

LECTURE 17

QUICKSORT

MCS 275 Spring 2021

Emily Dumas

LECTURE 17: QUICKSORT

Course bulletins:

- Starting with Quiz 6, you will have 48 hours for quizzes (Noon Sunday to Noon Tuesday).
- [Project 2 description](#) updated with sample data and modules policy.
- Project 2 due 6pm CST Friday, February 26.
- Check out the [recursion sample code](#).
- Worksheet 7 will explore recursive maze solver / generator in more depth.

PLAN

- Discuss mergesort a bit more
- Introduce quicksort
- Implement quicksort

WHY DISCUSS ALGORITHMS?

Python lists have built-in `.sort()` method. Why talk about sorting?

1. Study cases of easy-to-explain problems solved in clever ways.
2. See patterns of thinking that work in other settings.

MERGESORT

Last time we discussed and implemented mergesort.

History: Developed by von Neumann (1945) and Goldstine (1947).

But is it a good way to sort a list?

EFFICIENCY

Theorem: If you measure the time cost of mergesort in any of these terms

- Number of comparisons made
- Number of assignments (e.g. $L[i] = x$ counts as 1)
- Number of Python statements executed

then the cost to sort a list of length n is less than $Cn \log(n)$, for some constant C that only depends on which expense measure you chose.

ASYMPTOTICALLY OPTIMAL

$Cn \log(n)$ is pretty efficient for an operation that needs to look at all n elements. It's not linear in n , but it only grows a little faster than linear functions.

Furthermore, $Cn \log(n)$ is the best possible time for comparison sort of n elements (though different methods might have better C).

QUICKSORT

Another comparison sort typically implemented using recursion. Developed by Hoare, 1959.

Unlike mergesort, it uses very little temporary storage, and only ever **swaps** pairs of elements.

QUICKSORT SUMMARY

Starting with an unsorted list:

- If the list has 0 or 1 elements, return immediately.
- Partition: Choose an element (the **pivot**). Rearrange so elements smaller than the pivot come before it, elements larger than the pivot come after it.
- Quicksort the part of the list before the pivot.
- Quicksort the part of the list after the pivot.

It's divide and conquer, but with no merge step. The hard work is instead in partitioning.

QUICKSORT VISUALIZATION

76235814

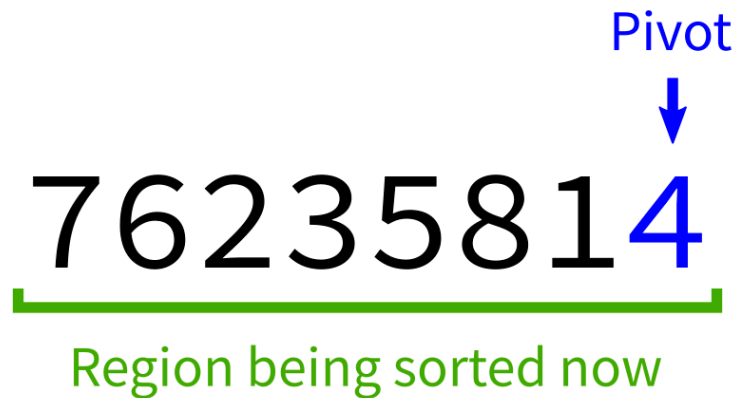
QUICKSORT VISUALIZATION

76235814

Region being sorted now

QUICKSORT VISUALIZATION

Pivot
↓
76235814
Region being sorted now

The image shows a sequence of numbers: 7, 6, 2, 3, 5, 8, 1, 4. The number 4 is highlighted in blue and labeled 'Pivot' with a blue arrow pointing down to it. A green horizontal line is drawn under the numbers 7, 6, 2, 3, 5, 8, and 1, with the text 'Region being sorted now' written below it in green.

