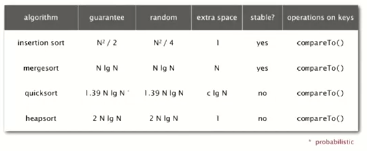
Key-Indexed Counting

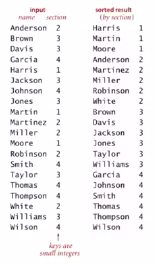
Summary of sorting algorithm performance:



Frequency of operations = key compares

Lower bound: ~N log N compares required by any compare-based algorithm

**If we don’t depend on compares, we can do better than the lower bound.**



Assumption: keys are between 0 and R-1

Implication: Can use key as an array index

Applications:

* Sort string by first letter
* Sort class roster by section
* Sort phone numbers by area code
* Subroutine in a sorting algorithm

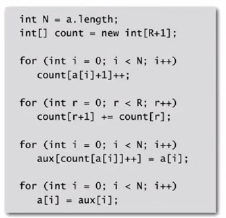
Note: Keys may have associated data ->   
Can’t just count up number of keys of each value

Index based counting

Goal: sort an array a[] of N integers between 90 and R -1.

* Count frequencies of each letter using key as index
* Compute frequency cumulates which specify destinations
* Access cumulates using key as index to move items
* Copy back into original array

Index based counting demo Java implementation



Proposition:

* Key-indexed counting uses ~11 N + 4 R array accesses to sort N items whose keys are   
  between 0 and R -1
* Key indexed counting uses extra space proportional to N + R

This is a stable algorithm