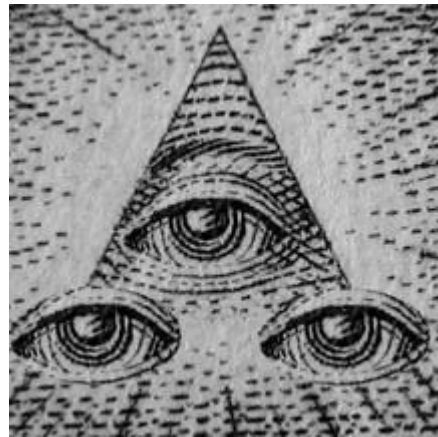


CS61B



Lecture 35: Radix Sort

- Counting Sort
- LSD Radix Sort
- LSD Radix Sort vs. Comparison Sorting
- MSD Radix Sort

Comparison Based Sorting

The key idea from our previous sorting lecture: Sorting requires $\Omega(N \log N)$ compares in the worst case.

- Thus, the ultimate comparison based sorting algorithm has a worst case runtime of $\Theta(N \log N)$.

From an asymptotic perspective, that means no matter how clever we are, we can never beat Merge Sort's worst case runtime of $\Theta(N \log N)$.

- ...but what if we don't compare at all?

Example #1: Sleep Sort (for Sorting Integers) (not actually good)

For each integer x in array A , start a new program that:

- Sleeps for x seconds.
- Prints x .

All start at the same time.

Runtime:

- $N + \max(A)$



The catch: On real machines, scheduling execution of programs must be done by an operating system. In practice requires list of running programs sorted by sleep time.

Genius sorting algorithm: Sleep sort

1 Name: **Anonymous** 2011-01-20 12:22

Man, am I a genius. Check out this sorting algorithm I just invented.

```
#!/bin/bash
function f() {
    sleep "$1"
    echo "$1"
}
while [ -n "$1" ]
do
    f "$1" &
    shift
done
wait
```

example usage:

```
./sleepsort.bash 5 3 6 3 6 3 1 4 7
```

Invented by 4chan.

Example #2: Counting Sort: Exploiting Space Instead of Time

#			
5	Sandra	Vanilla	Grimes
0	Lauren	Mint	Jon Talabot
11	Lisa	Vanilla	Blue Peter
9	Dave	Chocolate	Superpope
4	JS	Fish	The Filthy Reds
7	James	Rocky Road	Robots are Supreme
3	Edith	Vanilla	My Bloody Valentine
6	Swimp	Chocolate	Sef
1	Delbert	Strawberry	Ronald Jenkees
2	Glaser	Cardamom	Rx Nightly
8	Lee	Vanilla	La(r)va
10	Bearman	Butter Pecan	Extrobophile

Assuming keys are unique integers 0 to 11.

Idea:

- Create a new array.
- Copy item with key i into i th entry of new array.

Example #2: Counting Sort: Exploiting Space Instead of Time

#			
5	Sandra	Vanilla	Grimes
0	Lauren	Mint	Jon Talabot
11	Lisa	Vanilla	Blue Peter
9	Dave	Chocolate	Superpope
4	JS	Fish	The Filthy Reds
7	James	Rocky Road	Robots are Supreme
3	Edith	Vanilla	My Bloody Valentine
6	Swimp	Chocolate	Sef
1	Delbert	Strawberry	Ronald Jenkees
2	Glaser	Cardamom	Rx Nightly
8	Lee	Vanilla	La(r)va
10	Bearman	Butter Pecan	Extrobophile

#			
5	Sandra	Vanilla	Grimes

Example #2: Counting Sort: Exploiting Space Instead of Time

#			
5	Sandra	Vanilla	Grimes
0	Lauren	Mint	Jon Talabot
11	Lisa	Vanilla	Blue Peter
9	Dave	Chocolate	Superpope
4	JS	Fish	The Filthy Reds
7	James	Rocky Road	Robots are Supreme
3	Edith	Vanilla	My Bloody Valentine
6	Swimp	Chocolate	Sef
1	Delbert	Strawberry	Ronald Jenkees
2	Glaser	Cardamom	Rx Nightly
8	Lee	Vanilla	La(r)va
10	Bearman	Butter Pecan	Extrobophile

