arquivo FilaPrioridade.h
int insere_FilaPrio(FilaPrio* fp, char *nome, int prio);
//Arquivo FilaPrioridade.c
int insere_FilaPrio(FilaPrio* fp, char *nome, int prio){
    if(fp == NULL)
        return 0;
    if(fp->qtd == MAX)//fila cheia
        return 0;
    strcpy(fp->dados[fp->qtd].nome,nome);
    fp->dados[fp->qtd].prio = prio;
    promoverElemento(fp,fp->qtd);
    fp->qtd++;
    return 1;
}
```

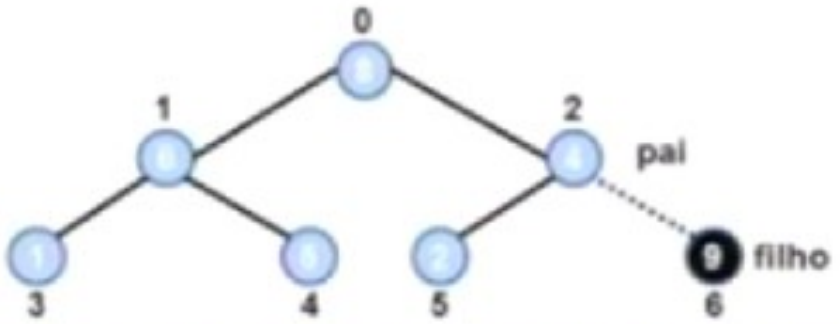
Fila de Prioridade usando heap binária

```
void promoverElemento(FilaPrio* fp, int filho){
    int pai;
    struct paciente temp;
    pai = (filho - 1) / 2;
    while((filho > 0) &&
          (fp->dados[pai].prio <= fp->dados[filho].prio))

        temp = fp->dados[filho];
        fp->dados[filho] = fp->dados[pai];
        fp->dados[pai] = temp;

        filho = pai;
        pai = (pai - 1) / 2;
    }
}
```

Fila de Prioridade usando heap binária

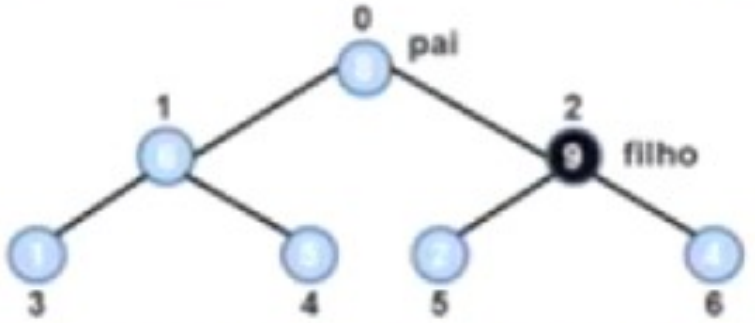
```
//Programa principal
int x = insere_FilaPrio(fp, "Banca, 10);
```

Insere elemento com prioridade 9



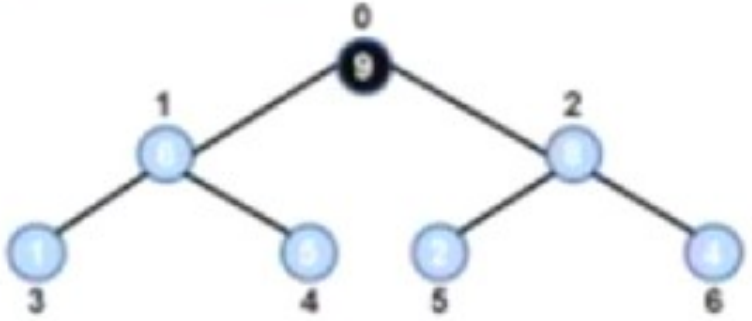
Prioridade do pai
é menor do que a
Prioridade do filho:

Trocar os dois de lugar



Prioridade do pai
é menor do que a
Prioridade do filho:

Trocar os dois de lugar



Elemento é o primeiro
da Heap

Finalizar processo

Fila de Prioridade usando heap binária

Remoção:

- Remover o elemento que está no topo da heap, ou seja no início do array
- Copiar o elemento do final para o início do array
- Levar o elemento que foi colocado no topo da heap para a sua respectiva posição de acordo com a sua prioridade

Fila de Prioridade usando heap binária

```
//Programa principal
int x = remove_FilaPrio(fp);
//Arquivo FilaPrioridade.h
int remove_FilaPrio(FilaPrio* fp);
//Arquivo FilaPrioridade.c
int remove_FilaPrio(FilaPrio* fp){
    if(fp == NULL)
        return 0;
    if(fp->qtd == 0)
        return 0;

    fp->qtd--;
    fp->dados[0] = fp->dados[fp->qtd];
    rebaixarElemento(fp,0);
    return 1;
}
```

Fila de Prioridade usando heap binária

