Outline
• Insert-sort
• Going half-sies
• Sorted binary trees
• Quicker-sort
• WWII Codebreaking

One-Slide Summary
• Insert-sort is $\Theta(n^2)$ worst case (reverse list), but is $\Theta(n)$ best case (sorted list).
• A recursive function that divides its input in half each time is often in $\Theta(\log n)$.
• If we could divide our input list in half rapidly, we could do a quicker sort: $\Theta(n \log n)$.
• Sorted binary trees are an efficient data structure for maintaining sorted sets.
• British codebreakers used cribs (guesses), brute force, and analysis to break the Lorenz cipher. Guessed wheel settings were likely to be correct if they resulted in a message with the right linguistic properties for German (e.g., repeated letters).

How much work is insert-sort?
def insert_sort(lst, cf):
    if not lst: return []
    return insert_one(lst[0], insert_sort(lst[1:], cf))

def insert_one(elt, lst, cf):
    if not lst: return [elt]
    if cf(elt, lst[0]): return [elt] + lst
    return [lst[0]] + insert_one(elt, lst[1:], cf)

How many times does insert-sort evaluate insert-one? running time of insert-one is in $\Theta(n)$
$n$ times (once for each element)

insert-sort has running time in $\Theta(n^2)$ where $n$ is the number of elements in the input list

Can we do better?
insert_one(88, [1,2,3,5,6,22,63,77,89,90], ascending)

Suppose we had procedures
first_half(lst)
second_half(lst)
that quickly divided the list in two halves?
**quicker-insert using halves**

```python
def quicker_insert(elt, lst, cf):
    if not lst: return [elt]  # just like insert_one
    if len(lst) == 1:
        # handle 1 element by hand
        return [elt] + lst if cf(elt, lst[0]) else lst + [elt]
    front = first_half(lst)
    back = second_half(lst)
    if cf(elt, back[0]):
        # insert into front half
        return quicker_insert(elt, front, cf) + back
    else:
        # insert into back half
        return front + quicker_insert(elt, back, cf)
```

**Evaluating quicker-sort**

```python
def quicker_sort(elt, lst, cf):
    if not lst: return [elt]  # just like insert_one
    if len(lst) == 1:
        # handle 1 element
        return [elt] + lst if cf(elt, lst[0]) else lst + [elt]
    front = first_half(lst)
    back = second_half(lst)
    if cf(elt, back[0]):
        # insert into front half
        return quicker_sort(elt, front, cf) + back
    else:
        # insert into back half
        return front + quicker_sort(elt, back, cf)
```

**How much work is quicker-sort?**

Each time we call quicker-insert, the size of the list only increases by 1.

<table>
<thead>
<tr>
<th>List Size</th>
<th># quicker_insert applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
</tr>
</tbody>
</table>

**Liberal Arts Trivia: Scandinavian Studies**

- This capital of and largest city in Denmark is situated on the islands of Zealand and Amager. It is the birthplace of Neils Bohr, Søren Kierkegaard, and Victor Borge. The city's origin as a harbor and a place of commerce is reflected in its name. Its original designation, from which the contemporary Danish name is derived, was Køpmannæhafn, "merchants' harbor". The English name for the city is derived from its (similar) Low German name.

- The argan tree, found primarily in Morocco, has a knobby, twisted trunk that allows these animals to climb it easily. The animals eat the fruit, which has an indigestible nut inside, which is collected by farmers and used to make argan oil: handy in cooking and cosmetics, but pricey at $45 per 500 ml.
Remembering Logarithms

\[
\log_b n = x \text{ means } b^x = n
\]

What is \(\log_2 1024\)?
What is \(\log_{10} 1024\)?
Is \(\log_{10} n\) in \(\Theta(\log_2 n)\)?

Changing Bases

\[
\log_b n = (1/\log_k b) \log_k n
\]

If \(k\) and \(b\) are constants, this is constant

\[\Theta(\log_2 n) \equiv \Theta(\log_{10} n) \equiv \Theta(\log n)\]

No need to include a constant base within asymptotic operators.

Number of Applications

Assuming the list is well-balanced, the number of applications of quicker-insert is in \(\Theta(\log n)\) where \(n\) is the number of elements in the input list.

quicker-sort?

\[
def \text{quicker\_sort}(\text{lst}, \text{cf}):
\]

\[
def \text{quicker\_insert}(\text{elt}, \text{lst}, \text{cf}):
\]

Is there a fast first-half procedure?

