BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!!
The Shift Cipher
Shift Cipher: Encryption, Decryption, Cracking
The Shift Cipher

Consider encrypting English text.

Associate 'a' with 0; 'b' with 1; ...; 'z' with 25.

\[ s \in \{0, \ldots, 25\} \] (or could think of \( s \in \{a, \ldots, z\} \)).

To encrypt using key \( s \), shift every letter of the plaintext by \( s \) positions (with wraparound).
Consider encrypting English text.
The Shift Cipher

- Consider encrypting English text.

- Associate ‘a’ with 0; ‘b’ with 1; . . . ; ‘z’ with 25.
Consider encrypting English text.

Associate ‘a’ with 0; ‘b’ with 1; . . . ; ‘z’ with 25.

$s \in \{0, \ldots, 25\}$ (or could think of $s \in \{a, \ldots, z\}$).
The Shift Cipher

- Consider encrypting English text.

- Associate ‘a’ with 0; ‘b’ with 1; . . . ; ‘z’ with 25.

- $s \in \{0, \ldots, 25\}$ (or could think of $s \in \{a, \ldots, z\}$).

- To encrypt using key $s$, shift every letter of the plaintext by $s$ positions (with wraparound).
I want to encode *Bill works at a zoo!* with a shift-3.
I want to encode Bill works at a zoo! with a shift-3.

1. Do usual preprocessing: blocks of five, etc to get:
   billw orksa tazoo
The Shift Cipher: Examples of Encryption

I want to encode *Bill works at a zoo!* with a shift-3.

1. Do usual preprocessing: blocks of five, etc to get:
   billw orksa tazoo

2. Convert letters to numbers to get:
   1-8-11-11-22     14-17-10-18-0     19-0-25-14-14
The Shift Cipher: Examples of Encryption

I want to encode *Bill works at a zoo!* with a shift-3.

1. Do usual preprocessing: blocks of five, etc to get:
   billw orksa tazoo

2. Convert letters to numbers to get:
   1-8-11-11-22 14-17-10-18-0 19-0-25-14-14

3. Add three to each number (wrap around) to get:
   4-11-14-14-25 17-20-13-21-3 22-3-2-17-17
The Shift Cipher: Examples of Encryption

I want to encode **Bill works at a zoo!** with a shift-3.

1. Do usual preprocessing: blocks of five, etc to get:
   \[
   \text{billw orksa tazoo}
   \]

2. Convert letters to numbers to get:
   \[
   1-8-11-11-22 \quad 14-17-10-18-0 \quad 19-0-25-14-14
   \]

3. Add three to each number (wrap around) to get:
   \[
   4-11-14-14-25 \quad 17-20-13-21-3 \quad 22-3-2-17-17
   \]

4. Convert numbers to letters to get:
   \[
   \text{elooz runvd wdcrr}
   \]
The Shift Cipher: How do Decrypt

Bob knows Alice used shift-3. How does he decrypt?

He does shift by $-3$ or can view as shift by $26 - 3 = 23$. 
The Shift Cipher: How do Decrypt

Bob knows Alice used shift-3. How does he decrypt?
He does shift by $-3$ or can view as shift by $26 - 3 = 23$. 
The Shift Cipher: An Example of Decrypt

Bob has to decode mrvkx dolnh vpo which was coded by shift-3.

1. Convert letters to numbers to get:

   12-17-21-10-23 3-14-11-13-7 21-15-14

2. Subtract 3 from each number (wrap around) to get:

   9-14-18-7-20 0-11-8-10-4 18-12-11

3. Convert numbers to letters to get:

   joshu alike sml

4. Figure out spacing to get:

   Joshua likes ML
The Shift Cipher: An Example of Decrypt

Bob has to decode mrvkx dolnh vpo which was coded by shift-3.
Bob has to decode mrvkx dolnh vpo which was coded by shift-3.

1. Convert letters to numbers to get:

2. Subtract 3 from each number (wrap around) to get:
   9-14-18-7-20  0-11-8-10-4  18-12-11.

3. Convert numbers to letters to get:
   joshu alike sml.

4. Figure out spacing to get:
   Joshua likes ML.
Bob has to decode **mrkvx dolnh vpo** which was coded by shift-3.

1. Convert letters to numbers to get:

2. Subtract 3 from each number (wrap around) to get:
   9-14-18-7-20 0-11-8-10-4 18-12-11.
The Shift Cipher: An Example of Decrypt

Bob has to decode *mrvkx dolnh vpo* which was coded by shift-3.

1. Convert letters to numbers to get:
   

2. Subtract 3 from each number (wrap around) to get:
   
   9-14-18-7-20   0-11-8-10-4   18-12-11.

3. Convert numbers to letters to get: *joshu alike sml*.
Bob has to decode mrvnx dolnh vpo which was coded by shift-3.

1. Convert letters to numbers to get:

2. Subtract 3 from each number (wrap around) to get:
   9-14-18-7-20 0-11-8-10-4 18-12-11.

3. Convert numbers to letters to get: joshu alike sml.

4. Figure out spacing to get: Joshua likes ML.
“Wrap Around” is Modular Arithmetic: Definitions

[9x251] $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

$[x \mod N] =$ the remainder when $x$ is divided by $N$.

i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x \equiv y \pmod{N}$.

