COS 140: Foundations of Computer Science

Hamming Codes

Fall 2018

Homework, etc.

- New book chapter (25) online
- Slides online at website
- Exercises at end of the chapter, due 12/5.

The problem

- Want a way to determine if data is correct:
  - from memory
  - across the network
  - from an I/O device
- Even better, want a way to fix the data if it is incorrect

Why Not Parity Bits?

- Parity bit just says “an error occurred” – not *where* it occurred.
- For messages:
  - ○ If only know an error occurred, will need to resend message.
  - ○ It can take a great deal of time to resend, so better if you can fix the error.
- For memory: can’t do equivalent of resending message!
- Know from RAID that can correct the error if know the location of the bit.

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?
- For every bit, have a parity bit associated with it. Then have a parity bit for the parity bits, to identify whether bit or parity bit has an error.

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?
- For every bit, have a parity bit associated with it. Then have a parity bit for the parity bits, to identify whether bit or parity bit has an error.
- An example byte: 0110 1111

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?
- For every bit, have a parity bit associated with it. Then have a parity bit for the parity bits, to identify whether bit or parity bit has an error.
- An example byte: 0110 1111
- Coded data (with odd parity): 0110 1111 1001 0000 1

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?
- For every bit, have a parity bit associated with it. Then have a parity bit for the parity bits, to identify whether bit or parity bit has an error.
- An example byte: 0110 1111
- Coded data (with odd parity): 0110 1111 1001 0000 1
- 17 bits: 8 bits data, 8 bits parity bit for corresponding data bit, parity for parity bits

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?
- For every bit, have a parity bit associated with it. Then have a parity bit for the parity bits, to identify whether bit or parity bit has an error.
- An example byte: 0110 1111
- Coded data (with odd parity): 0110 1111 1001 0000 1
- 17 bits: 8 bits data, 8 bits parity bit for corresponding data bit, parity for parity bits
- Problem: More than doubles size of memory!

A Possible Scheme for Error Correction

- Can we extend the idea of parity bits to give more information about which bit is wrong?
- For every bit, have a parity bit associated with it. Then have a parity bit for the parity bits, to identify whether bit or parity bit has an error.
- An example byte: 0110 1111
- Coded data (with odd parity): 0110 1111 1001 0000 1
- 17 bits: 8 bits data, 8 bits parity bit for corresponding data bit, parity for parity bits
- Problem: More than doubles size of memory!
- (Question: what is this scheme equivalent to when using even parity?)

Hamming Codes

- Example of an *error-correcting code*: way to correct errors in memory, in transmission.
- Used in computer networks, internal memory and elsewhere in the computer.
- Goal: Identify which bit has the error, but use fewer bits to do it than the parity scheme

Hamming Codes: Basic idea

- Associate parity bits with different *subsets* of the data
- Any single bit is uniquely linked to some group of parity bits.
- Each of those bits will indicate an error when that bit is incorrect

