BILL RECORDED
LECTURE
REVIEW FOR MIDTERM
SHIFT CIPHER
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.
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 $T$.

\[ \vec{f}_T = (N_a, N_b, \ldots, N_z) \]
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

$$
\vec{f}_T = \left( \frac{N_a}{N}, \frac{N_b}{N}, \cdots, \frac{N_z}{N} \right)
$$
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 complicated 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 \(\vec{f}_T \cdot \vec{f}_E = 0.061\)?
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 \textbf{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

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

If \textit{shift cipher} used, this will never happen.
If \textit{simple ciphers} used, this will never happen.
If \textit{complicated cipher} used, we may use different IS-ENGLISH function.
Cracking Shift Cipher

- 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 $s$. 

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 $\text{Is English}(T_s) = \text{YES}$ 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 0.065.
Variants of the Shift Cipher

1. Different alphabets.
2. Different languages.
3. Different domains (e.g., Credit Card Numbers).
4. $\Sigma = \{0, \ldots, 9\}$ (e.g., Credit Cards).

These all have
1. Small key spaces.
2. Uneven distribution of symbols.

So can be cracked.
Variants of the Shift Cipher

1. Different alphabets.
Variants of the Shift Cipher

1. Different alphabets.
2. Different languages.

$\Sigma =$ \{0, ..., 9\} (e.g., Credit Cards).

These all have
1. Small key spaces.
2. Uneven distribution of symbols.

So can be cracked.
Variants of the Shift Cipher

1. Different alphabets.
2. Different languages.
3. Different domains (e.g., Credit Card Numbers).
Variants of the Shift Cipher

1. Different alphabets.
2. Different languages.
3. Different domains (e.g., Credit Card Numbers).
4. $\Sigma = \{0, \ldots, 9\}$ (e.g., Credit Cards).
Variants of the Shift Cipher

1. Different alphabets.
2. Different languages.
3. Different domains (e.g., Credit Card Numbers).
4. $\Sigma = \{0, \ldots, 9\}$ (e.g., Credit Cards).

These all have

1. Small key spaces.
2. Uneven distribution of symbols.

So can be cracked.
Kerckhoff’s principle

We made the comment *We KNOW that SHIFT was used.* More generally we will always use the following assumption.

*Kerckhoff’s principle:*

- Eve knows the encryption scheme.
- Eve knows the alphabet and the language.
- Eve does not know the key.
- The key is chosen at random.
Kerckhoff’s principle

We made the comment We KNOW that SHIFT was used. More generally we will always use the following assumption.

Kerckhoff’s principle:

▶ Eve knows The encryption scheme.
Kerckhoff’s principle

We made the comment We KNOW that SHIFT was used. More generally we will always use the following assumption. Kerckhoff’s principle:

- Eve knows The encryption scheme.
- Eve knows the alphabet and the language.
Kerckhoff’s principle

We made the comment \textbf{We KNOW that SHIFT was used.} More generally we will always use the following assumption.

\textbf{Kerckhoff’s principle:}

\begin{itemize}
  \item Eve knows \textbf{The encryption scheme.}
  \item Eve knows \textbf{the alphabet and the language.}
  \item Eve does not know \textbf{the key}
\end{itemize}
Kerckhoff’s principle

We made the comment **We KNOW that SHIFT was used**. More generally we will always use the following assumption.

**Kerckhoff’s principle:**

- Eve knows **The encryption scheme**.
- Eve knows **the alphabet and the language**.
- Eve does not know **the key**
- The key is chosen **at random**.
Other Single Letter Ciphers
Affine Cipher

**Def** The Affine cipher with $a$, $b$:

1. Encrypt via $x \rightarrow ax + b \pmod{26}$. ($a$ has to be rel prime to 26 so that $a^{-1} \pmod{26}$ exists.

2. Decrypt via $x \rightarrow a^{-1}(x - b) \pmod{26}$.

**Limit on Keys** $(a, b)$ must be such that $a$ has an inverse.

**Number of** $(a, b) \phi(|\Sigma|) \times |\Sigma|.$

**Easily cracked** Only 312 keys. Use **Is-English** for each key.
The Quadratic Cipher

**Def** The Quadratic cipher with $a, b, c$: Encrypt via \( x \rightarrow ax^2 + bx + c \mod 26 \).
The Quadratic Cipher

**Def** The Quadratic cipher with $a$, $b$, $c$: Encrypt via $x \rightarrow ax^2 + bx + c \pmod{26}$.

Does not work and was never used because:

No easy test for Invertibility (depends on def of easy).
Def **Gen Sub Cipher** with perm $f$ on $\{0, \ldots, 25\}$. 