$25 \equiv 35 \pmod{10}$

$5 = [35 \mod 10]$
“Wrap Around” is Modular Arithmetic: Definitions

▶ $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$. 

$25 \equiv 35 \pmod{10}$

$5 = [35 \pmod{10}]$
“Wrap Around” is Modular Arithmetic: Definitions

- $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

- $[x \mod N] = \text{the remainder when } x \text{ is divided by } N$. 

- $25 \equiv 35 \pmod{10}$

- $5 = [35 \mod 10]$
“Wrap Around” is Modular Arithmetic: Definitions

- $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

- $[x \mod N] = \text{the remainder when } x \text{ is divided by } N$.
  - i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x \equiv y \pmod{N}$.

- $25 \equiv 35 \pmod{10}$
- $25 \neq [35 \mod 10]$
- $5 = [35 \mod 10]$
“Wrap Around” is Modular Arithmetic: Definitions

▶ $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

▶ $[x \mod N] =$ the remainder when $x$ is divided by $N$.
  ▶ i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x \equiv y \pmod{N}$.

▶ $25 \equiv 35 \pmod{10}$
“Wrap Around” is Modular Arithmetic: Definitions

- $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

- $[x \mod N] = \text{the remainder when } x \text{ is divided by } N$.
  - i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x \equiv y \pmod{N}$.

- $25 \equiv 35 \pmod{10}$

- $25 \neq [35 \mod 10]$
“Wrap Around” is Modular Arithmetic: Definitions

- $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

- $[x \mod{N}] = \text{the remainder when } x \text{ is divided by } N$.
  - i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x \equiv y \pmod{N}$.

- $25 \equiv 35 \pmod{10}$

- $25 \neq [35 \mod{10}]$

- $5 = [35 \mod{10}]$
Common usage:

\[ 100 \equiv 2 \pmod{7} \]
Modular Arithmetic II: Convention

Common usage:

\[ 100 \equiv 2 \pmod{7} \]

Commonly if we are in Mod \( n \) we have a large number on the left and then a number between 0 and \( n - 1 \) on the right.
Common usage:

\[ 100 \equiv 2 \pmod{7} \]

Commonly if we are in Mod \( n \) we have a large number on the left and then a number between 0 and \( n - 1 \) on the right.

When dealing with mod \( n \) we assume the entire universe is \( \{0, 1, \ldots, n - 1\} \).
Modular Arithmetic: $+, -, \times$

$\equiv$ is Mod 26 for this slide.
Modular Arithmetic: \(\text{+, \(-, \times}\)

\(\equiv\) is Mod 26 for this slide.

1. Addition: \(x + y\) is easy: wrap around. E.g., \(20 + 10 \equiv 30 \equiv 4\). Only use the number 30 as an intermediary value on the way to the real answer.

2. \(\sim\) is \(\equiv\) because \(19 + 7 \equiv 0\) (mod 26).

3. Mult: \(xy\) is easy: wrap around. E.g., \(20 \times 10 \equiv 200 \equiv 18\).

4. Division: Next Slide
Modular Arithmetic: +, −, ×

≡ is Mod 26 for this slide.

1. Addition: \( x + y \) is easy: wrap around. E.g., \( 20 + 10 \equiv 30 \equiv 4 \). Only use the number 30 as an intermediary value on the way to the real answer.

2. \(-7 \equiv x\) where \( 0 \leq x \leq 25 \).
Modular Arithmetic: $+,-,\times$

≡ is Mod 26 for this slide.

1. Addition: $x + y$ is easy: wrap around. E.g., $20 + 10 \equiv 30 \equiv 4$. Only use the number 30 as an intermediary value on the way to the real answer.

2. $-7 \equiv x$ where $0 \leq x \leq 25$.
   Pedantic $-y$ is the number such that $y + (-y) \equiv 0$. 

3. $xy$ is easy: wrap around. E.g., $20 \times 10 \equiv 200 \equiv 18$. Shortcut to avoid big numbers:

   $20 \times 10 \equiv -6 \times 10 \equiv -2 \times 30 \equiv -2 \times 4 \equiv -8 \equiv 18$.
Modular Arithmetic: $+, -, \times$

≡ is Mod 26 for this slide.

1. Addition: $x + y$ is easy: wrap around. E.g., $20 + 10 \equiv 30 \equiv 4$. Only use the number 30 as an intermediary value on the way to the real answer.

2. $-7 \equiv x$ where $0 \leq x \leq 25$.
   Pedantic $-y$ is the number such that $y + (-y) \equiv 0$.
   $-7 \equiv 19 \pmod{26}$ because $19 + 7 \equiv 0 \pmod{26}$.

3. Mult: $xy$ is easy: wrap around. E.g., $20 \times 10 \equiv 200 \equiv 18$.
   Shortcut to avoid big numbers: $20 \times 10 \equiv -6 \times 10 \equiv -2 \times 30 \equiv -2 \times 4 \equiv -8 \equiv 18$.