Associating Data with Parity Bits

1 1 1 0

Data to be checked/corrected

Overview

General idea

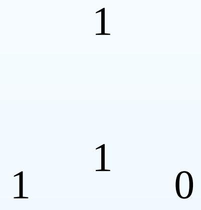
● Associating Data with Parity Bits

● Correcting a Bit

Hamming Codes

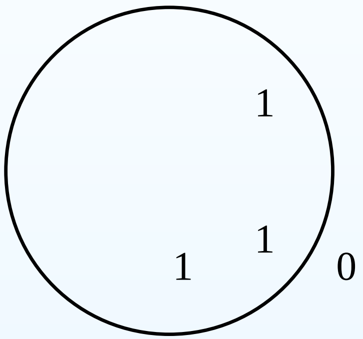
Error Correction

Associating Data with Parity Bits



Data to be checked/corrected

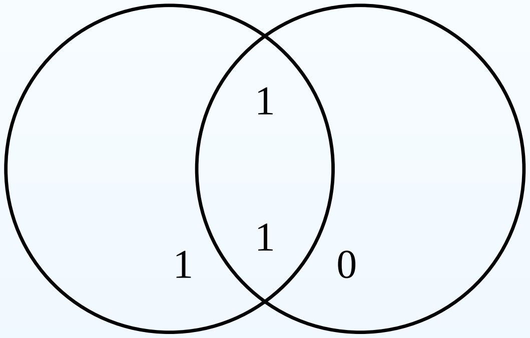
Associating Data with Parity Bits



Divide data into subsets

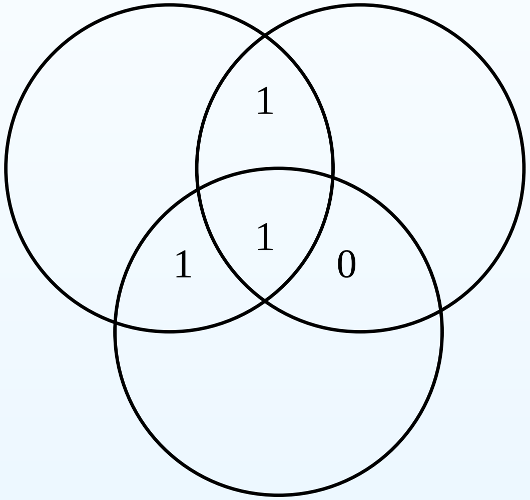
Computer Science Foundations

Associating Data with Parity Bits



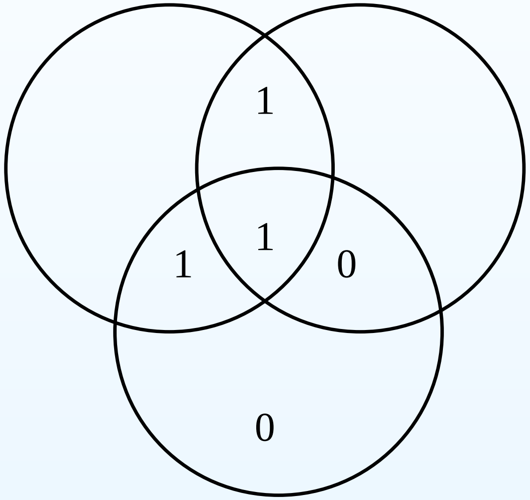
Divide data into subsets

Associating Data with Parity Bits



Divide data into subsets

Associating Data with Parity Bits



Compute even parity for bits in each circle

Overview

