CSE 373: Data Structures & Algorithms

Finish Sorting; Induction; Recurrence Relations

Riley Porter

Winter 2017

Course Logistics

- HW5 due today! Check your submission please (download and expand the zip)

- HW6 out already → sorting (and to a lesser degree reading specs/other files/tests)

Review: Sorting: The Big Picture

Simple algorithms: $O(n^2)$	Fancier algorithms: $O(n \log n)$	Comparison lower bound: $\Omega(n \log n)$	Specialized algorithms: $O(n)$	Handling huge data sets
Insertion sort	Heap sort		Bucket sort	External sorting
Selection sort	Merge sort		Radix sort	
Bubble sort	Quick sort (avg)			
Shell sort	…			
…				

Non-Comparison Sorts

- Bucket Sort:
  - every possible element in our domain is known
  - the domain is relatively small (ex: numbers 1-100, all sections AA-AF)
- Radix Sort:
  - we have a base we can use to bucket sort our domain in a series of passes (ex: 0-9 for decimal numbers, ascii values for chars)

Example: Bucket Sort

Quiz scores (0-5 possible):

- CJ Cregg – 5
- Donna Moss – 5
- Jed Bartlet – 4
- Josh Lyman – 4
- Leo McGarry – 5
- Sam Seaborn – 2
- Toby Ziegler – 3

`count` array	
0	0
1	0
2	0
3	0
4	0
5	0

Example: Bucket Sort

Quiz scores (0-5 possible):

~~CJ Cregg – 5~~

Donna Moss – 5

Jed Bartlet – 4

Josh Lyman – 4

Leo McGarry – 5

Sam Seaborn – 2

Toby Ziegler – 3

`count` array	
0	0
1	0
2	0
3	0
4	0
5	1

Example: Bucket Sort

Quiz scores (0-5 possible):

~~CJ Cregg – 5~~

~~Donna Moss – 5~~

Jed Bartlet – 4

Josh Lyman – 4

Leo McGarry – 5

Sam Seaborn – 2

Toby Ziegler – 3

`count` array	
0	0
1	0
2	0
3	0
4	0
5	2

Example: Bucket Sort

Quiz scores (0-5 possible):

~~CJ Cregg – 5~~

~~Donna Moss – 5~~

~~Jed Bartlet – 4~~

Josh Lyman – 4

Leo McGarry – 5

Sam Seaborn – 2

Toby Ziegler – 3

`count` array	
0	0
1	0
2	0
3	0
4	1
5	2

Example: Bucket Sort

Quiz scores (0-5 possible):

~~CJ Cregg – 5~~

~~Donna Moss – 5~~

~~Jed Bartlet – 4~~

~~Josh Lyman – 4~~

Leo McGarry – 5

Sam Seaborn – 2

Toby Ziegler – 3

`count` array	
0	0
1	0
2	0
3	0
4	2
5	2

Example: Bucket Sort

Quiz scores (0-5 possible):

~~CJ Cregg – 5~~

~~Donna Moss – 5~~

~~Jed Bartlet – 4~~

~~Josh Lyman – 4~~

~~Leo McGarry – 5~~

Sam Seaborn – 2

Toby Ziegler – 3

`count` array	
0	0
1	0
2	0
3	0
4	2
5	3

Example: Bucket Sort

Quiz scores (0-5 possible):

~~CJ Cregg – 5~~

~~Donna Moss – 5~~

~~Jed Bartlet – 4~~

~~Josh Lyman – 4~~

~~Leo McGarry – 5~~

~~Sam Seaborn – 2~~

Toby Ziegler – 3

`count` array	
0	0
1	0
2	1
3	0
4	2
5	3

Example: Bucket Sort

Quiz scores (0-5 possible):

- ~~CJ Cregg – 5~~
- ~~Donna Moss – 5~~
- ~~Jed Bartlet – 4~~
- ~~Josh Lyman – 4~~
- ~~Leo McGarry – 5~~
- ~~Sam Seaborn – 2~~
- ~~Toby Ziegler – 3~~

`count` array	
0	0
1	0
2	1
3	1
4	2
5	3

Example: Bucket Sort

`count` array	
0	0
1	0
2	1
3	1
4	2
5	3

Output, sorted scores: 2, 3, 4, 4, 5, 5, 5

Example: Bucket Sort

What if we wanted to keep track of who got which score?

Quiz scores (0-5 possible):

- CJ Cregg – 5
- Donna Moss – 5
- Jed Bartlet – 4
- Josh Lyman – 4
- Leo McGarry – 5
- Sam Seaborn – 2
- Toby Ziegler – 3

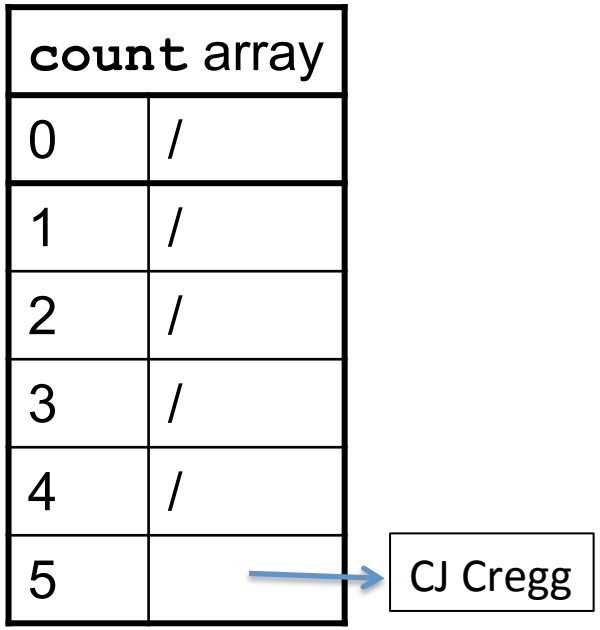
`count` array	
0	/
1	/
2	/
3	/
4	/
5	/

Example: Bucket Sort

What if we wanted to keep track of who got which score?

Quiz scores (0-5 possible):

- ~~CJ Cregg – 5~~
- Donna Moss – 5
- Jed Bartlet – 4
- Josh Lyman – 4
- Leo McGarry – 5
- Sam Seaborn – 2
- Toby Ziegler – 3

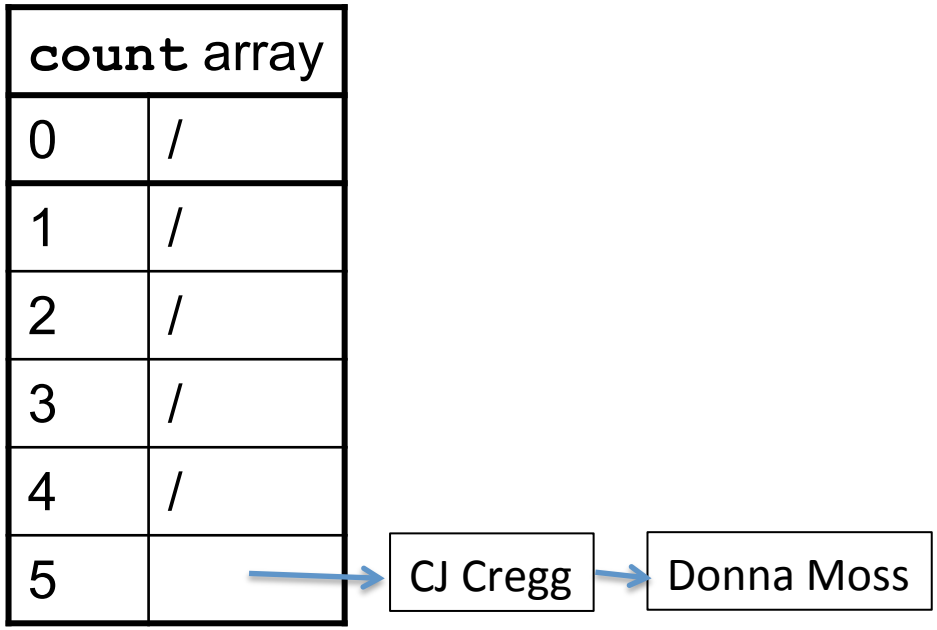


Example: Bucket Sort

What if we wanted to keep track of who got which score?

Quiz scores (0-5 possible):

- ~~CJ Cregg – 5~~
- ~~Donna Moss – 5~~
- Jed Bartlet – 4
- Josh Lyman – 4
- Leo McGarry – 5
- Sam Seaborn – 2
- Toby Ziegler – 3

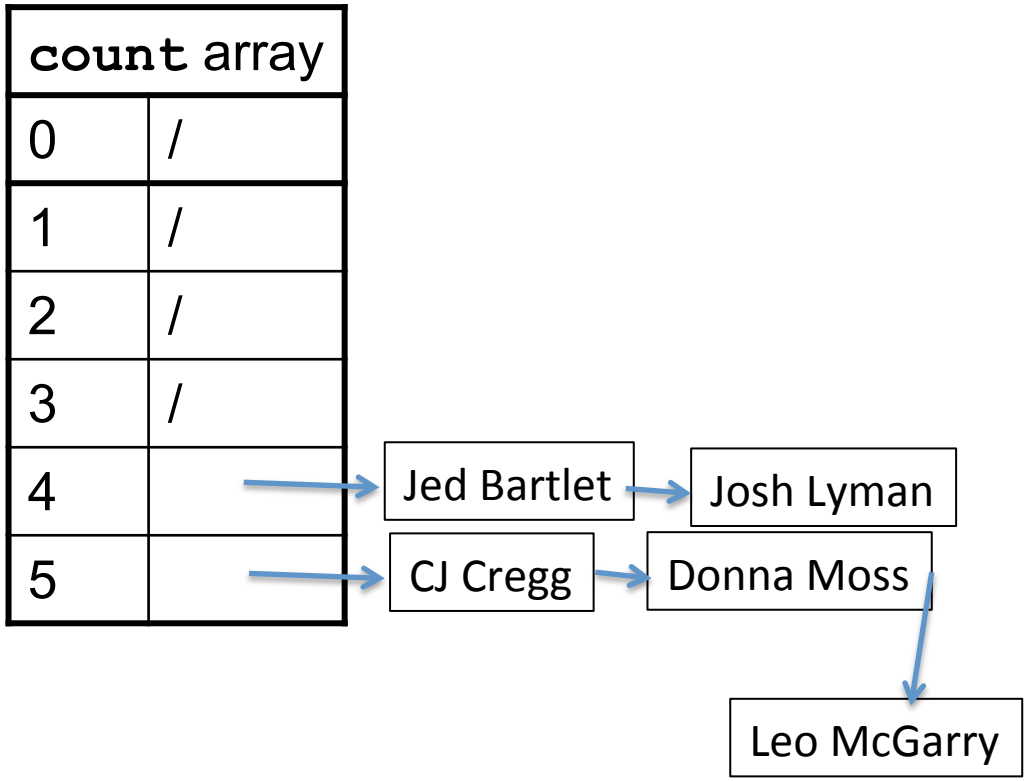


Example: Bucket Sort

What if we wanted to keep track of who got which score?

