

# Hibajavítás és hibajelzés

## Informatikai rendszerek alapjai

Horváth Árpád <horvath.arpad@amk.uni-obuda.hu>

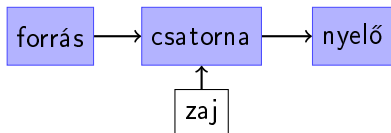
Óbudai Egyetem  
Alba Regia Műszaki Kar (AMK)  
Székesfehérvár

2016. november 24.

# Vázlat

- 1 Hibákról
- 2 Hibajavító és -jelző kódok

# Információátvitel diagrammja



Példák:

Beszélgetés

forrás: száj, csatorna: levegő, nyelő: fül

Földfelszíni rádióadás

forrás: a rádióadó vagy az átjátszó antennája, csatorna: az „éter”,

nyelő: a rádió antennája

Internet

forrás: router1, csatorna: üvegszál, nyelő: router2

# Hibák előfordulása

- Üvegszálakon ritkán
- Sodrott érpáron (pl. UTP kábel) gyakrabban
- Vezeték nélküli hálózatokban gyakran

# Hibák előfordulása

- Üvegszálakon ritkán
- Sodrott érpáron (pl. UTP kábel) gyakrabban
- Vezeték nélküli hálózatokban gyakran
- Egyes közegekben (pl. rádió) általában csoportosan
  - Előny: kevesebb adatcsomagot érint
  - Hátrány: nehezebb jelezni ill. javítani a hibát

# Vázlat

- 1 Hibákról
- 2 Hibajavító és -jelző kódok

# Hibajavítás és -jelzés

Két lehetőség:

- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy visszaállítható legyen az eredeti.

# Hibajavítás és -jelzés

Két lehetőség:

- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy visszaállítható legyen az eredeti.
- hibajavító kód – error-correcting code



# Hibajavítás és -jelzés

Két lehetőség:

- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy visszaállítható legyen az eredeti.
- hibajavító kód – error-correcting code
- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy kiderüljön, hogy hibás adatot kaptunk-e.

# Hibajavítás és -jelzés

Két lehetőség:

- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy visszaállítható legyen az eredeti.
- hibajavító kód – error-correcting code
- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy kiderüljön, hogy hibás adatot kaptunk-e.
- Ha hibás, újraküldjük.

# Hibajavítás és -jelzés

Két lehetőség:

- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy visszaállítható legyen az eredeti.
- hibajavító kód – error-correcting code
- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy kiderüljön, hogy hibás adatot kaptunk-e.
- Ha hibás, újraküldjük.
- hibajelző kód – error-detecting code

# Hibajavítás és -jelzés

Két lehetőség:

- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy visszaállítható legyen az eredeti.
- hibajavító kód – error-correcting code
- Annyi redundanciát adunk minden elküldött adatblokkhoz, hogy kiderüljön, hogy hibás adatot kaptunk-e.
- Ha hibás, újraküldjük.
- hibajelző kód – error-detecting code
- Az első akkor ha nem küldhető újra az adat (CD, DVD), illetve adatszórásnál (digitális rádió, TV).

# Alapfogalmak

Kódszavak (codeword):  $m$  üzenetbitből (message bit) és  $r$  redundáns ellenőrző bitből állnak.

A teljes hossz  $n = m + r$ .

## Definíció

Az olyan helyek számát, ahol két kódszóban különböző bitek állnak, a két kódszó **Hamming-távolság**ának nevezzük. (Hamming, 1950)

# Alapfogalmak

## Példa

a kódszó 1 1 0 1 1 0 1 0 1

b kódszó 0 1 1 1 0 1 1 0 1

$d(ab) = \square$  a Hamming-távolsága a két kódszónak.

Ha két kódszó Hamming-távolsága 4, akkor  $\square$  hiba kell ahhoz, hogy az egyik a másikba menjen.

## Alapfogalmak

## Példa

a kódszó 1 1 0 1 1 0 1 0 1

b kódszó 0 1 1 1 0 1 1 0 1

 $d(ab) = \boxed{4}$  a Hamming-távolsága a két kódszónak.

Ha két kódszó Hamming-távolsága 4, akkor  hiba kell ahhoz, hogy az egyik a másikba menjen.

## Alapfogalmak

## Példa

a kódszó 1 1 0 1 1 0 1 0 1

b kódszó 0 1 1 1 0 1 1 0 1

 $d(ab) = \boxed{4}$  a Hamming-távolsága a két kódszónak.

