Problem: \( U = \text{Universe of keys} \) \( \text{SMALL!} \)
\( \{0, 1, 2, \ldots, k\} \)
\( A = (a_1, a_2, \ldots, a_n) \) \( \text{sequence of keys to sort} \) \( \text{BIG} \)

\[
\begin{array}{cccccccc}
2^{(1)} & 5^{(1)} & 3^{(1)} & 0^{(1)} & 2^{(2)} & 3^{(2)} & 0^{(2)} & 3^{(3)} \\
\end{array}
\]

In most real world applications, is 3 an integer?

3\(^{(1)}\) is George \( 3\(^{(1)}\) is John \( 3\(^{(3)}\) is Tom

Other data is sometimes called a \text{Handle}

It's easy \( O(n) \) time to compute

\[
\begin{array}{cccccccc}
C & 2 & 0 & 2 & 3 & 0 & 1 \\
0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\( C[i] = \# \text{records in } A \) with \( \text{key} = i \)

Dream!

\[
\begin{array}{cccccccc}
0^{(1)} & 0^{(2)} & 2^{(1)} & 2^{(2)} & 3^{(1)} & 3^{(1)} & 3^{(3)} & 5^{(1)} \\
\end{array}
\]
Given a key like $[426, 295]$

the computer can quickly address memory to access an array element whose index is the key.
| 2
| 5
| 3
| 0
| 2
| 3
| 0
| 3
| 2
| 5
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Dream!

Idea 1: Ask where should the 5 go?

Bad idea?

| 2
| 5
| 3
| 0
| 2
| 3
| 0
| 3
| 2
| 5
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Dream!

input \( C[i] \): # of guys in \( A \)

that \( = i \)

output \( C[i] \): # of guys in \( A \) that are \( \leq i \)

\[ \text{Calculate (O}(k)\text{ time) } \]

\|
| 1 | 2 | 4 | 6 | 7 | 8 |
| 0 | 1 | 2 | 3 | 4 | 5 |

| 2
| 5
| 3
| 0
| 2
| 3
| 0
| 3
| 2
| 5
| 1 | 2 | 3 | 5 | 6 | 7 | 8 |

Dream!

input
Radix Sort

329  657
457  436  355

First idea: organize by highest order digit

329  355  457  436  657  720  839

have to sort the sublists separately
Clever idea: First organize by lowest order digit!
Bucket sort

.78
.17
.39
.26
.72
.94
.21
.12
.23
.68
.68
.94

One Bucket
What data structure should you choose?
And how will it be used so the bucket is
One Bucket eventually sorted?

Requirements:
1. Insert an unpredictable number of elements one by one.
2. End up with the elements sorted.

Option 1       Option 2
How much should be the replacement of each column when it fills up?

+16 +1 Squared 1

×2 ← we will analyze!
\[ \begin{align*}
V + 2V + 4V + & \quad 8V + \\
& \quad \cdots + 2^k V \\
2^k V &= N, \text{ so} \\
\text{Total} &= \\
N \left( \frac{1}{2^k} + \frac{1}{2^{k-1}} + \cdots + \frac{1}{2} + 1 \right) &< 2N
\end{align*} \]
Quick Sort

Input Set N keys

Pick one randomly
partition the rest

all ≤ .

all ≥ .

Recurse until done!

A little clever programming makes the computer do the partitioning in $O(N \log N)$, really small and no extra space (except for controlling the recursions)

Running time? $O(N \log N)$ if you are lucky
$O(N^2)$ if NOT 😞
The whole square above has \((N-1)^2\) dominoes. The sum equals approximately half of that number.
(Relates to feelings that result from knowledge) (Relates to teaching, learning, or discovering.)

"Intuitive" or "Heuristic" analysis

Very rough: When each split is about half and half, the same analysis as when sorting tells us the recursion depth is \( \log_2 N \).

\( O(N) \) work each level, total: \( CN \log_2 N \)

Less rough: If the split is 9:1 at worst,

\[
\text{depth} = \# \text{times} \left( \frac{9}{10} \right) \left( \frac{9}{10} \right) \left( \frac{9}{10} \right) \cdots \left( \frac{9}{10} \right) \cdot N \text{ to get } \leq 1
\]

\[
\left( \frac{9}{10} \right)^L N \approx 1 \quad N = \left( \frac{10}{9} \right)^L
\]

\[
\log_{\left( \frac{10}{9} \right)} N = L = \frac{\log_2 B}{\log_2 A} = \frac{\log_2 N}{\log_2 \left( \frac{10}{9} \right)}
\]

\[
\log_A B = L \iff A^L = B \text{ what's } L ?
\]