Quiz scores (0-5 possible):

- ~~CJ Cregg – 5~~
- ~~Donna Moss – 5~~
- ~~Jed Bartlet – 4~~
- ~~Josh Lyman – 4~~
- ~~Leo McGarry – 5~~
- Sam Seaborn – 2
- Toby Ziegler – 3

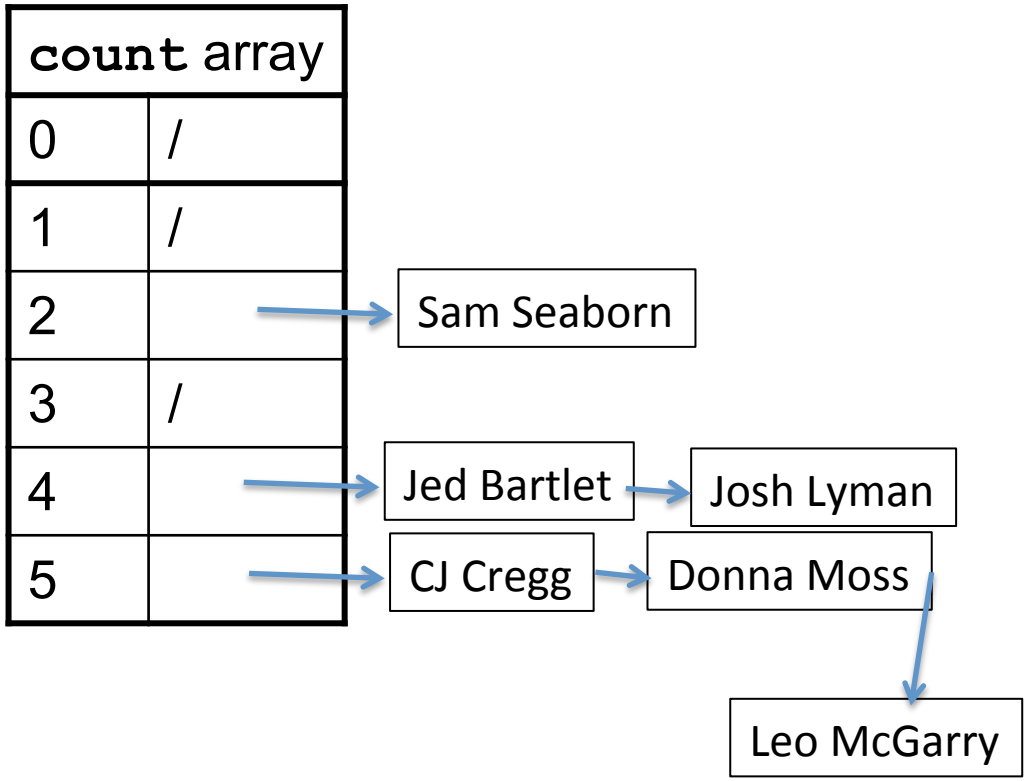


Example: Bucket Sort

What if we wanted to keep track of who got which score?

Quiz scores (0-5 possible):

- ~~CJ Cregg – 5~~
- ~~Donna Moss – 5~~
- ~~Jed Bartlet – 4~~
- ~~Josh Lyman – 4~~
- ~~Leo McGarry – 5~~
- ~~Sam Seaborn – 2~~
- Toby Ziegler – 3

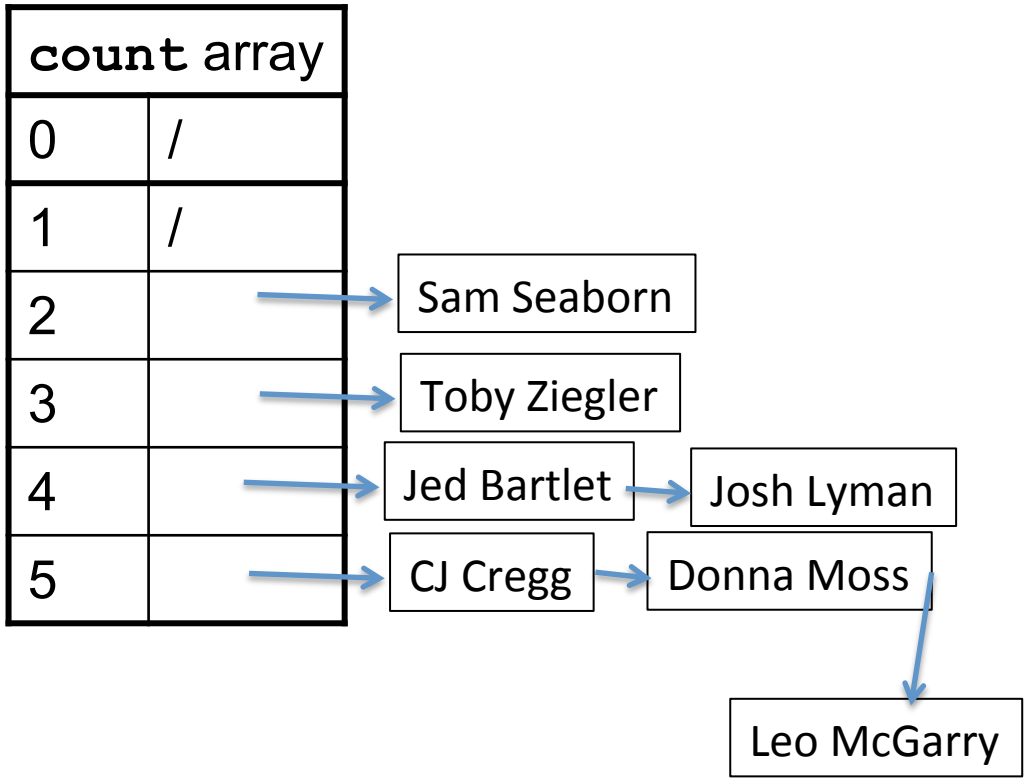


Example: Bucket Sort

What if we wanted to keep track of who got which score?

Quiz scores (0-5 possible):

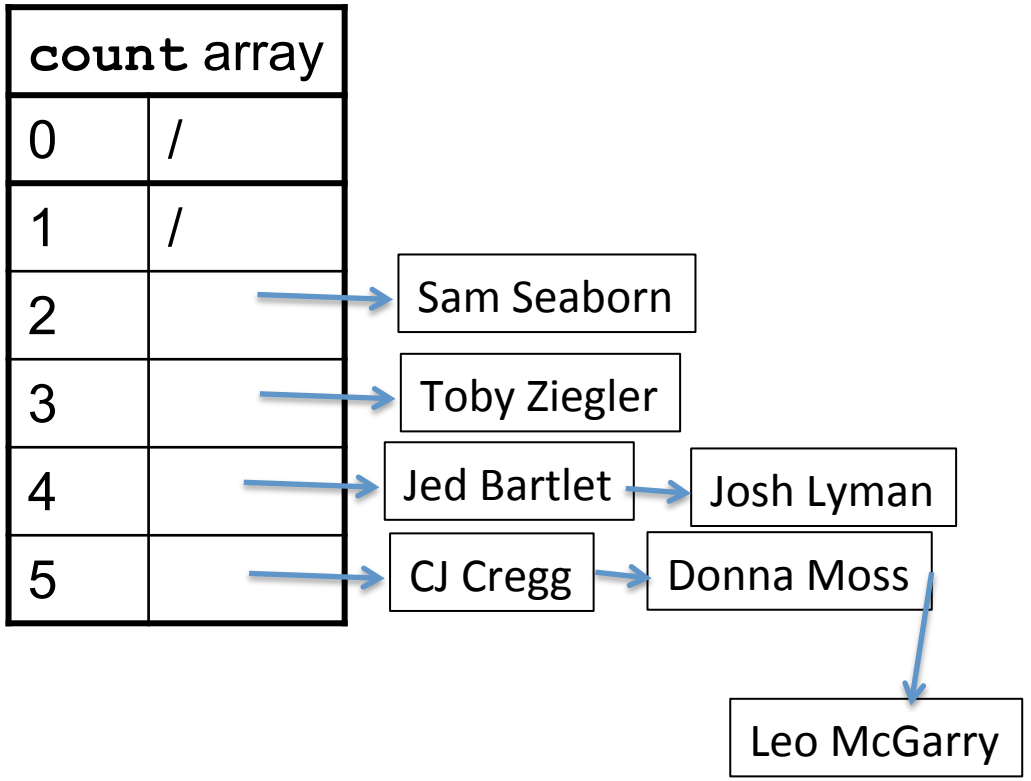
- ~~CJ Cregg – 5~~
- ~~Donna Moss – 5~~
- ~~Jed Bartlet – 4~~
- ~~Josh Lyman – 4~~
- ~~Leo McGarry – 5~~
- ~~Sam Seaborn – 2~~
- ~~Toby Ziegler – 3~~



Example: Bucket Sort

Final output:

(Sam Seaborn, 2), (Toby Ziegler, 3), (Jed Bartlet, 4), (Josh Lyman, 4), (CJ Cregg, 5), (Donna Moss, 5), (Leo McGarry, 5)



Example: Radix Sort

What if these scores were in the range (0-300)?