#			
0	Lauren	Mint	Jon Talabot
5	Sandra	Vanilla	Grimes

Example #2: Counting Sort: Exploiting Space Instead of Time

#			
5	Sandra	Vanilla	Grimes
0	Lauren	Mint	Jon Talabot
11	Lisa	Vanilla	Blue Peter
9	Dave	Chocolate	Superpope
4	JS	Fish	The Filthy Reds
7	James	Rocky Road	Robots are Supreme
3	Edith	Vanilla	My Bloody Valentine
6	Swimp	Chocolate	Sef
1	Delbert	Strawberry	Ronald Jenkees
2	Glaser	Cardamom	Rx Nightly
8	Lee	Vanilla	La(r)va
10	Bearman	Butter Pecan	Extrobophile

#			
0	Lauren	Mint	Jon Talabot
5	Sandra	Vanilla	Grimes
11	Lisa	Vanilla	Blue Peter

Example #2: Counting Sort: Exploiting Space Instead of Time

#			
5	Sandra	Vanilla	Grimes
0	Lauren	Mint	Jon Talabot
11	Lisa	Vanilla	Blue Peter
9	Dave	Chocolate	Superpope
4	JS	Fish	The Filthy Reds
7	James	Rocky Road	Robots are Supreme
3	Edith	Vanilla	My Bloody Valentine
6	Swimp	Chocolate	Sef
1	Delbert	Strawberry	Ronald Jenkees
2	Glaser	Cardamom	Rx Nightly
8	Lee	Vanilla	La(r)va
10	Bearman	Butter Pecan	Extrobophile

#			
0	Lauren	Mint	Jon Talabot
1	Delbert	Strawberry	Ronald Jenkees
2	Glaser	Cardamom	Rx Nightly
3	Edith	Vanilla	My Bloody Valentine
4	JS	Fish	The Filthy Reds
5	Sandra	Vanilla	Grimes
6	Swimp	Chocolate	Sef
7	James	Rocky Road	Robots are Supreme
8	Lee	Vanilla	La(r)va
9	Dave	Chocolate	Superpope
10	Bearman	Butter Pecan	Extrobophile
11	Lisa	Vanilla	Blue Peter

Generalizing Counting Sort

We just sorted N items in $\Theta(N)$ worst case time.

- Avoiding yes/no questions lets us dodge our lower bound based on puppy, cat, dog!

Simplest case:

- Keys are unique integers from 0 to $N-1$.

More complex cases:

- Non-unique keys.
- Non-consecutive keys.
- Non-numerical keys.

Counting Sort: <http://yellkey.com/carry>

Alphabet case: Keys belong to a finite ordered alphabet.

- Example: {♣, ♠, ♥, ♦} (in that order)

♠	Lauren
♥	Delbert
♦	Glaser
♣	Edith
♠	JS
♦	Sandra
♥	Swimp
♥	James
♣	Lee
♥	Dave
♣	Bearman
♦	Lisa

Question: What will be the index of the first ♥?

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Sorted

Counting Sort

Alphabet case: Keys belong to a finite ordered alphabet.

- Example: {♣, ♠, ♥, ♦} (in that order)

♠	Lauren
♥	Delbert
♦	Glaser
♣	Edith
♠	JS
♦	Sandra
♥	Swimp
♥	James
♣	Lee
♥	Dave
♣	Bearman
♦	Lisa

Question: What will be the index of the first ♥?

0	♣	
1	♣	
2	♣	
3	♠	
4	♠	
5		
6		
7		
8		
9		
10		
11		

Sorted

Implementing Counting Sort with Counting Arrays

Counting sort:

- Count number of occurrences of each item.
- Iterate through list, using count array to decide where to put everything.
- Interactive [Demo](#)

Bottom line, we can use counting sort to sort N objects in $\Theta(N)$ time.

Counting Sort Runtime

Counting Sort vs. Quicksort: <http://yellkey.com/slack>

For sorting an array of the 100 largest cities by population, which sort do you think has a better expected worst case runtime in seconds?

