Aplique manualmente o Heap Sort no vetor V = {5, 13, 2, 25, 7, 17, 20, 8, 4}

```
[ 0,  1,  2,  3,  4,  5,  6,  7,  8] Posições no vetor
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Encontrando os pais (n/2 - 1)
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Começando o Heapify pelo 25
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Encontrando seus filhos 2 * i + 1 e 2 * i + 2
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Vendo se o filho da esquerda é maior que o pai
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Vendo se o filho da direita é maior que o pai
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Nenhum filho é maior que o pai, continua a execução
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Executando o Heapify no 2
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Encontrando seus filhos 2 * i + 1 e 2 * i + 2
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Vendo se o filho da esquerda é maior que o pai
[ 5, 13,  2, 25,  7, 17, 20,  8,  4] Vendo se o filho da direita é maior que o da esquerda
[ 5, 13, 20, 25,  7, 17,  2,  8,  4] Troca o filho da direita com o pai
[ 5, 13, 20, 25,  7, 17,  2,  8,  4] Executa Heapify no 2 em sua nova posição (sem filhos)
[ 5, 13, 20, 25,  7, 17,  2,  8,  4] Executa Heapify no 13
[ 5, 13, 20, 25,  7, 17,  2,  8,  4] Encontrando seus filhos
[ 5, 13, 20, 25,  7, 17,  2,  8,  4] Vendo se o filho da esquerda é maior que o pai
[ 5, 13, 20, 25,  7, 17,  2,  8,  4] Vendo se o filho da direita é maior que o da esquerda
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Troca o filho da esquerda com o pai
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Executa o Heapify no 13 em sua nova posição
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Encontrando seus filhos
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Vendo se o filho da esquerda é maior que o pai
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Vendo se o filho da direita é maior que o pai
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Nenhum filho é maior que o pai, continua a execução
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Executa Heapify no 5
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Encontrando seus filhos
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Vendo se o filho da esquerda é maior que o pai
[ 5, 25, 20, 13,  7, 17,  2,  8,  4] Vendo se o filho da direita é maior que o da esquerda
[25,  5, 20, 13,  7, 17,  2,  8,  4] Troca o filho da esquerda com o pai
[25,  5, 20, 13,  7, 17,  2,  8,  4] Executa o Heapify no 5 em sua nova posição
[25,  5, 20, 13,  7, 17,  2,  8,  4] Encontrando seus filhos
[25,  5, 20, 13,  7, 17,  2,  8,  4] Vendo se o filho da esquerda é maior que o pai
[25,  5, 20, 13,  7, 17,  2,  8,  4] Vendo se o filho da direita é maior que o da esquerda
[25, 13, 20,  5,  7, 17,  2,  8,  4] Troca o filho da esquerda com o pai
[25, 13, 20,  5,  7, 17,  2,  8,  4] Executa novamente o Heapify no 5
[25, 13, 20,  5,  7, 17,  2,  8,  4] Encontra seus filhos
[25, 13, 20,  5,  7, 17,  2,  8,  4] Vendo se o filho da esquerda é maior que o pai
[25, 13, 20,  5,  7, 17,  2,  8,  4] Vendo se o filho da direita é maior que o da esquerda
[25, 13, 20,  8,  7, 17,  2,  5,  4] Troca o filho da esquerda com o pai
[25, 13, 20,  8,  7, 17,  2,  5,  4] Executa Heapify no 5 em sua nova posição (sem filhos)

[25, 13, 20,  8,  7, 17,  2,  5,  4] Agora o vetor tem a propriedade de Heap, Heap Sort:

[ 4, 13, 20,  8,  7, 17,  2,  5, 25] Trocamos o primeiro com o último, ajustando o 25
[ 4, 13, 20,  8,  7, 17,  2,  5, 25] Executamos o Heapify no 4
[ 4, 13, 20,  8,  7, 17,  2,  5, 25] Encontrando seus filhos
[ 4, 13, 20,  8,  7, 17,  2,  5, 25] Vendo se o filho da esquerda é maior que o pai
[ 4, 13, 20,  8,  7, 17,  2,  5, 25] Vendo se o filho da direita é maior que o da esquerda
[20, 13,  4,  8,  7, 17,  2,  5, 25] Trocamos o da direita com o pai
[20, 13,  4,  8,  7, 17,  2,  5, 25] Executa o Heapify no 4 em sua nova posição
[20, 13,  4,  8,  7, 17,  2,  5, 25] Encontra seus filhos
[20, 13,  4,  8,  7, 17,  2,  5, 25] Vendo se o filho da esquerda é maior que o pai
[20, 13,  4,  8,  7, 17,  2,  5, 25] Vendo se o filho da direita é maior que o da esquerda
[20, 13, 17,  8,  7,  4,  2,  5, 25] Trocamos o da esquerda com o pai
[20, 13, 17,  8,  7,  4,  2,  5, 25] Executa Heapify no 4 em sua nova posição (sem filhos)
[ 5, 13, 17,  8,  7,  4,  2, 20, 25] Trocamos o primeiro com o penúltimo, ajustando o 20
[ 5, 13, 17,  8,  7,  4,  2, 20, 25] Executamos o Heapify no 5
[ 5, 13, 17,  8,  7,  4,  2, 20, 25] Encontrando seus filhos
