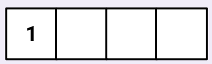
## Selection Sort Stability

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

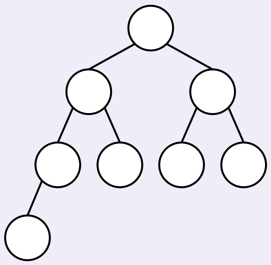
**Selection sort is not stable. Give an example.**






## Heapifying

Give the result of bottom-up max-heapifying the array [9, 1, 1, 3, 5, 5, 6, 8].





**?**: What does it mean to have a stable sorting algorithm?

**Q1**: Selection sort is not stable. Give an example.

**Q1**: Give the result of bottom-up max-heapifying the array [9, 1, 1, 3, 5, 5, 6, 8].

## Q Recursive Merge Sorting

Give the two arrays that will be merged by the final step of merge sort on [9, 1, 1, 3, 5, 5, 6, 8].



## Insertion Sort

32,15,2,17,19,26,41,17,17

Unstable sorting algorithms (selection sort, heapsort) 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 subsection but use only left-neighbor swaps to maintain 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.



**Q1**: Give the two arrays that will be merged by the final step of merge sort on [9, 1, 1, 3, 5, 5, 6, 8].

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





## Inversions: 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).

- A sorted array has 0 inversions.
- A reverse-sorted array has about $\frac{1}{2}N^2$ inversions.

**Each local swap fixes 1 inversion. Insertion sort runtime is in Θ(N + K) with K inversions.**

## Q Worst-Case Inversions

If we add 10 unknown items to the end of a sorted array, at most how many inversions can there be in the array?

11

**Q1**: If we add 10 unknown items to the end of a sorted array, at most how many inversions can there be in the array?

## Q Monotonically-Improving

For each algorithm, can the inversion count increase at some point in its execution?

**Insertion sort**.

**Selection sort**.

**Bottom-up max-heapification**.

**Bottom-up min-heapification**.

**Merge sort**.

13

For each algorithm, can the inversion count increase at some point in its execution?

**Q1**: Insertion sort

**Q2**: Selection sort

**Q3**: Bottom-up max-heapification

**Q4**: Bottom-up min-heapification

**Q5**: Merge sort