4. Division: Next Slide
Modular Arithmetic: $+, -, \times$

≡ is Mod 26 for this slide.

1. Addition: $x + y$ is easy: wrap around. E.g., $20 + 10 \equiv 30 \equiv 4$. Only use the number 30 as an intermediary value on the way to the real answer.

2. $-7 \equiv x$ where $0 \leq x \leq 25$.
   Pedantic $-y$ is the number such that $y + (-y) \equiv 0$.
   $-7 \equiv 19 \pmod{26}$ because $19 + 7 \equiv 0 \pmod{26}$.
   Shortcut: $-y \equiv 26 - y$.
Modular Arithmetic: \( +, -, \times \)

\( \equiv \) is Mod 26 for this slide.

1. Addition: \( x + y \) is easy: wrap around. E.g., \( 20 + 10 \equiv 30 \equiv 4 \).
   Only use the number 30 as an intermediary value on the way to the real answer.

2. \(-7 \equiv x\) where \(0 \leq x \leq 25\).
   Pedantic \(-y\) is the number such that \( y + (-y) \equiv 0 \).
   \(-7 \equiv 19 \pmod{26}\) because \(19 + 7 \equiv 0 \pmod{26}\).
   Shortcut: \(-y \equiv 26 - y\).

3. Mult: \( xy \) is easy: wrap around. E.g., \( 20 \times 10 \equiv 200 \equiv 18 \).
Modular Arithmetic: $+,-,\times$

$\equiv$ is Mod 26 for this slide.

1. Addition: $x + y$ is easy: wrap around. E.g., $20 + 10 \equiv 30 \equiv 4$. Only use the number 30 as an intermediary value on the way to the real answer.

2. $-7 \equiv x$ where $0 \leq x \leq 25$.
   
   Pedantic $-y$ is the number such that $y + (-y) \equiv 0$.
   
   $-7 \equiv 19 \pmod{26}$ because $19 + 7 \equiv 0 \pmod{26}$.
   
   Shortcut: $-y \equiv 26 - y$.

3. Mult: $xy$ is easy: wrap around. E.g., $20 \times 10 \equiv 200 \equiv 18$.
   
   Shortcut to avoid big numbers:
Modular Arithmetic: $+, -, \times$

$\equiv$ is Mod 26 for this slide.

1. Addition: $x + y$ is easy: wrap around. E.g., $20 + 10 \equiv 30 \equiv 4$.

Only use the number 30 as an intermediary value on the way to the real answer.

2. $-7 \equiv x$ where $0 \leq x \leq 25$.

Pedantic $-y$ is the number such that $y + (-y) \equiv 0$.

$-7 \equiv 19 \pmod{26}$ because $19 + 7 \equiv 0 \pmod{26}$.

Shortcut: $-y \equiv 26 - y$.

3. Mult: $xy$ is easy: wrap around. E.g., $20 \times 10 \equiv 200 \equiv 18$.

Shortcut to avoid big numbers:

$$20 \times 10 \equiv -6 \times 10 \equiv -2 \times 30 \equiv -2 \times 4 \equiv -8 \equiv 18.$$
Modular Arithmetic: $+,-,\times$

\[\equiv\] is Mod 26 for this slide.

1. Addition: \(x + y\) is easy: wrap around. E.g., \(20 + 10 \equiv 30 \equiv 4\). Only use the number 30 as an intermediary value on the way to the real answer.

2. \(-7 \equiv x\) where \(0 \leq x \leq 25\).
   
   Pedantic \(-y\) is the number such that \(y + (-y) \equiv 0\). 
   
   \(-7 \equiv 19 \pmod{26}\) because \(19 + 7 \equiv 0 \pmod{26}\).
   
   Shortcut: \(-y \equiv 26 - y\).

3. Mult: \(xy\) is easy: wrap around. E.g., \(20 \times 10 \equiv 200 \equiv 18\).
   
   Shortcut to avoid big numbers:
   
   \[
   20 \times 10 \equiv -6 \times 10 \equiv -2 \times 30 \equiv -2 \times 4 \equiv -8 \equiv 18.
   \]

4. Division: Next Slide
Modular Arithmetic: \( \div \)

\[ \equiv \text{ is Mod 26 for this slide.} \]
\[ \frac{1}{3} \equiv x \text{ where } 0 \leq x \leq 25. \]

Fact: A number \( y \) has an inverse mod 26 if \( y \) and 26 have no common factors. Numbers that have an inverse mod 26:

\[ \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\} \]
Modular Arithmetic: \( \equiv \)

\[ \equiv \text{ is Mod 26 for this slide.} \]

\[ \frac{1}{3} \equiv x \text{ where } 0 \leq x \leq 25. \]

**Pedantic** \( \frac{1}{y} \) is the number such that \( y \times \frac{1}{y} \equiv 1. \)
Modular Arithmetic: $\equiv$

$\equiv$ is Mod 26 for this slide.
$\frac{1}{3} \equiv x$ where $0 \leq x \leq 25$.