Ha két kódszó Hamming-távolsága 4, akkor  $\boxed{4}$  hiba kell ahhoz, hogy az egyik a másikba menjen.



# Üzenetek száma

$n = m + r$  (kódhossz = üzenet + redundáns)

lehetséges üzenetünk van. Az  $n$  bitből  bitsorozatot alkothatunk, de ezek közül csak  lesz érvényes.

# Üzenetek száma

$n = m + r$  (kódhossz = üzenet + redundáns)

$2^m$  lehetséges üzenetünk van. Az  $n$  bitből  $\square$  bitsorozatot alkothatunk, de ezek közül csak  $\square$  lesz érvényes.

# Üzenetek száma

$n = m + r$  (kódhossz = üzenet + redundáns)

$2^m$  lehetséges üzenetünk van. Az  $n$  bitből  $2^n$  bitsorozatot alkothatunk, de ezek közül csak  $\square$  lesz érvényes.

# Üzenetek száma

$n = m + r$  (kódhossz = üzenet + redundáns)

$2^m$  lehetséges üzenetünk van. Az  $n$  bitből  $2^n$  bitsorozatot alkothatunk, de ezek közül csak  $2^m$  lesz érvényes.

# Kód és kódszó

## Definíció

Az érvényes  $n$  bites bitsorozatokat kódszavaknak nevezzük.  
A kódszavak összességét kódnak nevezzük.

# Alapfogalmak

## Definíció

Egy **kód Hamming-távolságán** a minimális távolságot értjük, amelyet két kódszava között találunk.

## Példa

Az alábbi kód Hamming-távolsága .

a 01010101

b 00001111

c 00110010

$d(ab) =$

$d(bc) =$

$d(ac) =$

# Alapfogalmak

## Definíció

Egy **kód Hamming-távolságán** a minimális távolságot értjük, amelyet két kódszava között találunk.

## Példa

Az alábbi kód Hamming-távolsága .

a 01010101

b 00001111

c 00110010

$d(ab) =$

$d(bc) =$

$d(ac) =$

# Alapfogalmak

## Definíció

Egy **kód Hamming-távolságán** a minimális távolságot értjük, amelyet két kódszava között találunk.

## Példa

Az alábbi kód Hamming-távolsága .

a 01010101

b 00001111

c 00110010

$d(ab) =$

$d(bc) =$

$d(ac) =$



# Alapfogalmak

## Definíció

Egy **kód Hamming-távolságán** a minimális távolságot értjük, amelyet két kódszava között találunk.

## Példa

Az alábbi kód Hamming-távolsága .

a 01010101

b 00001111

c 00110010

$$d(ab) = \boxed{4}$$

$$d(bc) = \boxed{5}$$

$$d(ac) = \boxed{5}$$

# Alapfogalmak

## Definíció

Egy **kód Hamming-távolságán** a minimális távolságot értjük, amelyet két kódszava között találunk.

## Példa

Az alábbi kód Hamming-távolsága 4.

a 01010101

b 00001111

c 00110010

$$d(ab) = \text{4}$$

$$d(bc) = \text{5}$$

$$d(ac) = \text{5}$$

# Hamming-kódozás (1950), Richard Hamming (1915-1998)



Egy kód Hamming-távolsága 4.

- Legfeljebb  hiba lehet, hogy biztosan ne kapjunk másik kódszót az eredeti helyett.
- Legfeljebb  hiba lehet, hogy biztosan ki tudjuk találni, melyik volt az eredeti kódszó.

Egy kód Hamming-távolsága 4.

- Legfeljebb  hiba lehet, hogy biztosan ne kapjunk másik kódszót az eredeti helyett.
- Legfeljebb  hiba lehet, hogy biztosan ki tudjuk találni, melyik volt az eredeti kódszó.



Egy kód Hamming-távolsága 4.

- Legfeljebb  hiba lehet, hogy biztosan ne kapjunk másik kódszót az eredeti helyett.
- Legfeljebb  hiba lehet, hogy biztosan ki tudjuk találni, melyik volt az eredeti kódszó.



Egy kód Hamming-távolsága 4.

- Legfeljebb 3 hiba lehet, hogy biztosan ne kapjunk másik kódszót az eredeti helyett.
- Legfeljebb 1 hiba lehet, hogy biztosan ki tudjuk találni, melyik volt az eredeti kódszó.