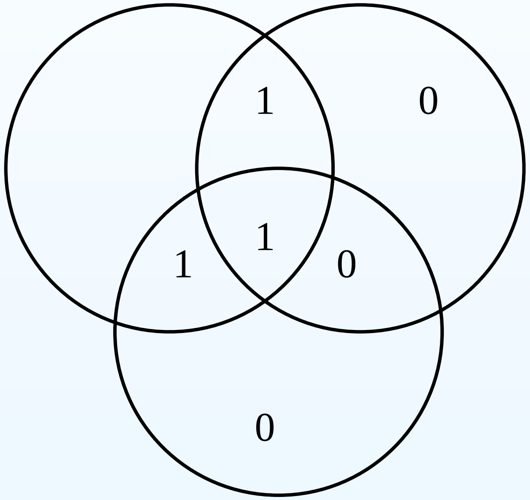
General idea

- Associating Data with Parity Bits
- Correcting a Bit

Hamming Codes

Error Correction

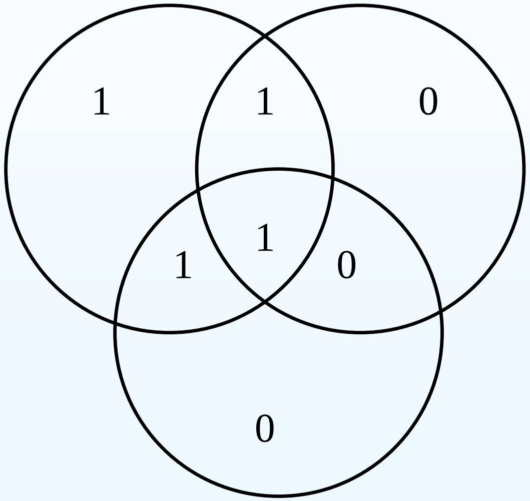
Associating Data with Parity Bits



Compute even parity for bits in each circle

Computer Science Foundations

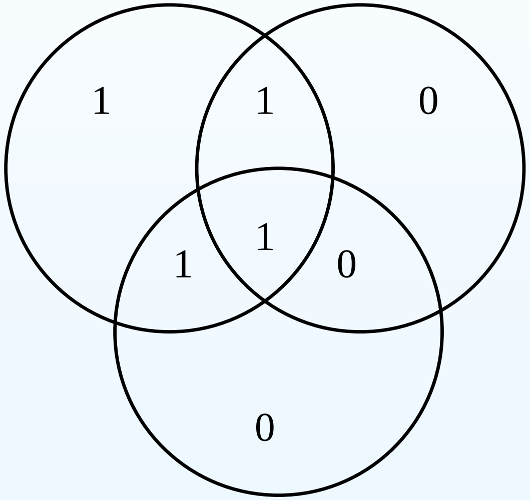
Associating Data with Parity Bits



Compute even parity for bits in each circle

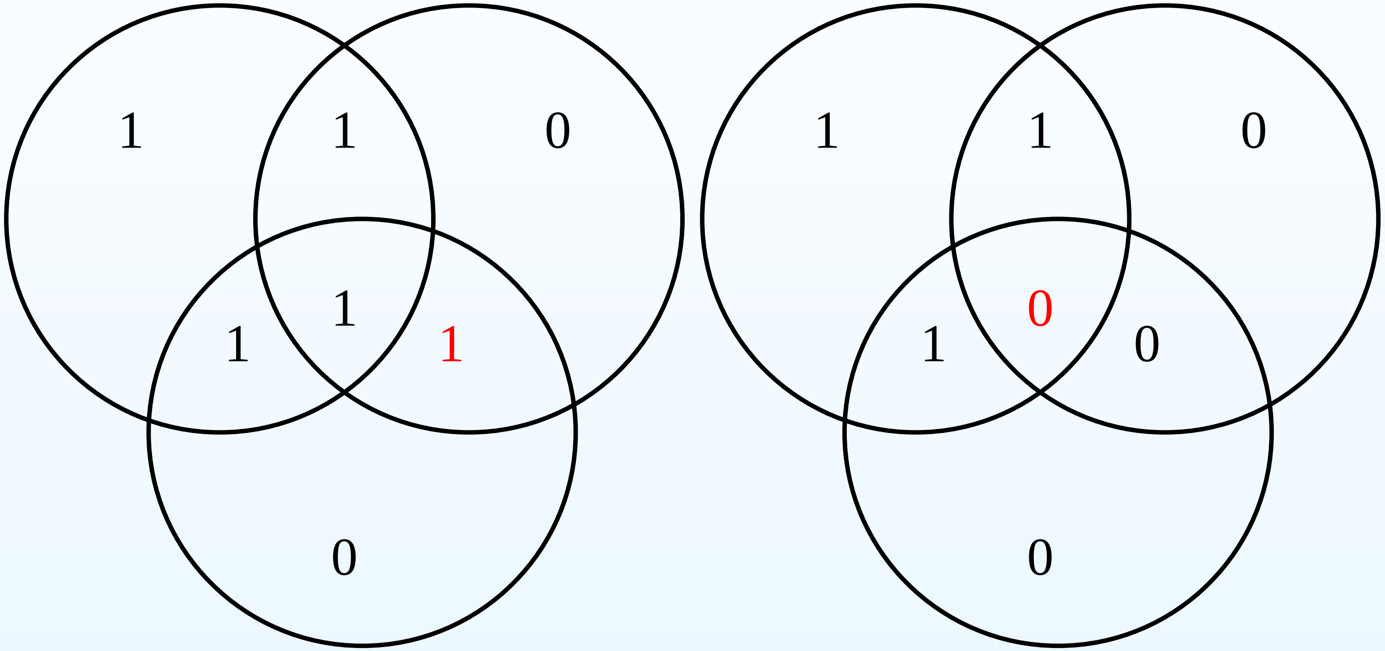
Computer Science Foundations

Associating Data with Parity Bits



Now each bit is checked by unique combination of parity bits

Correcting a Bit



- When get data, check all the parity bits.
- Left figure: two parity bits wrong.
- Right: all three wrong.
- Data to be corrected is at the intersection of all sets where parity bit is wrong.

