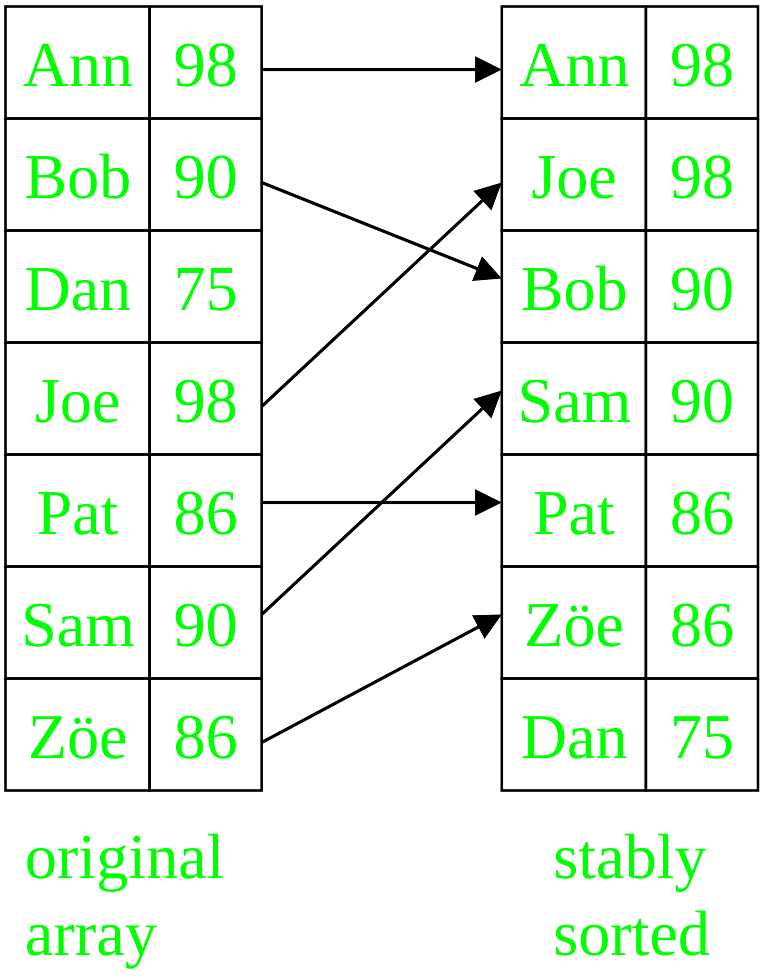
More on Sorting

Some Definitions

- In Place Sorting
  - The amount of extra space required to sort the data is constant with the input size.
  - Some devices don’t have enough space
    ex) Embedded system like PDA, cellphone
  - Reducing space usage is important
- Not in place (out of place) sorting
  - The opposite of in place sorting

