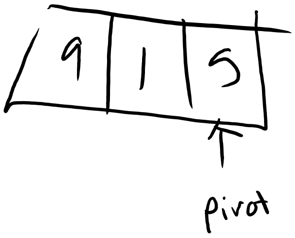
2019-05-02 Quicksort

Thursday, May 2, 2019 9:01 AM

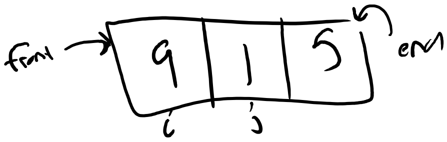
Quicksort

- General idea: find a "pivot" (reference value). Put items smaller than the pivot to the "left" of the pivot and items larger than the pivot to the "right" of the pivot.
- Recursively do this for each subsection of the array
  - "left" side smaller values
  - "right" side larger values
- The selection of the pivot is the most important factor when determining quicksort's runtime
  - The idea pivot is the array's median value because it perfectly partitions our data into two equal segments
    - Unfortunately, it's too expensive to always be finding the median
  - Compromise is to find "good enough" value
  - Richard Weiss suggests "best of 3" approach
    - Select middle-most value from array[0], array[midpoint], array[end]
    - This guarantees that we avoid the worst case selection of picking largest or smallest value as our pivot

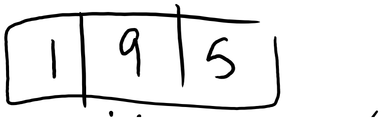
Most basic array sorting example

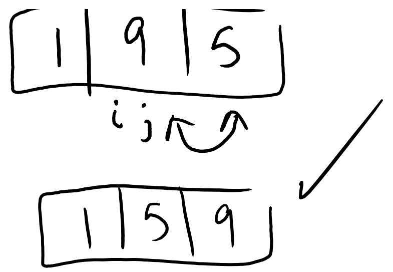


- After selecting the pivot, move it to "end" of the array
- Idea: get the pivot out of the way