- No! (at least not on lists)
- To produce the first half of a list length \(n\), we need to walk down the first \(n/2\) elements
- So, first-half on lists has running time in \(\Theta(n/2) = \Theta(n)\)

Orders of Growth

Is there a fast first-half procedure?
Making it faster

We need to either:

1. Reduce the number of applications of insert-one in insert-sort
   Impossible – need to consider each element

2. Reduce the number of applications of quicker-insert in quicker-insert
   Unlikely… each application already halves the list

3. Reduce the time for each application of quicker-insert
   Need to make first-half, second-half and append faster than $\Theta(n)$

Sorted Binary Trees

- A tree containing all elements $x$ such that $cf(x, el)$ is true
- A tree containing all elements $x$ such that $cf(x, el)$ is false

Tree Example

- $cf$ is $<$

Representing Trees

```python
def make_tree(left, el, right):
    return [el, left, right]

def get_element(tree):
    return tree[0]

def get_left(tree):
    return tree[1]

def get_right(tree):
    return tree[2]
```

left and right are trees
(  is a tree)

tree must be a non-null tree

tree must be a non-null tree

tree must be a non-null tree
Representing Trees

A = make_tree([], 1, [])
B = make_tree(A, 2, [])
C = make_tree([], 8, [])
D = make_tree(B, 5, C)

insert-one-tree

```python
def insert_one_tree(cf, new_elt, tree):
    if not tree:
        return make_tree([], new_elt, [])
    element_here = get_element(tree)
    if cf(new_elt, element_here):
        return make_tree(
            insert_one_tree(cf, new_elt, get_left(tree)),
            element_here,
            get_right(tree))
    else:
        return make_tree(
            get_left(tree),
            element_here,
            insert_one_tree(cf, new_elt, get_right(tree)))
```

How much work is insert-one-tree?

```python
def insert_one_tree(cf, new_elt, tree):
    if not tree:
        return make_tree([], new_elt, [])
    element_here = get_element(tree)
    if cf(new_elt, element_here):
        return make_tree(
            insert_one_tree(cf, new_elt, get_left(tree)),
            element_here,
            get_right(tree))
    else:
        return make_tree(
            get_left(tree),
            element_here,
            insert_one_tree(cf, new_elt, get_right(tree)))
```

The running time of `insert_one_tree` is in \( \Theta(\log n) \) where \( n \) is the number of elements in the input tree, which must be well-balanced.

quicker-insert-one

```python
def quicker_insert_one(cf, lst):
    if not lst: return []
    return quicker_insert_one(cf, lst[1:])
```

Lorenz Cipher Machine

Liberal Arts Trivia: Classics

• This ancient Greek epic poem, traditionally attributed to Homer, is widely believed to be the oldest extant work of Western literature. It describes the events of the final year of the Trojan War. The plot follows Achilles and his anger at Agamemnon, king of Mycenae. It is written in dactylic hexameter and comprises 15,693 lines of verse. It begins:
  - μῆνιν ᾠδείς θέα Πηληϊάδεω χιλ ος
  - η ά ὰ Ἀ ῆ
  - ο λομένην, μυρί' χαιο ς λγε' θηκεν
  - ῥη \( \alpha \chi λ ι\)ς ῥ α λ γε’ έθηκεν
Liberal Arts Trivia: Chemistry

• This violet variety of quartz, often used in jewelry, takes its name from the ancient Greek (α “not”) and methustos (“intoxicated”)), a reference to the belief that it protected its own from drunkenness; ancient Greeks and Romans made drinking vessels of it to prevent intoxication.

Liberal Arts Trivia: Literature

• Name the author of the Age of Innocence (1920). The novel describes the upper class in New York city in the 1870s and questions the mores and assumptions of society. The title is an ironic comment on the polished outward manners of New York society, when compared to its inward machinations. The authors was the first woman to win the Pulitzer Prize for Literature.

Lorenz Wheels

12 wheels
501 pins
total (set to control wheels)

Work to break in $\Theta(p^n)$ so real Lorenz is $41^{12/5^3} \sim 1$ quintillion ($10^{18}$) times harder!

Code Breaking Intuition

• Suppose we are using a simple letter substitution cipher (i.e., replace every A with Q, etc.)
• You intercept these two messages:
  - PF1120: Vagebqhpgvba gb Pbzchgvat: Rkcybengvbaf va Ynathntr, Ybtvp, naq Znpuvarf
• What does the first one say? What hints did you have?