- A. Counting Sort (as described in our demo)
- B. Quicksort

First question to ask yourself: What is the alphabet for counting sort here?

Counting Sort vs. Quicksort

For sorting an array of the 100 largest cities by population, which sort do you think has a better expected worst case runtime in seconds?

- A. Counting Sort (as described in our demo)
- B. Quicksort**

Counting sort requires building an array of size 37,832,892 (population of Tokyo).

6352254	Ahmedabad
4778000	Alexandria
5346518	Ankara
6555956	Atlanta
8495928	Bandung
12517749	Bangalore
...	...



...	...
4777999	0
4778000	1
4778001	0
4778002	0
...	...
37832892	1



...

Counts

Counting Sort Runtime Analysis

What is the runtime for counting sort on N keys with alphabet of size R ?

- Treat R as a variable, not a constant.

Counting Sort Runtime Analysis

Total runtime on N keys with alphabet of size R : $\Theta(N+R)$

- Create an array of size R to store counts: $\Theta(R)$
- Counting number of each item: $\Theta(N)$
- Calculating target positions of each item: $\Theta(R)$
- Creating an array of size N to store ordered data: $\Theta(N)$
- Copying items from original array to ordered array: Do N times:
 - Check target position: $\Theta(1)$
 - Update target position: $\Theta(1)$
- Copying items from ordered array back to original array: $\Theta(N)$

For ordered array.

For counts and starting points.

Memory usage: $\Theta(N+R)$

Empirical experiments needed to compare vs. Quicksort on practical inputs.

Bottom line: If $N \geq R$, then we expect reasonable performance.

Counting Sort vs. Quicksort: <http://yellkey.com/improve>

For sorting really really really big collections of items from some alphabet, which algorithm will be fastest?

- A. Counting Sort
- B. Quicksort

Counting Sort vs. Quicksort

For sorting really really really big collections of items from some alphabet, which algorithm will be fastest?

A. **Counting Sort:** $\Theta(N+R)$

B. Quicksort: $\Theta(N \log N)$

For sufficiently large collections, counting sort will simply be faster.

Sort Summary

	Memory	Runtime	Notes	Stable?
Heapsort	$\Theta(1)$	$\Theta(N \log N)$	Bad caching (61C)	No
Insertion	$\Theta(1)$	$\Theta(N^2)$	Small N, almost sorted	Yes
Mergesort	$\Theta(N)$	$\Theta(N \log N)$	Fastest stable	Yes
Random Quicksort	$\Theta(\log N)$	$\Theta(N \log N)$ expected	Fastest compare sort	No
Counting Sort	$\Theta(N+R)$	$\Theta(N+R)$	Alphabet keys only	Yes

N: Number of keys. R: Size of alphabet.

Counting sort is nice, but alphabetic restriction limits usefulness.

- No obvious way to sort hard-to-count things like Strings.

LSD Radix Sort

Radix Sort

Not all keys belong to finite alphabets, e.g. Strings.

- However, Strings consist of characters from a finite alphabet.

horse	Lauren
elf	Delbert
cat	Glaser
crab	Edith
monkey	JS
rhino	Sandra
raccoon	Swimp
cat	James
fish	Lee
tree	Dave
virus	Bearman
human	Lisa

♠♠	Lauren
♥♦	Delbert
♦♣	Glaser
♣♥	Edith
♠♥	JS
♦♣	Sandra
♥♠	Swimp
♥♦	James
♣♠	Lee
♥♣	Dave
♣♠	Bearman
♦♠	Lisa

4238xz	Lauren
34163	Delbert
123	Glaser
43415	Edith
9918	JS
767	Sandra
3	Swimp
634	James
724	Lee
2346	Dave
457	Bearman
312	Lisa

LSD (Least Significant Digit) Radix Sort -- Using Counting Sort

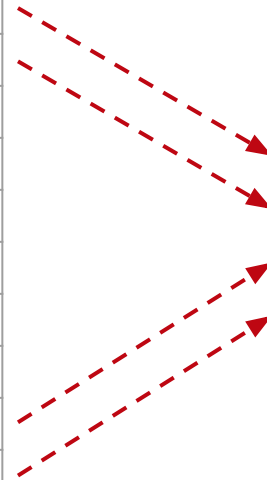
Sort each digit independently from rightmost digit towards left.