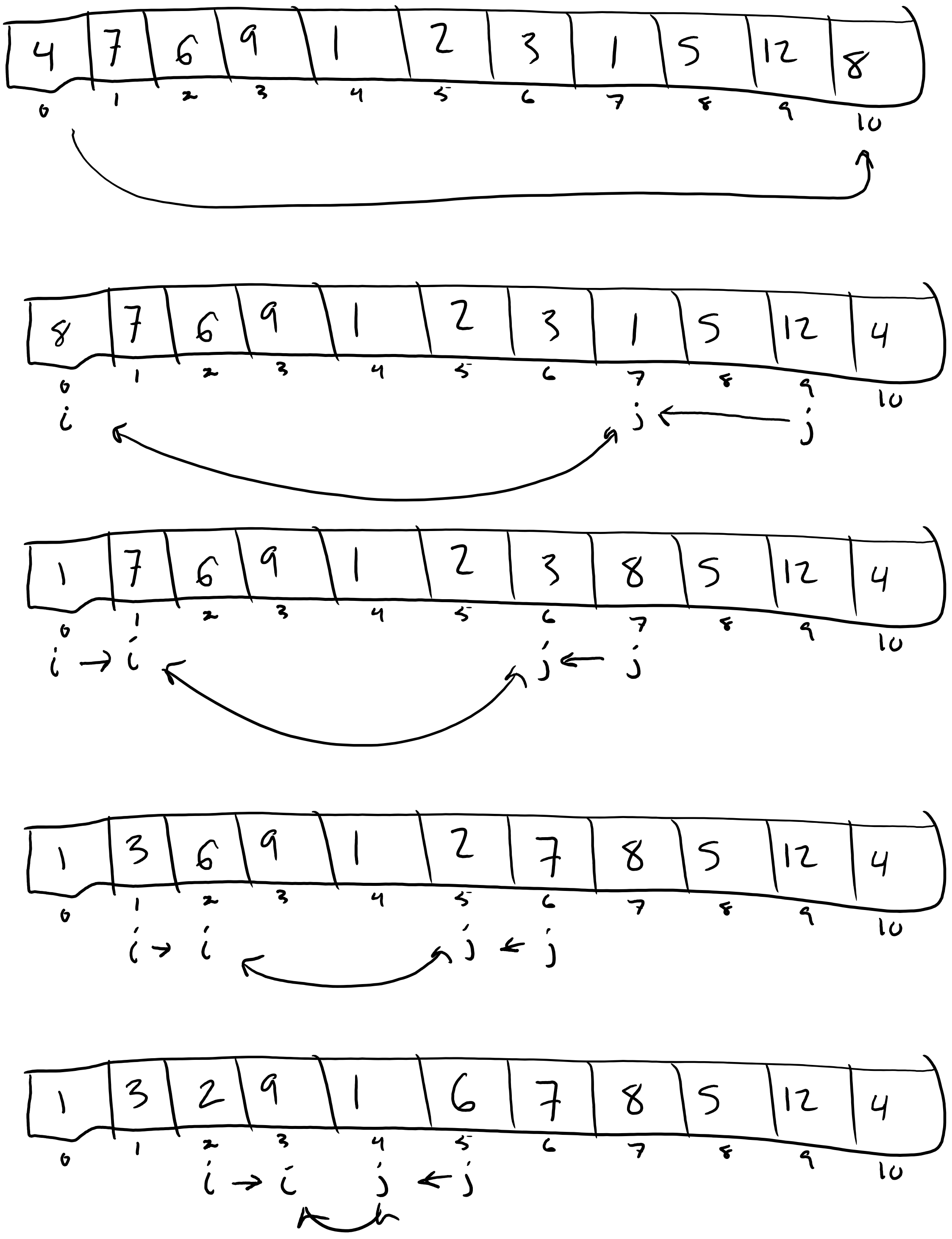


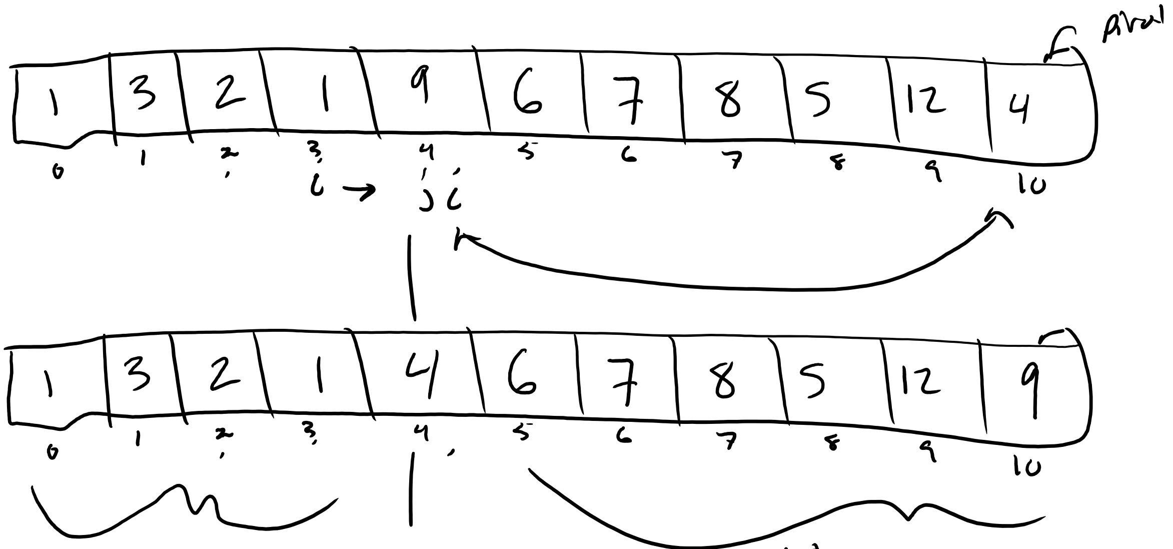
1. Define i = front; j = end - 1
2. WHILE data[i] < pivot AND i < j
   a. Increment i
3. WHILE data[j] > pivot AND i < j
   a. Decrement j
4. If i != j:
   a. Swap numbers[i] with numbers[j]
   b. Go back to step #2
5. Loop is done. Swap numbers[end] with numbers[i]
6. Recursively repeat this process on the divided array