QUICKSORT VISUALIZATION

Pivot

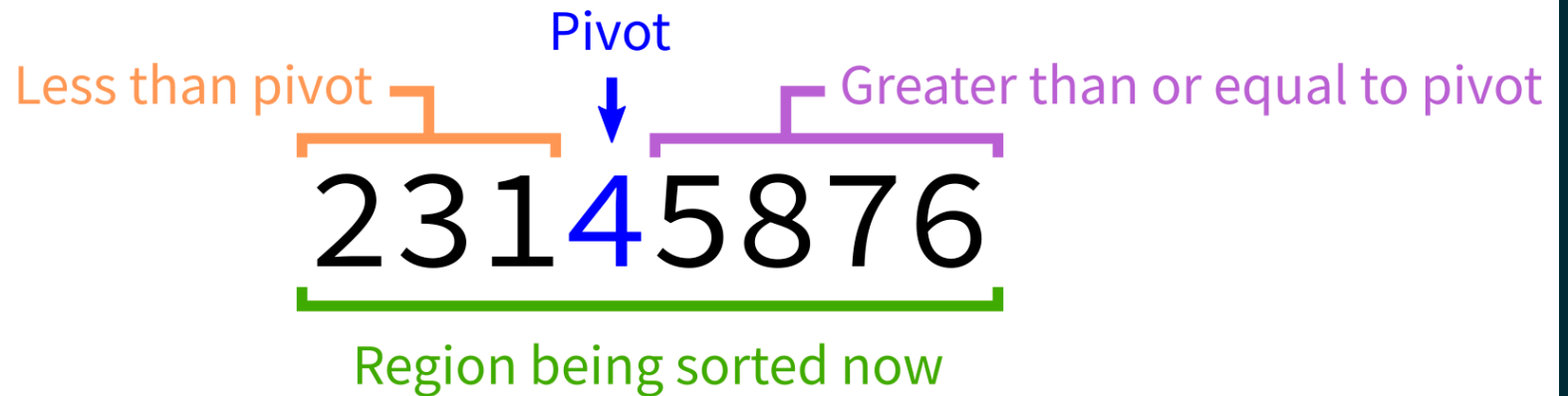


23145876



Region being sorted now

QUICKSORT VISUALIZATION



QUICKSORT VISUALIZATION

23145876

Region being sorted now

QUICKSORT VISUALIZATION

23145876



QUICKSORT VISUALIZATION

23145876

QUICKSORT VISUALIZATION

13245876

QUICKSORT VISUALIZATION

13245876



QUICKSORT VISUALIZATION

13245876

QUICKSORT VISUALIZATION

13245876

QUICKSORT VISUALIZATION

12345876

QUICKSORT VISUALIZATION

12345876



QUICKSORT VISUALIZATION

12345876



QUICKSORT VISUALIZATION

12345876

QUICKSORT VISUALIZATION

12345876


QUICKSORT VISUALIZATION

12345876



QUICKSORT VISUALIZATION

12345876



QUICKSORT VISUALIZATION

12345678

QUICKSORT VISUALIZATION

12345678



QUICKSORT VISUALIZATION

12345678



QUICKSORT VISUALIZATION

12345678



QUICKSORT VISUALIZATION

12345678

QUICKSORT VISUALIZATION

12345678



QUICKSORT VISUALIZATION

12345678

QUICKSORT VISUALIZATION

12345678



QUICKSORT VISUALIZATION

12345678



QUICKSORT VISUALIZATION

12345678

QUICKSORT VISUALIZATION

12345678

ASSUME WE HAVE PARTITION

For the moment, we'll take the partition step as a "black box", assuming we already have:

```
def partition(L, start, end):  
    """Look at L[start:end]. Take the last element as a pivot.  
    Move elements around so that any value less than the pivot  
    appears before it, and any element greater than or equal to  
    the pivot appears after it. L is modified in place. The  
    final location of the pivot is returned."""  
    # TODO: Add code here
```

Note this function uses the last element as a pivot.
Later we'll discuss other options.

CODING TIME

Let's implement `quicksort` in Python.

Algorithm quicksort:

Input: list L and indices $start$ and end .

Goal: reorder elements of L so that $L[start:end]$ is sorted.

1. If $(end - start)$ is less than or equal to 1, return immediately.
2. Otherwise, call $partition(L)$ to partition the list, letting m be the final location of the pivot.
3. Call $quicksort(L, start, m)$ and $quicksort(L, m+1, end)$ to sort the parts of the list on either side of the pivot.

PARTITION

How to write `partition(L, start, end)`?

Recall we plan to make a version that uses the last element of `L[start:end]` as the pivot.