CJ Cregg – 51
Donna Moss – 125
Jed Bartlet – 243
Josh Lyman – 34
Leo McGarry – 299
Sam Seaborn – 103
Toby Ziegler – 9

The count array we were using before would be a lot larger and the complexity for the final pass would be O(K + N) where K = 301 and N =7

Radix Sort: Example

Radix = 10
(digits are 0-9)

0	1	2	3	4	5	6	7	8	9

Input:

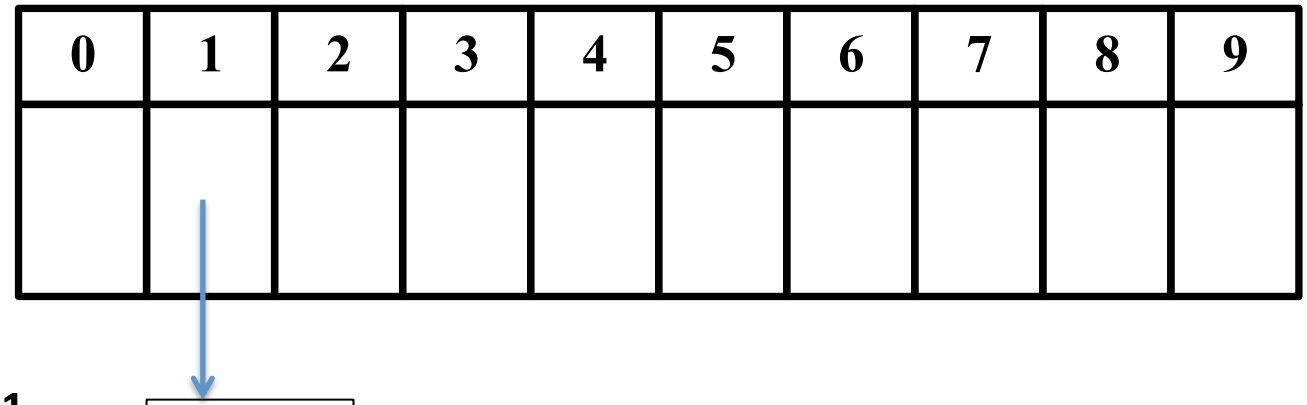
CJ Cregg – 05**1**
Donna Moss – 12**5**
Jed Bartlet – 24**3**
Josh Lyman – 03**4**
Leo McGarry – 29**9**
Sam Seaborn – 10**3**
Toby Ziegler – 00**9**

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Input:

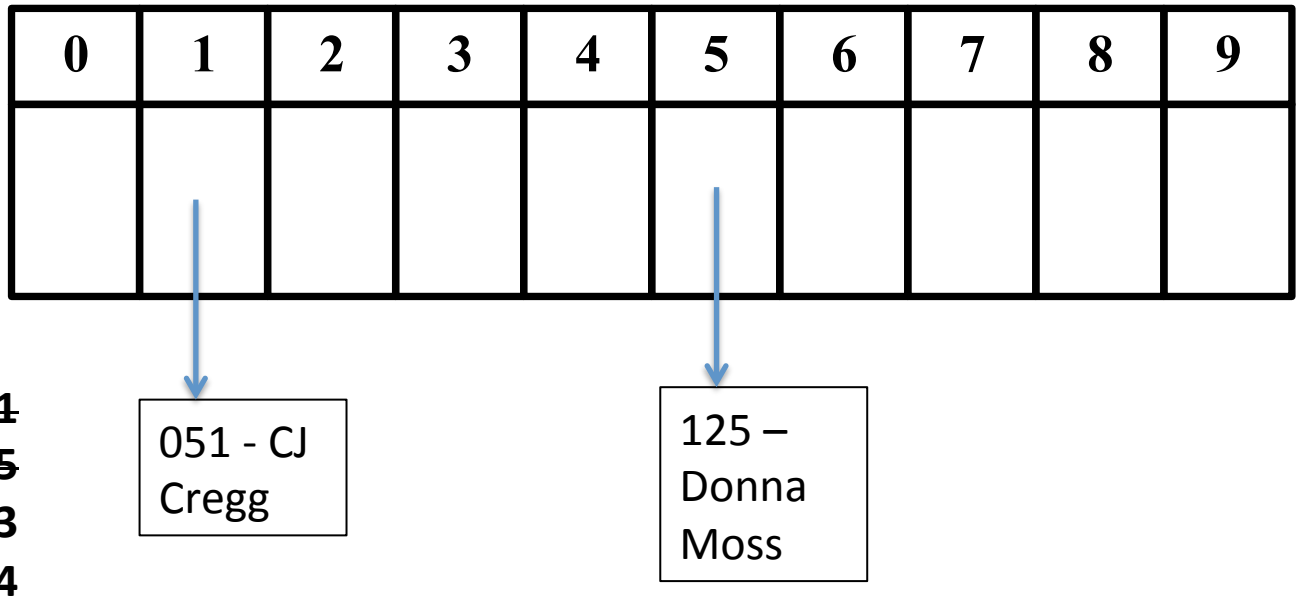
~~CJ Cregg – 051~~
Donna Moss – 12**5**
Jed Bartlet – 24**3**
Josh Lyman – 03**4**
Leo McGarry – 29**9**
Sam Seaborn – 10**3**
Toby Ziegler – 00**9**

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Input:

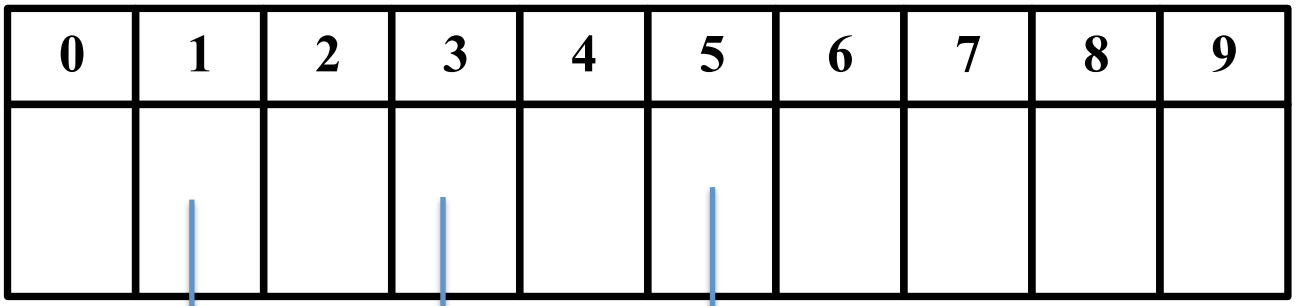
~~CJ Cregg – 051~~
~~Donna Moss – 125~~
Jed Bartlet – 24**3**
Josh Lyman – 03**4**
Leo McGarry – 29**9**
Sam Seaborn – 10**3**
Toby Ziegler – 00**9**

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



051 - CJ Cregg

243 – Jed Bartlet

125 – Donna Moss

Input:

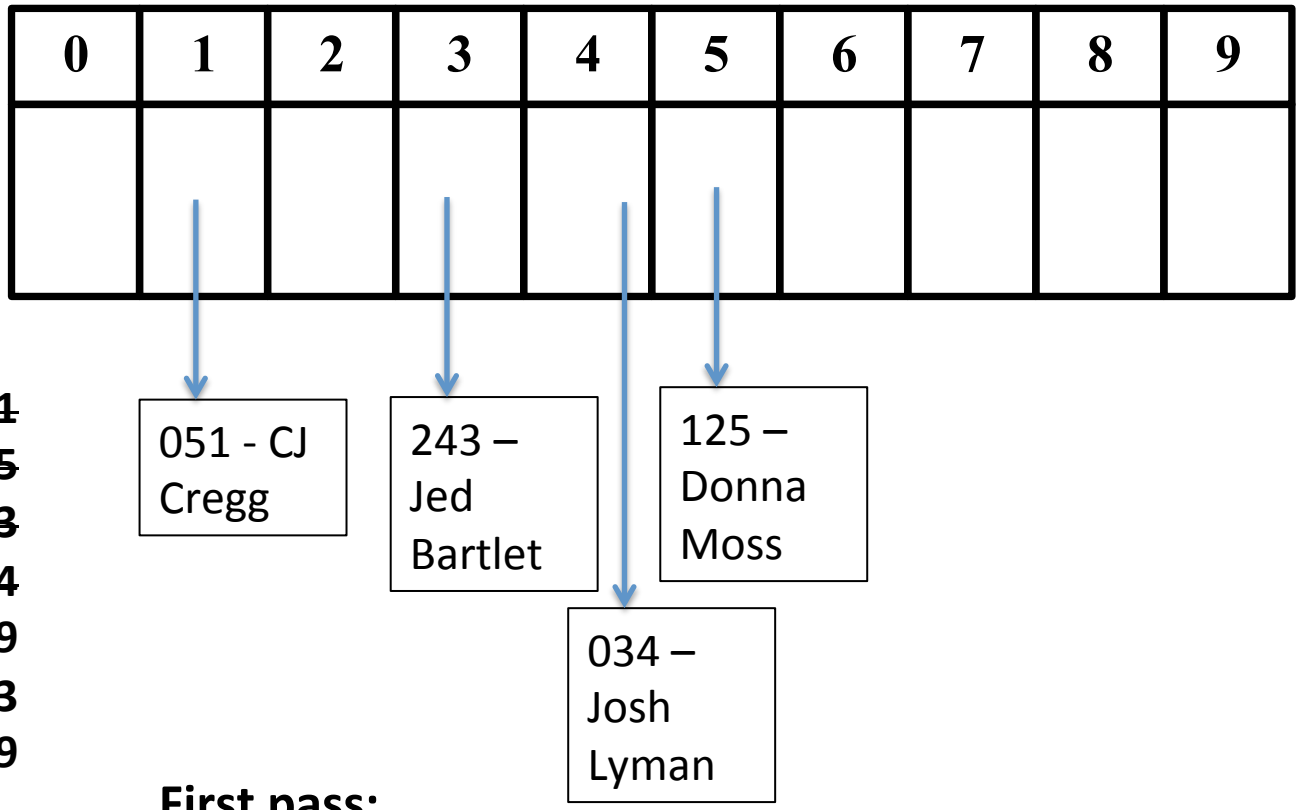
~~CJ Cregg – 051~~
~~Donna Moss – 125~~
~~Jed Bartlet – 243~~
Josh Lyman – 034
Leo McGarry – 299
Sam Seaborn – 103
Toby Ziegler – 009

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Input:

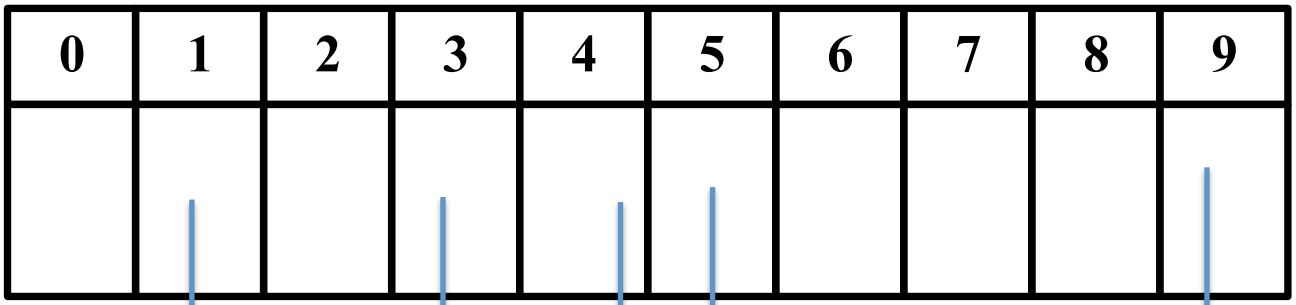
~~CJ Cregg – 051~~
~~Donna Moss – 125~~
~~Jed Bartlet – 243~~
~~Josh Lyman – 034~~
Leo McGarry – 299
Sam Seaborn – 103
Toby Ziegler – 009

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Input:

~~CJ Cregg – 051~~
~~Donna Moss – 125~~
~~Jed Bartlet – 243~~
~~Josh Lyman – 034~~
~~Leo McGarry – 299~~
Sam Seaborn – 103
Toby Ziegler – 009

051 - CJ Cregg

243 – Jed Bartlet

125 – Donna Moss

034 – Josh Lyman

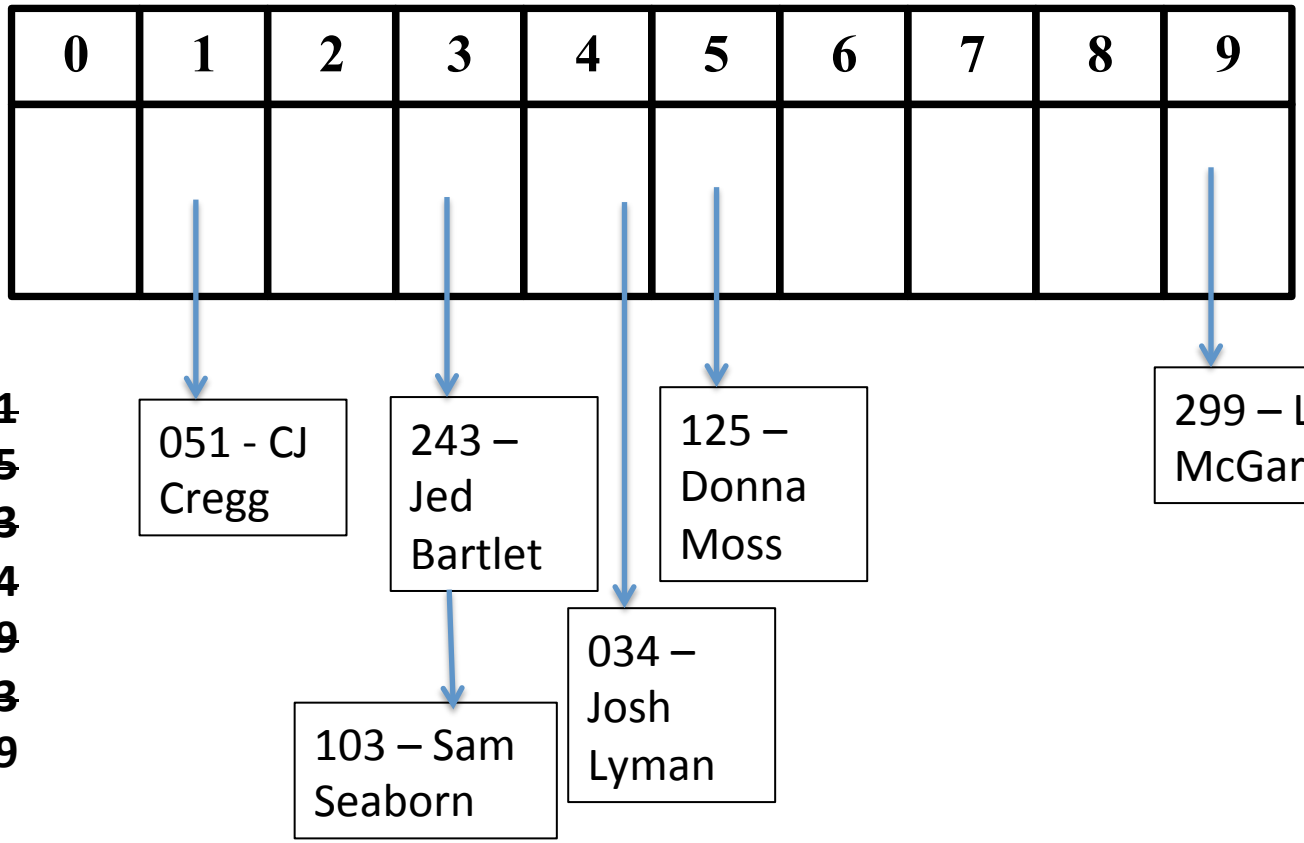
299 – Leo McGarry

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Input:

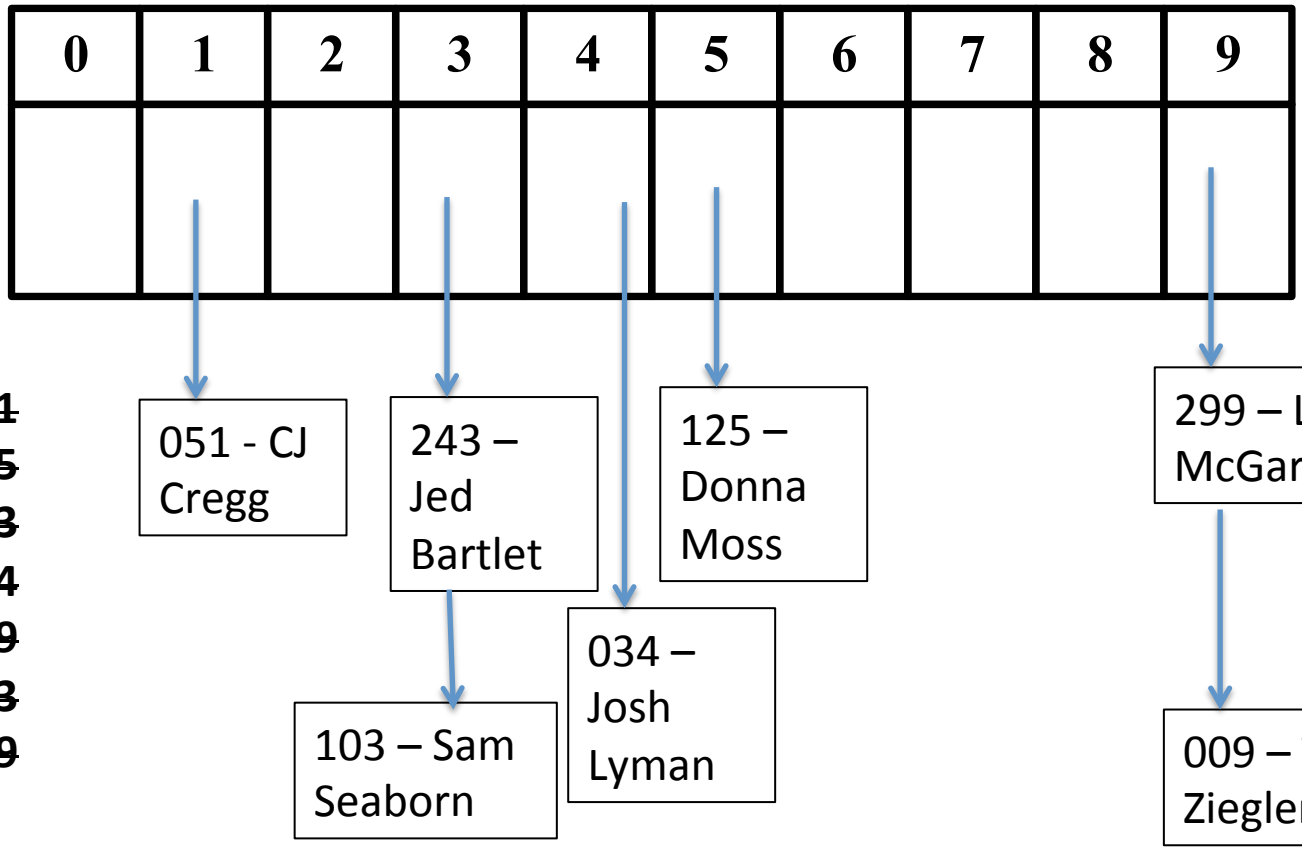
~~CJ Cregg – 051~~
~~Donna Moss – 125~~
~~Jed Bartlet – 243~~
~~Josh Lyman – 034~~
~~Leo McGarry – 299~~
~~Sam Seaborn – 103~~
Toby Ziegler – 009

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Input:

~~CJ Cregg – 051~~
~~Donna Moss – 125~~
~~Jed Bartlet – 243~~
~~Josh Lyman – 034~~
~~Leo McGarry – 299~~
~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~

First pass:

bucket sort input in given order by ones digit

Radix Sort: Example

Radix = 10
(digits are 0-9)

0	1	2	3	4	5	6	7	8	9

New Input Order:

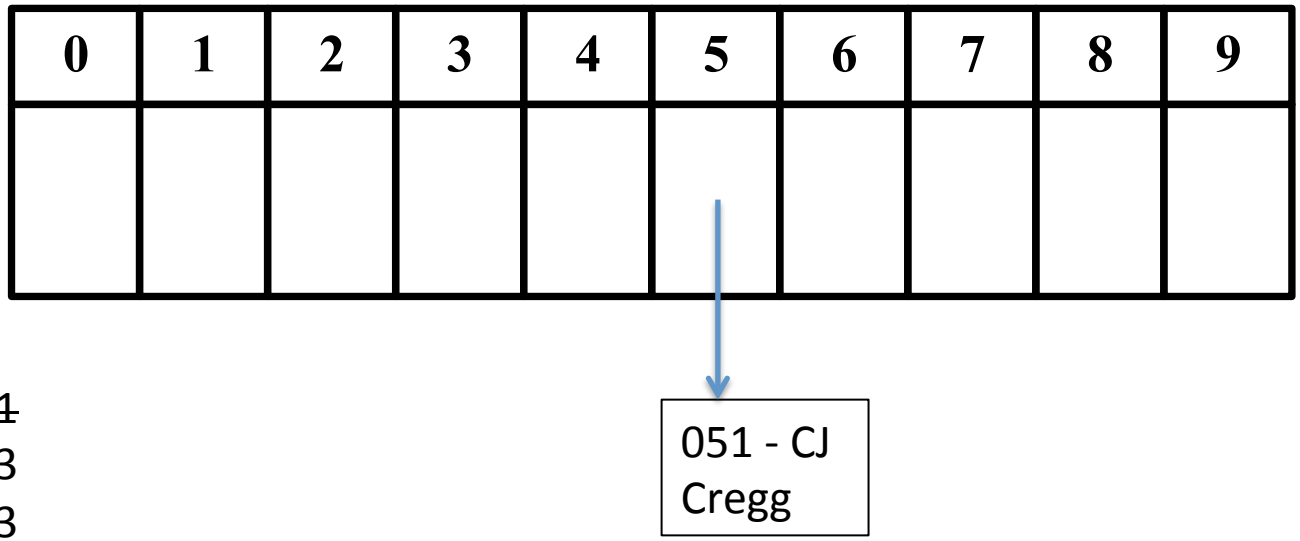
CJ Cregg – 0**5**1
Jed Bartlet – 2**4**3
Sam Seaborn – 1**0**3
Josh Lyman – 0**3**4
Donna Moss – 1**2**5
Leo McGarry – 2**9**9
Toby Ziegler – 0**0**9

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

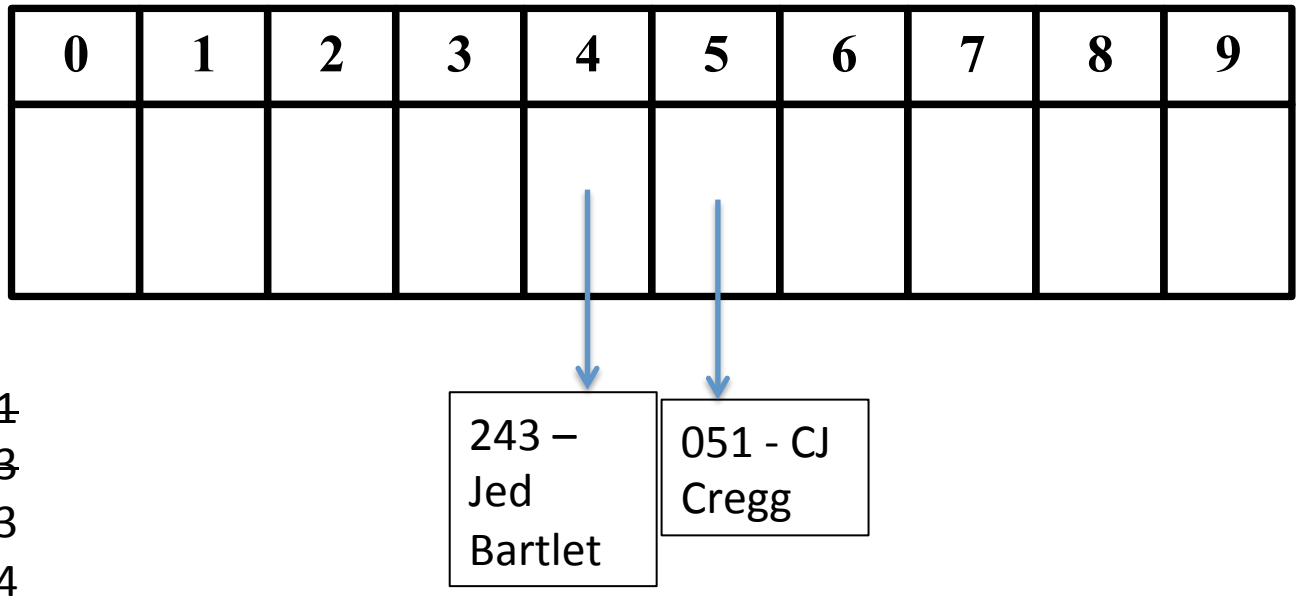
~~CJ Cregg – 0**5**1~~
Jed Bartlet – 2**4**3
Sam Seaborn – 1**0**3
Josh Lyman – 0**3**4
Donna Moss – 1**2**5
Leo McGarry – 2**9**9
Toby Ziegler – 0**0**9

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

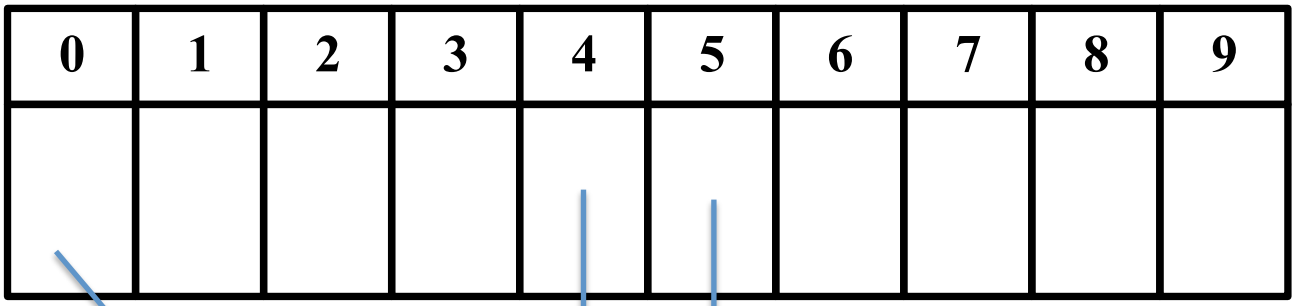
~~CJ Cregg – 0**5**1~~
~~Jed Bartlet – 2**4**3~~
Sam Seaborn – 1**0**3
Josh Lyman – 0**3**4
Donna Moss – 1**2**5
Leo McGarry – 2**9**9
Toby Ziegler – 0**0**9

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~CJ Cregg – 051~~
~~Jed Bartlet – 243~~
~~Sam Seaborn – 103~~
Josh Lyman – 034
Donna Moss – 125
Leo McGarry – 299
Toby Ziegler – 009

103 – Sam Seaborn

243 – Jed Bartlet

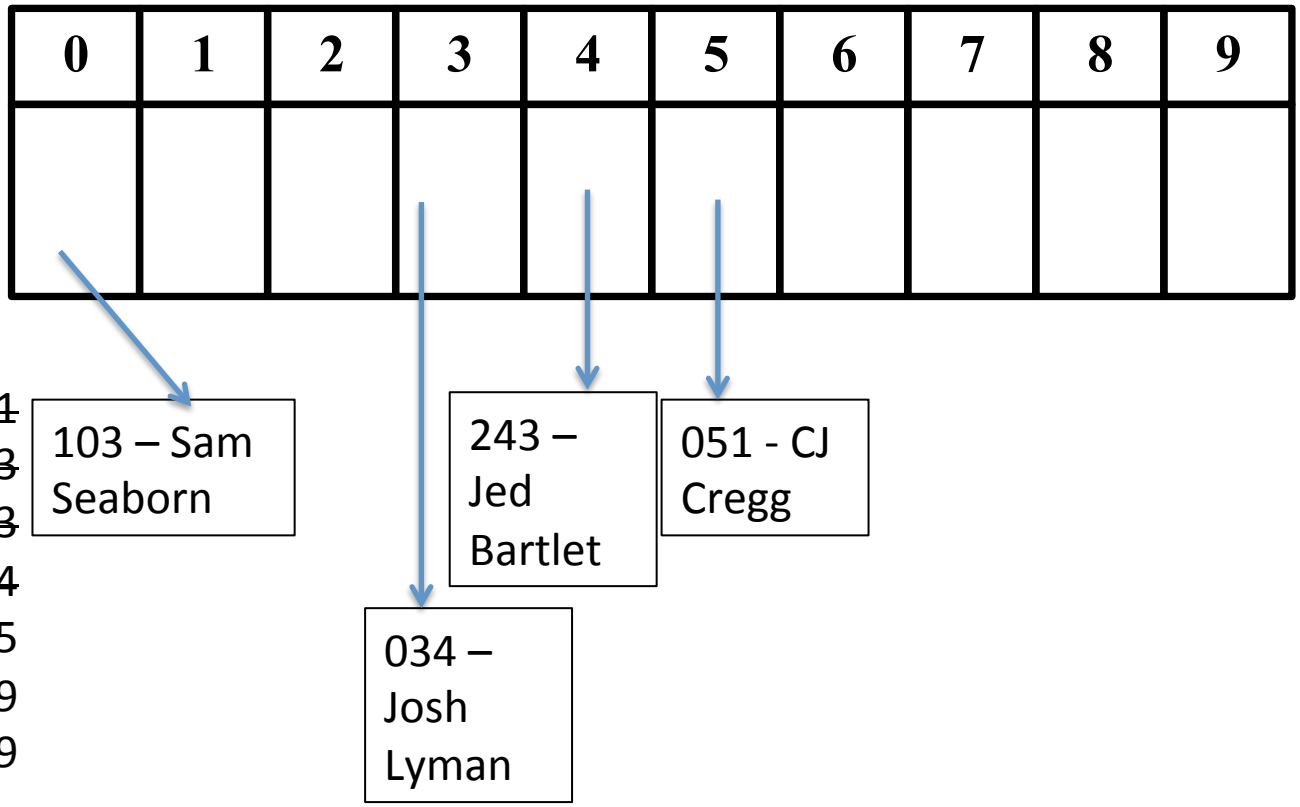
051 - CJ Cregg

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

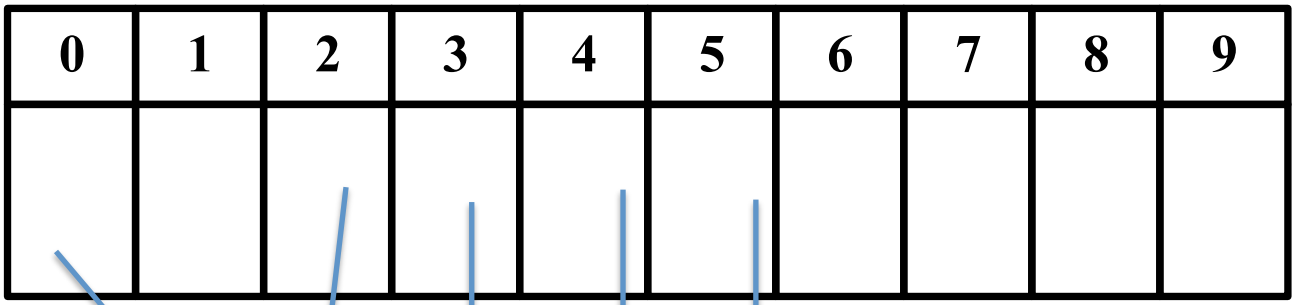
~~CJ Cregg – 0**5**1~~
~~Jed Bartlet – 2**4**3~~
~~Sam Seaborn – 1**0**3~~
~~Josh Lyman – 0**3**4~~
Donna Moss – 1**2**5
Leo McGarry – 2**9**9
Toby Ziegler – 0**0**9