[ 5, 13, 17,  8,  7,  4,  2, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 5, 13, 17,  8,  7,  4,  2, 20, 25] Vendo se o filho da direita é maior que o da esquerda
[17, 13,  5,  8,  7,  4,  2, 20, 25] Trocamos o da direita com o pai
[17, 13,  5,  8,  7,  4,  2, 20, 25] Executa o Heapify no 5 em sua nova posição
[17, 13,  5,  8,  7,  4,  2, 20, 25] Encontra seus filhos
[17, 13,  5,  8,  7,  4,  2, 20, 25] Vendo se o filho da esquerda é maior que o pai
[17, 13,  5,  8,  7,  4,  2, 20, 25] Vendo se o filho da direita é maior que o pai
[17, 13,  5,  8,  7,  4,  2, 20, 25] Nenhum filho é maior que o pai, continua a execução
```

```
[ 2, 13,  5,  8,  7,  4, 17, 20, 25] Trocamos o primeiro com o próximo, ajustando o 17
[ 2, 13,  5,  8,  7,  4, 17, 20, 25] Executamos o Heapify no 2
[ 2, 13,  5,  8,  7,  4, 17, 20, 25] Encontra seus filhos
[ 2, 13,  5,  8,  7,  4, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 2, 13,  5,  8,  7,  4, 17, 20, 25] Vendo se o filho da direita é maior que o da esquerda
[13,  2,  5,  8,  7,  4, 17, 20, 25] Trocamos o da esquerda com o pai
[13,  2,  5,  8,  7,  4, 17, 20, 25] Executamos o Heapify no 2 novamente
[13,  2,  5,  8,  7,  4, 17, 20, 25] Encontramos seus filhos
[13,  2,  5,  8,  7,  4, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[13,  2,  5,  8,  7,  4, 17, 20, 25] Vendo se o filho da direita é maior que o da esquerda
[13,  8,  5,  2,  7,  4, 17, 20, 25] Trocamos o da esquerda com o pai
[13,  8,  5,  2,  7,  4, 17, 20, 25] Executa Heapify no 2 em sua nova posição (sem filhos)
[ 4,  8,  5,  2,  7, 13, 17, 20, 25] Trocamos o primeiro com o próximo, ajustando o 13
[ 4,  8,  5,  2,  7, 13, 17, 20, 25] Executamos o Heapify no 4
[ 4,  8,  5,  2,  7, 13, 17, 20, 25] Encontramos seus filhos
[ 4,  8,  5,  2,  7, 13, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 4,  8,  5,  2,  7, 13, 17, 20, 25] Vendo se o filho da direita é maior que o da esquerda
[ 8,  4,  5,  2,  7, 13, 17, 20, 25] Trocamos o da esquerda com o pai
[ 8,  4,  5,  2,  7, 13, 17, 20, 25] Executamos o Heapify no 4 em sua nova posição
[ 8,  4,  5,  2,  7, 13, 17, 20, 25] Encontramos seus filhos
[ 8,  4,  5,  2,  7, 13, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 8,  4,  5,  2,  7, 13, 17, 20, 25] Vendo se o filho da direita é maior que o pai
[ 8,  7,  5,  2,  4, 13, 17, 20, 25] Trocamos o da direita com o pai
[ 8,  7,  5,  2,  4, 13, 17, 20, 25] Executa Heapify no 4 em sua nova posição (sem filhos)
[ 4,  7,  5,  2,  8, 13, 17, 20, 25] Trocamos o primeiro com o próximo, ajustando o 8
[ 4,  7,  5,  2,  8, 13, 17, 20, 25] Executamos o Heapify no 4
[ 4,  7,  5,  2,  8, 13, 17, 20, 25] Encontramos seus filhos
[ 4,  7,  5,  2,  8, 13, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 4,  7,  5,  2,  8, 13, 17, 20, 25] Vendo se o filho da direita é maior que o da esquerda
[ 7,  4,  5,  2,  8, 13, 17, 20, 25] Trocamos o da esquerda com o pai
[ 7,  4,  5,  2,  8, 13, 17, 20, 25] Executamos o Heapify no 4 em sua nova posição
[ 7,  4,  5,  2,  8, 13, 17, 20, 25] Encontramos seus filhos
[ 7,  4,  5,  2,  8, 13, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 7,  4,  5,  2,  8, 13, 17, 20, 25] Nenhum filho é maior que o pai, continua a execução
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Trocamos o primeiro com o próximo, ajustando o 7
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Executamos Heapify no 2
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Encontramos seus filhos
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Vendo se o filho da direita é maior que o da esquerda
[ 5,  4,  2,  7,  8, 13, 17, 20, 25] Trocamos o da direita com o pai
[ 5,  4,  2,  7,  8, 13, 17, 20, 25] Executa Heapify no 2 em sua nova posição (sem filhos)
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Trocamos o primeiro com o próximo, ajustando o 5
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Executamos Heapify no 2
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Encontramos seus filhos
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Vendo se o filho da esquerda é maior que o pai
[ 4,  2,  5,  7,  8, 13, 17, 20, 25] Trocamos o da esquerda com o pai
[ 4,  2,  5,  7,  8, 13, 17, 20, 25] Executa Heapify no 2 em sua nova posição (sem filhos)
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Trocamos o primeiro com o próximo, ajustando o 4
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Executa Heapify no 2 em sua nova posição (sem filhos)
[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Ele é o último, então está na posição correta

[ 2,  4,  5,  7,  8, 13, 17, 20, 25] Assim termina a execução
```