1. Encrypt via $x \rightarrow f(x)$.
2. Decrypt via $x \rightarrow f^{-1}(x)$. 

**PRO** Very Large Key Space: $26!$, so brute force not an option.

**CON** 100 years ago Hard to use, so we will look at alternatives that take a short seed and get a random looking perm.

**CON** today Crackable. We discuss how later.
General Substitution Cipher

**Def Gen Sub Cipher** with perm $f$ on $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow f(x)$.
General Substitution Cipher

**Def Gen Sub Cipher** with perm $f$ on $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow f(x)$.
2. Decrypt via $x \rightarrow f^{-1}(x)$.

PRO

Very Large Key Space: $26!$, so brute force not an option.

CON

100 years ago

Hard to use, so we will look at alternatives

CON today

Crackable. We discuss how later.
General Substitution Cipher

**Def Gen Sub Cipher** with perm $f$ on $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow f(x)$.
2. Decrypt via $x \rightarrow f^{-1}(x)$.

**PRO** Very Large Key Space: 26!, so brute force not an option.
Def Gen Sub Cipher with perm \( f \) on \( \{0, \ldots, 25\} \).

1. Encrypt via \( x \rightarrow f(x) \).
2. Decrypt via \( x \rightarrow f^{-1}(x) \).

**PRO** Very Large Key Space: 26!, so brute force not an option.

**CON** 100 years ago Hard to use, so we will look at alternatives that take a short seed and get a random looking perm.
General Substitution Cipher

Def Gen Sub Cipher with perm \( f \) on \{0,\ldots,25\}.
1. Encrypt via \( x \rightarrow f(x) \).
2. Decrypt via \( x \rightarrow f^{-1}(x) \).

PRO Very Large Key Space: 26!, so brute force not an option.
CON 100 years ago Hard to use, so we will look at alternatives that take a short seed and get a random looking perm.
CON today Crackable. We discuss how later.
Keyword-Shift Cipher. Key is (Word, Shift)

\[ \Sigma = \{a, \ldots, k\} \text{. Key: (jack, 4).} \]
Keyword-Shift Cipher. Key is (Word, Shift)

\[ \Sigma = \{ a, \ldots, k \} \]. **Key:** (jack, 4).

Alice then does the following:
Keyword-Shift Cipher. Key is (Word, Shift)

\( \Sigma = \{a, \ldots, k\} \). **Key:** (jack, 4).

Alice then does the following:

1. List out the key word and then the remaining letters:

\[
\begin{array}{|l|l|l|l|l|l|l|l|l|}
\hline
j & a & c & k & b & d & e & f & g & h & i \\
\hline
\end{array}
\]
Keyword-Shift Cipher. Key is (Word, Shift)

Σ = \{a, \ldots, k\}. **Key:** (jack, 4).

Alice then does the following:

1. List out the key word and then the remaining letters:

\[
\begin{array}{cccccccccc}
j & a & c & k & b & d & e & f & g & h & i \\
\end{array}
\]

2. Now do Shift 4 on this:

\[
\begin{array}{cccccccccc}
f & g & h & i & j & a & c & k & b & d & e \\
\end{array}
\]
Keyword-Shift Cipher. Key is (Word, Shift)

Σ = \{a, \ldots, k\}. Key: (jack, 4).

Alice then does the following:
1. List out the key word and then the remaining letters:

   \[ j \ a \ c \ k \ b \ d \ e \ f \ g \ h \ i \]

2. Now do Shift 4 on this:

   \[ f \ g \ h \ i \ j \ a \ c \ k \ b \ d \ e \]

   This is where a, b, c, \ldots go, so:

   \[ a \ b \ c \ d \ e \ f \ g \ h \ i \ j \ k \]
   \[ f \ g \ h \ i \ j \ a \ c \ k \ b \ d \ e \]
UPSHOT

1000 years ago

1. General Sub Student was hard to use and hard to crack.
2. Keyword Shift cipher was easy to use and hard to crack since it looked random.

