Problem: \( U = \text{Universe of keys} \) SMALL!

\[ = \{0, 1, 2, \ldots, k\} \]

\( A = (a_1, a_2, \ldots, a_n) \) sequence of keys (to sort) 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 3 George, 3^{(1)} is 3 Andy, 3^{(3)} is 3 Tom.

Other data is sometime called a "Handle"

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

\[
C \begin{bmatrix} 2 & 0 & 2 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \quad C[i] = \# \text{records in } A \text{ with key } = i
\]
Given key like $[426, 295]$

The computer can quickly address memory to access an array element whose index is the key.
Idea 1 Ask where should the 5 go?

Bad idea 2

input $C[i] =$ # of friends in A
that $= i$

Output $C[i] =$ # of friends in A that are $\leq i$

Where should $g^{(3)}$ be copied?
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

<table>
<thead>
<tr>
<th>Value</th>
<th>Bucket</th>
</tr>
</thead>
<tbody>
<tr>
<td>.78</td>
<td>0</td>
</tr>
<tr>
<td>.17</td>
<td>1</td>
</tr>
<tr>
<td>.39</td>
<td>1</td>
</tr>
<tr>
<td>.26</td>
<td>2</td>
</tr>
<tr>
<td>.72</td>
<td>3</td>
</tr>
<tr>
<td>.94</td>
<td>4</td>
</tr>
<tr>
<td>.21</td>
<td>5</td>
</tr>
<tr>
<td>.12</td>
<td>6</td>
</tr>
<tr>
<td>.23</td>
<td>7</td>
</tr>
<tr>
<td>.68</td>
<td>8</td>
</tr>
<tr>
<td>.68</td>
<td>9</td>
</tr>
</tbody>
</table>

One Bucket
What data structure should you choose? And how will it be used so the bucket is 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 amoy be when it fills up?

\[ +16 \quad +1 \quad \text{Squared} \quad ! \]

\[ \times 2 \quad \text{we will analyze!} \]
\[ V + 2V + 4V + 8V + \cdots + 2^k V = N \]
\[ 2^k V = N, \text{ so} \]
\[ \text{Total} = N \left( \frac{1}{2^k} + \frac{1}{2^{k-1}} + \cdots + \frac{1}{2} + 1 \right) \leq 2N \]
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? \( \Theta(N \log N) \) if you are lucky
\( \Theta(N^2) \) 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 merge sort tells us the recursion depth $= \lceil \log_2 N \rceil$. $O(N)$ work each level, total $= CN \log_2 N$.

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

$$\text{Depth} = \text{times} \left( \frac{9}{10} \right) \left( \frac{9}{10} \right) \left( \frac{9}{10} \right) \ldots \left( \frac{9}{10} \right) \cdot N \approx \frac{\log N}{\log \left( \frac{9}{10} \right)} \approx \frac{\log N}{-0.05}. \quad \text{if } N \approx 10^{10} \Rightarrow (9,1) \approx \frac{\log^2 N}{(9,1) \cdot N}$$

$$\log \left( \frac{9}{10} \right) \approx -0.05 \quad \Rightarrow \quad L = \frac{\log_2 B}{\log_2 A} = \frac{\log_2 N}{\log_2 \left( \frac{10}{9} \right)}$$

$$\log_2 B = L \iff A^L = B \quad \text{Whats } L?$$

Street Clobber:

fight: $A$ with a $\log_2 A$:

$$\left( 2^{(\log_2 A)} \right)^L = B \quad 2^{(\log_2 A) \cdot L} = B$$

Now clobber $B$: $(\log_2 A) \cdot L = \log_2 B$
\[
\log_{10} N = L = \frac{\log_2 B}{\log_2 A} - \frac{\log_2 N}{\log_2 (\frac{10}{a})}
\]

\[
\frac{10}{a} = \frac{a}{q} + \frac{1}{q} = \frac{1}{q} = 1.\overline{1111} \\
0.15...
\]

\[
2^C = 1.0000... \\
C = \frac{1}{0.15} = 6.67
\]

a little more than 0

\[
2^C = a \text{ little more than 1.}
\]

\[
2^{0.15...} = 1.\overline{1111}...
\]
Towards a precise analysis:

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

- Gets compared to all these
- The first random choice gets compared to all these

But

one 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.

\[
P( a'_i \text{ or } a'_j \text{ carry}) = \frac{2}{(i, j)(j - i + 1) \text{ for the } (i, j) \text{ pair}}
\]

Expected number of comparisons = \( \sum_{i < j} \frac{2}{j - i + 1} \) by math

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

all pairs \#x, \#y, \#z, etc
Big Digit Fact:

Number \( \times \) to write \( \frac{X}{\text{a million}} \) in base \( \text{Base} \) like \( \frac{2}{\text{Ten}} \),

\[ \sqrt{1,000,000} \sim 6 \]

\( X \) million 10 10 10 100 1000 10000 100000 1000000 6 + 60 = 96

So \( \log_{\text{Ten}} (1 \text{ million}) = 6 \)

Cryptography

Rivest

Shamir

Adleman

\( (M, E) \) RSA

100 - 200 Decim Digits

\( M = F_1, F_2, F_1 + F_2 \) and \( \sim 50 - 100 \text{ Decimal Digits} \)

Primes
\[ f_N = 1234 \times 4321 \]
\[
\begin{array}{c}
1234 \\
2468 \\
3702 \\
4938 \\
\hline
576 \quad 74 \quad 2944
\end{array}
\]

\[ \Theta(N) \quad 2944 \times 10^2 \quad \text{or something} \]

Missing derivation

\[ 1234 = (12) \times 10^2 + 34 \]
\[ 4321 = (43) \times 10^2 + 21 \]

\[ 12 - 43 \cdot 10^4 + \left[ (12 \times 21) + (43 \times 34) \right] \cdot 10^2 + (34 \times 21) \]

\[ \left[ (12 + 34) \times (43 + 21) - 12 \times 34 - 43 \times 21 \right] \]
\[
\begin{align*}
\text{FN} & = 1234 \\
\times & \quad \text{4321} \\
\hline
\text{1234} & \text{2468} \\
\text{2702} & \text{4938} \\
\hline
\Theta(N^2) & = 5332114
\end{align*}
\]

\[
\begin{align*}
12 + 34 & = 46 \\
43 + 21 & = 64 \\
576 & = 714 \\
2944 & = 5332114
\end{align*}
\]

\[
\begin{align*}
576 \cdot 10,000 & + (2944 - 576 - 714) \cdot 100 \\
& + 714 = ?
\end{align*}
\]

\text{Missing derivative}

\[
\begin{align*}
1234 & = (12) \cdot 10^2 + 34 \\
4321 & = (43) \cdot 10^2 + 21
\end{align*}
\]

\[
\begin{align*}
12 - 43 \cdot 10^4 & + \left[ (12 \cdot 21) + (43 \cdot 34) \right] \cdot 10 + (34 \times 21)
\end{align*}
\]

\[
\begin{align*}
\left[ (12 + 34) \times (21 + 43) - 12 \times 34 - 43 \times 21 \right]
\end{align*}
\]
\[ T(n) = 3T(n/2) + CN \]

\[ \frac{n}{2} \quad \frac{n}{2} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{3CN}{2} \]

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

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

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

\[ 3 = 2 \cdot \log_3 N \quad C \cdot 2^{\log_2 3 \cdot N^2} = CN^{\frac{1}{3}} \]
\[ \log_2 3 = ? \]

By calculator, this is \( \approx (1.6) \)

\[ \log_2 1 = 0 \quad \log_2 2 = 1 \quad (\text{Between}) \quad \log_2 4 = 2 \]

\[ 100 \quad 128 = 2^7 \]
\[ 64 = 2^6 \quad 82 = 2^m \quad 256 = 2^8 \]

\[ 50 \quad 64 \quad 2^5 \]

\[ 25 \quad \log_2 100 = 7 \]

\[ 13 \quad \ell \]

\[ 7 \quad \ell \]
\[ 4 \quad \ell \]
\[ 2 \quad \ell \]
\[ 1 \quad \ell \]
Extendable array data structure

Problem:
Input: N elements GIVEN sequentially, BUT N is NOT KNOWN ahead of time.
Output: N and the N elements stored in the sequence they were inputted.

Conventional wisdom (?)

Array?
Linked List?
Java Vector? Java ArrayList?
Linked list solution

next el of data D

1. allocate a new node
2. copy D into new node
3. update last

new C++ Java, walker called in C
At the end of the day, when all N data items have been stored, HOW MUCH TIME HAD BEEN CONSUMED FOR ALL THE WORK??
Problem:
Input: Array A of numbers, length N.
Output: Indexes and value \((lowi, hii, val)\)
so \(val\) is the MAXIMUM SUM OF
some ADJACENT ELEMENTS, and those
adjacent elements are in \(A[lowi...hii]\).

3 Solutions:
Bad:
Ugly:
Good:
\[ T(N) = 2T(N/2) + CN \]
\[ U(N) = 2U(N/2) + C \]
\[ C + 2^1(C + 2^1(C + 2^1( \ldots ))) \]
\[ = C + 2^1C + 2^22^1C + \ldots + 2^\log_2 N C \]
\[ = CN(1/2^{\text{don't care}} + \ldots + 1/2 + 1) \]