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

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

\( A = (a_1, a_2, \ldots, a_N) \) sequence of keys (to sort)

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

In most real world applications, is 3 an integer?

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

\( 3^{(2)} \) is Andy

\( 3^{(3)} \) is Tom

Other data is sometimes called a handle

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

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

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

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

Dream!

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

the computer can quickly
address memory to access an array

element whose index is the key.
<table>
<thead>
<tr>
<th>2^n</th>
<th>5^n</th>
<th>3^n</th>
<th>O^n</th>
<th>2^e</th>
<th>3^e</th>
<th>O^e</th>
<th>3^e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Idea 1: Ask where should the 5 go?

Bad idea:

<table>
<thead>
<tr>
<th>2^n</th>
<th>5^n</th>
<th>3^n</th>
<th>O^n</th>
<th>2^e</th>
<th>3^e</th>
<th>O^e</th>
<th>3^e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Dream!

Input: C[i] = # of keys in A that = i

Output: C[i] = # of keys in A where should be copied?

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

Calculate (O(k) time)
Radix Sort

329  657
457  436  355

First idea: organize by highest order digit

329  355  457  436  657  720  839

329  355  436  457  657  720  839

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

.78  0
.17  1
.39  1
.26  2
.72  3
.94  4
.21  5
.12  6
.23  7
.68  8
.94  9

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

array be when it

squared

fills up?

we will analyze

1.2

1/16

16
\[ \begin{align*}
V &= V \\
V + V &= V + V \\
4V &= 2V + 2V \\
V + 2V + 4V + \ldots + 2^k V &= N \\
2^k V &= N, \text{ so} \\
\text{Total} &= N \left( \frac{1}{2^k} + \frac{1}{2^{k-1}} + \ldots + \frac{1}{2} + 1 \right) < 2N
\end{align*} \]
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.
Intuitive or "heuristic" analysis

Very rough: When each split is about half and half, the same analysis as mergesort 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,

\[ \text{Depth} = \text{number} \left( \frac{9}{10}, \frac{9}{10}, \frac{9}{10}, \ldots, \frac{9}{10} \right) \cdot N \text{ to get } \leq 1 \]

\( \left( \frac{9}{10} \right)^2 N \approx 1 \quad N = \left( \frac{10}{9} \right)^2 \)

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

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

Street clobber fight: A with a \( \log_2 \) A

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

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

\( \Rightarrow \) Solve for L
\[
\log_{\frac{1}{2}} N = L = \frac{\log_2 B}{\log_2 A} - \frac{\log_2 N}{\log_2 \left(\frac{10}{9}\right)}
\]

\[
\frac{10}{9} = \frac{9}{9} + \frac{1}{9} = 1\frac{1}{9} = 1.\overline{111111}
\]

\[
2^0 = 1.0000\ldots
\]

\[
C \log_2 N
\]

\[
C = \frac{1}{0.15}
\]

\[
= 6.67
\]

a little more than 0

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

\[
2^{0.15}\ldots
\]

\[
2 = 1.\overline{111111}\ldots
\]
Towards a precise analysis:

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

The first random choice gets compared to all these, but none of them 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 bag made red.

\[
P( a_i \lor a_j \text{ one carry}) = \frac{2}{(i-i+1) \text{ else of the } (i,i)}
\]

Expected # of comparisons =

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

all pairs \((i,i), (i,j), \ldots\) etc.
Big Digit Fact:

Number $X$ to write $X$ in base $\text{base}$

$$\sqrt{1,000,000} \quad \# \text{digits} \approx \log_{\text{base}} X$$

$$\text{million} \quad 10^6 \quad \text{r.o.}$$

6 times multit.

So $\log_{10} (1 \text{ million}) = 6$

Cryptography

Rivest

Shamir

Adleman

Key $(M, E)$

RSA

100-200 Decimal Digits

$M = F_1, F_2, F_1 + F_2 \text{ and } \approx 50-100 \text{ Decimal Digit Primes }$
\[
\begin{align*}
\text{TN} &= 1234 \\
\times 4321 \\
\hline
1234 & \ \ \ 1234 \\
2468 & \ \ \ 2468 \\
3702 & \ \ \ 3702 \\
4938 & \ \ \ 4938 \\
\hline
\end{align*}
\]

\[12 + 34 = 46\]

\[43 + 21 = 64\]

\[576 \times 714 = 2944\]

\[\Theta(N) \quad 2944 \times 10^4 \quad \text{or something?}\]

