Filas de prioridades

Prof. Paulo Henrique Pisani

abril/2022

Fila de prioridade

- Uma fila de prioridade é uma estrutura de dados em que cada elemento possui uma prioridade associada;
- Essa estrutura de dados possui algumas operações importantes:
  - **Enfileirar** elemento;
  - **Obter** elemento com maior prioridade;
  - **Desenfileirar** elemento com maior prioridade;
  - **Alterar** prioridade de um elemento.

Fila de prioridade

- Existem diversas formas de implementar uma fila de prioridade:
  - Lista não ordenada;
  - Lista ordenada;
  - Heap.
- Contudo, há diferenças no custo para execução de cada uma das operações, dependendo da implementação.

Lista não ordenada

- Custo das operações:
  - Enfileirar: O(1)
  - Desenfileirar: O(n)

Lista ordenada

- Custo das operações:
  - Enfileirar: O(n)
  - Desenfileirar: O(1)

Heap

- Custo das operações:
  - Enfileirar: O(log(n))
  - Desenfileirar: O(log(n))

Filas de prioridades

- Fila de prioridade máxima
- Fila de prioridade mínima

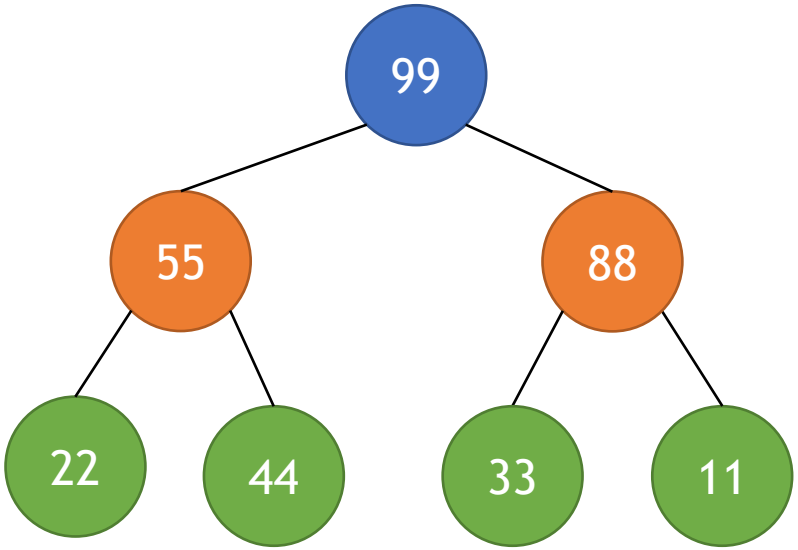
Nesta aula, será discutida a fila de prioridade máxima utilizando um heap de máximo.

Operações

- A seguir, serão discutidas as seguintes operações:
  - Desenfileirar;
  - Enfileirar;
  - Alterar.

Heap

- Armazenamento em vetor:



$$i_{esq} = 2.i + 1$$

$$i_{dir} = 2.i + 2$$

$$i_{pai} = \lfloor (i-1)/2 \rfloor$$

Nós folha ficam nos índices $\lfloor n/2 \rfloor$ até $n-1$, em que n é a quantidade de elementos no heap.

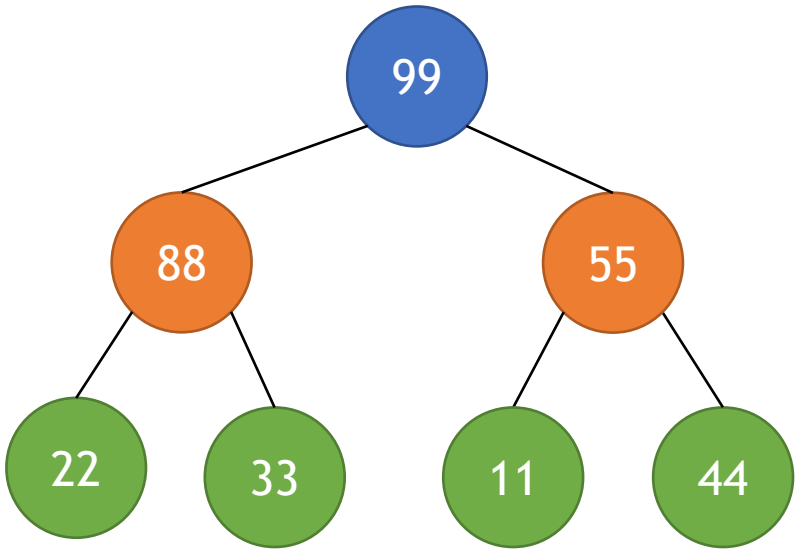
Desenfileirar

- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;

- Retornar elemento salvo no primeiro passo.