- Example: Over {♣, ♠, ♥, ♦}

♠♠	Lauren
♥♦	Delbert
♦♣	Glaser
♣♥	Edith
♠♥	JS
♦♣	Sandra
♥♠	Swimp
♥♦	James
♣♠	Lee
♥♣	Dave
♣♠	Bearman
♦♠	Lisa



♦♣	Glaser
♦♣	Sandra
♥♣	Dave
♥♠	Swimp
♠♠	Lauren
♣♠	Lee
♣♠	Bearman
♦♠	Lisa
♠♥	JS
♣♥	Edith
♥♦	James
♥♦	Delbert



♣♠	Lee
♣♠	Bearman
♣♥	Edith
♠♠	Lauren
♠♥	JS
♥♣	Dave
♥♠	Swimp
♥♦	James
♥♦	Delbert
♦♣	Glaser
♦♣	Sandra
♦♠	Lisa

LSD (Least Significant Digit) Radix Sort -- Using Counting Sort

Sort each digit independently from rightmost digit towards left.

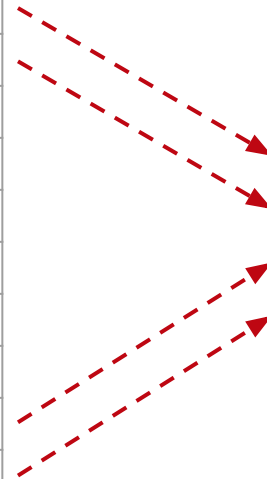
- Example: Over {1, 2, 3, 4}

Note: Lauren and Swimp are backwards in middle column, same with Delbert and James, same with Edith and JS

22	Lauren
34	Delbert
41	Glaser
13	Edith
23	JS
41	Sandra
32	Swimp
34	James
12	Lee
31	Dave
12	Bearman
42	Lisa



41	Glaser
41	Sandra
31	Dave
32	Swimp
22	Lauren
12	Lee
12	Bearman
42	Lisa
23	JS
13	Edith
34	James
34	Delbert



12	Lee
12	Bearman
13	Edith
22	Lauren
23	JS
31	Dave
32	Swimp
34	James
34	Delbert
41	Glaser
41	Sandra
42	Lisa

LSD Runtime

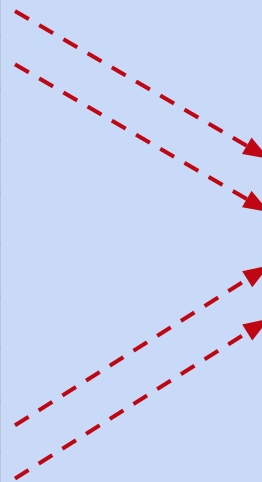
What is the runtime of LSD sort?

- Pick appropriate letters to represent non-constant terms.

22	Lauren
34	Delbert
41	Glaser
13	Edith
23	JS
41	Sandra
32	Swimp
34	James
12	Lee
31	Dave
12	Bearman
42	Lisa



41	Glaser
41	Sandra
31	Dave
32	Swimp
22	Lauren
12	Lee
12	Bearman
42	Lisa
23	JS
13	Edith
34	James
34	Delbert



12	Lee
12	Bearman
13	Edith
22	Lauren
23	JS
31	Dave
32	Swimp
34	James
34	Delbert
41	Glaser
41	Sandra
42	Lisa

LSD Runtime

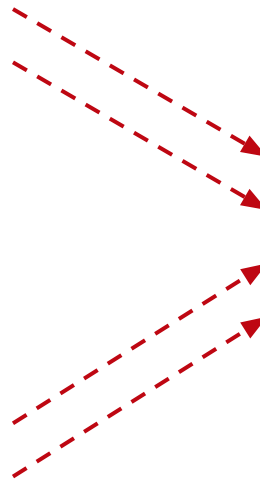
What is the runtime of LSD sort?