**Pedantic** $\frac{1}{y}$ is the number such that $y \times \frac{1}{y} \equiv 1$.

$\frac{1}{3} \equiv 9$ since $9 \times 3 = 27 \equiv 1$. 
Modular Arithmetic: ÷

≡ is Mod 26 for this slide.

\( \frac{1}{3} \equiv x \) where \( 0 \leq x \leq 25 \).

**Pedantic** \( \frac{1}{y} \) is the number such that \( y \times \frac{1}{y} \equiv 1 \).

\( \frac{1}{3} \equiv 9 \) since \( 9 \times 3 = 27 \equiv 1 \).

Shortcut:
Modular Arithmetic: ÷

≡ is Mod 26 for this slide.
\( \frac{1}{3} \equiv x \) where \( 0 \leq x \leq 25 \).

Pedantic \( \frac{1}{y} \) is the number such that \( y \times \frac{1}{y} \equiv 1 \).

\( \frac{1}{3} \equiv 9 \) since \( 9 \times 3 = 27 \equiv 1 \).

Shortcut: there is an algorithm that finds \( \frac{1}{y} \) quickly.
Modular Arithmetic: \( \equiv \)

\( \equiv \) is Mod 26 for this slide.

\( \frac{1}{3} \equiv x \) where \( 0 \leq x \leq 25 \).

**Pedantic** \( \frac{1}{y} \) is the number such that \( y \times \frac{1}{y} \equiv 1 \).

\( \frac{1}{3} \equiv 9 \) since \( 9 \times 3 = 27 \equiv 1 \).

Shortcut: there is an algorithm that finds \( \frac{1}{y} \) quickly.

We will study the algorithm later.
Modular Arithmetic: \[\equiv\]

\[\equiv\] is Mod 26 for this slide.
\[
\frac{1}{3} \equiv x \text{ where } 0 \leq x \leq 25.
\]

**Pedantic** \[
\frac{1}{y} \text{ is the number such that } y \times \frac{1}{y} \equiv 1.
\]
\[
\frac{1}{3} \equiv 9 \text{ since } 9 \times 3 = 27 \equiv 1.
\]

Shortcut: there is an algorithm that finds \(\frac{1}{y}\) quickly. We will study the algorithm later.

\[
\frac{1}{2} \equiv x \text{ where } 0 \leq x \leq 25.
\]
Modular Arithmetic: \( \equiv \)

\( \equiv \) is Mod 26 for this slide.
\( \frac{1}{3} \equiv x \) where \( 0 \leq x \leq 25. \)

**Pedantic** \( \frac{1}{y} \) is the number such that \( y \times \frac{1}{y} \equiv 1. \)

\( \frac{1}{3} \equiv 9 \) since \( 9 \times 3 = 27 \equiv 1. \)

Shortcut: there is an algorithm that finds \( \frac{1}{y} \) quickly.
We will study the algorithm later.

\( \frac{1}{2} \equiv x \) where \( 0 \leq x \leq 25. \) Think about.
Modular Arithmetic: \[ \equiv \]

\[ \equiv \] is Mod 26 for this slide.
\[ \frac{1}{3} \equiv x \text{ where } 0 \leq x \leq 25. \]

**Pedantic** \[ \frac{1}{y} \] is the number such that \[ y \times \frac{1}{y} \equiv 1. \]

\[ \frac{1}{3} \equiv 9 \text{ since } 9 \times 3 = 27 \equiv 1. \]

Shortcut: there is an algorithm that finds \[ \frac{1}{y} \] quickly. We will study the algorithm later.

\[ \frac{1}{2} \equiv x \text{ where } 0 \leq x \leq 25. \text{ Think about.} \]

No such \( x \) exists.
Modular Arithmetic: \( \equiv \)

\( \equiv \) is Mod 26 for this slide.
\( \frac{1}{3} \equiv x \) where \( 0 \leq x \leq 25. \)

**Pedantic** \( \frac{1}{y} \) is the number such that \( y \times \frac{1}{y} \equiv 1. \)
\( \frac{1}{3} \equiv 9 \) since \( 9 \times 3 = 27 \equiv 1. \)
Shortcut: there is an algorithm that finds \( \frac{1}{y} \) quickly.
We will study the algorithm later.

\( \frac{1}{2} \equiv x \) where \( 0 \leq x \leq 25. \) Think about.
No such \( x \) exists.

**Fact** A number \( y \) has an inverse mod 26 if \( y \) and 26 have no common factors. Numbers that have an inverse mod 26:

\[ \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\} \]
The Shift Cipher, Formally

- $\mathcal{M} = \{\text{all texts in lowercase English alphabet}\}$
  $\mathcal{M}$ for Message space.
  All arithmetic mod 26.

- Choose uniform $s \in \mathcal{K} = \{0, \ldots, 25\}$. $\mathcal{K}$ for Keyspace.

- Encode $(m_1 \ldots m_t)$ as $(m_1 + s \ldots m_t + s)$.

- Decode $(c_1 \ldots c_t)$ as $(c_1 - s \ldots c_t - s)$.

- Can verify that correctness holds.
Cracking the Shift Cipher
Is the Shift Cipher Secure?

