Encoding/Decoding

Russell Impagliazzo and Miles Jones
Thanks to Janine Tiefenbruck

http://cseweb.ucsd.edu/classes/sp16/cse21-bd/
May 9, 2016
A **permutation** of \( r \) elements from a set of \( n \) *distinct* objects is an **ordered** arrangement of them. There are

\[
P(n,r) = n(n-1)(n-2) \ldots (n-r+1)
\]

many of these.

A **combination** of \( r \) elements from a set of \( n \) distinct objects is an **unordered** selection of them. There are

\[
C(n,r) = \frac{n!}{r!(n-r)!}
\]

many of these.

**Review: Terminology**

Rosen p. 407-413
How many length $n$ binary strings contain $k$ ones?

Density is number of ones

**Objects:** all strings made up of $0_1, 0_2, 1_1, 1_2, 1_3, 1_4$  \( n! \)

**Categories:** strings that agree except subscripts

**Size of each category:**  \( k!(n-k)! \)

\[
\# \text{ categories} = \frac{\# \text{ objects}}{\text{size of each category}} = \frac{n!}{k!(n-k)!} = \binom{n}{k}
\]

Rosen p. 413
What's the smallest number of bits that we need to specify a binary string if we know it has $k$ ones and $n-k$ zeros?

A. $n$
B. $k$
C. $\log_2(C(n,k))$
D. ??

Encoding Fixed-density Binary Strings

Rosen p. 413
Data Compression

Store / transmit information in as little space as possible
Data Compression: Video

**Video:** stored as sequence of still frames.

**Idea:** instead of storing each frame fully, record change from previous frame.
**Data Compression: Run-Length Encoding**

**Image:** described as grid of pixels, each with RED, GREEN, BLUE values.

**Idea:** instead of storing RGB value of each pixel, store run-length of run of same color.

When is this a good coding mechanism? Will there be any loss in this compression?
Lossy Compression: Singular Value Decomposition

**Image:** described as grid of pixels, each with **RED**, **GREEN**, **BLUE** values.

**Idea:** use Linear Algebra to compress data to a fraction of its size, with minimal loss.
Complicated compression scheme

... save storage space
... may take a long time to encode / decode
Palindrome: string that reads the same forward and backward.

Which of these are binary palindromes?

A. The empty string.
B. 0101.
C. 0110.
D. 101.
E. All but one of the above.
Encoding: Binary Palindromes

**Palindrome:** string that reads the same forward and backward.

How many length $n$ binary palindromes are there?

A. $2^n$
B. $n$
C. $n/2$
D. $\log_2 n$
E. None of the above
Encoding: Binary Palindromes

**Palindrome:** string that reads the same forward and backward.

How many bits are (optimally) required to encode length n binary palindromes?

A. n  
B. n-1  
C. n/2  
D. $\log_2 n$  
E. None of the above.

Is there an algorithm that achieves this?
Goal: encode a length $n$ binary string that we know has $k$ ones (and $n-k$ zeros).

*How would you represent such a string with $n-1$ bits?*
**Encoding: Fixed Density Strings**

**Goal:** encode a length $n$ binary string that we know has $k$ ones (and $n-k$ zeros).

*How would you represent such a string with $n-1$ bits?*

*Can we do better?*
Encoding: Fixed Density Strings

Goal: encode a length $n$ binary string that we know has $k$ ones (and $n-k$ zeros).

*How would you represent such a string with $n-1$ bits?*

Can we do better?

Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.
Encoding: Fixed Density Strings

Idea: give positions of 1s in the string within some smaller window.
   - Fix window size.
   - If there is a 1 in the current "window" in the string, record its position and move the window over.
   - Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ?
Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ?

Output:

There's a 1! What's its position?
Encoding: Fixed Density Strings

Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 01

There's a 1! What's its position?
Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode $s = 01\underline{1}000000010$? Output: 01

There's a 1! What's its position?
Encoding: Fixed Density Strings

Idea: give positions of 1s in the string within some smaller window.
   - Fix window size.
   - If there is a 1 in the current "window" in the string, record its position and move the window over.
   - Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 01100000010? Output: 0100

There's a 1! What's its position?
Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size \( \frac{n}{k} = 4 \).

How do we encode \( s = 011000000010 \) ? Output: 0100

No 1s in this window.
Encoding: Fixed Density Strings

Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 01000

No 1s in this window.
Encoding: Fixed Density Strings

Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 01000

There's a 1! What's its position?
Idea: give positions of 1s in the string within some smaller window.
   - Fix window size.
   - If there is a 1 in the current "window" in the string, record its position and move the window over.
   - Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 0100011

There's a 1! What's its position?
Idea: give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 0100011

No 1s in this window.
**Idea:** give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

**Example** n=12, k=3, window size n/k = 4.

How do we encode $s = 011000000010$ ? Output: 01000110.

*No 1s in this window.*
**Encoding: Fixed Density Strings**

**Idea:** give positions of 1s in the string within some smaller window.
- Fix window size.
- If there is a 1 in the current "window" in the string, record its position and move the window over.
- Otherwise, record a 0 and move the window over.