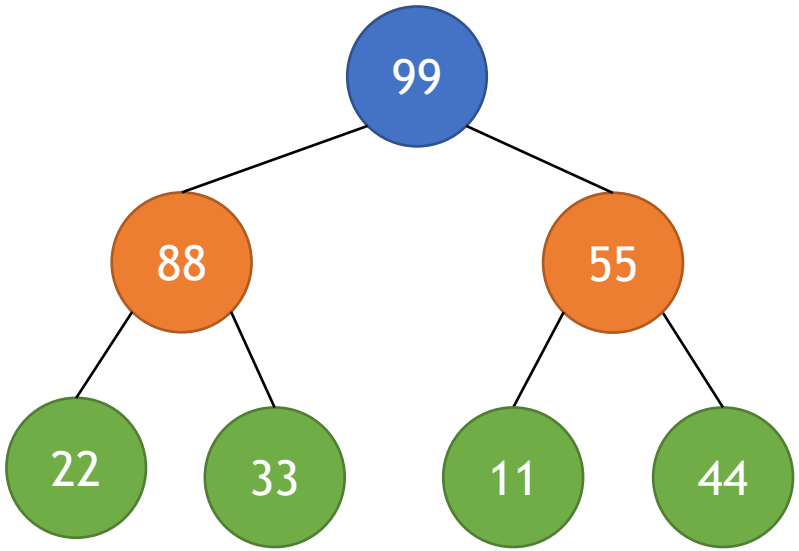
Desenfileirar

- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;
- Retornar elemento salvo no primeiro passo.



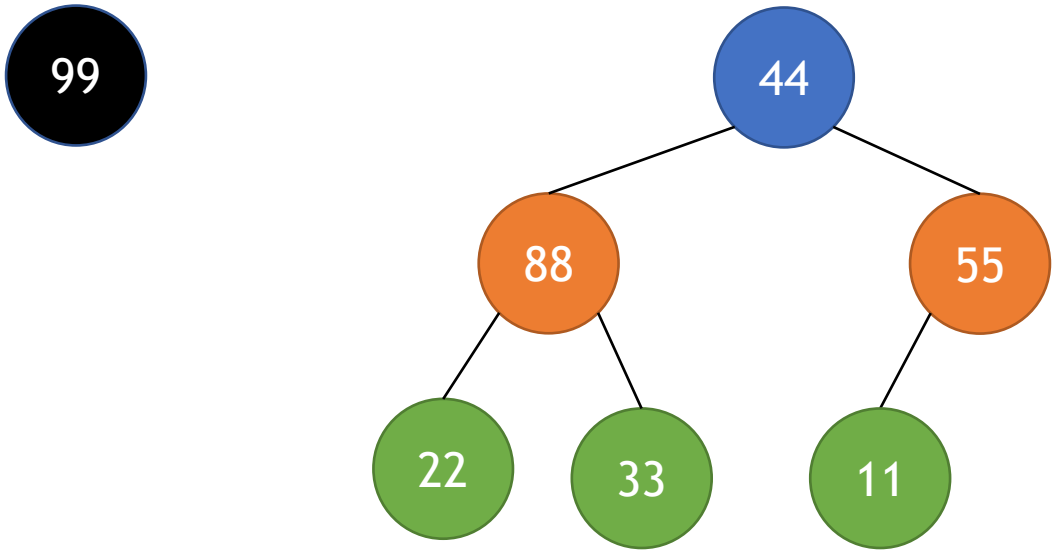
Desenfileirar

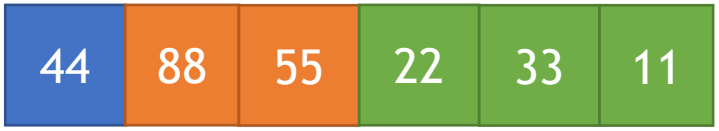
- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;
- Retornar elemento salvo no primeiro passo.