# Feladat

Maximum  $e$  hibára számítunk.

Mennyi legyen a kód  $d$  Hamming-távolsága, hogy ki tudjuk mutatni a hibát?



# Feladat

Maximum  $e$  hibára számítunk.

Mennyi legyen a kód  $d$  Hamming-távolsága, hogy ki tudjuk mutatni a hibát?

$$d \geq e + 1$$

# Feladat

Maximum  $e$  hibára számítunk.

Mennyi legyen a kód  $d$  Hamming-távolsága, hogy ki tudjuk mutatni a hibát?

$$d \geq e + 1$$

Mennyi legyen a kód  $d$  Hamming-távolsága, hogy javítani tudjuk a hibát?

# Feladat

Maximum  $e$  hibára számítunk.

Mennyi legyen a kód  $d$  Hamming-távolsága, hogy ki tudjuk mutatni a hibát?

$$d \geq e + 1$$

Mennyi legyen a kód  $d$  Hamming-távolsága, hogy javítani tudjuk a hibát?

$$d \geq 2e + 1$$

## Gondolkozzunk! Biztosan sikerül megoldani.

Hogyan lehetne olyan kódot előállítani, hogy egy hibát ki tudjunk mutatni?

**Azaz**

jönnek **adott hosszúságú üzenetek (bitsorozatok)**, és mindegyik helyett egy másik bitsorozatot küldünk, úgy, hogy meg tudjuk mondani, hogy **hibásan ment-e át** az üzenet, és ha nem, **meg tudjuk határozni az üzenetet**.

Több bit kell? Elég ugyanannyi? Vagy kevesebb is?

# Na és ezt sikerül?

Hogyan lehetne olyan kódot előállítani, hogy egy hibát ki tudjunk javítani?

Egyelőre nem érdekel, hogy milyen hosszúak a kódszavak.

# Paritásbit

## Definíció

Ha a kódszóhoz egy olyan bitet fűzünk, hogy a kódszóban levő 1-esek száma páros (illetve páratlan) legyen. Ezt a bitet nevezzük páros (páratlan) **paritásbit**nek.

Továbbiakban páros paritásbitet használunk.

Pl. 11011  $\Rightarrow$

Pl. 10000  $\Rightarrow$



# Paritásbit

## Definíció

Ha a kódszóhoz egy olyan bitet fűzünk, hogy a kódszóban levő 1-esek száma páros (illetve páratlan) legyen. Ezt a bitet nevezzük páros (páratlan) **paritásbit**nek.

Továbbiakban páros paritásbitet használunk.

Pl. 11011  $\Rightarrow$  11011**0**

Pl. 10000  $\Rightarrow$



# Paritásbit

## Definíció

Ha a kódszóhoz egy olyan bitet fűzünk, hogy a kódszóban levő 1-esek száma páros (illetve páratlan) legyen. Ezt a bitet nevezzük páros (páratlan) **paritásbit**nek.

Továbbiakban páros paritásbitet használunk.

Pl. 11011  $\Rightarrow$  11011**0**

Pl. 10000  $\Rightarrow$  10000**1**

A paritásbit hozzáfűzésével kapott kód Hamming-távolsága:



# Paritásbit

## Definíció

Ha a kódszóhoz egy olyan bitet fűzünk, hogy a kódszóban levő 1-esek száma páros (illetve páratlan) legyen. Ezt a bitet nevezzük páros (páratlan) **paritásbit**nek.

Továbbiakban páros paritásbitet használunk.

Pl. 11011  $\Rightarrow$  11011**0**

Pl. 10000  $\Rightarrow$  10000**1**

A paritásbit hozzáfűzésével kapott kód Hamming-távolsága: 2

Alkalmas-e hibajavításra,

jelzésre?

# Paritásbit

## Definíció

Ha a kódszóhoz egy olyan bitet fűzünk, hogy a kódszóban levő 1-esek száma páros (illetve páratlan) legyen. Ezt a bitet nevezzük páros (páratlan) **paritásbit**nek.

Továbbiakban páros paritásbitet használunk.