- No – only 26 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - Only one possibility will “make sense”

- Example of a “brute-force” or “exhaustive-search” attack
Example

- Ciphertext uryyb jbeyq
- Try every possible key...
  - tqxxa iadxp
  - spwwz hzcwo
  - ...
  - hello world

Question: We can tell that hello world is correct but how can a computer do that. Can we mechanize the process of picking out the right one?
Example

- Ciphertext uryyb jbeyq
- Try every possible key...
  - tqxxa iadxp
  - spwwz hzcwo
  - ...
  - hello world

**Question:** We can tell that **hello world** is correct but how can a computer do that. Can we mechanize the process of picking out the right one?
Letter Frequencies

The chart shows the percentage frequency of each letter in a given text.

- The highest frequency is 'e' at 12.7%
- Other high frequencies include 't' at 9.1%, 's' at 6.3%, and 'n' at 6.0%
- Lower frequencies include 'y' and 'z' at 0.1%

The y-axis represents the percentage, while the x-axis lists the letters from 'a' to 'z'.
Freq Vectors

Let $T$ be a long text. Length $N$. May or may not be coded.

Let $N_a$ be the number of $a$'s in $T$.
Let $N_b$ be the number of $b$'s in $T$. 
: 

...
Freq Vectors

Let $T$ be a long text. Length $N$. May or may not be coded.

Let $N_a$ be the number of $a$'s in $T$.
Let $N_b$ be the number of $b$'s in $T$.

The **Freq Vector of** $T$ is

$$\vec{f}_T = \left( \frac{N_a}{N}, \frac{N_b}{N}, \cdots, \frac{N_z}{N} \right)$$
How to Tell Is-English

Given a Text $T$ you want to tell if it's English or a Shift of English. You do not want to read all 26 possible shifts of $T$. 

$\vec{f}_E$ is the Frequency Vector for English. $\vec{f}_T$ is the Frequency Vector for $T$. 

How to tell if $\vec{f}_T$ is close to $\vec{f}_E$? 

Ideas:

$\sum_{i=0}^{25} |f_E,i - f_T,i|^2$

These are good ideas but do not seem to work.
How to Tell Is-English

Given a Text $T$ you want to tell if it’s English or a Shift of English. You do not want to read all 26 possible shifts of $T$.

Let $\vec{f}_E$ be Freq Vector for English.
Let $\vec{f}_T$ be Freq Vector for $T$. 
How to Tell Is-English

Given a Text $T$ you want to tell if it’s English or a Shift of English. You do not want to read all 26 possible shifts of $T$.

Let $\vec{f}_E$ be Freq Vector for English.
Let $\vec{f}_T$ be Freq Vector for $T$.
How to tell if $\vec{f}_T$ is close to $\vec{f}_E$?
How to Tell Is-English

Given a Text $T$ you want to tell if it’s English or a Shift of English. You do not want to read all 26 possible shifts of $T$.

Let $\vec{f}_E$ be Freq Vector for English.
Let $\vec{f}_T$ be Freq Vector for $T$.
How to tell if $\vec{f}_T$ is close to $\vec{f}_E$? Ideas?
How to Tell Is-English

Given a Text $T$ you want to tell if it’s English or a Shift of English. You do not want to read all 26 possible shifts of $T$.

Let $\vec{f}_E$ be Freq Vector for English.
Let $\vec{f}_T$ be Freq Vector for $T$.
How to tell if $\vec{f}_T$ is close to $\vec{f}_E$? Ideas?

$\sum_{i=0}^{25} |f_E,i - f_T,i|$
$\sum_{i=0}^{25} (f_E,i - f_T,i)^2$

These are good ideas but do not seem to work.
Given a Text $T$ you want to tell if it’s **English** or a **Shift of English**. You do not want to read all 26 possible shifts of $T$.

Let $\vec{f}_E$ be Freq Vector for English.
Let $\vec{f}_T$ be Freq Vector for $T$.

How to tell if $\vec{f}_T$ is close to $\vec{f}_E$? Ideas?

- $\sum_{i=0}^{25} |f_{E,i} - f_{T,i}|$
- $\sum_{i=0}^{25} (f_{E,i} - f_{T,i})^2$

These are good ideas but do not seem to work.
Vorlons Alphabet: \( \{a, b, c, d\} \)

- Vorlon freq shifted by 0 is \( \vec{f}_0 = \{0.5, 0.3, 0.1, 0.1\} \).
- Vorlon freq shifted by 1 is \( \vec{f}_1 = \{0.1, 0.5, 0.3, 0.1\} \).
- Vorlon freq shifted by 2 is \( \vec{f}_2 = \{0.1, 0.1, 0.5, 0.3\} \).
- Vorlon freq shifted by 3 is \( \vec{f}_3 = \{0.3, 0.1, 0.1, 0.5\} \).
Vorlons Alphabet: \( \{ a, b, c, d \} \)