**Missing derivative**

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

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

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

\[
\left[\frac{(12 + 34) \times (43 + 21) - 12 \times 34 - 43 \times 21}{11}\right]
\]
\[
\begin{align*}
\text{Missing derivative:} \\
1234 &= (12) \times 10^2 + 34 \\
4321 &= (43) \times 10^2 + 21
\end{align*}
\]

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

\[
\left[ (12 + 34) \times (21 + 43) - (12 \times 34 - 43 \times 21) \right]
\]

\[
\begin{align*}
\text{Regular expression: } &12 + 34 = 46 \\
&43 + 21 = 64
\end{align*}
\]
\[ T(n) = 3T(n/2) + CN \]

\[ \frac{n}{2} \]

\[ \frac{n}{4} \]

\[ \frac{n}{4} \]

\[ \frac{n}{4} \]

\[ \frac{n}{4} \]

\[ \frac{n}{4} \]

\[ \frac{n}{4} \]

\[ 3\frac{CN}{2^2} \]

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

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

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

\[ 3 = 2^{\log_2 3} \]

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

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

\[ \log_2 1 = 0 \quad \log_2 2 = 1 \quad \log_2 4 = 2 \]

\[
\begin{array}{c|c|c|c|c}
100 & 128 & 7 & 256 & \mathbb{R} \\
50 & 64 & 6 & 256 & 2 \\
25 & 2 & 2 & & \\
13 & & & & \\
7 & & & & \\
4 & & & & \\
2 & & & & \\
1 & & & & \\
\end{array}
\]
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 endl of data D

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

new C++/Java
caller
At the end of the day, when all N data items have been stored, HOW MUCH TIME HAD BEEN CONSUMED FOR ALL THE WORK??
Towards the end of class, we read from the Java class library documentation comparing ArrayList and Vector. Vector is synchronized, but grows the size by adding a constant amount. ArrayList is not synchronized, but does not guarantee any size-growing algorithm, but it does not specify any size-growing algorithm, but it DOES give a small O(n) amortized running time.

Prepare to do this QUIZ on Tuesday.

Do the math that shows the TOTAL TIME used for all this work, in terms of N, is O(N). In other words, all of the times on the right added up to a number that is < some constant * N.

\[
\begin{align*}
N &= 2^{11} \\
1 &= 2^{12} \\
+4 &= 2 + 4 \\
+8 &= 2 + 4 + 8 \\
+16 &= 2 + 4 + 8 + 16 \\
+32 &= 2 + 4 + 8 + 16 + 32 \\
+64 &= 2 + 4 + 8 + 16 + 32 + 64
\end{align*}
\]
Problem:
Input: Array A of numbers, length N.
Output: Indexes and value (lo, hi, val) so that val is the MAXIMUM SUM OF some ADJACENT ELEMENTS, and those adjacent elements are in A[lo...hi].

3 Solutions:
Bad:
Ugly:
Good:

First, calculate:
\[ S_{lo, hi} = \sum_{l=lo}^{hi} A[l] \] for all possible \( 0 \leq lo \leq hi \leq N-1 \)

Number of times the ugly algo calculates on \( S_{lo, hi} \) is \( O(N^2) \)

Second, calculate the minimum of those sums, keep track of \( lo, hi \).
ugly: you don’t yet have a really great idea so you see the divide and conquer idea

Blue[Green
Blue1 > Green
Blue2 > G2

get $l_1, h_1, m_1, l_2, h_2, m_2$

So cut in: might be $m_1$, might be $m_2$

Blue1 + Blue2 > Green

> G2

Green1 + G2

can be found the max of 3
\[ \begin{bmatrix} 1 & 2 & 7 & -6 & 4 & 10 & 11 & 13 & 14 \\ 1 & 2 & -2 \end{bmatrix} \]
\[ \begin{align*}
T(N) &= 2T(N/2) + CN \\
U(N) &= 2U(N/2) + C \\
C + 2(C + 2(C + 2(...))) &= C + 2C + 2^2C + ... + 2^{\log_2 N}C \\
&= CN(1/2^{\text{don't care}} + ... + 1/2 + 1)
\end{align*} \]