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~CJ Cregg – 051~~
~~Jed Bartlet – 243~~
~~Sam Seaborn – 103~~
~~Josh Lyman – 034~~
~~Donna Moss – 125~~
Leo McGarry – 299
Toby Ziegler – 009

103 – Sam Seaborn

125 – Donna Moss

034 – Josh Lyman

243 – Jed Bartlet

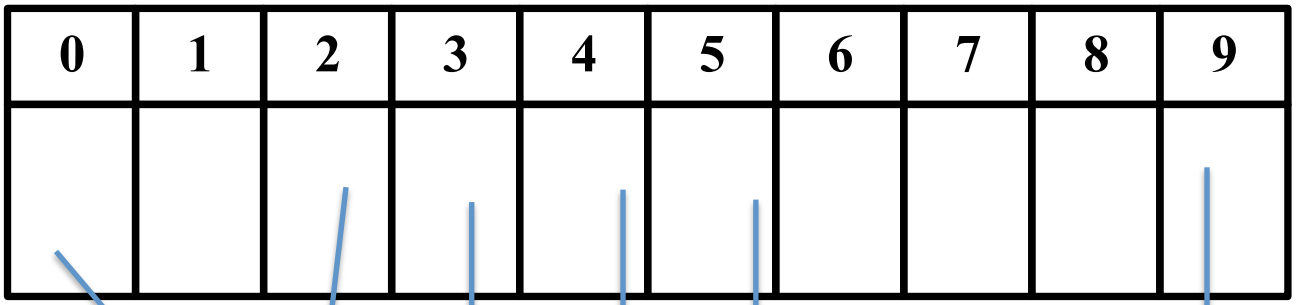
051 - CJ Cregg

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~CJ Cregg – 051~~
~~Jed Bartlet – 243~~
~~Sam Seaborn – 103~~
~~Josh Lyman – 034~~
~~Donna Moss – 125~~
~~Leo McGarry – 299~~
Toby Ziegler – 009

103 – Sam Seaborn

125 – Donna Moss

034 – Josh Lyman

243 – Jed Bartlet

051 - CJ Cregg

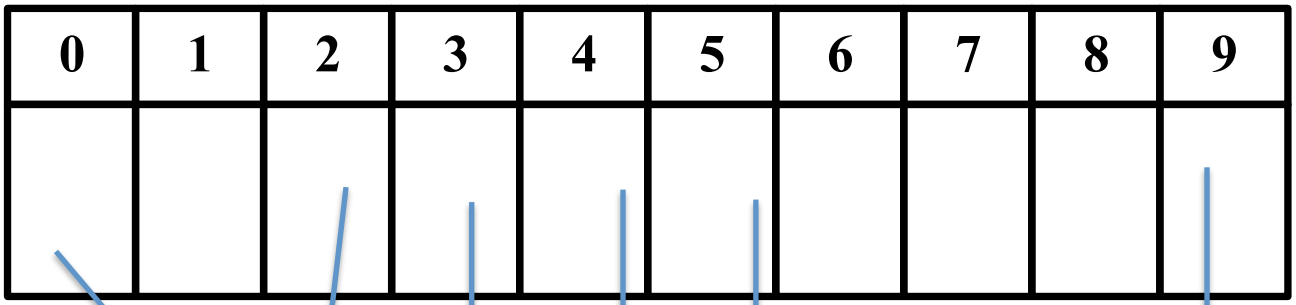
299 – Leo McGarry

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~CJ Cregg – 051~~
~~Jed Bartlet – 243~~
~~Sam Seaborn – 103~~
~~Josh Lyman – 034~~
~~Donna Moss – 125~~
~~Leo McGarry – 299~~
~~Toby Ziegler – 009~~

103 – Sam Seaborn

125 – Donna Moss

243 – Jed Bartlet

051 - CJ Cregg

299 – Leo McGarry

009 – Toby Ziegler

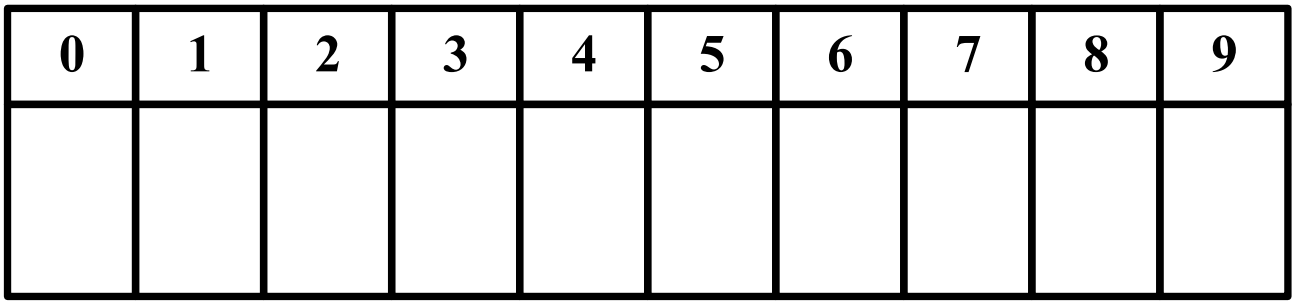
034 – Josh Lyman

Ready for second pass:

bucket sort input in given order by tens digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

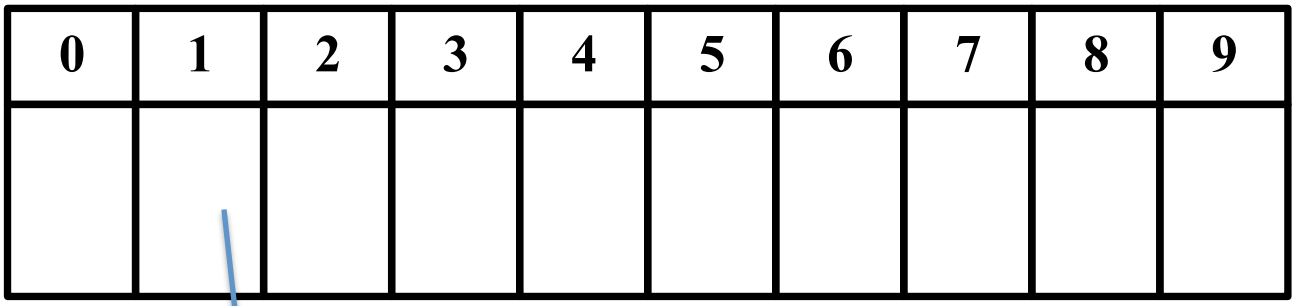
Sam Seaborn – 103
Toby Ziegler – 009
Donna Moss – 125
Josh Lyman – 034
Jed Bartlet – 243
CJ Cregg – 051
Leo McGarry – 299

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

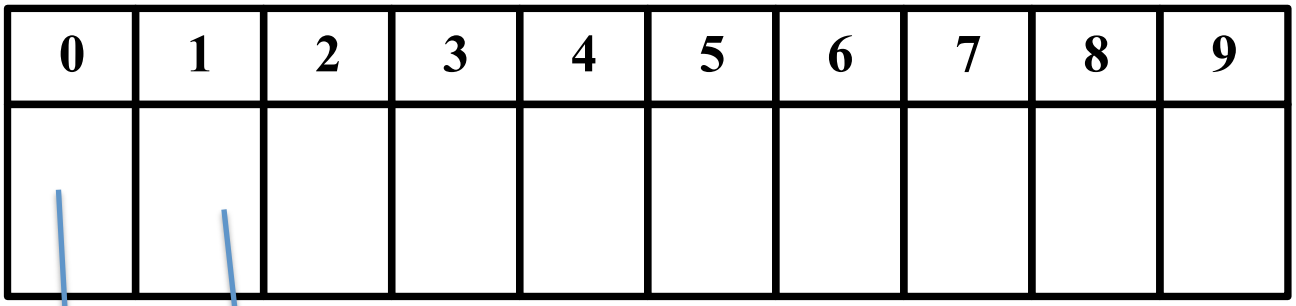
~~Sam Seaborn – 103~~
Toby Ziegler – 009
Donna Moss – 125
Josh Lyman – 034
Jed Bartlet – 243
CJ Cregg – 051
Leo McGarry – 299

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~
Donna Moss – 125
Josh Lyman – 034
Jed Bartlet – 243
CJ Cregg – 051
Leo McGarry – 299