Hamming Codes in Transmitted Data

- Have data in a stream, not in Venn diagrams.
- Divide data into segments that can be checked with Hamming codes.
- Need on the order of $\log_2 n$ checkbits for n bits.
- Add one check bit each time the number of data bits doubles.
- So: segments should be fairly large (but small enough that don't have unused bits).

Overview

General idea

Hamming Codes
- Transmitted data
- Data subsets
- Parity
- Code for 8-bit data

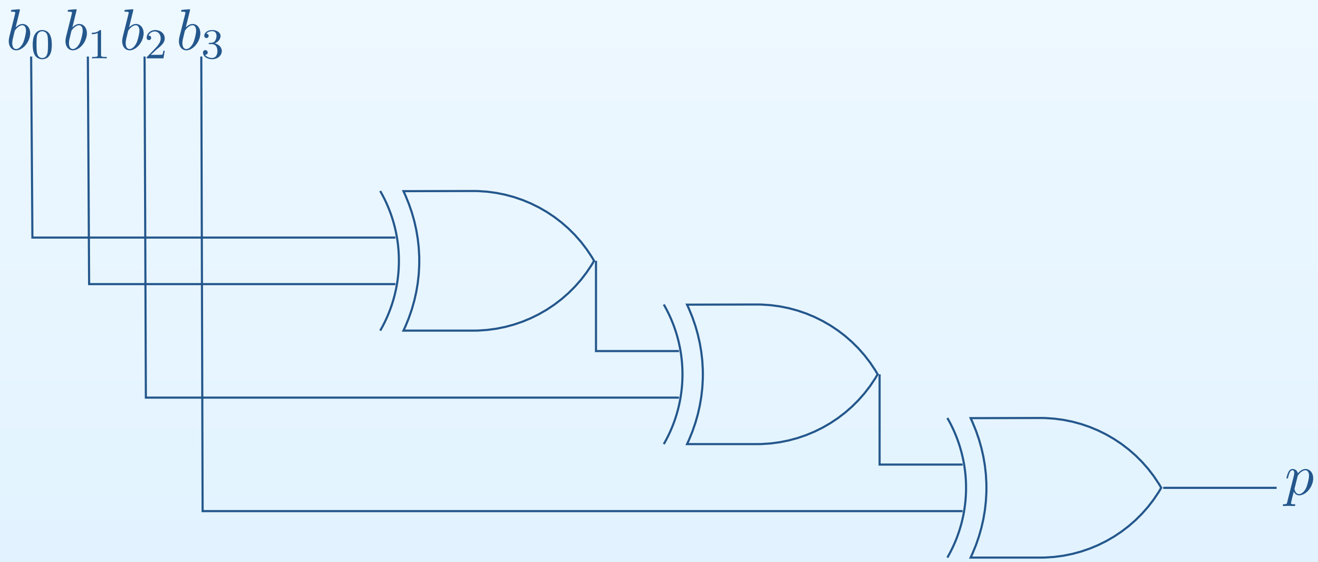
Error Correction

Dividing the Data into Subsets

- Recognize that each bit has a unique position in the stream (starting at position 1).
- The position can be written as a binary number.
- In the binary representation of the position, some digits are 1 and some are 0.
- Put check bits in the positions that correspond to powers of 2 (1,2,4,8, etc.)
- Put data bits in the other positions.
- Check bits check parity for all positions that have a 1 for the corresponding digit (e.g., the bit at position 12 (1100) is checked by parity bits at positions 8 and 4).
- Because all binary numbers are combinations of different powers of 2, each position is checked by a different set of bits.
- Need enough check bits to “address” data bits plus check bits.