PARTITION VISUALIZATION

76235814

PARTITION VISUALIZATION

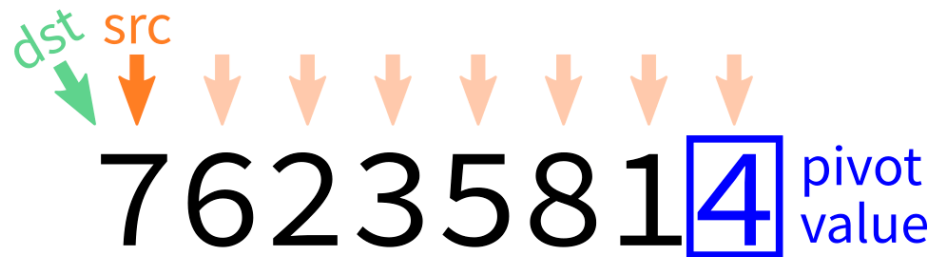
76235814 pivot
value

PARTITION VISUALIZATION

dst src
76235814 pivot
value

The diagram shows a sequence of numbers: 7, 6, 2, 3, 5, 8, 1, and 4. Above the number 7 is the label 'dst' with a green arrow pointing down to it. Above the number 6 is the label 'src' with an orange arrow pointing down to it. The number 4 is enclosed in a blue square box, and to its right are the labels 'pivot' and 'value' stacked vertically.

PARTITION VISUALIZATION



PARTITION VISUALIZATION

dst src
7 6 2 3 5 8 1 4 pivot
value

The diagram shows a sequence of numbers: 7, 6, 2, 3, 5, 8, 1, 4. The number 7 is highlighted in pink. Above it, a green arrow labeled 'dst' points down, and an orange arrow labeled 'src' points down. The number 4 is enclosed in a blue square box, and the text 'pivot value' is written to its right in blue. The numbers 6, 2, 3, 5, 8, and 1 are in black.

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 4 pivot
value

$L[\text{src}] \geq \text{pivot}$: do nothing

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 4 pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 4 pivot
value

L[src] ≥ pivot: do nothing

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

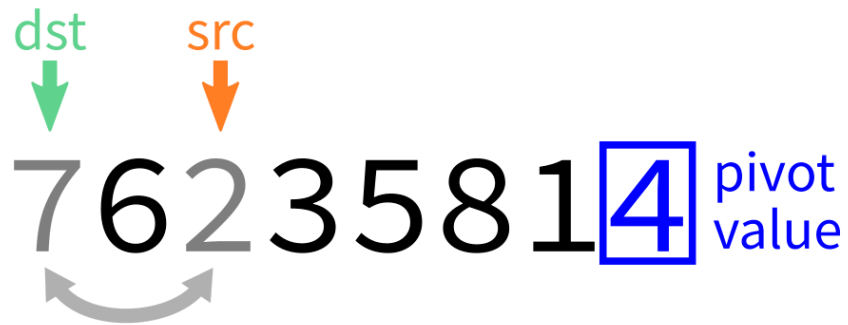
dst src
↓ ↓
7 6 2 3 5 8 1 4 pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
7 6 2 3 5 8 1 4 pivot
value

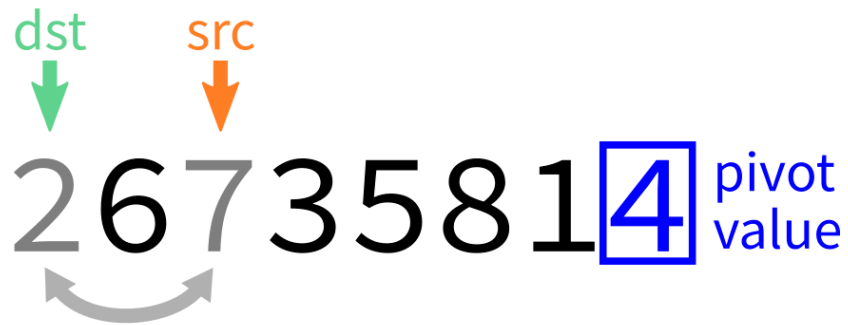
$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION



$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION



$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION

dst src
↓ ↓
2 6 7 3 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 6 7 3 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

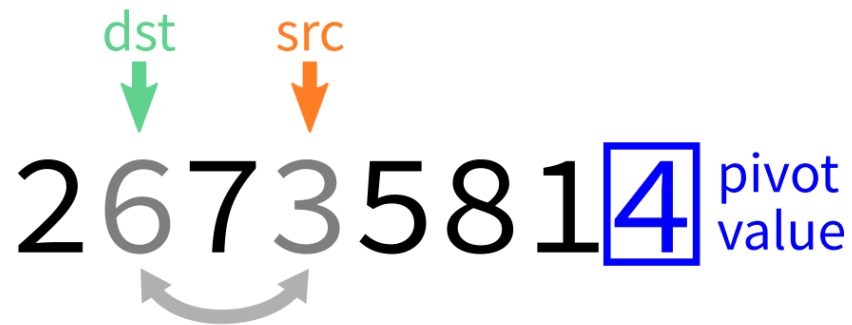
dst src
↓ ↓
2 6 7 3 5 8 1 4 pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 6 7 3 5 8 1 4 pivot
value