Desenfileirar

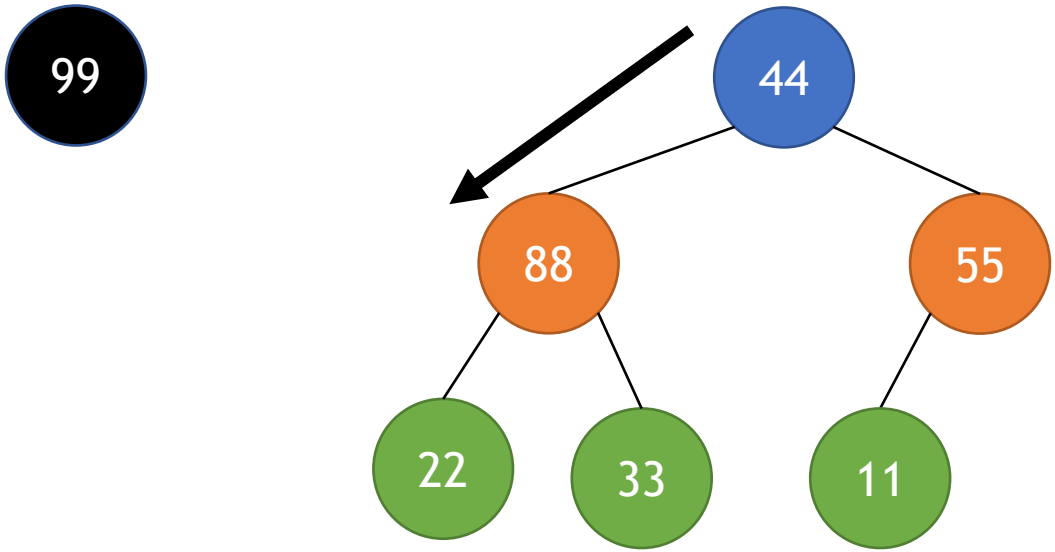
- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;
- Retornar elemento salvo no primeiro passo.

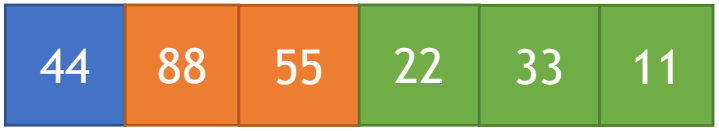




Desenfileirar

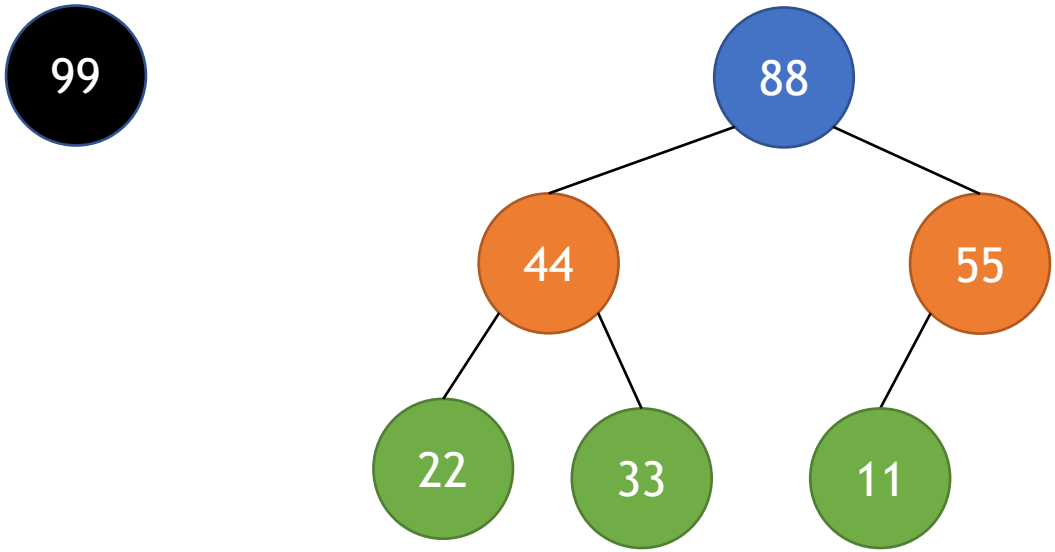
- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;
- Retornar elemento salvo no primeiro passo.