- Vorlon freq shifted by 0 is \( \vec{f}_0 = \{0.5, 0.3, 0.1, 0.1\} \).
- Vorlon freq shifted by 1 is \( \vec{f}_1 = \{0.1, 0.5, 0.3, 0.1\} \).
- Vorlon freq shifted by 2 is \( \vec{f}_2 = \{0.1, 0.1, 0.5, 0.3\} \).
- Vorlon freq shifted by 3 is \( \vec{f}_3 = \{0.3, 0.1, 0.1, 0.5\} \).

\[ \vec{f}_0 \cdot \vec{f}_0 = 0.5^2 + 0.3^2 + 0.1^2 + 0.1^2 = 0.36 \]
Vorlons Alphabet: \(\{a, b, c, d\}\)

- Vorlon freq shifted by 0 is \(\vec{f}_0 = \{0.5, 0.3, 0.1, 0.1\}\).
- Vorlon freq shifted by 1 is \(\vec{f}_1 = \{0.1, 0.5, 0.3, 0.1\}\).
- Vorlon freq shifted by 2 is \(\vec{f}_2 = \{0.1, 0.1, 0.5, 0.3\}\).
- Vorlon freq shifted by 3 is \(\vec{f}_3 = \{0.3, 0.1, 0.1, 0.5\}\).

\[
\vec{f}_0 \cdot \vec{f}_0 = 0.5^2 + 0.3^2 + 0.1^2 + 0.1^2 = 0.36
\]

\[
\vec{f}_0 \cdot \vec{f}_1 = 0.5 \times 0.1 + 0.3 \times 0.5 + 0.1 \times 0.3 + 0.1 \times 0.1 = 0.24
\]
Vorlons Alphabet: \{a, b, c, d\}

- Vorlon freq shifted by 0 is \(\vec{f}_0 = \{0.5, 0.3, 0.1, 0.1\}\).
- Vorlon freq shifted by 1 is \(\vec{f}_1 = \{0.1, 0.5, 0.3, 0.1\}\).
- Vorlon freq shifted by 2 is \(\vec{f}_2 = \{0.1, 0.1, 0.5, 0.3\}\).
- Vorlon freq shifted by 3 is \(\vec{f}_3 = \{0.3, 0.1, 0.1, 0.5\}\).

\[ \vec{f}_0 \cdot \vec{f}_0 = 0.5^2 + 0.3^2 + 0.1^2 + 0.1^2 = 0.36 \]
\[ \vec{f}_0 \cdot \vec{f}_1 = 0.5 \times 0.1 + 0.3 \times 0.5 + 0.1 \times 0.3 + 0.1 \times 0.1 = 0.24 \]
\[ \vec{f}_0 \cdot \vec{f}_2 = 0.5 \times 0.1 + 0.3 \times 0.1 + 0.1 \times 0.5 + 0.1 \times 0.3 = 0.16 \]
**Vorlons Alphabet:** \{a, b, c, d\}

- Vorlon freq shifted by 0 is \(\vec{f}_0 = \{0.5, 0.3, 0.1, 0.1\}\).
- Vorlon freq shifted by 1 is \(\vec{f}_1 = \{0.1, 0.5, 0.3, 0.1\}\).
- Vorlon freq shifted by 2 is \(\vec{f}_2 = \{0.1, 0.1, 0.5, 0.3\}\).
- Vorlon freq shifted by 3 is \(\vec{f}_3 = \{0.3, 0.1, 0.1, 0.5\}\).

\[
\vec{f}_0 \cdot \vec{f}_0 = 0.5^2 + 0.3^2 + 0.1^2 + 0.1^2 = 0.36
\]
\[
\vec{f}_0 \cdot \vec{f}_1 = 0.5 \times 0.1 + 0.3 \times 0.5 + 0.1 \times 0.3 + 0.1 \times 0.1 = 0.24
\]
\[
\vec{f}_0 \cdot \vec{f}_2 = 0.5 \times 0.1 + 0.3 \times 0.1 + 0.1 \times 0.5 + 0.1 \times 0.3 = 0.16
\]
\[
\vec{f}_0 \cdot \vec{f}_3 = 0.5 \times 0.3 + 0.3 \times 0.1 + 0.1 \times 0.1 + 0.1 \times 0.5 = 0.24
\]
Vorlons Alphabet: \{a, b, c, d\}

- Vorlon freq shifted by 0 is $\vec{f}_0 = \{0.5, 0.3, 0.1, 0.1\}$.
- Vorlon freq shifted by 1 is $\vec{f}_1 = \{0.1, 0.5, 0.3, 0.1\}$.
- Vorlon freq shifted by 2 is $\vec{f}_2 = \{0.1, 0.1, 0.5, 0.3\}$.
- Vorlon freq shifted by 3 is $\vec{f}_3 = \{0.3, 0.1, 0.1, 0.5\}$.

\[
\vec{f}_0 \cdot \vec{f}_0 = 0.5^2 + 0.3^2 + 0.1^2 + 0.1^2 = 0.36
\]
\[
\vec{f}_0 \cdot \vec{f}_1 = 0.5 \cdot 0.1 + 0.3 \cdot 0.5 + 0.1 \cdot 0.3 + 0.1 \cdot 0.1 = 0.24
\]
\[
\vec{f}_0 \cdot \vec{f}_2 = 0.5 \cdot 0.1 + 0.3 \cdot 0.1 + 0.1 \cdot 0.5 + 0.1 \cdot 0.3 = 0.16
\]
\[
\vec{f}_0 \cdot \vec{f}_3 = 0.5 \cdot 0.3 + 0.3 \cdot 0.1 + 0.1 \cdot 0.1 + 0.1 \cdot 0.5 = 0.24
\]