Street Clobber fight: A with \( \log_2 \) \( A = 2 \log_2 A \)

\[
(2^{\log_2 A})^L = B \quad 2^{( \log_2 A ) \cdot L} = B
\]

Now clobber B: \( (\log_2 A) \cdot L = \log_2 B \)

\( \iff \text{ solve for } L \)
\[
\log_{(10)} N = L = \frac{\log_2 B}{\log_2 A} - \frac{\log_2 N}{\log_2 (1/9)}
\]

\[
\frac{10}{9} = \frac{9}{9} + \frac{1}{9} = \frac{1}{9} = 1.\overline{11111} \approx 0.15...
\]

\[
2^0 = 1.00000 \cdots \quad C = \frac{1}{9} \quad \approx 0.15
\]

\[
2^{\text{a little more than 0}} = 2^{0.15 \cdots} = \text{a little more than 1}.
\]

\[
2^0 = 1.\overline{11111} \cdots
\]
Towards a precise analysis:

Dream: Imagine A is sorted (it really isn’t!)

- Gets compared to all these
- Gets compared to all these

But

- Are NEVER compared!

How Quicksort acts.

Beautiful fact from probability:

Expected value (Sum of Random Variables) = Sum of their expected values

Even if the many random variables are correlated!
Will \( a_i \) and \( a_j \) ever get compared to each other?

IF and ONLY IF in that interval,
either \( a_i \) or \( a_j \) was the first
day made Red

\[
P( a_i \neq a_j \text{ one carrier} ) = \frac{2}{(j-i+1) \text{ length of the } (i,j)}
\]

Expected # comparisons =

\[
\leq \frac{2}{j-i+1} = O(N \log N)
\]

all pairs \( i \), \( i+1 \), \( \ldots \), \( N \) etc
\[ \begin{array}{r}
1234 \\
\times 4321 \\
\hline
1234 \\
2468 \\
3702 \\
4938 \\
\hline
576 \\
744 \\
2944 \\
\end{array} \]

\[ 12 + 34 = 46 \]

\[ 43 + 21 = 64 \]

\[ \Theta(N) \]

2944 \times 10,000 \text{ or something?}

\[ (516 + 714) \times 100 + 34 \cdot 21 \]

\[ \text{Missing derivative} \]

\[ 1234 = (12) \times 10^2 + 34 \]

\[ 4321 = (43) \times 10^2 + 21 \]

\[ 12 - 43 \cdot 10^4 + \left[ \frac{1}{11} \left( (12 \times 21) + (43 \times 34) \right) \right] \cdot 10 + (34 \times 21) \]

\[ \left( (12 + 34) \times (43 + 21) - 12 \times 34 - 43 \times 21 \right) \]
\[ \begin{array}{c}
1234 \\
\times 4321
\end{array} \]

\[ \begin{array}{c}
1234 \\
2468 \\
3702 \\
4938
\end{array} \]

\[ 12 + 34 = 46 \]

\[ 43 + 21 = 64 \]

\[ 576 \]

\[ 714 \]

\[ 2944 \]

\[ 576 \times 10,000 \]

\[ + (2944 - 576 - 714) \times 100 \]

\[ + 714 = ? \]

**Missing derivation**

\[ 1234 = (12) \times 10^2 + 34 \]

\[ 4321 = (43) \times 10^2 + 21 \]

\[ 12 - 43 \cdot 10^4 + \left[ (12 \times 21) + (43 \times 34) \right] \cdot 10 + (34 \times 21) \]

\[ \left[(12 + 34) \times (43 + 21) - 12 \times 34 - 43 \times 21 \right] \]
\[ T(N) = 3T(N/2) + CN \]

\[ \begin{array}{c}
N \\
\frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \\
\frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \\
\frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \\
\frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \\
\frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \quad \frac{N}{4} \\
\frac{N}{4} \quad \frac{N}{4} \\
\frac{N}{4} \\
\frac{N}{4} \\
\frac{N}{4} \\
\end{array} \]

\[ CN \left( 1 + \frac{3}{2} + \left( \frac{3}{2} \right)^2 + \ldots + \left( \frac{3}{2} \right)^{\log_2 N} \right) \]

\[ = CN \left( \frac{3}{2} \right)^{\log_2 N} \left( \frac{(2)^{\log_2 N}}{2} + \frac{(2)^{\log_2 N - 1}}{3} + \ldots + \frac{2}{3} + 1 \right) \]

\[ \leq \frac{1}{1 - 2/3} = \frac{1}{1/3} = 3 \]

\[ 3 = 2^{\log_2 3} \quad CN \cdot 2^{\log_2 3} \cdot N = CN^{3/2} \]