- $\Theta(WN+WR)$
- N: Number of items, R: size of alphabet, W: Width of each item in # digits

22	Lauren
34	Delbert
41	Glaser
13	Edith
23	JS
41	Sandra
32	Swimp
34	James
12	Lee
31	Dave
12	Bearman
42	Lisa



41	Glaser
41	Sandra
31	Dave
32	Swimp
22	Lauren
12	Lee
12	Bearman
42	Lisa
23	JS
13	Edith
34	James
34	Delbert



12	Lee
12	Bearman
13	Edith
22	Lauren
23	JS
31	Dave
32	Swimp
34	James
34	Delbert
41	Glaser
41	Sandra
42	Lisa

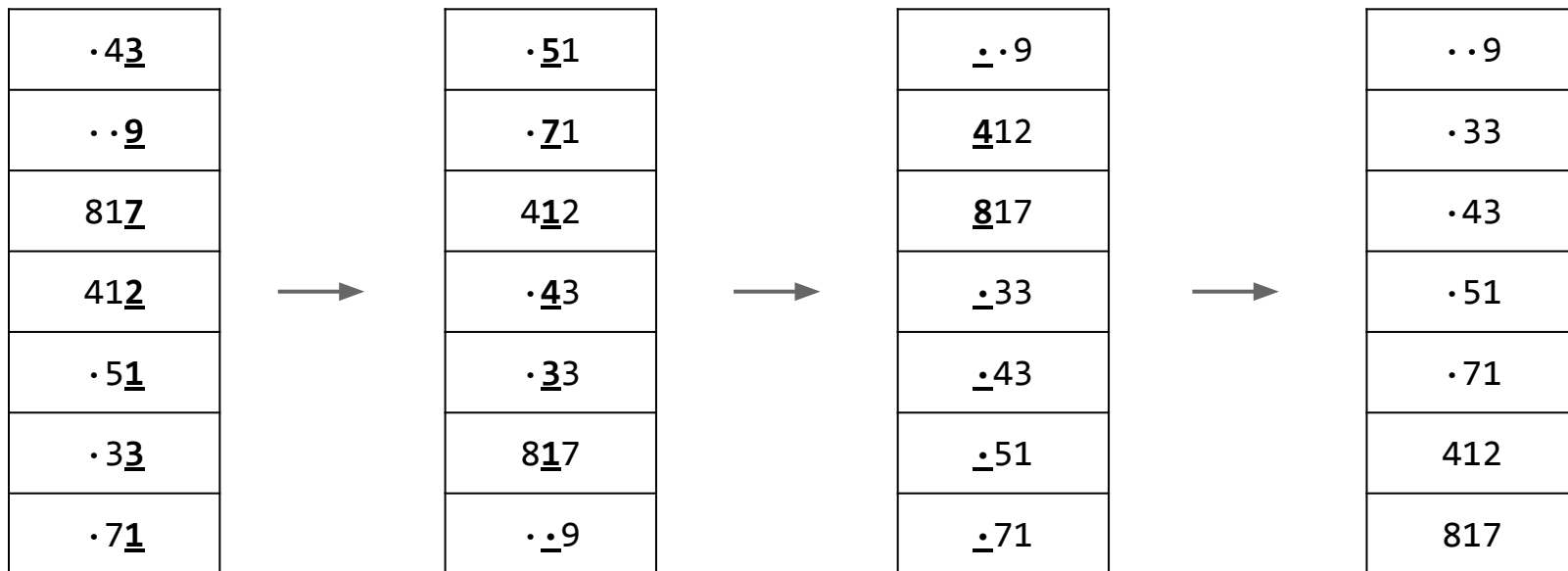
Non-equal Key Lengths

After processing least significant digit, we have array shown below. Now what?

4 <u>3</u>	→	51
<u>9</u>		71
81 <u>7</u>		412
41 <u>2</u>		43
5 <u>1</u>		33
3 <u>3</u>		817
7 <u>1</u>		9

Non-equal Key Lengths

When keys are of different lengths, can treat empty spaces as less than all other characters.



