Analysis of Algorithms: Sorting algorithms (Bubble sort and Quicksort)

Garrett Dancik, PhD

Fall 2021

Course Notes: https://gdancik.github.io

What do we mean by Sorting?

- One of the most common operations in computer science is to sort data numerically or alphabetically
- We have seen previously that sorted data can be searched much more efficiently than unsorted data. Why?
- In addition, for presentation purposes, elements such as names, states, ages, GPAs, etc, are often displayed in sorted order (numeric data may be sorted from low to high or high to low; when we say that numeric data is sorted we will mean low to high)

11	21	18	3	15	19

- The list above in sorted order is: 3, 11, 15, 18, 19, and 21

Bubble sort

- Find the maximum element in the list (all *n* elements)
  - Swap this maximum element with the last element in the list
- Find the maximum element in the list (first *n* – 1 elements)
  - Swap this maximum element with the second to last element in the list
- Find the maximum element in the list (first *n* – 2 elements)
  - Swap this maximum element with the third to last element in the list
- This process repeats until we are down to the first element. This is the minimum element, which is now the first element in the list

11	21	18	3	15	19

Bubble sort (example)

- We search all $n = 6$ elements for the maximum, and swap this maximum element with the last one in the list (the 6^{th} one)

The max is 21 → swap this with the last element

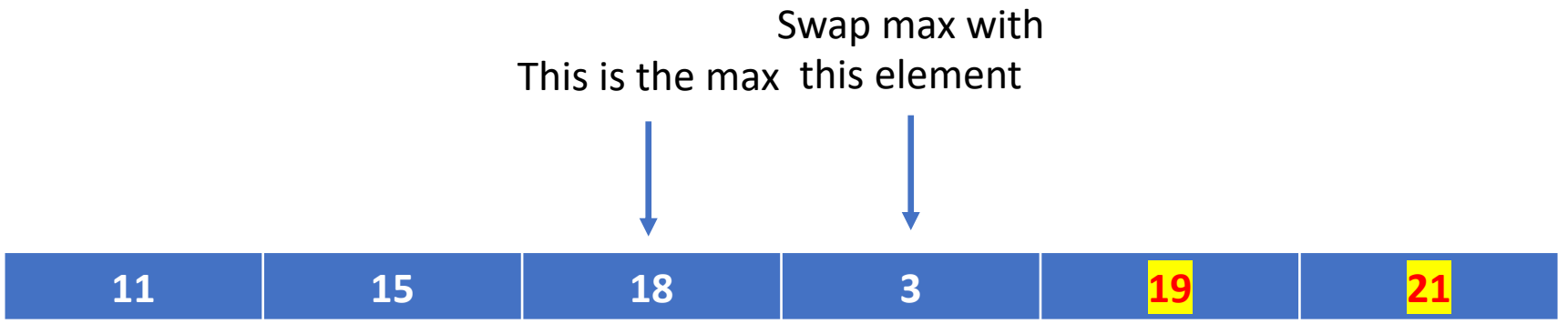
Bubble sort (example)

- We search the first *n-1* = 5 elements for the maximum, and swap this maximum element with the 5th one (or the second to last one)

The max is 19 → swap this with the 5th element

Bubble sort (example)

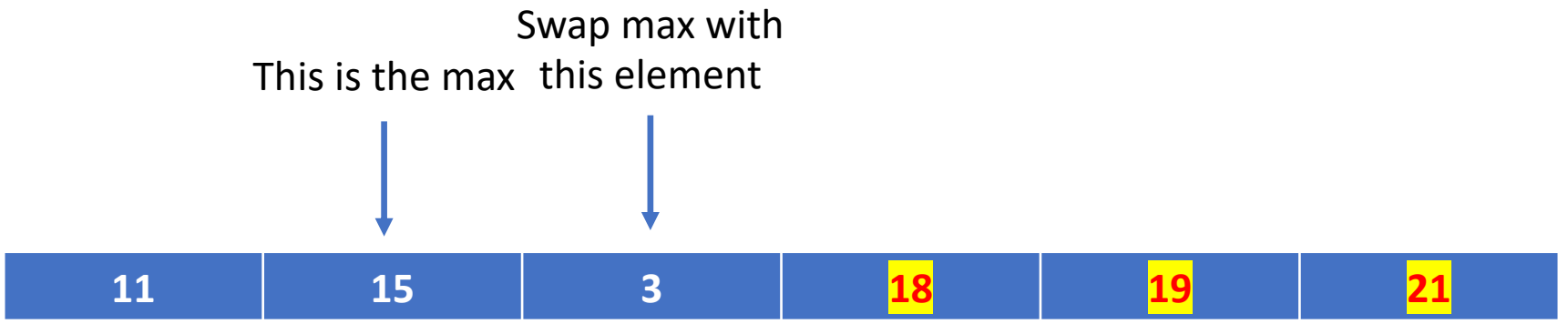
- We search the first *n-2* = 4 elements for the maximum, and swap this maximum element with the 4th one