**Example** $n=12$, $k=3$, window size $n/k = 4$.

How do we encode $s = 011000000010$? Output: 01000110.

Compressed to 8 bits!

*But can we recover the original string? Decoding ...*
Encoding: Fixed Density Strings

With $n=12$, $k=3$, window size $n/k = 4$. Output: 01000110

Can be parsed as the (intended) input: $s = 011000000010$ ?

But also:

01: one in position 1
0: no ones
00: one in position 0
11: one in position 3
0: no ones

$s' = 010000100010$

Problem: two different inputs with same output. Can't uniquely decode.
A valid compression algorithm must:

- Have outputs of shorter (or same) length as input.
- Be uniquely decodable.
Can we modify this algorithm to get unique decodability?

**Idea:** use *marker bit* to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.
Idea: use marker bit to indicate when to interpret output as a position.

- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output:
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output:
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ?

What output corresponds to these first few bits?
A. 0       C. 01
B. 1       D. 101
E. None of the above.
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ?

Output: 101

Interpret next bits as position of 1; this position is 01
Idea: use marker bit to indicate when to interpret output as a position.

- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 101
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode $s = 01\underbrace{1000000010}_?$ Output: 101100

Interpret next bits as position of 1; this position is 00
Encoding: Fixed Density Strings

Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode $s = 011000000010$ ? Output: 101100
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ?

Output: 1011000

No 1s in this window.
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 1011000
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example $n=12$, $k=3$, window size $n/k = 4$.

How do we encode $s = 0110000000\underline{1}0$? Output: 1011000111

Interpret next bits as position of 1; this position is 11
Idea: use marker bit to indicate when to interpret output as a position.
   - Fix window size.
   - If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
   - Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ? Output: 1011000111
Encoding: Fixed Density Strings

Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example $n=12$, $k=3$, window size $n/k = 4$.

How do we encode $s = 011000000010$? Output: 10110001110

No 1s in this window.
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example n=12, k=3, window size n/k = 4.

How do we encode s = 011000000010 ?

Output: 10110001110

Compare to previous output: 01000110

Output uses more bits than last time. Any redundancies?
Idea: use marker bit to indicate when to interpret output as a position.
- Fix window size.
- If there is a 1 in the current "window" in the string, record a 1 to interpret next bits as position, then record its position and move the window over.
- Otherwise, record a 0 and move the window over.

Example \( n=12, k=3, \) window size \( n/k = 4 \).

How do we encode \( s = 011000000010 \)?

Output: \( 10110001110 \)

Compare to previous output: \( 01000110 \)

* After see the last 1, don't need to add 0s to indicate empty windows. *
procedure WindowEncode (input: \( b_1b_2...b_n \), with exactly \( k \) ones and \( n-k \) zeros)

1. \( w := \text{floor} \left( \frac{n}{k} \right) \)
2. \( \text{count} := 0 \)
3. \( \text{location} := 1 \)
4. While \( \text{count} < k \):
5. If there is a 1 in the window starting at current location
6. Output 1 as a marker, then output position of first 1 in window.
7. Increment count.
8. Update location to immediately after first 1 in this window.
9. Else
10. Output 0.
11. Update location to next index after current window.

Uniquely decodable?
Decoding: Fixed Density Strings

procedure WindowDecode (input: x_1 x_2 ... x_m, target is exactly k ones and n-k zeros)
1. w := floor ( n/k )
2. b := floor ( log_2(w))
3. s := empty string
4. i := 0
5. While i < m
6.   If x_i = 0
7.      s += 0...0 (j times)
8.      i += 1
9.   Else
10.      p := decimal value of the bits x_{i+1}...x_{i+b}
11.      s += 0...0 (p times)
12.      s += 1
13.      i := i+b+1
14. If length(s) < n
15.      s += 0...0 ( n-length(s) times )
16. Output s.
**Correctness?**

\[ E(s) = \text{result of encoding string } s \text{ of length } n \text{ with } k \text{ 1s, using } \text{WindowEncode}. \]

\[ D(t) = \text{result of decoding string } t \text{ to create a string of length } n \text{ with } k \text{ 1s, using WindowDecode}. \]

**Well-defined functions?**

**Inverses?**

**Goal:** For each \( s \), \( D(E(s)) = s \).

**Strong Induction!**
Output size?

Assume n/k is a power of two. Consider s a binary string of length n with k 1s.

How long is E(s)?

A. n-1
B. log₂(n/k)
C. Depends on where 1s are located in s
Output size?

Assume \( n/k \) is a power of two. Consider \( s \) a binary string of length \( n \) with \( k \) 1s.

For which strings is \( E(s) \) shortest?

A. More 1s toward the beginning.
B. More 1s toward the end.
C. 1s spread evenly throughout.
Output size?

Assume $n/k$ is a power of two. Consider $s$ a binary string of length $n$ with $k$ 1s.

