

COMP20003
Algorithms and Data Structures
Distribution Counting

Kris Ehinger
Department of Computing and
Information Systems
University of Melbourne
Semester 2



Why Sorting?

Census 1890 led to IBM

- 1960: ¼ CPU resources worldwide spent on sorting

- Pervasive for human experience:

- **Digital**: FB, Tweet, Email, Reedit, google search, etc.
- **Physical**: Bills, papers, books, socks, etc.
- **Societal**: Sports - grades - college ranking, credit rating, etc.
- **Animal Behaviour**: group sorting, bigger groups -> confrontations, fade-out with time

Sort Is **Prophylaxis** for Search:

- CS: should you sort at all? All about trade-offs

Sort is a **preemptive** strike for future search

- Mess can be a bliss

Estimate ahead of time the future usage

433-521 Algorithms and Complexity

2-2

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**.

433-521 Algorithms and Complexity

2-3

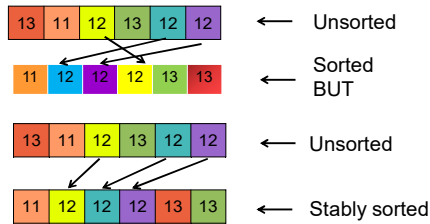
Sorting by Counting: Approach

- 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**

433-521 Algorithms and Complexity

2-4

Segue: What is a stable sort?



COMP 20003 Algorithms and Data Structures

1-5

Stable sorting: definition

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

COMP 20003 Algorithms and Data Structures

1-6

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

no longer sorted by time

still sorted by time

COMP 20003 Algorithms and Data Structures

1-7

Example from Sedgwick 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.

433-521 Algorithms and Complexity

2-8

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** [, , , ,]

	Pos	0	1	2	3	4		
•	CumulativeCount	=	[]	(#items < pos_key)
	Pos	0	1	2	3	4		
- **Redistribute**
 - Create **auxiliary array**
 - [, , , , , , , , , , , , , , ,]
 - Pos 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
 - **traverse original array copying each item to position:**
 - `aux_array[cumulativeCount[item.key]] = item`
 - Increase `cumulativeCount[itemkey] + 1`

2-9

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`

433-521 Algorithms and Complexity

2-10

Distribution Counting: Analysis

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

433-521 Algorithms and Complexity

2-11

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 **$r > n$** ...

COMP 20003 Algorithms and Data Structures

1-12

But what about theory?

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

COMP 20003 Algorithms and Data Structures

1-13

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

COMP 20003 Algorithms and Data Structures

1-14