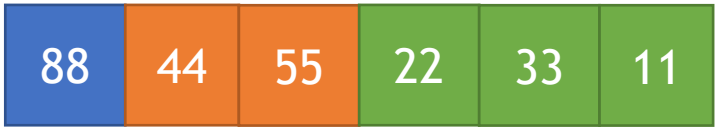




Desenfileirar

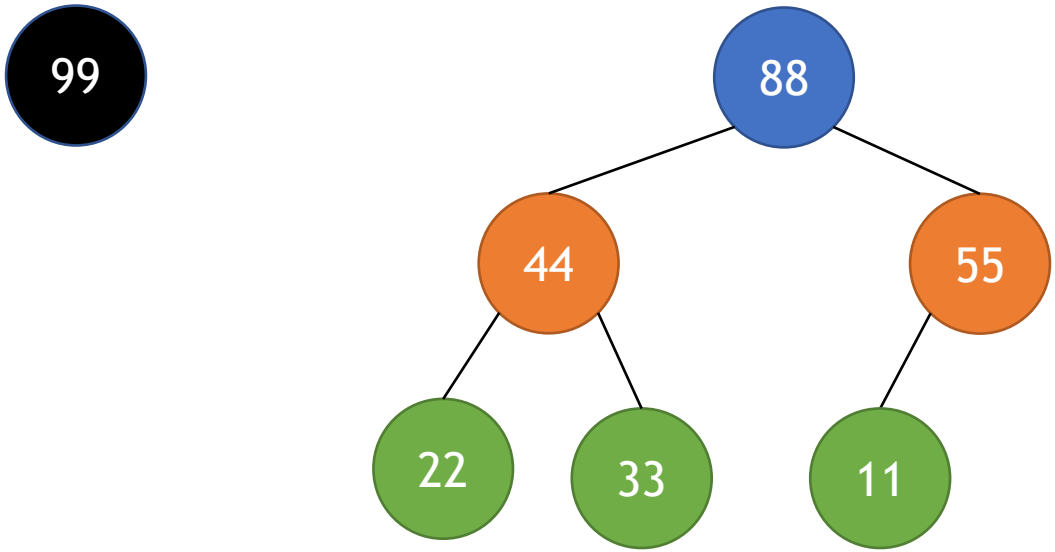
- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;
- Retornar elemento salvo no primeiro passo.

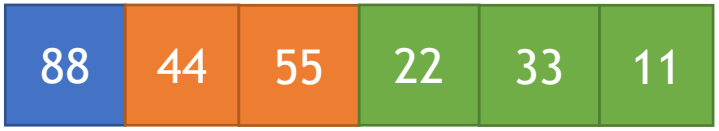




Desenfileirar

- Salvar elemento no índice 0;
- Mover o último elemento para o índice 0;
- Reduzir contador de quantidade de elementos;
- Aplicar max_heapfy no índice 0;
- Retornar elemento salvo no primeiro passo.