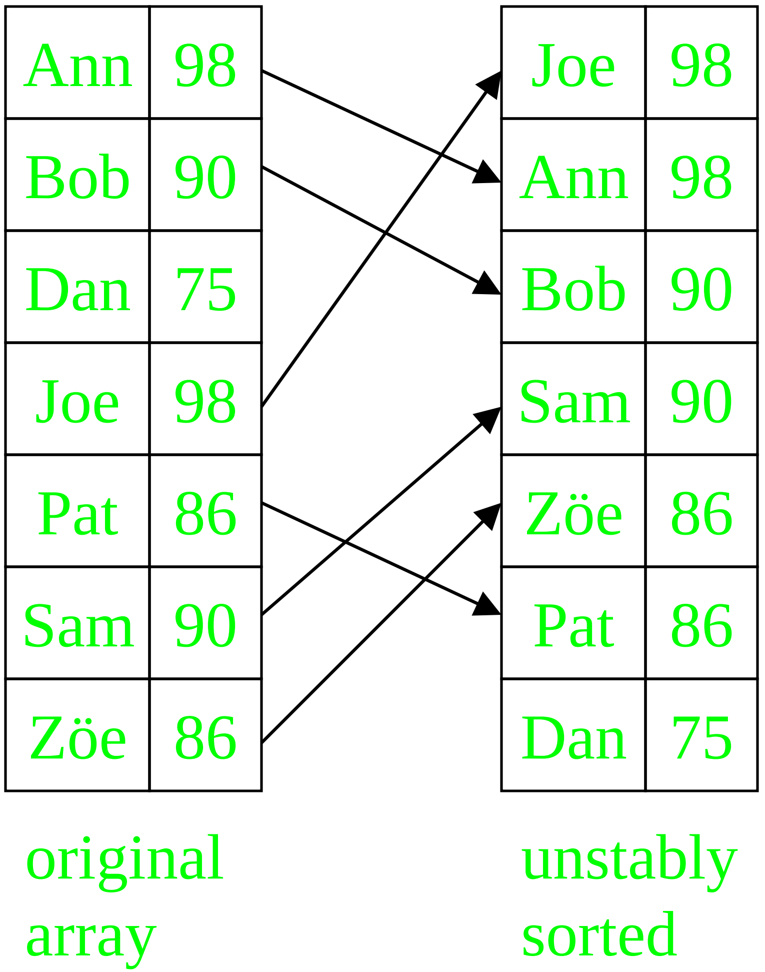
Stable sort algorithms

- A <u>stable</u> sort keeps equal elements in the same order
- This may matter when you are sorting data according to some characteristic
- Example: sorting students by test scores



Unstable sort algorithms

- An <u>unstable</u> sort may or may not keep equal elements in the same order
- Stability is usually not important, but sometimes it is important



Types of Sorting Algorithms

- There are many, many different types of sorting algorithms, but the primary ones are:
- Sorting
  - O(n^2) sorting algorithms:
    Insertion, Selection, Bubble
  - O($n$log$n$) sorting algorithms:
    HeapSort, MergeSort, QuickSort

When to use which sorting algorithm?

- Large arrays: merge sort, quick sort.
- Small arrays: insertion sort, selection sort.
  - Recursion is expensive.
- Merge sort or quick sort in an average case?
  - Cost of comparing elements
  - Cost of moving/switching elements

Small Array in QuickSort

- When S is small, recursive calls become expensive (*overheads*)
- General strategy
  - When size < threshold, use a sort more efficient for small arrays (e.g., InsertionSort)
  - Good thresholds range from 5 to 20
  - Also avoids issue with finding median-of-three pivot for array of size 2 or less
  - Has been shown to reduce running time by 15%

MergeSort vs. QuickSort

- both are divide and conquer algorithms (recursive)
- both are O($n$log$n$) in the average case
- Mergesort is O($n$log$n$) in the worst case
  Quicksort is O(n^2) in the worst case
- Mergesort requires 2n space
- Quicksort requires no extra space

MergeSort vs. QuickSort

- MergeSort
  - Lowest number of comparisons among popular algorithms
  - Lots of data movements/copying (merging)
- Quick sort
  - More comparisons
  - Fewer data movements

MergeSort vs. QuickSort

- Main problem with quicksort:
  - QuickSort may end up dividing the input array into subproblems of size 1 and N-1 in the worst case, at every recursive step (unlike merge sort which always divides into two halves)
    - When can this happen?
    - Leading to O(n^2) performance

  ⇨ <u>Need to choose pivot wisely (but efficiently)</u>
- MergeSort is typically implemented using a temporary array (for merge step)
  - QuickSort can partition the array "in place"

Sorting methods

- ***Comparison based sorting***
  - $O(n^2)$ methods E.g., Insertion, bubble
  - Average time $O(n \log n)$ methods E.g., quick sort
  - $O(n \log n)$ methods E.g., Merge sort, heap sort
- ***Non-comparison based sorting***
  - Integer sorting
  - Counting sorting
  - Bucket sorting
  - Radix sorting

Internal sorting vs. External sorting

- Internal Sort
  - The data to be sorted is all stored in the computer’s main memory.
- External Sort
  - Some of the data to be sorted might be stored in some external, slower device.

Sorting: Summary

- Comparison vs. Non comparison
- In-place vs. not in-place
- Stable vs. Non Stable
- Internal sorting vs. External sorting
- Need for sorting is ubiquitous in software
- Optimizing the sort algorithm to the domain is essential
- Good general-purpose algorithms available:
  - QuickSort
- Optimizations continue…
  - Sort benchmarks
    http://sortbenchmark.org/