Certifique-se de que você tentou fazer por um tempo antes de ver as próximas páginas!

8. Mostre como implementar uma fila usando heap. Especificamente, implemente funções ENFILEIRA e DESENFILEIRA considerando que você já possui funções implementadas para heap. Você não deve mexer nas implementações das funções de heap (use-as como caixa preta).

Na fila, os elementos precisam ser armazenados de forma que o DESENFILEIRA devolva o elemento que está armazenado lá mais tempo.
Vamos manter um contador, que deve ser incrementado sempre que um novo elemento for enfileirado e associado a esse elemento (sua prioridade). Esse contador nunca deve ser decrementado, de forma que indicará o instante de tempo que cada elemento chega na estrutura. Com isso, tem prioridade na remoção quem tem o valor de tempo associado menor.
Para aproveitar as estruturas vistos em aula, basta colocar o valor de prioridade negativo.

```
ENFILEIRA(A, x)
1  x.prioridade = -(contador)
2  contador = contador + 1
3  INSERENAHEAP(A, x)
```

Observe que aqui não podemos colocar A.tamanho

```
DESENFILEIRA(A)
1  devolve REMOVEDAHEAP(A)
```

Os dois algoritmos levam tempo $O(\lg n)$, que é o tempo de inserção e remoção em heap.

7. Prove que o algoritmo HEAPSORT está correto, isto é, que ele corretamente ordena qualquer vetor dado na entrada. Para isso, use a frase $P(x)$ = "Antes da x-ésima iteração começar, vale que (a) $i = n - x + 1$, (b) o vetor $A[x+1..n]$ está ordenado de modo não-decrescente e contém os $n-x$ maiores elementos de A; (b) $A.tamanho = x$ e o vetor $A[1..A.tamanho]$ é um heap.". Considere que as funções da estrutura heap estão corretas.

Está resolvido no meu livro.

12. Escreva um algoritmo que ordena uma lista de n itens dividindo-a em três sublistas de aproximadamente $n/3$ itens, ordenando cada sublista recursivamente e intercalando as três sublistas ordenadas.

```
MERGESORT++ ( A, ini, fim )
    se  fim > ini
        parte1 = ⌊(2ini + fim)/3⌋
        parte2 = ⌊(ini + 2fim)/3⌋
        MERGESORT++ ( A, ini, parte1)
        MERGESORT++ ( A, parte1+1, parte2)
        MERGESORT++ ( A, parte2+1, fim)
        COMBINA ( A, ini, parte1, parte2)
        COMBINA ( A, ini, parte2, fim)
```

Vamos provar por indução no tamanho n do vetor que $P(n)$ = "MERGESORT++ (A, ini, fim), com $n = fim - ini + 1$, ordena o vetor $A[ini..fim]$"

BASE: $n \leq 1$. Então $fim - ini + 1 \leq 1$, ou $fim \leq ini$. Nesse caso, o algoritmo não faz nada e, de fato, o vetor já está ordenado.

HIPÓTESE: $P(K)$ vale $\forall\ 1 \leq K < n$.

PASSO: Seja $n > 1$. Então $fim - ini + 1 > 1$, ou $fim > ini$. O algoritmo começa calculando parte1 e parte2 e faz a chamada MERGESORT++ (A, ini, parte1). Note que o vetor $A[ini..parte1]$ tem tamanho

$$\begin{aligned} parte1 - ini + 1 &= \lfloor (2ini + fim)/3 \rfloor - ini + 1 \\ &\leq (2ini + fim)/3 - 3ini/3 + 3/3 \\ &= (fim - ini + 3)/3 = (n+2)/3 \end{aligned}$$

e $(n+2)/3 < n$ sempre que $n > 1$. Logo, por HI, essa chamada ordena $A[ini..parte1]$.

O algoritmo então chama MERGESORT++(A, parte1+1, parte2). Note que o vetor A[parte1+1 .. parte2] tem tamanho

$$\begin{aligned} parte2 - (parte1+1) + 1 &= parte2 - parte1 \\ &= \lfloor (ini + 2fim)/3 \rfloor - \lfloor (2ini + fim)/3 \rfloor \\ &< (ini + 2fim)/3 - ((2ini + fim)/3 - 1) \\ &= \frac{fim}{3} - \frac{ini}{3} + \frac{3}{3} = (n+2)/3 \end{aligned}$$

e $(n+2)/3 < n$ sempre que $n > 1$. Logo, por HI, essa chamada ordena A[parte1+1 .. parte2].

O algoritmo então chama MERGESORT++(A, parte2+1, fim).

O vetor A[parte2 + 1 .. fim] tem tamanho

$$\begin{aligned} fim - (parte2+1) + 1 &= fim - parte2 \\ &= fim - \lfloor (ini + 2fim)/3 \rfloor \\ &< fim - ((ini + 2fim)/3 - 1) \\ &= fim/3 - ini/3 + 3/3 = (n+2)/3. \end{aligned}$$

Novamente, por HI, essa chamada ordena A[parte2 + 1 .. fim].

O próximo comando é COMBINA(A, ini, parte1, parte2). Como A[ini .. parte1] e A[parte1+1 .. parte2] estão ordenados, já sabemos que essa função deixará A[ini .. parte2] ordenado.

O último comando é COMBINA(A, ini, parte2, fim), que também sabemos que ordenará A[ini .. fim].

Portanto P(n) vale.

O tempo T(n) de execução é dado por

$$T(n) = 3T(\tfrac{n}{3}) + \theta(n)$$

Pelo método mestre, com $a = 3$, $b = 3$ e $k = 1$, temos que $T(n)$ é $\theta(n \log n)$.