COMP20003 Algorithms and Data Structures Distribution Counting

Nir Lipovetzky
Department of Computing and Information Systems
University of Melbourne
Semester 2

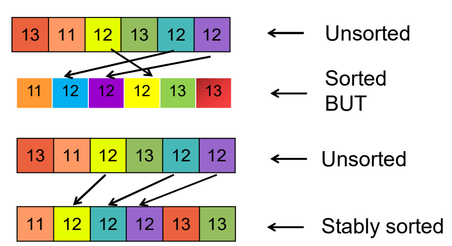
Sorting by Counting

- Distribution counting:
  - unusual approach to sorting
- Later we will look at more common approaches
- Distribution counting requires:
  - Key values to be within a certain range, *lower* to *upper*.

Sorting by Counting: Approach

- Steps in distribution counting:
  - Start with array of:
    - Records, or
    - Keys + pointers to records
  - Count number of records associated with each key value (*lower* to *upper*)
  - Redistribute array elements
- Net result:
  - Sorted array
  - Stable sort

Segue: What is a stable sort?

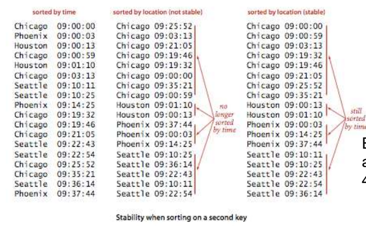


Stable sorting: definition

Stable sorting algorithms maintain relative order of records with equal key values.

Stable sorting: Applications

- Want file sorted on one key, and within each group, sorted on another key:



sorted by time		sorted by location (not stable)		sorted by location (stable)	
Chicago	09:00:00	Chicago	09:25:52	Chicago	09:00:00
Phoenix	09:00:03	Chicago	09:03:13	Chicago	09:00:59
Houston	09:00:13	Chicago	09:21:05	Chicago	09:03:13
Chicago	09:00:59	Chicago	09:19:46	Chicago	09:19:32
Houston	09:01:10	Chicago	09:19:32	Chicago	09:19:46
Chicago	09:03:13	Chicago	09:00:00	Chicago	09:21:05
Seattle	09:10:11	Chicago	09:35:21	Chicago	09:25:52
Seattle	09:10:25	Chicago	09:00:59	Chicago	09:35:21
Phoenix	09:14:25	Houston	09:01:10	Houston	09:00:13
Chicago	09:19:32	Houston	09:00:13	Houston	09:01:10
Chicago	09:19:46	Phoenix	09:37:44	Phoenix	09:00:03
Chicago	09:21:05	Phoenix	09:00:03	Phoenix	09:14:25
Seattle	09:22:43	Phoenix	09:14:25	Phoenix	09:37:44
Seattle	09:22:54	Seattle	09:10:25	Seattle	09:10:11
Chicago	09:25:52	Seattle	09:36:14	Seattle	09:10:25
Chicago	09:35:21	Seattle	09:22:43	Seattle	09:22:43
Seattle	09:36:14	Seattle	09:10:11	Seattle	09:22:54
Phoenix	09:37:44	Seattle	09:22:54	Seattle	09:36:14

Stability when sorting on a second key

Example from Sedgewick and Wayne, Algorithms, 4th Edition, 2011

Back to Distribution Counting: Approach

- Steps in distribution counting:
  - Input: array of:
    - records, or
    - keys + pointers to records
  - Count number of records associated with each key value (*lower* to *upper*).
  - Redistribute array elements.
  - Output: stably sorted array.

Back to Distribution Counting: Example:

- Sort [4,4,2,2,0,2,1,3,2,4,3,1,4,3,1,4]
- Count records for each key [1,3,4,3,5]
  - CumulativeCount = [0,1,4,8,11]
- Redistribute
  - Create auxiliary array
  - traverse original array copying each item to position:
    - aux_array[cumulativeCount[item.key]] = item
    - Increase cumulativeCount[itemkey] + 1

Distribution Counting: Analysis

- Time:
  - Worst-case:
  - Average-case:
- Space:

Does the key range influence the complexity?

- *O(n)* if range *r* of keys is in *O(n)*
  - **`count[]`** array size is *r*
  - Initialization and shuffling are *O(r)*
  - So if *r > n*…

But what about theory?

- we said weeks ago:
  - Comparison-based sorting is Ω(n log n)
- Does distribution counting contradict that statement?

Sorting without comparing

- Other non-comparison-based sorting algorithms include:
  - LSD Radix sort
  - MSD Radix sort
  - Several others
- Drawbacks:
  - Take extra space
  - Generally less flexible than comparison-based
  - Can be fiddly if keys are not the same length, *e.g.* variable length strings in MSD radix