103 – Sam Seaborn

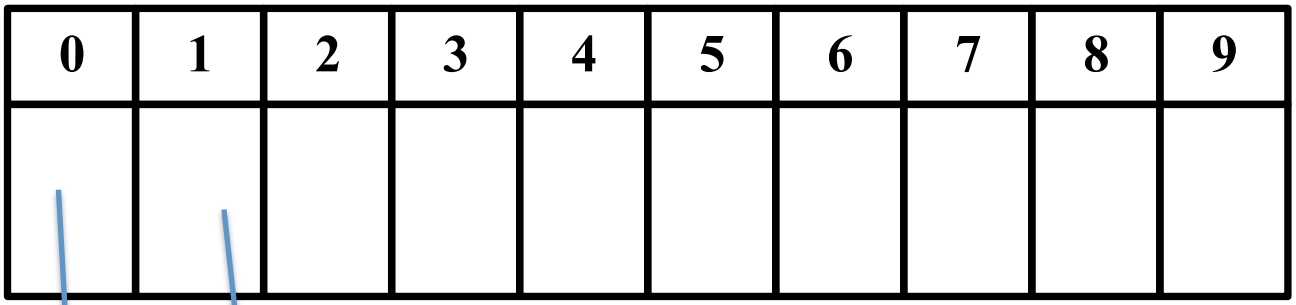
009 – Toby Ziegler

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~
~~Donna Moss – 125~~
Josh Lyman – 034
Jed Bartlet – 243
CJ Cregg – 051
Leo McGarry – 299

103 – Sam Seaborn

009 – Toby Ziegler

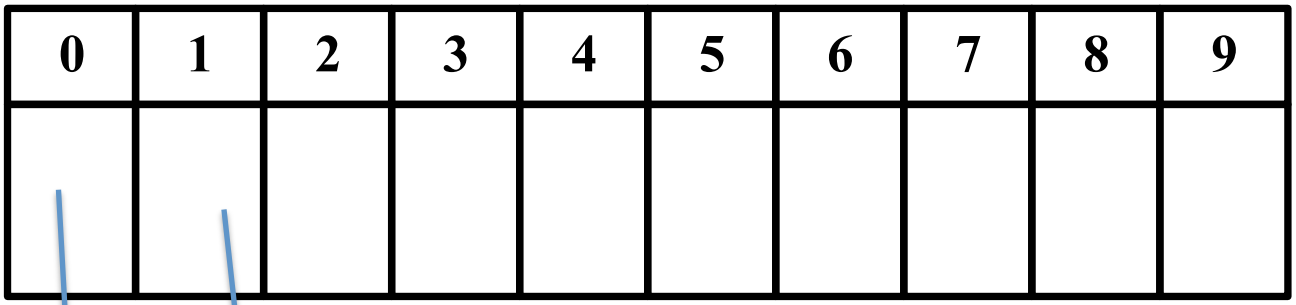
125 – Donna Moss

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~
~~Donna Moss – 125~~
~~Josh Lyman – 034~~
Jed Bartlet – 243
CJ Cregg – 051
Leo McGarry – 299

103 – Sam Seaborn

009 – Toby Ziegler

125 – Donna Moss

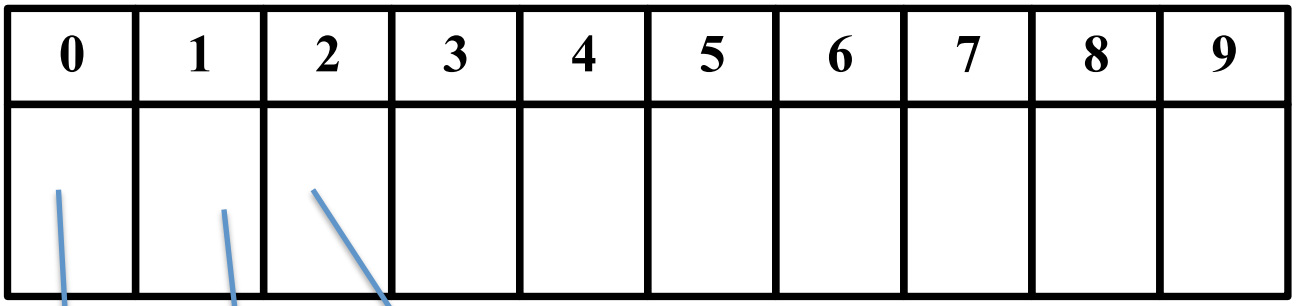
034 – Josh Lyman

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~
~~Donna Moss – 125~~
~~Josh Lyman – 034~~
~~Jed Bartlet – 243~~
CJ Cregg – 051
Leo McGarry – 299

103 – Sam Seaborn

243 – Jed Bartlet

009 – Toby Ziegler

125 – Donna Moss

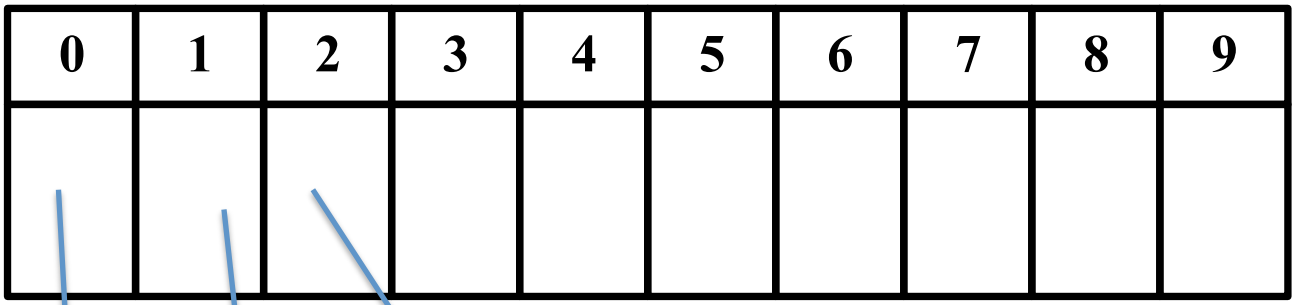
034 – Josh Lyman

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~
~~Donna Moss – 125~~
~~Josh Lyman – 034~~
~~Jed Bartlet – 243~~
~~CJ Cregg – 051~~
Leo McGarry – 299

103 – Sam Seaborn

243 – Jed Bartlet

009 – Toby Ziegler

125 – Donna Moss

034 – Josh Lyman

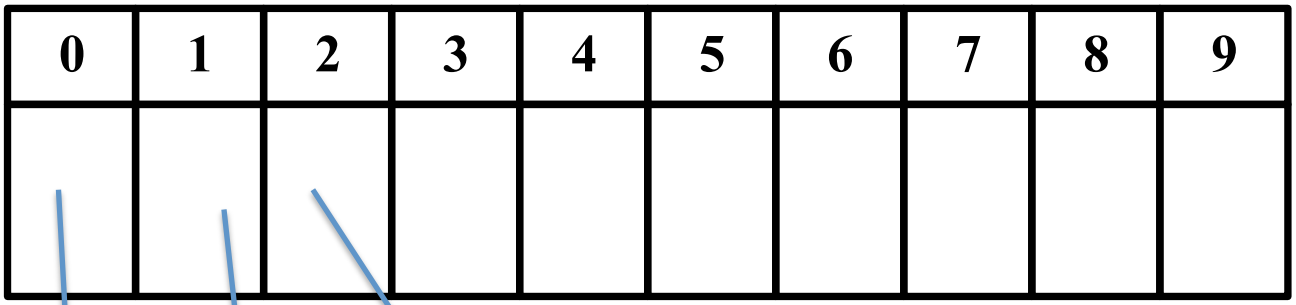
051 - CJ Cregg

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



New Input Order:

~~Sam Seaborn – 103~~
~~Toby Ziegler – 009~~
~~Donna Moss – 125~~
~~Josh Lyman – 034~~
~~Jed Bartlet – 243~~
~~CJ Cregg – 051~~
~~Leo McGarry – 299~~

103 – Sam Seaborn

243 – Jed Bartlet

299 – Leo McGarry

009 – Toby Ziegler

125 – Donna Moss

034 – Josh Lyman

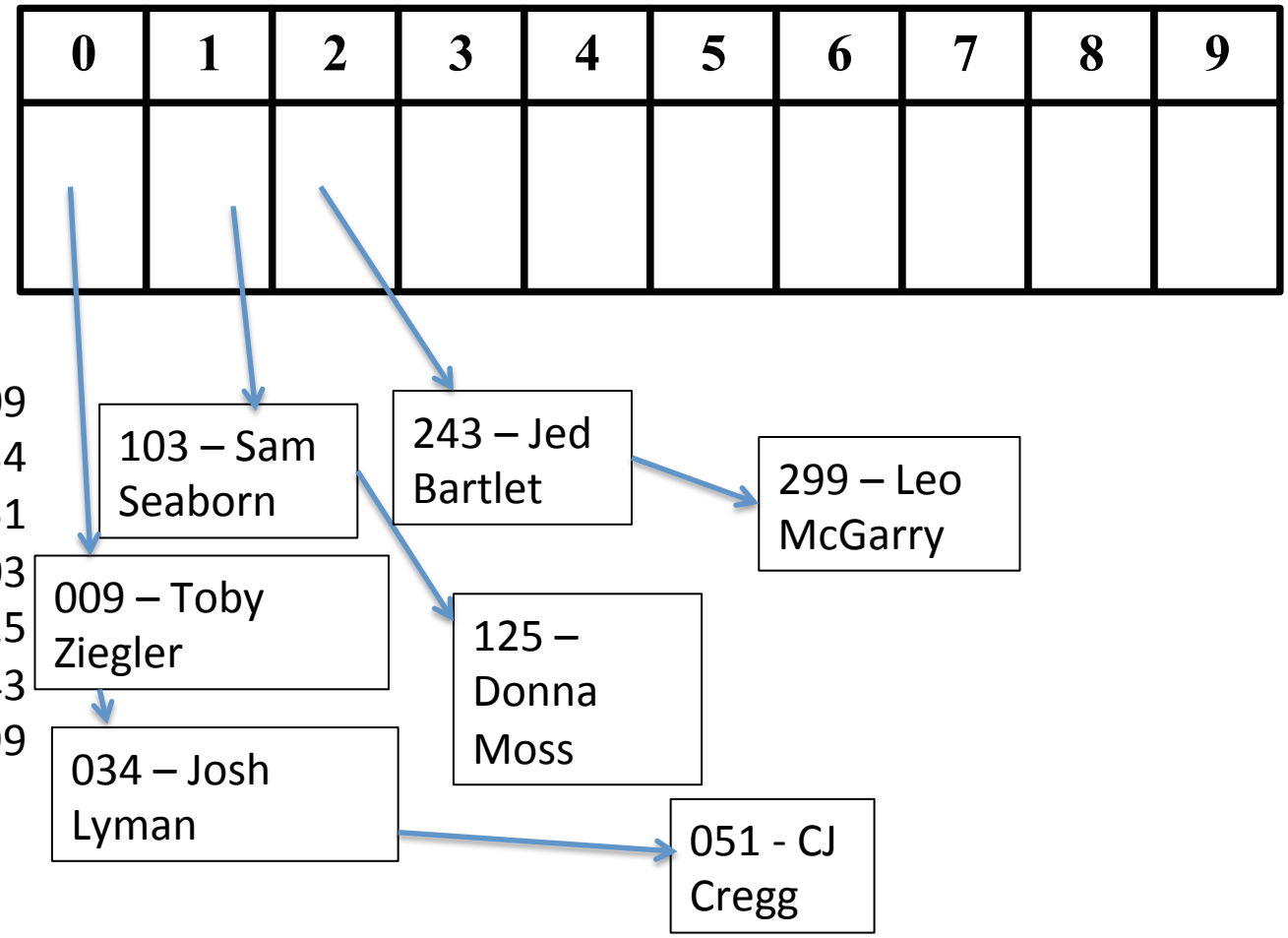
051 - CJ Cregg

Ready for third pass:

bucket sort input in given order by hundreds digit

Radix Sort: Example

Radix = 10
(digits are 0-9)