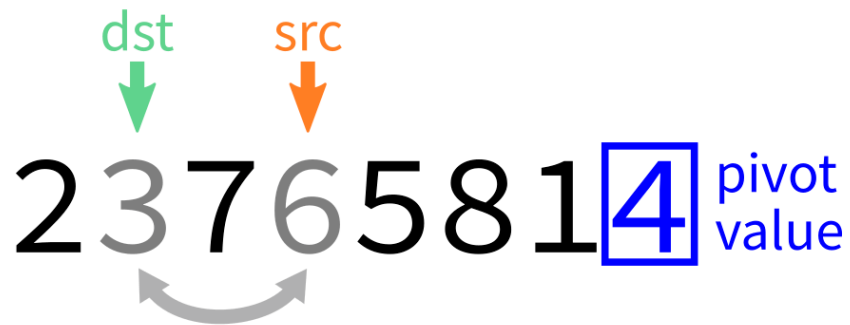
$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION



$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION



$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 4 pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 4 pivot
value

$L[\text{src}] \geq \text{pivot}$: do nothing

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 4 pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 4 pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 4 pivot
 value

$L[\text{src}] \geq \text{pivot}$: do nothing

PARTITION VISUALIZATION

dst src
↓ ↓
2 3 7 6 5 8 1 **4** pivot
value

PARTITION VISUALIZATION

dst src
↓ ↓
23765814 pivot
value

PARTITION VISUALIZATION

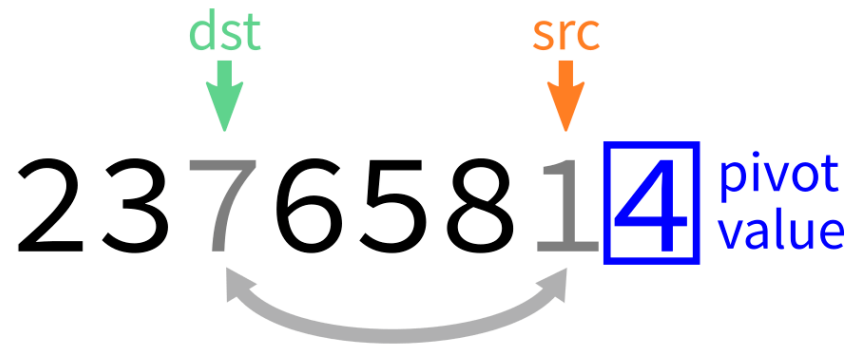
dst
↓
23765814 pivot
src
↓
value

PARTITION VISUALIZATION

dst
↓
23765814 pivot
src
↓
value

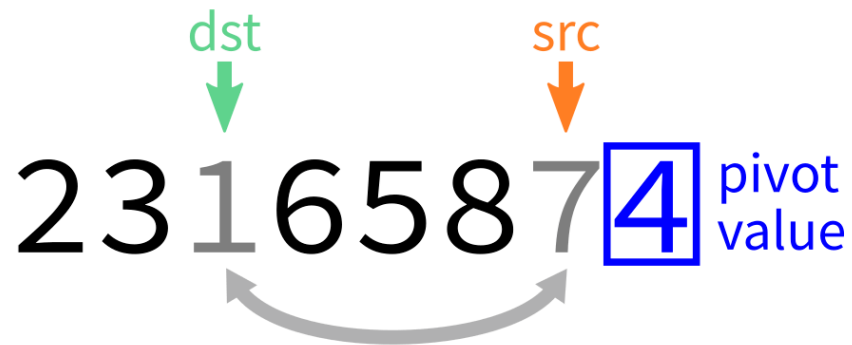
$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION



$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION



$L[\text{src}] < \text{pivot}$: **swap** $L[\text{src}], L[\text{dst}]$

PARTITION VISUALIZATION

dst src
↓ ↓
2316587 **4** pivot
value

PARTITION VISUALIZATION

23165874 pivot value

The diagram illustrates the partitioning step of a sorting algorithm. The array is [2, 3, 1, 6, 5, 8, 7, 4]. The pivot value is 4, which is highlighted with a blue box. A green arrow labeled 'dst' points to the element 6, indicating its target position. An orange arrow labeled 'src' points to the pivot element 4, indicating its source position. The text 'pivot value' is written in blue next to the pivot element.