Today

1. General Sub Student is easy to use and easy to crack.
2. Keyword Shift cipher is a pedagogical example of a pseudo-random generator.
3. A pseudo-random generator takes a short random string and produces a long pseudo-random string.
4. Pseudo-random generators are important in modern crypto to use a pseudo-one-time-pad.
5. We will see examples of modern pseudo-random generators later in the course.
UPSHOT

1000 years ago

1. General Sub Student was hard to use and hard to crack
UPSHOT

1000 years ago
1. General Sub Student was hard to use and hard to crack
2. Keyword Shift cipher was easy to use and hard to crack since looked random..

Today
UPSHOT

1000 years ago
1. General Sub Student was hard to use and hard to crack
2. Keyword Shift cipher was easy to use and hard to crack since looked random..

Today
1. General Sub Student easy to use and easy to crack.
UPSHOT

1000 years ago
1. General Sub Student was hard to use and hard to crack
2. Keyword Shift cipher was easy to use and hard to crack since looked random..

Today
1. General Sub Student easy to use and easy to crack.
2. Keyword Shift cipher is a pedagogical example of a psuedo-random generator.
UPSHOT

1000 years ago
1. General Sub Student was hard to use and hard to crack
2. Keyword Shift cipher was easy to use and hard to crack since **looked** random..

Today
1. General Sub Student easy to use and easy to crack.
2. Keyword Shift cipher is a pedagogical example of a psuedo-random generator.
3. A psuedo-random generator takes a **short random string** and produces a **long psuedo-random string**.
UPSHOT

1000 years ago

1. General Sub Student was hard to use and hard to crack
2. Keyword Shift cipher was easy to use and hard to crack since looked random.

Today

1. General Sub Student easy to use and easy to crack.
2. Keyword Shift cipher is a pedagogical example of a psuedo-random generator.
3. A psuedo-random generator takes a short random string and produces a long psuedo-random string.
4. Psuedo-random generators are important in modern crypto to use a psuedo-one-time-pad.
UPSHOT

1000 years ago
1. General Sub Student was hard to use and hard to crack
2. Keyword Shift cipher was easy to use and hard to crack since looked random..

Today
1. General Sub Student easy to use and easy to crack.
2. Keyword Shift cipher is a pedagogical example of a psuedo-random generator.
3. A psuedo-random generator takes a short random string and produces a long psuedo-random string.
4. Psuedo-random generators are important in modern crypto to use a psuedo-one-time-pad.
5. We will see examples of modern psuedo-random generators later in the course.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
2. The **2-grams** of $T$ are just the contiguous pairs of letters in $T$, counting repeats. Also called **bigrams**.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
2. The **2-grams** of $T$ are just the contiguous pairs of letters in $T$, counting repeats. Also called **bigrams**.
3. The **3-grams** of $T$ you can guess. Also called **trigrams**.
**Terminology: 1-Gram, 2-Gram, 3-Gram**

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
2. The **2-grams** of $T$ are just the contiguous pairs of letters in $T$, counting repeats. Also called **bigrams**.
3. The **3-grams** of $T$ you can guess. Also called **trigrams**.
4. One usually talks about the freq of $n$-grams.
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!

3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq’s of $n$-grams in $\sigma(T)$. 

Notation and Parameter for a Family of Algorithms

**Notation** Let \( \sigma \) be a perm and \( T \) a text.

1. \( f_E \) is freq of \( n \)-grams. It is a \( 26^n \) long vector. (Formally we should use \( f_E(n) \). We omit the \( n \). The value of \( n \) will be clear from context.)
2. \( \sigma(T) \) is taking \( T \) and applying \( \sigma \) to it. If \( \sigma^{-1} \) was used to encrypt, then \( \sigma(T) \) will be English!
3. \( f_{\sigma(T)} \) is the \( 26^n \)-long vector of freq’s of \( n \)-grams in \( \sigma(T) \).
4. \( I \) and \( R \) will be parameters we discuss later.
Notation and Parameter for a Family of Algorithms

Notation

Let \( \sigma \) be a perm and \( T \) a text.

