Naive Heapsort with a **Max-Heap**

Demo

Idea. Instead of rescanning entire array for min, maintain a heap so that finding min is fast!

Heapsort. Selection sort with a **max-oriented heap**–neat trick for saving memory later.

1. **O(1)**. Create output array.
2. **O(N log N)**. Insert all items into a new max-heap (separate array for the heap).
3. Repeat N times:
    a. **O(log N)**. Delete max item from the max-heap.
    b. **O(1)**. Put max item at the end of the unused part of the output array.

O(N log N) time complexity.

Θ(N) space complexity to create Θ(N) separate array heap and Θ(N) output array.

In-place Heapsort Runtime

Idea. Save ~2N memory by treating the input array as a heap. Avoid extra copies of data.

Bottom-up heapification. Efficient heap construction by sinking nodes in reverse level order.

Once heap-ified, algorithm is almost the same as naive heap sort.

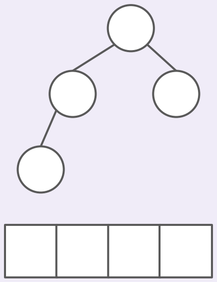
1. **Bottom-up heapification**.
2. Repeat N times:
    a. **O(log N)**. Delete max item from the max-heap.
    b. **O(1)**. Put max item at the end the array.

Give the tight asymptotic time complexity of in-place heapsort in big-O notation.

Heapsort Stability

1. **Bottom-up heapification**.
2. Repeat N times:
    a. Delete max item from the max-heap.
    b. Put max item at the end the array.

Heapsort is not stable. Give an example.



Selection Sort Stability

Selection sort. Repeatedly select the smallest remaining item and swap it to its proper index.

1. Find the smallest item in the array, and swap it with the first item.
2. Find the next smallest item in the array, and swap it with the next item.
3. Continue until all items in the array are sorted.

Selection sort is not stable. Give an example.

Insertion Sort

Demo

Unstable sorting algorithms (heapsort, selection sort) use long-distance swaps.

Merge sort, a stable sort, uses the fact that left-half items come before right-half items.

Idea. Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Insertion sort. Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Purple	Item that we're swapping left.
Black	Item swapped with purple item.
Grey	Not considered this iteration.

```
P O T A T O
P O T A T O  (0 swaps)
O P T A T O  (1 swap )
O P T A T O  (0 swaps)
A O P T T O  (3 swaps)
A O P T T O  (0 swaps)
A O O P T T  (3 swaps)
```

```
S O R T E X A M P L E
S O R T E X A M P L E  (0 swaps)
O S R T E X A M P L E  (1 swap )
O R S T E X A M P L E  (1 swap )
O R S T E X A M P L E  (0 swaps)
E O R S T X A M P L E  (4 swaps)
E O R S T X A M P L E  (0 swaps)
A E O R S T X M P L E  (6 swaps)
A E M O R S T X P L E  (5 swaps)
A E M O P R S T X L E  (4 swaps)
A E L M O P R S T X E  (7 swaps)
A E E L M O P R S T X  (8 swaps)
```

Insertion Sort Examples

Algorithms (Robert Sedgewick, Kevin Wayne/Princeton)

Quantifying Sortedness

Inversion. A pair of keys that are out of order.

A E E L M O T R X P S

1	2	3	4	5	6
T–R	T–P	T–S	R–P	X–P	X–S

Partially sorted. An array of length N is partially sorted if the number of inversions is in O(N).

- Sorted array has 0 inversions.
- Sorted subarray with 10 random items at the end has at most 10N + 45 inversions.

Each insertion sort local swap fixes 1 inversion. Θ(N + K) where K is number of inversions.

COS 226: Elementary Sorts (Robert Sedgewick, Kevin Wayne/Princeton)

Sort	Best-Case	Worst-Case	Space	Stable	Notes
Selection Sort	$\Theta(N^2)$	$\Theta(N^2)$	Θ(1)	No	
Heapsort	Θ(N)	Θ(N log N)	Θ(1)	No	**Slow in practice.**
Merge sort	Θ(N log N)	Θ(N log N)	Θ(N)	Yes	**Fastest stable sort.**
Insertion Sort	Θ(N)	$\Theta(N^2)$	Θ(1)	Yes	**Best for small or almost sorted inputs.**