The max is 18 → swap this with the 4th element

Bubble sort (example)

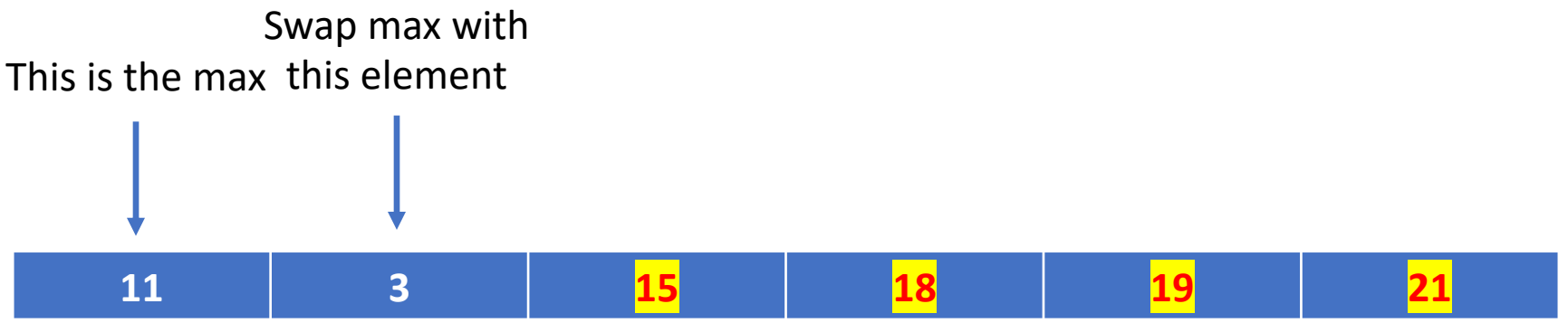
- We search the first n – 3 = 3 elements for the maximum, and swap this maximum element with the 3rd one



The max is 15 → swap this with the 3rd element

Bubble sort (example)

- We search the first n – 4 = 2 elements for the maximum, and swap the maximum element with the 2nd one



The max is 11 → swap this with the 2nd element

Bubble sort (example)

- Once we have only 1 element left (we are finding the max of just the 1st element), then we are done. The list is now sorted.

Bubble sort algorithm

- *end* = n – 1
- while *end* > 0 :
  - Set *max_index* to the index of the maximum element between *values[0]* through *values[end]*
  - Swap *values[max_index]* and *values[end]*
  - Set *end* = *end* - 1

Bubble sort algorithm

- *end* = n – 1
- while *end* > 0 :
  - Set *max_index* to the index of the maximum element between *values[0]* through *values[end]*
  - Swap *values[max_index]* and *values[end]*
  - Set *end* = *end* - 1

    - Set *max_index* to 0
    - Set *i* to 0
    - While *i* <= end:
      - If values[i] > values[max_index]*:*
        - set *max_index* to *I*
      - set *i* = *i +1*

Bubble sort algorithm

- *end* = n – 1
- while *end* > 0 :
  - Set *max_index* to 0
  - Set *i* to 0
  - While *i* <= end:
    - If values[*i*] > values[*max_index*]*:*
      - set *max_index* to *I*
    - Set *i* = *i* + 1
  - Swap *values[max_index]* and *values[end]*
  - Set *end* = *end* - 1

Executed $n - 1$ times (while loop)

Executed up to $n - 1$ times each time (while loop)

This suggests an order of magnitude of n^2

Bubble sort algorithm

- *end* = n – 1
- while *end* > 0 :
  - Set *max_index* to 0
  - Set *i* to 0
  - While *i* <= end:
    - If values[*i*] > values[*max_index*]*:*
      - set *max_index* to *I*
    - Set *i* = *i* + 1
  - Swap *values[max_index]* and *values[end]*
  - Set *end* = *end* - 1

Assume that *n* = 4

end	# iterations of inner while loop
3	4
2	3
1	2
0	-

In general, for a list of size *n*, the total number of inner loop iterations is:

2 + 3 + 4 + …. + *n*

This is $n(n+1)/2 - 1$, which has an order of magnitude of n^2.

Quicksort

- STAY TUNED!