```
HeapSort (A[1..n])
1  Constrói-heap (A[1..n])
2  m ← n
3  Enquanto m > 1
4      A[1] ↔ A[m]
5      m ← m - 1
6      Descer (1, m)
```

Invariante: $A[1..m]$ é heap de máximo contendo os m menores elementos de A e $A[m+1..n]$ contém os $n-m$ maiores elementos de A ordenados.

Inicialização: Antes do início da execução do laço, $m = n$. $A[1..n]$ é heap de máximo pela execução do procedimento da linha 1. $A[1..n]$ contém todos os elementos de A e portanto os n menores. $A[m+1..n]$ é o vetor vazio, e portanto o restante do invariante é válido por vacuidade.

```
HeapSort (A[1..n])
1   Constrói-heap (A[1..n])
2   m ← n
3   Enquanto m > 1
4       A[1] <-> A[m]
5       m ← m-1
6       Descer (1, m)
```

Invariante: $A[1..m]$ é heap de mínimo i contendo os m menores elementos de A e $A[m+1..n]$ contém os $n-m$ maiores i elementos de A ordenados.

Manutenção: Suponha que a invariante é válida para $m = i$.
Na linha 4, o elemento da raiz é trocado com o elemento da última posição. Como o elemento da raiz era o maior em $A[1..i]$, e os elementos de $A[i+1..n]$ eram todos maiores que ele (pela invariante), ao colocar esse elemento na posição i, temos $A[i..n]$ ordenado contendo os $n-i+1$ maiores elementos de A.

```
HeapSort (A[1..n])
1   Constrói-heap (A[1..n])
2   m ← n
3   Enquanto m > 1
4       A[1] <-> A[m]
5       m ← m-1
6       Descer (1, m)
```

Invariante: $A[1..m]$ é heap de máximo contendo os m menores elementos de A e $A[m+1..n]$ contém os $n-m$ maiores elementos de A ordenados.

Na linha 5, $m \leftarrow i-1$.
Então a segunda parte do invariante fica válida para o novo valor de m.
Na linha 6, passamos como parâmetro para a rotina Descer os valores $(1, i-1)$
↳ valor de m atualizado.
Assumindo que Descer consertа o heap corretamente, depois da sua execução, $A[1..i-1] = A[1..m]$ é um heap de máximo. Como os $i-1$ menores elem. de A...

Término (do laço):

Inicialmente, m possui valor n. Dentro do laço, m é sempre decrementado de 1 unidade e necessariamente vai atingir o valor 1 em algum momento, e o laço para de ser executado.

```
HeapSort (A[1..n])
1  Constrói-heap (A[1..n])
2  m ← n
3  Enquanto m > 1
4      A[1] ↔ A[m]
5      m ← m-1
6      Descer (1, m)
```

$A[2..n]$ possui os $n-1$ maiores elementos de A ordenados.

Logo, o vetor inteiro está ordenado. ▣

Corretude do algoritmo:

O laço para quando $m=1$.

Pela invariante, $A[1]$ possui o menor elemento de A e