PARTITION VISUALIZATION

23165874 pivot value

The diagram illustrates the partitioning step of a sorting algorithm. The array is [2, 3, 1, 6, 5, 8, 7, 4]. The pivot value is 4, which is highlighted in a blue box. A green arrow labeled 'dst' points to the element 6, and an orange arrow labeled 'src' points to the pivot value 4. The text 'pivot value' is written in blue to the right of the pivot value.

PARTITION VISUALIZATION

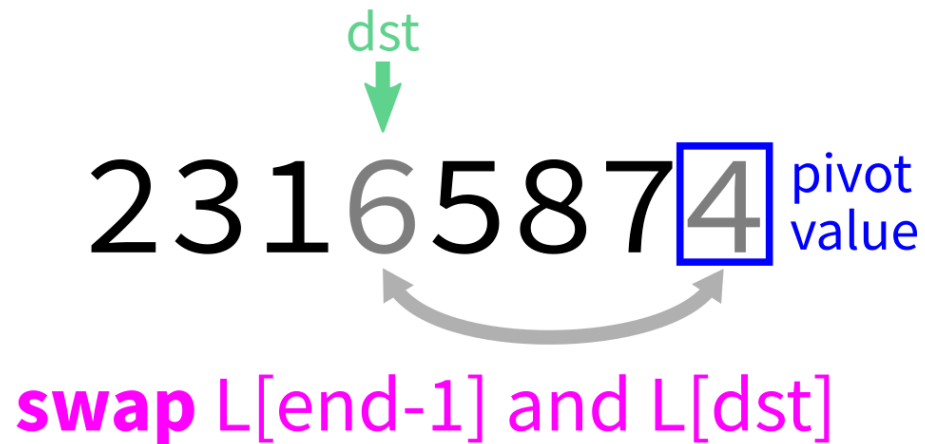
dst
↓
23165874 pivot
src
↓
value

$L[\text{src}] \geq \text{pivot}$: do nothing

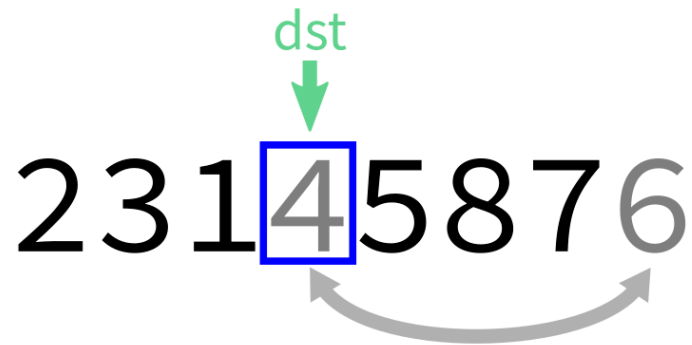
PARTITION VISUALIZATION

dst
↓
23165874 pivot
value

PARTITION VISUALIZATION



PARTITION VISUALIZATION



swap $L[\text{end}-1]$ and $L[\text{dst}]$

PARTITION VISUALIZATION

23145876

PARTITION VISUALIZATION

23145876

The image shows the array [2, 3, 1, 4, 5, 8, 7, 6] with the pivot element 4 highlighted by a blue square. An orange bracket is positioned under the elements 2, 3, and 1, and a purple bracket is positioned under the elements 5, 8, 7, and 6. This visualizes the partitioning step of a sorting algorithm where elements less than the pivot are on the left and elements greater than the pivot are on the right.

PARTITION VISUALIZATION

23145876

<4 ≥4

The diagram illustrates the partitioning of the array [2, 3, 1, 4, 5, 8, 7, 6] around the pivot element 4. The pivot is highlighted with a blue square. An orange bracket under the elements 2, 3, and 1 is labeled <4, indicating they are less than the pivot. A purple bracket under the elements 5, 8, 7, and 6 is labeled ≥4, indicating they are greater than or equal to the pivot.

REFERENCES

- You can refer to the [general references about recursion](#) that have appeared in several recent lectures. The rest of this list is specific to mergesort and quicksort.
- Making nice visualizations of sorting algorithms is a cottage industry in CS education. Some you might like to check out:
 - [2D visualization through color sorting](#) by Linus Lee
 - [Animated bar graph visualization of many sorting algorithms](#) by Alex Macy
 - [Slanted line animated visualizations of mergesort and quicksort](#) by Mike Bostock

REVISION HISTORY

- 2021-02-22 Moved unused slides to Lecture 18 and fix partition visualization
- 2021-02-17 Fix description of partition step in quicksort preview
- 2021-02-17 Initial publication

