SORTING BASICS

David Kauchak
CS 62 – Spring 2021

1

Admin

Erin's office hours today 7-8pm (going forward)

My office hours today 3:30-4pm

Compression assignment hints/observations

2

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?

3

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 mergsort	Library sort	Spaghetti sort
Cocktail sort	**Mergesort**	**Treesort**

4

Selection sort

Divide the data into two parts: sorted and unsorted

Repeat:

5

Selection sort

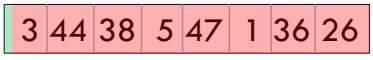
Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

6

Selection sort



sorted unsorted

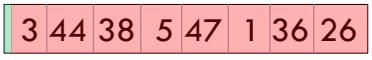
Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

7

Selection sort



sorted unsorted

Smallest?

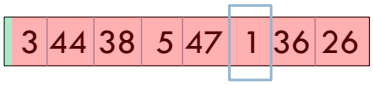
Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

8

Selection sort

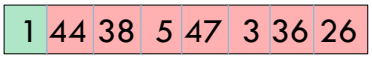


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

9

Selection sort

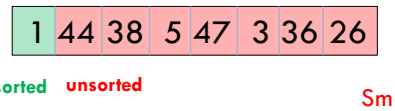


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

10

Selection sort

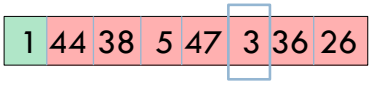


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

11

Selection sort

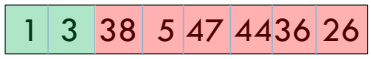


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

12

Selection sort

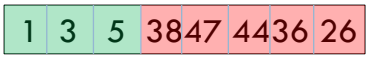


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

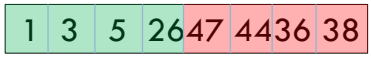


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

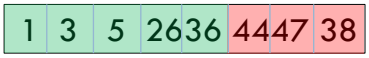


Divide the array into two parts: a sorted part on the left and an unsorted part on the right

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

Best case?

Worst case?

Average case?

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

17

Running time to find the smallest element

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

18

Overall runtime

3	44	38	5	47	1	36	26

sorted **unsorted**

size_of_unsorted_array

How big is this for the first iteration?

19

Overall runtime

3	44	38	5	47	1	36	26

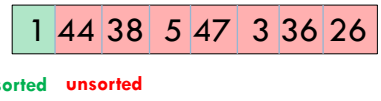
sorted **unsorted**

size_of_unsorted_array

How big is this for the first iteration? n

20

Overall runtime

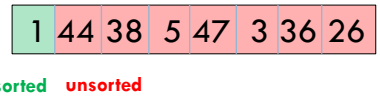


size_of_unsorted_array

How big is this for the second iteration?

21

Overall runtime

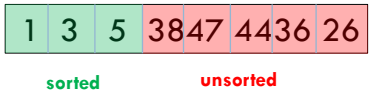


size_of_unsorted_array

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

22

Overall runtime

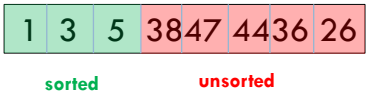


size_of_unsorted_array

How big is this for the second iteration?

23

Overall runtime

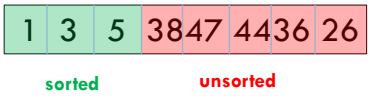


size_of_unsorted_array

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

24

Overall runtime

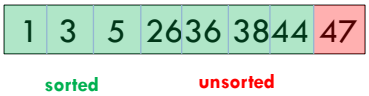


size_of_unsorted_array

How big is this for the last iteration?

25

Overall runtime



size_of_unsorted_array

How big is this for the last iteration? 1

26

Overall runtime

$$runtime = 1 + 2 + 3 + n - 2 + n - 1 + n = \sum_{i=1}^{n} i$$

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

27

Overall runtime

$$runtime = \sum_{i=1}^{n} i$$?

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

28

Overall runtime

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

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

29

Overall runtime

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

30

Selection sort: overall runtime

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

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

31

Insertion sort

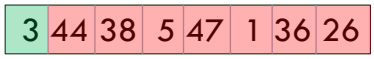
3	44	38	5	47	1	36	26

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

32

Insertion sort



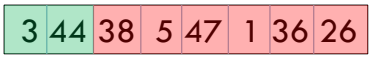
sorted unsorted

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

33

Insertion sort



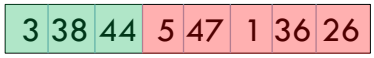
sorted unsorted

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

34

Insertion sort



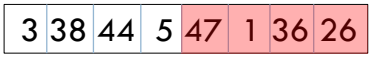
sorted unsorted

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

35

Insertion sort



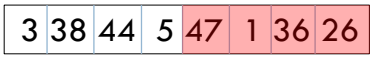
sorted unsorted

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

36

Insertion sort



sorted unsorted

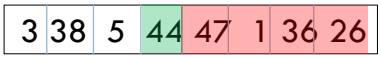
Is 5 in the correct spot?

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

37

Insertion sort



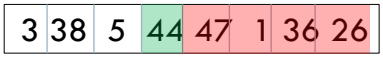
sorted unsorted

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

38

Insertion sort



sorted unsorted

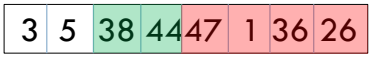
Is 5 in the correct spot?

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

39

Insertion sort



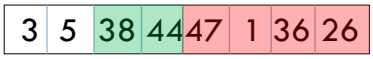
sorted unsorted

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

40

Insertion sort



sorted unsorted

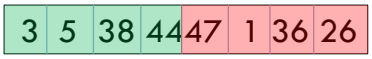
Is 5 in the correct spot?

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

41

Insertion sort



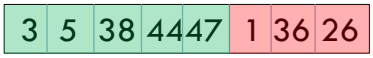
sorted unsorted

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

42

Insertion sort



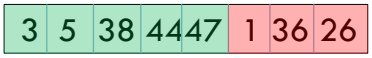
sorted unsorted

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

43

Insertion sort



sorted unsorted

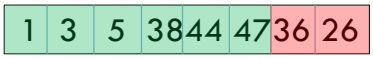
Was that fast or slow?

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

44

Insertion sort



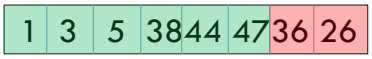
sorted unsorted

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

45

Insertion sort



sorted unsorted

Was that fast or slow?

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

46

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

47

Running time to find the correct spot

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 case: 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

48

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

49

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

50