Sorting Summary

W passes of counting sort: $\Theta(WN+WR)$ runtime.

- Annoying feature: Runtime depends on length of longest key.

	Memory	Runtime	Notes	Stable?
Heapsort	$\Theta(1)$	$\Theta(N \log N)^*$	Bad caching (61C)	No
Insertion	$\Theta(1)$	$\Theta(N^2)^*$	Small N, almost sorted	Yes
Mergesort	$\Theta(N)$	$\Theta(N \log N)^*$	Fastest stable sort	Yes
Random Quicksort	$\Theta(\log N)$	$\Theta(N \log N)^*$ expected	Fastest compare sort	No
Counting Sort	$\Theta(N+R)$	$\Theta(N+R)$	Alphabet keys only	Yes
LSD Sort	$\Theta(N+R)$	$\Theta(WN+WR)$	Strings of alphabetical keys only	Yes

N: Number of keys. R: Size of alphabet. W: Width of longest key.

*: Assumes constant compareTo time.

MSD Radix Sort

MSD (Most Significant Digit) Radix Sort

Basic idea: Just like LSD, but sort from leftmost digit towards the right.

Pseudopseudohypoparathyroidism
Floccinaucinihilipilification
Antidisestablishmentarianism
Honorificabilitudinitatibus
Pneumonoultramicroscopicsilicovolcanoconiosis

MSD Sort Question: <http://yellkey.com/yourself>

Suppose we sort by topmost digit, then middle digit, then rightmost digit. Will we arrive at the correct result? A. Yes, B. No

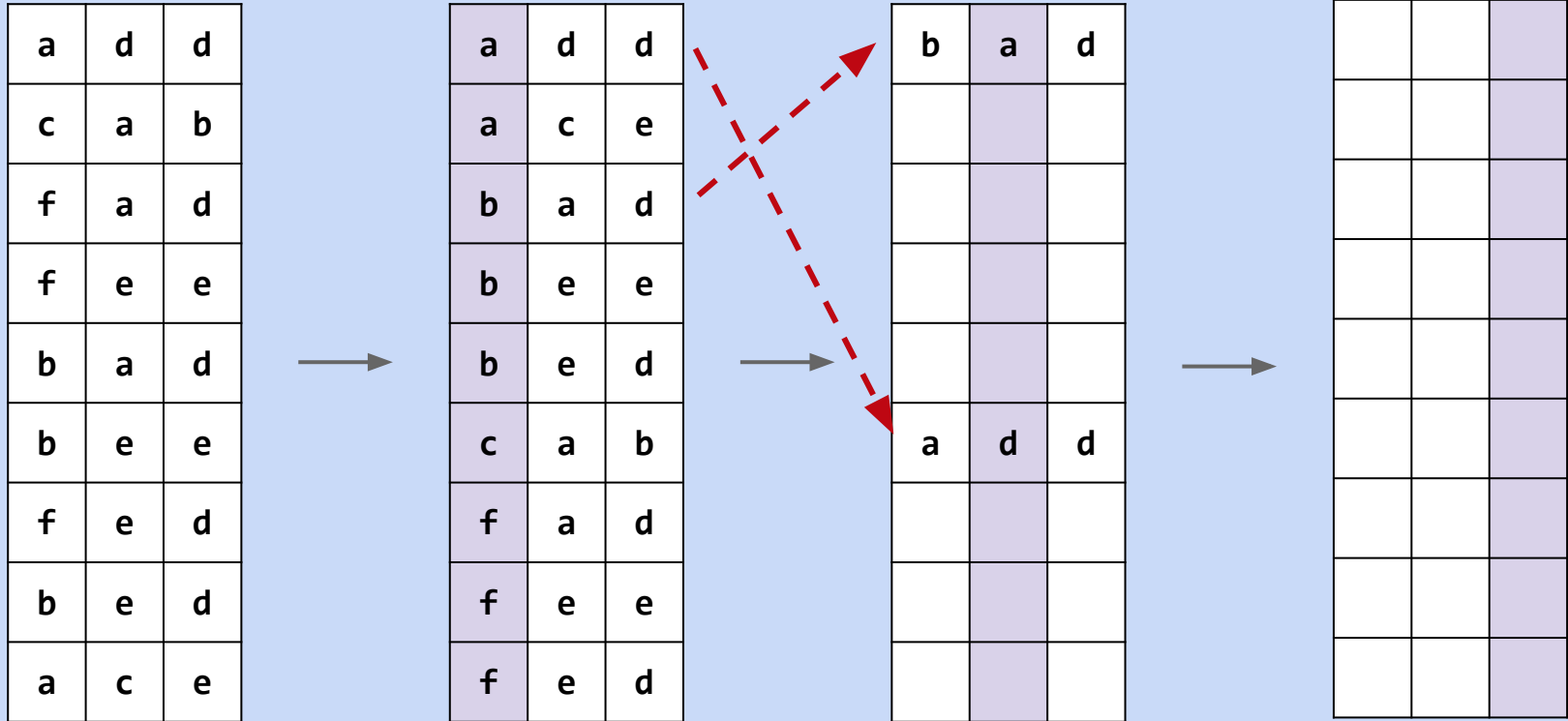
a	d	d
c	a	b
f	a	d
f	e	e
b	a	d
b	e	e
f	e	d
b	e	d
a	c	e

