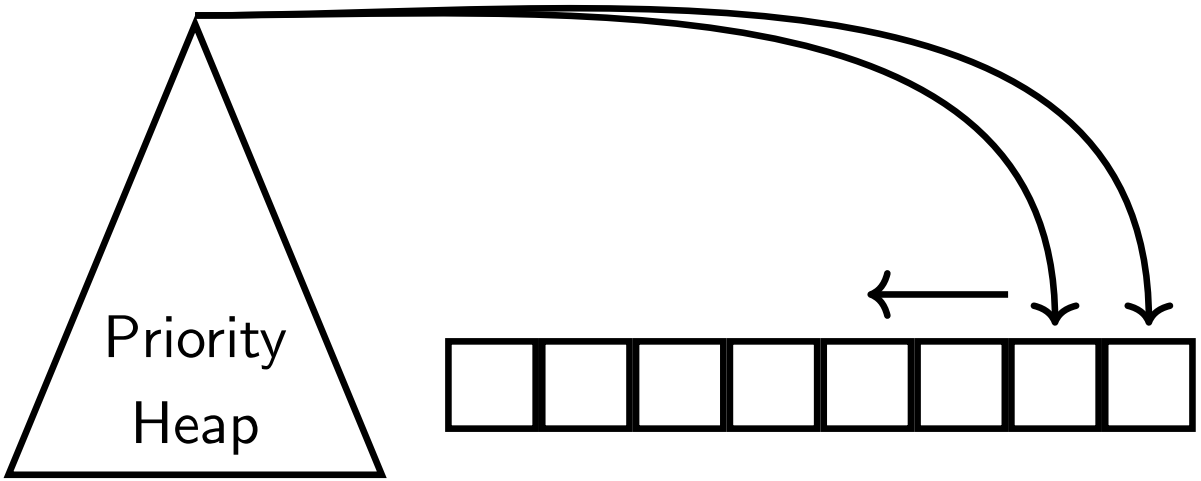
Heap Sort (Selection)

Heap Sort

A priority heap structure allows efficient selecting and removal of the highest priority (i.e. largest value) from a collection of values. Heap sort uses a priority heap to sort all the values by first constructing a priority heap with the values to be sorted and then, repetitively removing the largest value from the heap and filling it in to the output array starting at the end of the array and working backwards towards the start of the array.



Heap Sort

If we use the standard array implementation of the Binary Heap, then every time we remove the highest priority element from the heap, we reduce the size of the heap by one and we are therefore using less of the array to hold the heap values. We can therefore use the *SAME* array to put the sorted output values into, thus avoiding the need to use a separate extra array. Of course, the final sorted elements will be in index locations 1 to n of the array, instead of 0 to $n-1$:

```
1 heapSort(array a, int n) {
2     heapify(a,n)
3     for( j = n ; j > 1 ; j-- ) {
4         swap a[1] and a[j]
5         bubbleDown(1,a,j-1)
6     }
7 }
```

Heap Sort Complexity and Stability

Heapify is $O(n)$. Then we have to do n bubble down operations, each of complexity $O(\log n)$ which gives a cost of $O(n \log n)$ in total.

Because of the reordering in the bubble down operations, this sort is *unstable*