Breaking Fish

• Gov’t Communications HQ learned about first Fish link (Tunny) in May 1941
  - British codebreakers used “Fish” to refer to German teleprinter traffic
  - Intercepted unencrypted Baudot-encoded test messages
• August 30, 1941: Big Break!
  - Operator retransmits failed message with same starting configuration
  - Gets lazy and uses some abbreviations, makes some mistakes
    • SPRUCHNUMMER/SPRUCHNR (Serial Number)

“Two Time” Pad

• Allies have intercepted:
  $C_1 = M_1 \oplus K_1$
  $C_2 = M_2 \oplus K_1$
  Same key used for both (same starting configuration)
• Breaking message:
  $C_1 \oplus C_2 = (M_1 \oplus K_1) \oplus (M_2 \oplus K_1)$
  $= (M_1 \oplus M_2) \oplus (K_1 \oplus K_1)$
  $= M_1 \oplus M_2$
“Cribs”
- Know: $C_1, C_2$ (intercepted ciphertext)
  $$C_1 \oplus C_2 = M_1 \oplus M_2$$
- Don’t know $M_1$ or $M_2$
  - But, can make some guesses (cribs)
    - SPRUCHNUMMER
    - Sometimes allies moved ships, sent out bombers to help the cryptographers get good cribs
- Given guess for $M_1$, calculate $M_2$
  $$M_2 = C_1 \oplus C_2 \oplus M_1$$
- Once guesses that work for $M_1$ and $M_2$
  $$K_1 = M_1 \oplus C_1 = M_2 \oplus C_2$$

Reverse Engineering Lorenz
- From the 2 intercepted messages, Col. John Tiltman worked on guessing cribs to find $M_1$ and $M_2$: 4000 letter messages, found 4000 letter key $K_1$
- Bill Tutte (recent Chemistry graduate) given task of determining machine structure
  - Already knew it was 2 sets of 5 wheels and 2 wheels of unknown function
  - Six months later new machine structure likely to generate $K_1$

Intercepting Traffic
- Set up listening post to intercept traffic from 12 Lorenz (Fish) links
  - Different links between conquered capitals
  - Slightly different coding procedures, and different configurations
- 600 people worked on intercepting traffic

Breaking Traffic
- Knew machine structure, but a different initial configuration was used for each message
- Need to determine wheel setting:
  - Initial position of each of the 12 wheels
  - 1271 possible starting positions
  - Needed to try them fast enough to decrypt message while it was still strategically valuable

Recognizing a Good Guess
- Intercepted Message (divided into 5 channels for each Baudot code bit)
  $$Z = z_0 Z_1 z_2 Z_3 z_4 Z_5 ...$$
  $$z_{c,i} = \text{ith bit of ciphertext}, z_{m,i} = \text{ith bit of message} \oplus \text{ith bit of key}$$
  key comes from all of the wheels (e.g., $S$-wheel, ...)

- Look for statistical properties
  - How many of the $z_{c,i}$'s are 0? $\frac{1}{2}$ (not useful)
  - How many of ($z_{c,i+1} \oplus z_{c,i}$) are 0? $\frac{1}{2}$

Double Delta
$$\Delta Z_{c,i} = Z_{c,i} \oplus Z_{c,i+1}$$
Combine two channels:
$$\Delta Z_{1,i} \oplus \Delta Z_{2,i} = \Delta M_{1,i} \oplus \Delta M_{2,i} \quad > \frac{1}{2} \quad \text{Yippee!}$$
$$\oplus \Delta X_{1,i} \oplus \Delta X_{2,i} \quad = \frac{1}{2} \quad (\text{key})$$
$$\oplus \Delta S_{1,i} \oplus \Delta S_{2,i} \quad > \frac{1}{2} \quad \text{Yippee!}$$

Why is $\Delta M_{1,i} \oplus \Delta M_{2,i} > \frac{1}{2}$
Message is in German, more likely following letter is a repetition than random
Why is $\Delta S_{1,i} \oplus \Delta S_{2,i} > \frac{1}{2}$
$S$-wheels only turn when $M$-wheel is 1
Actual Advantage: Linguistics

• Probability of repeating letters
  \( \text{Prob}[\Delta M_{1,i} \oplus \Delta M_{2,i} = 0] \approx 0.614 \)
  3.3% of German digraphs are repeating
• Probability of repeating S-keys
  \( \text{Prob}[\Delta S_{1,i} \oplus \Delta S_{2,i} = 0] \approx 0.73 \)

\[
\text{Prob}[\Delta Z_{1,i} \oplus \Delta Z_{2,i} \oplus \Delta X_{1,i} \oplus \Delta X_{2,i} = 0] \\
= 0.614 \times 0.73 + (1-0.614) \times (1-0.73) \\
= 0.55 \quad \text{if the wheel settings guess is correct (0.5 otherwise)}
\]

Using the Advantage

• If the guess of \( X \) is correct, should see higher than \( \frac{1}{2} \) of the double deltas are 0
• Try guessing different configurations to find highest number of 0 double deltas

Problem:

\[ \text{# of double delta operations to try one config} \]
\[ = \text{length of } Z \times \text{length of } X \]
\[ = \text{for 10,000 letter message} = 12 \text{ M for each setting} \]
\[ \times 7 \oplus \text{ per double delta} \]
\[ = 89 \text{ M} \oplus \text{ operations} \]

Homework

• Problem Set 4
• Study for Exam 1
  - Out Soon