a	d	d
a	c	e
b	a	d
b	e	e
b	e	d
c	a	b
f	a	d
f	e	e
f	e	d

[illegible][illegible]

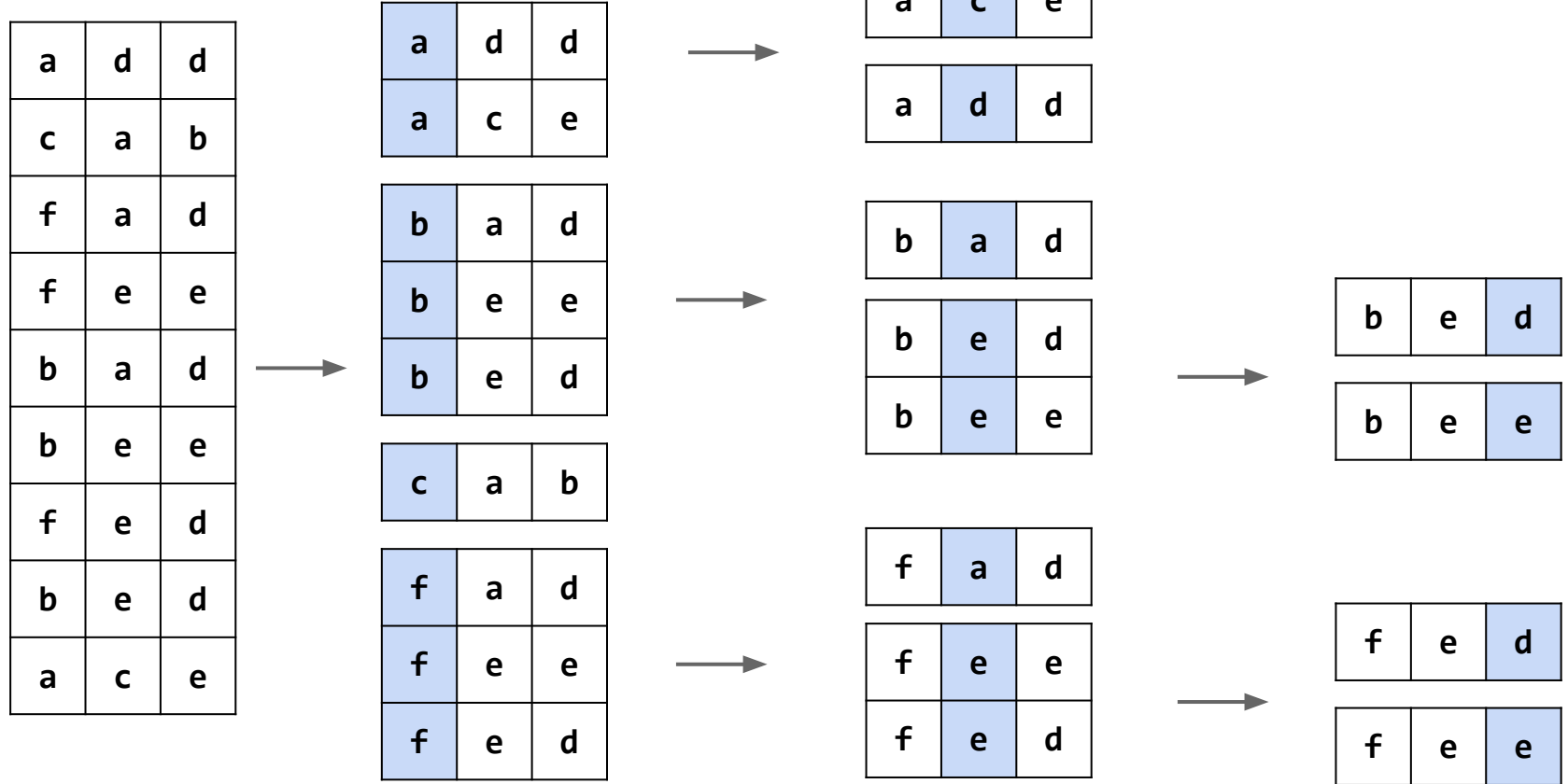
MSD Sort Question

Suppose we sort by topmost digit, then middle digit, then rightmost digit. Will we arrive at the correct result? **A. Yes, B. No.** How do we fix?



MSD Radix Sort (correct edition)

Key idea: Sort each subproblem separately.



Runtime of MSD

What is the Best Case of MSD sort (in terms of N , W , R)?

What is the Worst Case of MSD sort (in terms of N , W , R)?

Runtime of MSD

Best Case.

- We finish in one counting sort pass, looking only at the top digit: $\Theta(N + R)$

Worst Case.

- We have to look at every character, degenerating to LSD sort: $\Theta(WN + WR)$

Sorting Runtime Analysis

	Memory	Runtime (worst)	Notes	Stable?
Heapsort	$\Theta(1)$	$\Theta(N \log N)^*$	Bad caching (61C)	No
Insertion	$\Theta(1)$	$\Theta(N^2)^*$	Fastest for small N, almost sorted data	Yes
Mergesort	$\Theta(N)$	$\Theta(N \log N)^*$	Fastest stable sort	Yes
Random Quicksort	$\Theta(\log N)$	$\Theta(N \log N)^*$ expected	Fastest compare sort	No
Counting Sort	$\Theta(N+R)$	$\Theta(N+R)$	Alphabet keys only	Yes
LSD Sort	$\Theta(N+R)$	$\Theta(WN+WR)$	Strings of alphabetical keys only	Yes
MSD Sort	$\Theta(N+WR)$	$\Theta(N+R)$ (best) $\Theta(WN+WR)$ (worst)	Bad caching (61C)	Yes

N: Number of keys. R: Size of alphabet. W: Width of longest key.

*: Assumes constant compareTo time.

Sounds of Sorting Algorithms

Starts with selection sort: <https://www.youtube.com/watch?v=kPRA0W1kECg>

Insertion sort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=0m9s>

Quicksort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=0m38s>

Mergesort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=1m05s>

Heapsort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=1m28s>

LSD sort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=1m54s>

MSD sort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=2m10s>

Shell's sort: <https://www.youtube.com/watch?v=kPRA0W1kECg&t=3m37s>

Questions to ponder (later... after class):

- How many items are sorted in the video for selection sort?
- Why does insertion sort take longer / more compares than selection sort?
- At what time stamp does the first partition complete for Quicksort?
- Could the size of the input used by mergesort in the video be a power of 2?
- What do the colors mean for heapsort?
- How many characters are in the alphabet used for the LSD sort problem?
- How many digits are in the keys used for the LSD sort problem?

Citations

Creepy eye thing, title slide:

[http://photos3.meetupstatic.com/photos/event/a/3/f/4/highres_335141972.jp
eg](http://photos3.meetupstatic.com/photos/event/a/3/f/4/highres_335141972.jpg)