1. \( f_E \) is freq of \( n \)-grams. It is a \( 26^n \) long vector. (Formally we should use \( f_E(n) \). We omit the \( n \). The value of \( n \) will be clear from context.)

2. \( \sigma(T) \) is taking \( T \) and applying \( \sigma \) to it. If \( \sigma^{-1} \) was used to encrypt, then \( \sigma(T) \) will be English!

3. \( f_{\sigma(T)} \) is the \( 26^n \)-long vector of freq’s of \( n \)-grams in \( \sigma(T) \).

4. \( I \) and \( R \) will be parameters we discuss later. \( I \) stands for Iterations and will be large (like 2000).


Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!

3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq’s of $n$-grams in $\sigma(T)$.

4. I and R will be parameters we discuss later. I stands for Iterations and will be large (like 2000). R stands for Redos and will be small (like 5).
Input $T$. Find Freq of 1-grams and $n$-grams.
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

Candidates for $\sigma$ are $\sigma_1, ..., \sigma_R$

Pick the $\sigma_r$ with min good $r$ or have human look at all $\sigma_r(T)$.

The parameters $R$ and $I$ need to be picked carefully.
Input $T$. Find Freq of 1-grams and $n$-grams. 

$\sigma_{\text{init}}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$$\sigma_r \leftarrow \sigma_{\text{init}}$$
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

\[ \sigma_r \leftarrow \sigma_{\text{init}} \]

For $i = 1$ to $I$ ($I$ is large, about 2000)
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$\sigma_r \leftarrow \sigma_{\text{init}}$

For $i = 1$ to $I$ ($I$ is large, about 2000)

Pick $j, k \in \{0, \ldots, 25\}$ at Random.
Input $T$. Find Freq of 1-grams and $n$-grams. 

$\sigma_{\text{init}}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

\[
\sigma_r \leftarrow \sigma_{\text{init}}
\]

For $i = 1$ to $I$ ($I$ is large, about 2000)

- Pick $j, k \in \{0, \ldots, 25\}$ at Random.
- Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped
$n$-Gram Algorithm

Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

\[ \sigma_r \leftarrow \sigma_{\text{init}} \]

For $i = 1$ to $I$ ($I$ is large, about 2000)

- Pick $j, k \in \{0, \ldots, 25\}$ at Random.
- Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped
- If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for $\sigma$ are $\sigma_1, \ldots, \sigma_R$

Pick the $\sigma_r$ with min good $r$ or have human look at all $\sigma_r(T)$

The parameters $R$ and $I$ need to be picked carefully.
**n-Gram Algorithm**

Input $T$. Find Freq of 1-grams and $n$-grams.  
$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$$\sigma_r \leftarrow \sigma_{\text{init}}$$

For $i = 1$ to $I$ ($I$ is large, about 2000)

- Pick $j, k \in \{0, \ldots, 25\}$ at Random.
- Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped
- If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for $\sigma$ are $\sigma_1, \ldots, \sigma_R$
**n-Gram Algorithm**

Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$$\sigma_r \leftarrow \sigma_{\text{init}}$$

For $i = 1$ to $I$ ($I$ is large, about 2000)

Pick $j, k \in \{0, \ldots, 25\}$ at Random.

Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped

If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for $\sigma$ are $\sigma_1, \ldots, \sigma_R$

Pick the $\sigma_r$ with min $\text{good}_r$ or have human look at all $\sigma_r(T)$
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$$\sigma_r \leftarrow \sigma_{\text{init}}$$

For $i = 1$ to $I$ ($I$ is large, about 2000)

Pick $j, k \in \{0, \ldots, 25\}$ at Random.

Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped

If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for $\sigma$ are $\sigma_1, \ldots, \sigma_R$

Pick the $\sigma_r$ with min $\text{good}_r$ or have human look at all $\sigma_r(T)$

The parameters $R$ and $I$ need to be picked carefully.