Final Order:

Toby Ziegler – 009
Josh Lyman – 034
CJ Cregg – 051
Sam Seaborn – 103
Donna Moss – 125
Jed Bartlet – 243
Leo McGarry – 299

Sorting Takeaways

- Simple $O(n^2)$ sorts can be fastest for small n
  - Selection sort, Insertion sort (latter linear for mostly-sorted)
  - Good for "below a cut-off" to help divide-and-conquer sorts
- $O(n \log n)$ sorts
  - Heap sort, in-place but not stable nor parallelizable
  - Merge sort, not in place but stable and works as external sort
  - Quick sort, in place but not stable and $O(n^2)$ in worst-case
    - Often fastest, but depends on costs of comparisons/copies
- $\Omega(n \log n)$ is worst-case and average lower-bound for sorting by comparisons
- Non-comparison sorts
  - Bucket sort good for small number of possible key values
  - Radix sort uses fewer buckets and more phases
- Best way to sort? It depends!

Switching Topics: Induction and Recurrences

recursive cat is recursive

Background on Induction

- Type of mathematical proof
- Typically used to establish a given statement for all natural numbers (integers > 0)
- Proof is a sequence of deductive steps
  1. Show the statement is true for the first number.
  2. Show that the statement is true for any one arbitrary number and then, this implies the statement is true for the next arbitrary number.
  3. If so, we can infer that the statement is true for all numbers.

Motivation

- Induction turns out to be a useful technique
  - proof of AVL trees log(n) worst case height
  - Heaps
  - proof of Graph algorithms
  - Can also prove things like $3^n > n^3$ *for* $n \geq 4$

- Exposure to rigorous thinking and proofs. Helps with recursion and recursive reasoning.

Think about climbing a ladder

1. Show you can get to the first rung (base case)

2. Show you can get between rungs (inductive step)

3. Now you can climb forever on an infinity ladder


INDUCTIVE REASONING
FOX NEWS
LIVE
the KELLY file
IT'S RECURSION, ESSENTIALLY
imgflip.com

5 steps to inductive proofs

1. State what you're trying to prove.
   - Suppose that P(n) is some predicate (mention n)
   - Ex:

     "Let P(n) be …
     Will prove that P(n) is true for every n >= x"
2. Prove the "base case"
   - Show that P(x) is true
3. Inductive Hypothesis (IH)
   - Assume that P(k) is true for some arbitrary integer k in the set of integers you're looking at
4. Inductive Step
   - Show that P(k + 1) is true.
   - Be sure to use the Inductive Hypothesis, and point out where you use it!
5. Conclusion. Conclude that you did it.

More specifically

$$P(0)$$
$$\forall\, k\, (P(k) \rightarrow P(k+1))$$

$$\therefore\ \forall\, n\, P(n)$$

1. Prove P(0) — **Base Case**

2. Let k be an arbitrary integer ≥ 0
   3.1. Assume that P(k) is true — **Inductive Hypothesis**

   3.2. …
   3.3. Prove P(k+1) is true — **Inductive Step**

3. P(k) → P(k+1)	Direct Proof Rule
4. ∀k (P(k) → P(k+1))	Intro ∀: 2, 3
5. ∀n P(n)	Induction: 1, 4

Conclusion

Example

Prove: $1 + 2 + 4 + 8 + \ldots + 2^n = 2^{n+1}-1$ for all $n >= 0$

(see handwritten notes)

Let $P(n)$ be "$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$". We go by induction on n.

Base Case (n=0):

Note that $2^0 = 1 = 2 - 1 = 2^{0+1} - 1$, which is exactly $P(0)$.

Induction Hypothesis:

Suppose P(k) is true for some k $\in \mathbb{N}$.

Induction Step:

We want to show P($k+1$). That is, we want to show:

$$\sum_{i=0}^{k+1} 2^i = 2^{(k+1)+1} - 1$$

$\sum_{i=0}^{k+1} 2^i$ = ... = ... = ... $= 2^{n+1} - 1$.

One of these steps must use the IH.

So, the claim is true for all natural numbers by induction.

Let $P(n)$ be "$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$". We go by induction on n.

Base Case (n=0): Note that $2^0 = 1 = 2 - 1 = 2^{0+1} - 1$, which is exactly $P(0)$.

Induction Hypothesis: Suppose $P(k)$ is true for some $k \in \mathbb{N}$.

Induction Step: We want to show $P(k+1)$. That is, we want to show: $\sum_{i=0}^{k+1} 2^i = 2^{(k+1)+1} - 1$

Note that $\sum_{i=0}^{k+1} 2^i = \left(\sum_{i=0}^{k} 2^i\right) + 2^{k+1}$ [Splitting the summation]

$= (2^{k+1} - 1) + 2^{k+1}$ [By IH]

$= (2^{k+1} + 2^{k+1}) - 1$ [Assoc. of +]

$= (2(2^{k+1})) - 1$ [Factoring]

$= 2^{k+2} - 1$ [Simplifying]

Don't bother justifying the "obvious" steps. But make sure you say "by IH" somewhere.

This is exactly $P(k+1)$. So, $P(k) \rightarrow P(k+1)$.

So, the claim is true for all natural numbers by induction.

We know (by IH)...

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

We're trying to get...

$$\sum_{i=0}^{k+1} 2^i = 2^{(k+1)+1} - 1$$

Our goal is to find a sub-expression of the left that looks like the left side of the IH.

Asymptotic Runtime of Recursion

Recurrence Definition:

A recurrence is a recursive definition of a function in terms of smaller values.

Example: Fibonacci numbers.

To analyze the runtime of recursive code, we use a recurrence by splitting the work into two pieces:

- Non-Recursive Work
- Recursive Work

Recursive version of sum:

```
int sum(int[] arr){
    return help(arr,0,arr.length);
}
int help(int[] arr, int lo, int hi) {
    if(lo==hi)    return 0;
    if(lo==hi-1) return arr[lo];
    int mid = (hi+lo)/2;
    return help(arr,lo,mid) + help(arr,mid,hi);
}
```

What's the recurrence T(n)?

- Non-Recursive Work: O(1)
- Recursive Work: T(n/2) * 2 halves

T(n) = O(1) + 2*T(n/2)

Solving That Recurrence Relation

1. Determine the recurrence relation. What is the base case?
   - *If* $T(1) = 1$, then $T(n) = 1 + 2*T(n/2)$

2. "Expand" the original relation to find an equivalent general expression *in terms of the number of expansions*.
   - $T(n) = 1 + 2 * T(n / 2)$
     $= 1 + 2 + 2 * T(n / 4)$
     $= 1 + 2 + 4 + \ldots$ for log(n) times
     $= \ldots$
     $= 2^{(\log n)} - 1$

3. Find a closed-form expression by setting *the number of expansions* to a value which reduces the problem to a base case
   - So $T(n)$ is $O(n)$

- Explanation: it adds each number once while doing little else

Solving Recurrence Relations Example 2

1. Determine the recurrence relation. What is the base case?
   - *If* $T(n) = 10 + T(n/2)$ and $T(1) = 10$

2. "Expand" the original relation to find an equivalent general expression *in terms of the number of expansions*.
   - $T(n) = 10 + 10 + T(n/4)$
     $= 10 + 10 + 10 + T(n/8)$
     $= \ldots$
     $= 10k + T(n/(2^k))$

3. Find a closed-form expression by setting *the number of expansions* to a value which reduces the problem to a base case
   - $n/(2^k) = 1$ means $n = 2^k$ means $k = \log_2 n$
   - So $T(n) = 10 \log_2 n + 8$ (get to base case and do it)
   - So $T(n)$ is $O(\log n)$

Really common recurrences

You can recognize some really common recurrences:

$T(n) = O(1) + T(n\text{-}1)$	linear
$T(n) = O(1) + 2T(n/2)$	linear
$T(n) = O(1) + T(n/2)$	logarithmic $O(\log n)$
$T(n) = O(1) + 2T(n\text{-}1)$	exponential
$T(n) = O(n) + T(n\text{-}1)$	quadratic
$T(n) = O(n) + T(n/2)$	linear
$T(n) = O(n) + 2T(n/2)$	$O(n \log n)$ (divide and conquer sort)

Note big-Oh can also use more than one variable

- Example: can sum all elements of an n-by-m matrix in $O(nm)$

Parallelism teaser

- But suppose we could do two recursive calls *at the same time*
  - *Like having a friend do half the work for you!*

```
int sum(int[]arr){
   return help(arr,0,arr.length);
}
int help(int[]arr, int lo, int hi) {
   if(lo==hi)    return 0;
   if(lo==hi-1) return arr[lo];
   int mid = (hi+lo)/2;
   return help(arr,lo,mid) + help(arr,mid,hi);
}
```

- If you have as many "friends of friends" as needed the recurrence is now $T(n) = O(1) + 1T(n/2)$
  - $O(\log n)$: same recurrence as for `find`