**Best case**: 1s toward the beginning of the string. $E(s)$ has
- One bit for each 1 in $s$ to indicate that next bits denote positions in window.
- $\log_2(n/k)$ bits for each 1 in $s$ to specify position of that 1 in a window.
- $k$ such ones.
- No bits representing 0s because all 0s are "caught" in windows with 1s or after the last 1.

**Total** $|E(s)| = k \log_2(n/k) + k$
Output size?

Assume \( n/k \) is a power of two. Consider \( s \) a binary string of length \( n \) with \( k \) 1s.

**Worst case**: 1s toward the end of the string. \( E(s) \) has
- Some bits representing 0s since there are no 1s in first several windows.
- One bit for each 1 in \( s \) to indicate that next bits denote positions in window.
- \( \log_2(n/k) \) bits for each 1 in \( s \) to specify position of that 1 in a window.
- \( k \) such ones.

What's an upper bound on the number of these bits?

A. \( n \)  
B. \( n-k \)  
C. \( k \)  
D. 1  
E. None of the above.
Output size?

Assume \( \frac{n}{k} \) is a power of two. Consider \( s \) a binary string of length \( n \) with \( k \) 1s.

Worst case: 1s toward the end of the string. \( E(s) \) has
- At most \( k \) bits representing 0s since there are no 1s in first several windows.
- One bit for each 1 in \( s \) to indicate that next bits denote positions in window.
- \( \log_2 \left( \frac{n}{k} \right) \) bits for each 1 in \( s \) to specify position of that 1 in a window.
- \( k \) such ones.

Total \( |E(s)| \leq k \log_2 \left( \frac{n}{k} \right) + 2k \)
Output size?

Assume \( n/k \) is a power of two. Consider \( s \) a binary string of length \( n \) with \( k \) 1s.

\[
k \log_2 \left( \frac{n}{k} \right) + k \leq |E(s)| \leq k \log_2 \left( \frac{n}{k} \right) + 2k
\]

Using this inequality, there are at most ____ length \( n \) strings with \( k \) 1s.

A. \( 2^n \)  
B. \( n \)  
C. \( (n/k)^2 \)  
D. \( (n/k)^k \)  
E. None of the above.
Output size?

Assume $n/k$ is a power of two. Consider $s$ a binary string of length $n$ with $k$ 1s. Given $|E(s)| \leq k \log_2(n/k) + 2k$, we need at most $k \log_2(n/k) + 2k$ bits to represent all length $n$ binary strings with $k$ 1s. Hence, there are at most $2^{k \log_2(n/k) + 2k}$ many such strings.
Output size?

Assume \( n/k \) is a power of two. Consider \( s \) a binary string of length \( n \) with \( k \) 1s. Given \(|E(s)| \leq k \log_2(n/k) + 2k\), we need at most \( k \log_2(n/k) + 2k \) bits to represent all length \( n \) binary strings with \( k \) 1s. Hence, there are at most

\[
2^{(k \log(n/k) + 2k)} = 2^{(k \log(n/k))} \cdot 2^{(2k)} \\
= \left(2^{(\log(n/k))}\right)^k \cdot 2^{(2k)} \\
= (n/k)^k \cdot 4^k = (4n/k)^k
\]
Encoding/Decoding: Fixed Density Strings

Output size?

Assume \( n/k \) is a power of two. Consider \( s \) a binary string of length \( n \) with \( k \) 1s. Given \(|E(s)| \leq k \log_2(n/k) + 2k\), we need at most \( k \log_2(n/k) + 2k \) bits to represent all length \( n \) binary strings with \( k \) 1s. Hence, there are at most \( 2^{k \log(n/k) + 2k} \) many such strings.

\[
2^{(k \log(n/k) + 2k)} = 2^{(k \log(n/k))} \cdot 2^{(2k)} \\
= \left(2^{(\log(n/k))}\right)^k \cdot 2^{(2k)} \\
= \left(n/k\right)^k \cdot 4^k = (4n/k)^k
\]

\[ C(n,k) = \text{# Length } n \text{ binary strings with } k \text{ 1s} \leq (4n/k)^k \]
Bounds for Binomial Coefficients

Using \texttt{windowEncode}(): \[ \binom{n}{k} \leq (4n/k)^k \]

Lower bound?

\textbf{Idea:} find a way to count a \textit{subset} of the fixed density binary strings.

\begin{center}
\begin{tabular}{cccccccc}
\hline
\text{ } & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } \\
\hline
\text{ } & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } & \text{ } \\
\hline
\end{tabular}
\end{center}

Some fixed density binary strings have one 1 in each of k chunks of size n/k.

\textbf{How many such strings are there?}
A. \( n^n \) \quad B. \( k! \) \quad C. \( (n/k)^k \) \quad D. \( C(n,k)^k \) \quad E. None of the above.
Bounds for Binomial Coefficients

Using \( \text{windowEncode}() \): 
\[
\binom{n}{k} \leq \left( \frac{4n}{k} \right)^k
\]

Using evenly spread strings:
\[
\left( \frac{n}{k} \right)^k \leq \binom{n}{k}
\]

Counting helps us analyze our compression algorithm.

Compression algorithms help us count.