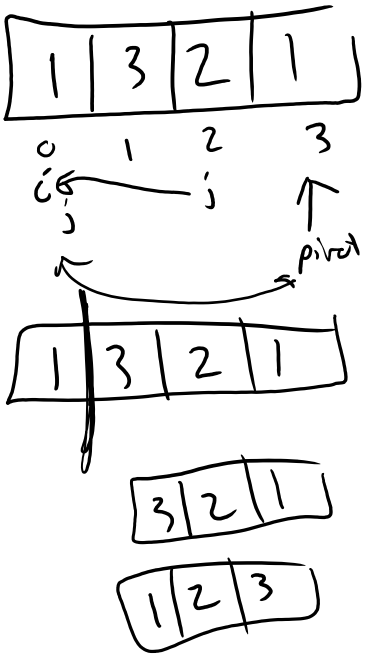


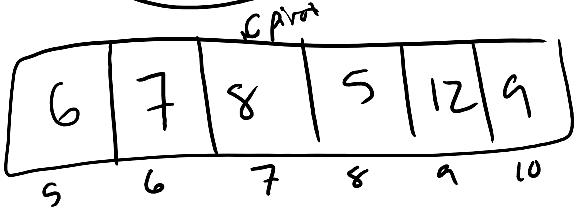


Another example

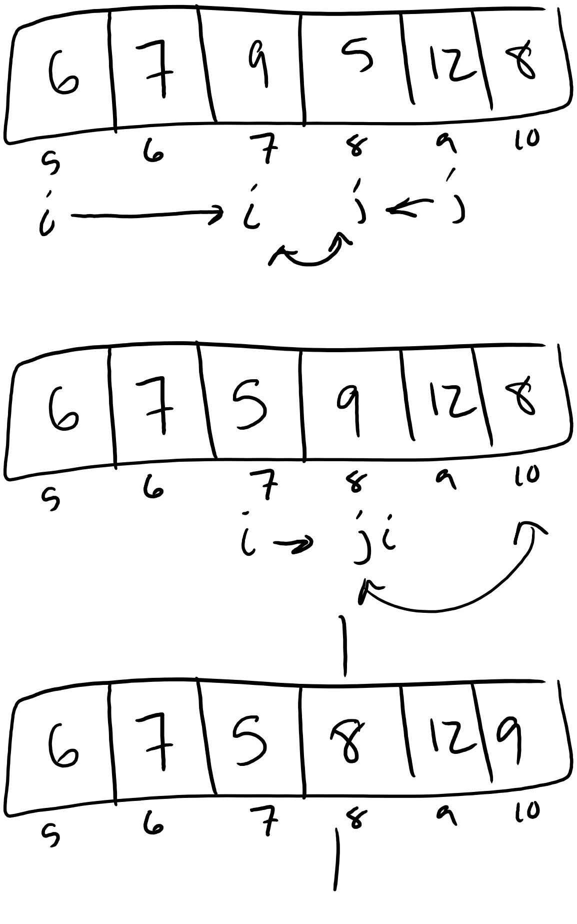






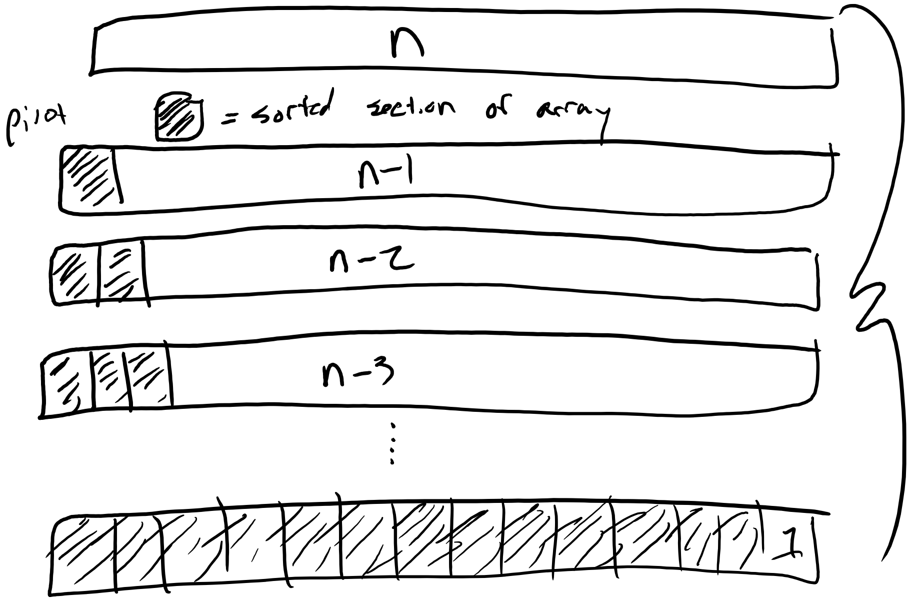


$$\text{midpoint} = \frac{(\text{beginning} + \text{end})}{2}$$



Analysis of Quicksort

- Consider if we were always to choose the worst pivot
  - I.e. select smallest item every time

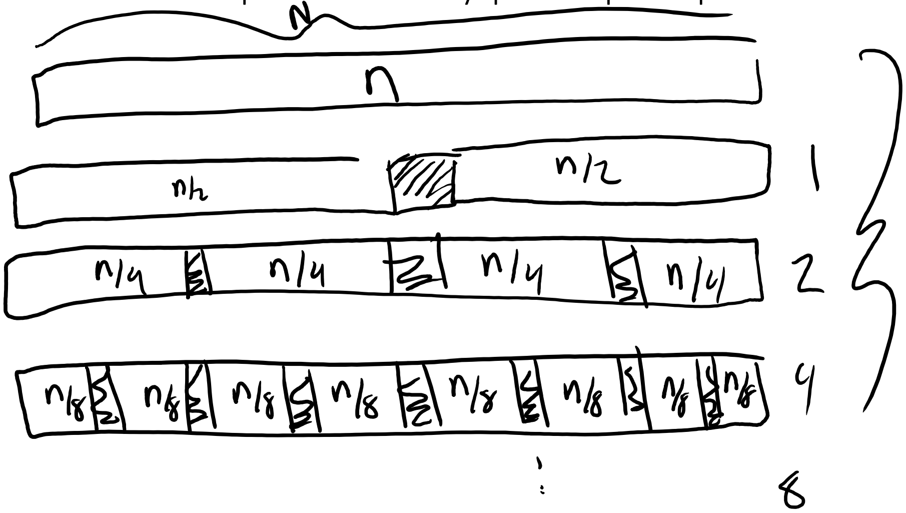


$$\sum_{i=0}^{n} n-i$$

$$\sum_{i=0}^{n} i = \frac{n(n-1)}{2}$$

$$= n^2$$

- Consider a quicksort that always picks the perfect pivot



height = $\lg N$

$N \lg N$

Exponential knowledge gain means logarithmic increase in time complexity