**Upshot**

$\vec{f}_0 \cdot \vec{f}_0$ big

For $i \in \{1, 2, 3\}$, $\vec{f}_0 \cdot \vec{f}_i$ small
English Alphabet: \{a, \ldots, z\}

- English freq shifted by 0 is $\vec{f}_0$
- For $1 \leq i \leq 25$, English freq shifted by $i$ is $\vec{f}_i$. 
English Alphabet: \( \{a, \ldots, z\} \)

- English freq shifted by 0 is \( \vec{f}_0 \)
- For \( 1 \leq i \leq 25 \), English freq shifted by \( i \) is \( \vec{f}_i \).

\[ \vec{f}_0 \cdot \vec{f}_0 \sim 0.065 \]
English Alphabet: \{a, \ldots, z\}

- English freq shifted by 0 is \( \vec{f}_0 \)
- For \( 1 \leq i \leq 25 \), English freq shifted by \( i \) is \( \vec{f}_i \).

\[
\vec{f}_0 \cdot \vec{f}_0 \sim 0.065
\]

\[
\max_{1 \leq i \leq 25} \vec{f}_0 \cdot \vec{f}_i \sim 0.038
\]
English Alphabet: \( \{a, \ldots, z\} \)

- English freq shifted by 0 is \( \vec{f}_0 \)
- For \( 1 \leq i \leq 25 \), English freq shifted by \( i \) is \( \vec{f}_i \).

\[
\vec{f}_0 \cdot \vec{f}_0 \sim 0.065
\]

\[
\max_{1 \leq i \leq 25} \vec{f}_0 \cdot \vec{f}_i \sim 0.038
\]

**Upshot**
\( \vec{f}_0 \cdot \vec{f}_0 \) **big**

For \( i \in \{1, \ldots, 25\} \), \( \vec{f}_0 \cdot \vec{f}_i \) **small**
English Alphabet: \( \{a, \ldots, z\} \)

- English freq shifted by 0 is \( \vec{f}_0 \)
- For \( 1 \leq i \leq 25 \), English freq shifted by \( i \) is \( \vec{f}_i \).

\[
\vec{f}_0 \cdot \vec{f}_0 \sim 0.065
\]

\[
\max_{1 \leq i \leq 25} \vec{f}_0 \cdot \vec{f}_i \sim 0.038
\]

**Upshot**

\( \vec{f}_0 \cdot \vec{f}_0 \) **big**

For \( i \in \{1, \ldots, 25\} \), \( \vec{f}_0 \cdot \vec{f}_i \) **small**

**Henceforth** \( \vec{f}_0 \) will be denoted \( \vec{f}_E \). \( E \) is for **English**
Is English

We describe a way to tell if a text Is English that we will use throughout this course.
Is English

We describe a way to tell if a text Is English that we will use throughout this course.

1. Input($T$) a text
2. Compute $\vec{f}_T$, the freq vector for $T$
3. Compute $\vec{f}_E \cdot \vec{f}_T$. If $\approx 0.065$ then output YES, else NO

Note: What if $\vec{f}_T \cdot \vec{f}_E = 0.061$? If shift cipher used, this will never happen. If 'simple' ciphers used, this will never happen. If 'difficult' cipher used, we may use different IS-ENGLISH function.
Is English

We describe a way to tell if a text is English that we will use throughout this course.

1. Input($T$) a text
2. Compute $\vec{f}_T$, the freq vector for $T$
3. Compute $\vec{f}_E \cdot \vec{f}_T$. If $\approx 0.065$ then output YES, else NO

Note: What if $\vec{f}_T \cdot \vec{f}_E = 0.061$?
Is English

We describe a way to tell if a text Is English that we will use throughout this course.

1. Input($T$) a text
2. Compute $\vec{f}_T$, the freq vector for $T$
3. Compute $\vec{f}_E \cdot \vec{f}_T$. If $\approx 0.065$ then output YES, else NO

Note: What if $\vec{f}_T \cdot \vec{f}_E = 0.061$?
If shift cipher used, this will never happen.
We describe a way to tell if a text **Is English** that we will use throughout this course.

1. Input($T$) a text
2. Compute $\vec{f}_T$, the freq vector for $T$
3. Compute $\vec{f}_E \cdot \vec{f}_T$. If $\approx 0.065$ then output YES, else NO

**Note:** What if $\vec{f}_T \cdot \vec{f}_E = 0.061$?

If shift cipher used, this will **never** happen.
If ‘simple’ ciphers used, this will **never** happen.
Is English

We describe a way to tell if a text Is English that we will use throughout this course.