```
void rebaixarElemento(FilaPrio* fp, int pai){
  struct paciente temp;
  int filho = 2 * pai + 1;
  while(filho < fp->qtd){
    if(filho < fp->qtd-1)
      if(fp->dados[filho].prio < fp->dados[filho+1].prio)
          filho++;

    if(fp->dados[pai].prio >= fp->dados[filho].prio)
       break;

    temp = fp->dados[pai];
    fp->dados[pai] = fp->dados[filho];
    fp->dados[filho] = temp;

    pai = filho;
    filho = 2 * pai + 1;
  }
}
```

Pai tem 2 filhos?
Quem é o maior?

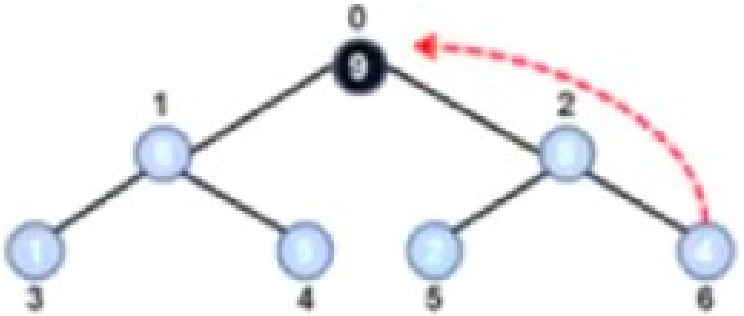
Pai >= Filho?
Terminar processo

Trocar Pai e Filho de lugar
Filho vira Pai
Calcular novo Filho

Fila de Prioridade usando heap binária

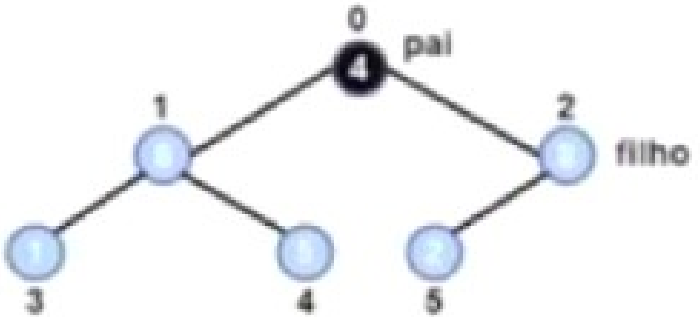
```
//Programa principal
int x = remove_FilaPrio(fp);
```

Remove elemento com maior prioridade



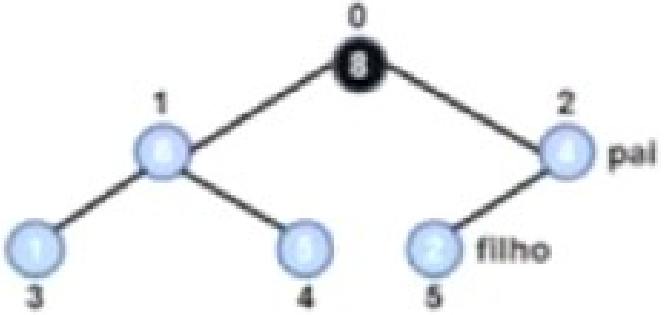
Copia o último elemento da Heap para o topo

Iniciar ajuste da Heap



Prioridade do pai é menor do que a Prioridade do filho:

Trocar os dois de lugar

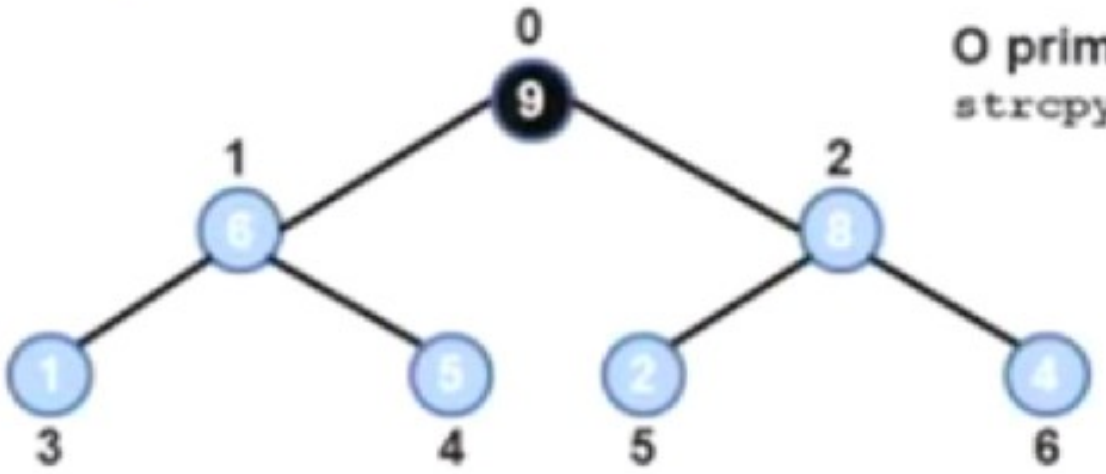


Prioridade do pai é maior do que a Prioridade do filho:

Finalizar processo

Fila de Prioridade usando heap binária

```
//Programa principal
int x = consulta_FilaPrio(fp, nome);
//Arquivo FilaPrioridade.h
int consulta_FilaPrio(FilaPrio* fp, char* nome);
//Arquivo FilaPrioridade.c
int consulta_FilaPrio(FilaPrio* fp, char* nome){
    if(fp == NULL || fp->qtd == 0)
        return 0;
    strcpy(nome,fp->dados[0].nome);
    return 1;
}
```



O primeiro da Fila é o primeiro da Heap

```
strcpy(nome,fp->dados[0].nome);
```

Referências

Backes, André. Slides de aulas: Fila de Prioridade