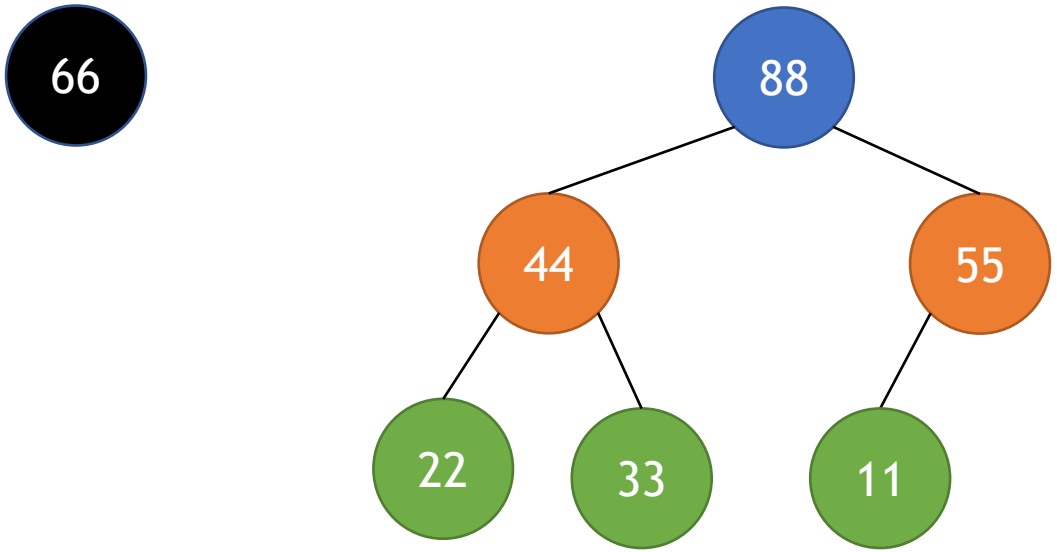


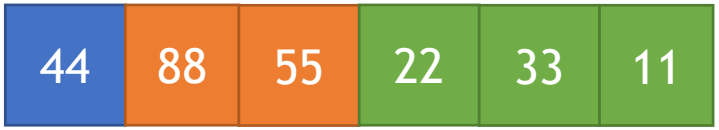
Enfileirar

- Inserir novo elemento após último da fila;
- Incrementar contador de quantidade de elementos;
- Subir novo elemento no heap até chegar na posição correta.

Enfileirar

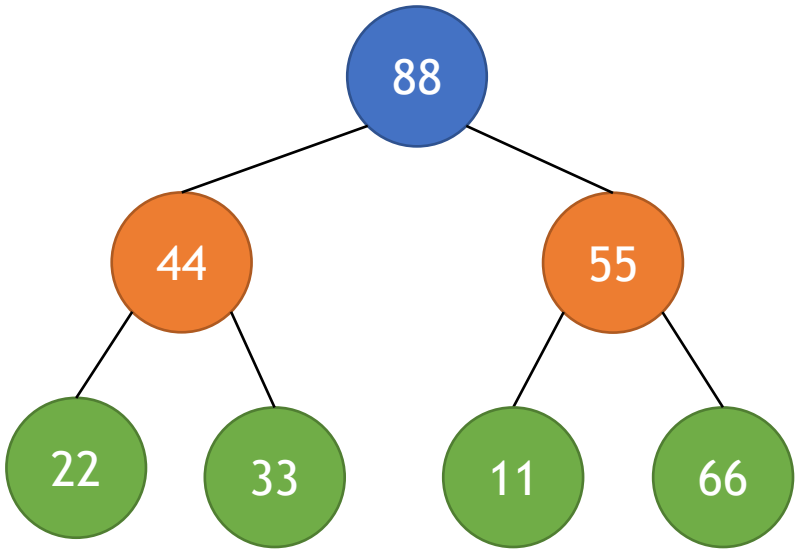
- Inserir novo elemento após último da fila;
- Incrementar contador de quantidade de elementos;
- Subir novo elemento no heap até chegar na posição correta.





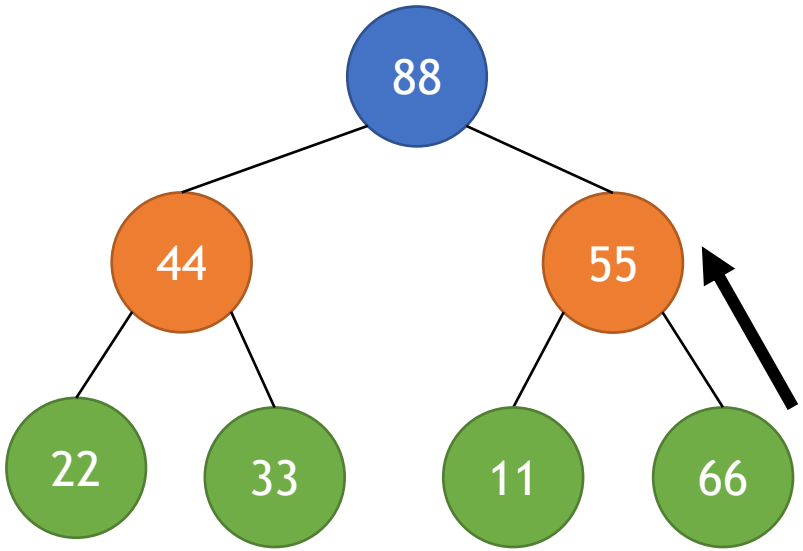
Enfileirar

- Inserir novo elemento após último da fila;
- Incrementar contador de quantidade de elementos;
- Subir novo elemento no heap até chegar na posição correta.