1. Input(\(T\)) a text
2. Compute \(\vec{f}_T\), the freq vector for \(T\)
3. Compute \(\vec{f}_E \cdot \vec{f}_T\). If \(\approx 0.065\) then output YES, else NO

Note: What if \(\vec{f}_T \cdot \vec{f}_E = 0.061\)?
If shift cipher used, this will never happen.
If ‘simple’ ciphers used, this will never happen.
If ‘difficult’ cipher used, we may use different IS-ENGLISH function.
Given $T$ a long text that you KNOW was coded by shift.
Cracking Shift Cipher

- Given $T$ a long text that you KNOW was coded by shift.
- For $s = 0$ to 25
  - Create $T_s$ which is $T$ shifted by $s$. 
  
  If Is English($T_s$) = YES then output $T_s$ and stop. Else try next value of $s$. 
  
  Note: No Near Misses. There will not be two values of $s$ that are both close to 0.
Cracking Shift Cipher

- Given $T$ a long text that you KNOW was coded by shift.
- For $s = 0$ to 25
  - Create $T_s$ which is $T$ shifted by $s$.
  - If $\text{Is English}(T_s) = \text{YES}$ then output $T_s$ and stop. Else try next value of $s$. 

Note: No Near Misses. There will not be two values of $s$ that are both close to 0.
Cracking Shift Cipher

- Given $T$ a long text that you KNOW was coded by shift.
- For $s = 0$ to 25
  - Create $T_s$ which is $T$ shifted by $s$.
  - If Is English($T_s$) = YES then output $T_s$ and stop. Else try next value of $s$.

**Note:** No Near Misses. There will not be two values of $s$ that are both close to 0.065.
Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Speeding Up Cracking of Shift Cipher

In the last slide we tried all shifts in order. Can do better: Most common letter is probably e. If not then 2nd most. . . .
In the last slide we tried all shifts in order. Can do better: Most common letter is probably e. If not then 2nd most...

- Given $T$ a long text that you KNOW was coded by shift.
Speeding Up Cracking of Shift Cipher

In the last slide we tried all shifts in order. Can do better: Most common letter is probably e. If not then 2nd most.

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$.

$\vec{g}$ · $\vec{f}$ $\approx$ 0.065 then stop: $T_i$ is your text. Else try next value of $i$. Note: Quite likely to succeed in the first try, or at least very early.
Speeding Up Cracking of Shift Cipher

In the last slide we tried all shifts in order. Can do better: Most common letter is probably e. If not then 2nd most. . . .

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$.
- Sort vector. So most common letter is $\sigma_0$, next is $\sigma_1$, etc.
Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order. Can do better: Most common letter is probably e. If not then 2nd most.

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$.
- Sort vector. So most common letter is $\sigma_0$, next is $\sigma_1$, etc.
- For $i = 0$ to 25

$\vec{g}$ is the freq vector for $T_i$ which is $T$ shifted as if $\sigma_i$ maps to e. Compute $\vec{g} \cdot \vec{f}$. If $\approx 0.065$ then stop: $T_i$ is your text. Else try next value of $i$.
Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order. Can do better: Most common letter is probably e. If not then 2nd most. . . .

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$.
- Sort vector. So most common letter is $\sigma_0$, next is $\sigma_1$, etc.
- For $i = 0$ to 25
  - Create $T_i$ which is $T$ shifted as if $\sigma_i$ maps to e.
Speeding Up Cracking of Shift Cipher

In the last slide we tried all shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most. . . .

▶ Given $T$ a long text that you KNOW was coded by shift.
▶ Find frequencies of all letters, form vector $\vec{f}$.
▶ Sort vector. So most common letter is $\sigma_0$, next is $\sigma_1$, etc.
▶ For $i = 0$ to 25
  ▶ Create $T_i$ which is $T$ shifted as if $\sigma_i$ maps to e.
  ▶ Compute $\vec{g}$, the freq vector for $T_i$. 
Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably *e*. If not then 2nd most. . . .

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$.
- Sort vector. So most common letter is $\sigma_0$, next is $\sigma_1$, etc.
- For $i = 0$ to 25
  - Create $T_i$ which is $T$ shifted as if $\sigma_i$ maps to $e$.
  - Compute $\vec{g}$, the freq vector for $T_i$.
  - Compute $\vec{g} \cdot \vec{f}_E$. If $\approx 0.065$ then stop: $T_i$ is your text. Else try next value of $i$. Note: Quite likely to succeed in the first try, or at least very early.
In the last slide we tried *all* shifts in order. Can do better: Most common letter is probably e. If not then 2nd most. . . .

- **Given** $T$ a long text that you KNOW was coded by shift.
- **Find** frequencies of all letters, form vector $\vec{f}$.
- **Sort** vector. So most common letter is $\sigma_0$, next is $\sigma_1$, etc.
- **For** $i = 0$ to 25
  - **Create** $T_i$ which is $T$ shifted as if $\sigma_i$ maps to e.
  - **Compute** $\vec{g}$, the freq vector for $T_i$.
  - **Compute** $\vec{g} \cdot \vec{f}_E$. If $\approx 0.065$ then stop: $T_i$ is your text. Else try next value of $i$.

**Note:** Quite likely to succeed in the first try, or at least very early.