Sorting

What sorting algorithms have you seen before?

If I gave you a deck of cards and asked you to sort it, how would you do it?

Sorting algorithms

Adaptive heapsort  Comb sort  Pancake sort
Bitonic sorter  Flashsort  Quicksort
Block sort  Gnome sort  Radix sort
Bubble sort  Heapsort  Selection sort
Bucket sort  Insertion sort  Shell sort
Cascade mergesort  Library sort  Spaghetti sort
Cocktail sort  Mergesort  Treesort
Selection sort

Divide the data into two parts: sorted and unsorted

Repeat:

```
Find the smallest element in the unsorted part
Swap it with the leftmost element of the unsorted array
The sorted array is now one element larger
```

3 44 38 5 47 1 36 26

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:

- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

3 44 38 5 47 1 36 26

Smallest?
Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

1 44 38 5 47 3 36 26

Sorted  Unsorted

9

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

1 44 38 5 47 3 36 26

Sorted  Unsorted

10

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

1 44 38 5 47 3 36 26

Sorted  Unsorted

11

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

1 44 38 5 47 3 36 26

Sorted  Unsorted

12
Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

13

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

14

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

15

Selection sort

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

16
### Running time to find the smallest element

<table>
<thead>
<tr>
<th>Best case?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst case?</td>
</tr>
<tr>
<td>Average case?</td>
</tr>
</tbody>
</table>

Divide the array into two parts: a sorted part on the left and an unsorted part on the right.

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

### All cases: size_of_unsorted_array – we have to search through the entire unsorted array to find it

Divide the array into two parts: a sorted part on the left and an unsorted part on the right.

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

### Overall runtime

3 44 38 5 47 1 36 26

- size_of_unsorted_array
- How big is this for the first iteration?

### Overall runtime

3 44 38 5 47 1 36 26

- size_of_unsorted_array
- How big is this for the first iteration? n
Overall runtime

1  4  4  3  8  5  4  7  3  6  2  6

sorted  unsorted

size_of_unsorted_array

How big is this for the second iteration?

Overall runtime

1  4  4  3  8  5  4  7  3  6  2  6

sorted  unsorted

size_of_unsorted_array

How big is this for the second iteration?  n-1

Overall runtime

1  3  5  3  8  4  7   4  4  3  6  2  6

sorted  unsorted

size_of_unsorted_array

How big is this for the second iteration?  n-2

Overall runtime

1  3  5  3  8  4  7   4  4  3  6  2  6

sorted  unsorted

size_of_unsorted_array

How big is this for the second iteration?  n-2
Divide the array into two parts: a sorted part on the left and an unsorted part on the right.

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

Overall runtime

\[
\text{runtime} = 1 + 2 + 3 + n - 2 + n - 1 + n = \sum_{i=1}^{n} i
\]
Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

Overall runtime

$$\text{runtime} = \sum_{i=1}^{n} \frac{n(n+1)}{2}$$

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

Overall runtime

$$\text{runtime} = \sum_{i=1}^{n} \frac{n(n+1)}{2} \in O(?)$$

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

Selection sort: overall runtime

$$\text{runtime} = \sum_{i=1}^{n} \frac{n(n+1)}{2} \in O(n^2)$$

Divide the array into two parts: a sorted part on the left and an unsorted part on the right

Repeat:
- Find the smallest element in the unsorted part
- Swap it with the leftmost element of the unsorted array
- The sorted array is now one element larger

Insertion sort

Divide the array into two parts:
- left part: left elements in sorted order
- right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger
Insertion sort

Divide the array into two parts:
- left part: left elements in sorted order
- right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

3 44 38 5 47 1 36 26

sorted  unsorted

3 38 44 5 47 1 36 26

sorted  unsorted

3 38 44 5 47 1 36 26

sorted  unsorted
Insertion sort

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

3 38 44 5 47 1 36 26

sorted  unsorted

Is 5 in the correct spot?

38

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

3 38 5 44 47 1 36 26

sorted  unsorted

39

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

3 38 5 44 47 1 36 26

sorted  unsorted

40
Insertion sort

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

Is 5 in the correct spot?

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

Was that fast or slow?
Insertion sort

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

1  3   5  3844 4736 26

Was that fast or slow?

Running time to find the correct spot

Best case?

Worst case?

Average case?

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

Best case: O(1), it's larger than any element to the left

Worst case: size_sorted_part, it's smaller than any element to the left

Average cases size_sorted_part/2

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger
Insertion sort: overall runtime

- Best case? When does this happen?
- Worst case? When does this happen?
- Average case?

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger

Overall runtime

- Best case: O(n), the array is already sorted
- Worst case: O(n^2), the array is reverse sorted (same sum as before)
- Average case: O(n^2), n iterations and still have to move n/2 entries on average

Divide the array into two parts:
left part: left elements in sorted order
right part: right elements in unsorted order

Repeat:
- Look at the next element in the unsorted part
- Find the correct location in the sorted part (by sliding each item right one at a time)
- The sorted array is now one element larger