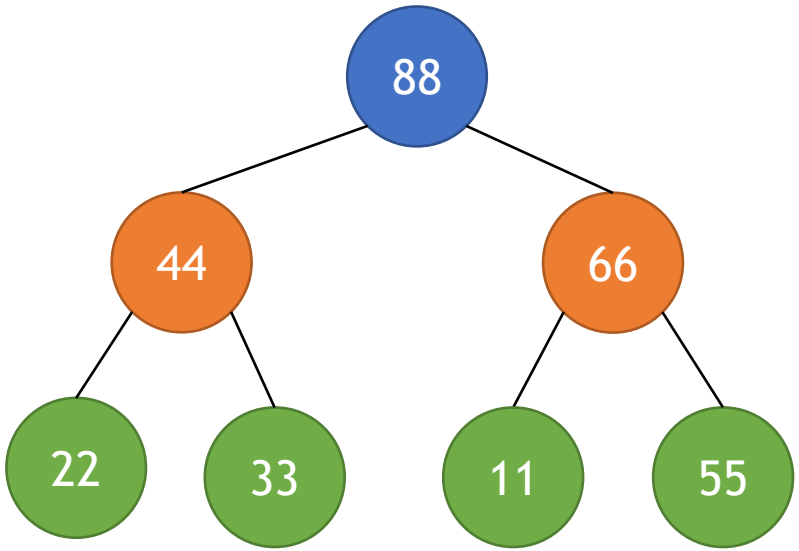
Enfileirar

- Inserir novo elemento após último da fila;
- Incrementar contador de quantidade de elementos;
- Subir novo elemento no heap até chegar na posição correta.



Enfileirar

- Inserir novo elemento após último da fila;
- Incrementar contador de quantidade de elementos;
- Subir novo elemento no heap até chegar na posição correta.



Alterar

- Depende se ocorrerá um aumento ou uma redução da prioridade:
  - **Aumento**: envolve subir elemento no heap até chegar na posição correta;
  - **Redução**: envolve descer elemento no heap até chegar na posição correta.

Exercícios

Exercício 1

- Para os arranjos a seguir, indique quais são heaps de máximo, quais são heaps de mínimo e quais não são nenhum dos dois tipos de heap:
  a) 85 77 84 73 75 66 52 65 62 52 49 47 44 10 50 7 55 23 54 26
  b) 68 20 55 95 9 88 6 13 77 38 31 26 77 63 36 25 20 14 71 35
  c) 10 22 11 40 85 82 79 96 97 86
  d) 96 76 72 28 60 27 60 4 14 8
  e) 10 22 11 90 85 82 79 81 97 86
  f) 5 14 6 22 43 27 46 96 50 94

Exercício 2

- Escreva uma versão sem recursão do algoritmo heap sort.

Exercício 3

- Escreva uma função para remover o maior elemento de um heap de máximo.

Exercício 4

- Escreva uma função para inserir um elemento em um heap de máximo.

Exercício 5

- Uma aplicação importante da estrutura de heap é para *filas de prioridades*. Implemente funções para as operações seguir de uma fila de prioridades para tarefas utilizando um *heap de máximo*:
  - *cria_fila_max*: inicializa uma fila de prioridade;
  - *obter_max*: retorna a próxima tarefa, ou seja, a com maior chave/prioridade na fila;
  - *desenfileirar_max*: remove e retorna a tarefa de maior chave/prioridade da fila;
  - *enfileirar_max*: insere uma nova tarefa na fila, junto com sua chave/prioridade;
  - *libera_fila_max*: libera uma fila de prioridade da memória.

Exercício 5

- As funções possuem o sufixo “max” pois serão usadas para uma *fila de prioridade máxima*, em que o elemento retornado ao desenfileirar é aquele com maior valor de prioridade;
- Cada tarefa possui uma descrição (string – máximo 100 caracteres);
- Além disso, sempre que uma tarefa é inserida, um valor de prioridade deve ser atribuído (será a chave do heap);
- **Sugestão**: defina uma struct para implementar a fila e outra struct para armazenar os dados de cada item do heap.

Referências

- Jayme L. Szwarcfiter, Lilian Markenzon. Estruturas de Dados e Seus Algoritmos. 3ª edição. LTC, 2012.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Algoritmos: Teoria e Prática. Elsevier, 2012.
- Robert Sedgewick, Kevin Wayne. Algorithms 4ª edição. Pearson, 2011.