Aside: Computing parity

- Computing parity is trivial!
- Suppose parity bit p checks bits b_0–b_3
- Using single-bit addition, $p = b_0 + b_1 + b_2 + b_3$
- Example: Data = 1011: 1 + 0 + 1 + 1 = 11, but no carry, so $\Rightarrow$ 1 – correct parity bit
- In hardware:
  - ○ Could use half-adder – but don't need the carry
  - ○ So just $p = b_0 \oplus b_1 \oplus b_2 \oplus b_3$



Overview

General idea

Hamming Codes
- Transmitted data
- Data subsets
- Parity
- Code for 8-bit data

Error Correction

Hamming Code for 8 Bits of Data

- Check (parity) bit locations: 2^0, 2^1, 2^2, 2^3, ⇒ word length = 12 bits
- Example: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
		1		1	0	1		1	1	1	0

- Data bit relations to check bits:

Data bit	Location in Word	Check Bits
1	3 (0011)	C1, C2
2	5 (0101)	C1, C4
3	6 (0110)	C2, C4
4	7 (0111)	C1, C2, C4
5	9 (1001)	C1, C8
6	10 (1010)	C2, C8
7	11 (1011)	C1, C2, C8
8	12 (1100)	C4, C8

Hamming Code for 8 Bits of Data

- Check (parity) bit locations: 2^0, 2^1, 2^2, 2^3, $\Rightarrow$ word length = 12 bits
- Example: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1		1		1	0	1		1	1	1	0
		^		^		^		^		^	

- Data bit relations to check bits:

Data bit	Location in Word	Check Bits
1	3 (0011)	C1, C2
2	5 (0101)	C1, C4
3	6 (0110)	C2, C4
4	7 (0111)	C1, C2, C4
5	9 (1001)	C1, C8
6	10 (1010)	C2, C8
7	11 (1011)	C1, C2, C8
8	12 (1100)	C4, C8

Hamming Code for 8 Bits of Data

- Check (parity) bit locations: 2^0, 2^1, 2^2, 2^3, ⇒ word length = 12 bits
- Example: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1		1	0	1		1	1	1	0
		^			^	^			^	^	

- Data bit relations to check bits:

Data bit	Location in Word	Check Bits
1	3 (0011)	C1, C2
2	5 (0101)	C1, C4
3	6 (0110)	C2, C4
4	7 (0111)	C1, C2, C4
5	9 (1001)	C1, C8
6	10 (1010)	C2, C8
7	11 (1011)	C1, C2, C8
8	12 (1100)	C4, C8

Hamming Code for 8 Bits of Data

- Check (parity) bit locations: 2^0, 2^1, 2^2, 2^3, $\Rightarrow$ word length = 12 bits
- Example: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1		1	1	1	0
				^	^	^					^

- Data bit relations to check bits:

Data bit	Location in Word	Check Bits
1	3 (0011)	C1, C2
2	5 (0101)	C1, C4
3	6 (0110)	C2, C4
4	7 (0111)	C1, C2, C4
5	9 (1001)	C1, C8
6	10 (1010)	C2, C8
7	11 (1011)	C1, C2, C8
8	12 (1100)	C4, C8

Hamming Code for 8 Bits of Data

- Check (parity) bit locations: 2^0, 2^1, 2^2, 2^3, $\Rightarrow$ word length = 12 bits
- Example: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	1	0
								^	^	^	^

- Data bit relations to check bits:

Data bit	Location in Word	Check Bits
1	3 (0011)	C1, C2
2	5 (0101)	C1, C4
3	6 (0110)	C2, C4
4	7 (0111)	C1, C2, C4
5	9 (1001)	C1, C8
6	10 (1010)	C2, C8
7	11 (1011)	C1, C2, C8
8	12 (1100)	C4, C8

Example of Hamming Code Correction

Overview

General idea

Hamming Codes

Error Correction

● Example

Actual data: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	1	0

Example of Hamming Code Correction

Actual data: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	1	0

Error in bit 7: 1101 1100

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	0	0

Computer Science Foundations

Overview

General idea

Hamming Codes

Error Correction

● Example

Example of Hamming Code Correction

Actual data: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	1	0

Error in bit 7: 1101 1100

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	0	0

Computed parities:

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
0	1	1	0	1	0	1	0	1	1	0	0

Example of Hamming Code Correction

Actual data: 1101 1110

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	1	0

Error in bit 7: 1101 1100

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
1	0	1	0	1	0	1	1	1	1	0	0

Computed parities:

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
C1	C2	D	C4	D	D	D	C8	D	D	D	D
0	1	1	0	1	0	1	0	1	1	0	0

Parity errors will appear when computing
C1, C2, C8 ⇒ bit 1011 in error.