Pl. 11011  $\Rightarrow$  11011**0**

Pl. 10000  $\Rightarrow$  10000**1**

A paritásbit hozzáfűzésével kapott kód Hamming-távolsága: 2

Alkalmas-e hibajavításra,

jelzésre? 1 bithiba jelzésére alkalmas (SED–single error detection)

## Egy bithiba javítása (SEC–single error correction)

Hány bites üzenetünk lehet, ha  $r$  többlet bitet adunk az üzenethez?  
kódhossz ( $n$ ) = üzenethossz ( $m$ ) + redundáns bitek száma ( $r$ )

## Egy bithiba javítása (SEC–single error correction)

Hány bites üzenetünk lehet, ha  $r$  többlet bitet adunk az üzenethez?  
kódhossz ( $n$ ) = üzenethossz ( $m$ ) + redundáns bitek száma ( $r$ )

- Egy kódszó és 1 bithibás szomszédai együtt  $n + 1$  bitsorozatot adnak.

## Egy bithiba javítása (SEC–single error correction)

Hány bites üzenetünk lehet, ha  $r$  többlet bitet adunk az üzenethez?  
kódhossz ( $n$ ) = üzenethossz ( $m$ ) + redundáns bitek száma ( $r$ )

- Egy kódszó és 1 bithibás szomszédai együtt  $n + 1$  bitsorozatot adnak.
- A  $2^m$  üzenethez legalább  $2^m(n + 1)$  különböző bitsorozat kell.

# Egy bithiba javítása (SEC–single error correction)

Hány bites üzenetünk lehet, ha  $r$  többlet bitet adunk az üzenethez?  
kódhossz ( $n$ ) = üzenethossz ( $m$ ) + redundáns bitek száma ( $r$ )

- Egy kódszó és 1 bithibás szomszédai együtt  $n + 1$  bitsorozatot adnak.
- A  $2^m$  üzenethez legalább  $2^m(n + 1)$  különböző bitsorozat kell.
- Ha  $n$  bites kódszavakat küldünk, akkor a  $2^n \geq 2^m(n + 1)$  egyenlőtlenségnek kell teljesülnie, hogy egy bithiba javítása lehetséges legyen.

# Egy bithiba javítása (SEC–single error correction)

Hány bites üzenetünk lehet, ha  $r$  többlet bitet adunk az üzenethez?  
kódhossz ( $n$ ) = üzenethossz ( $m$ ) + redundáns bitek száma ( $r$ )

- Egy kódszó és 1 bithibás szomszédai együtt  $n + 1$  bitsorozatot adnak.
- A  $2^m$  üzenethez legalább  $2^m(n + 1)$  különböző bitsorozat kell.
- Ha  $n$  bites kódszavakat küldünk, akkor a  $2^n \geq 2^m(n + 1)$  egyenlőtlenségnek kell teljesülnie, hogy egy bithiba javítása lehetséges legyen.
- Mivel  $n = r + m$ ,  $2^r \geq m + r + 1$  egyenlőtlenségnek kell teljesülnie.

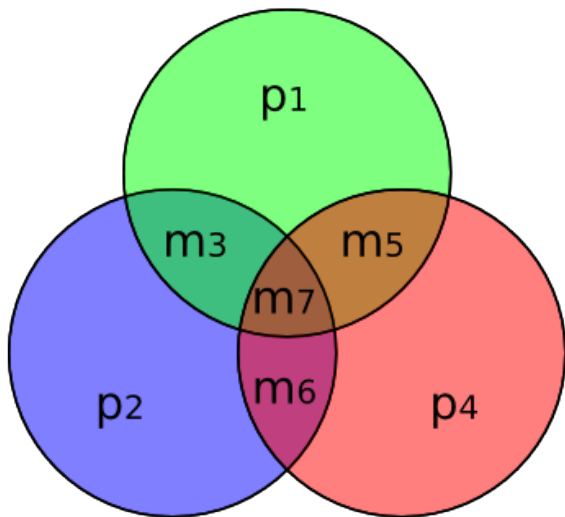
# Egy bithiba javítása (SEC–single error correction)

Hány bites üzenetünk lehet, ha  $r$  többlet bitet adunk az üzenethez?  
kódhossz ( $n$ ) = üzenethossz ( $m$ ) + redundáns bitek száma ( $r$ )

- Egy kódszó és 1 bithibás szomszédai együtt  $n + 1$  bitsorozatot adnak.
- A  $2^m$  üzenethez legalább  $2^m(n + 1)$  különböző bitsorozat kell.
- Ha  $n$  bites kódszavakat küldünk, akkor a  $2^n \geq 2^m(n + 1)$  egyenlőtlenségnek kell teljesülnie, hogy egy bithiba javítása lehetséges legyen.
- Mivel  $n = r + m$ ,  $2^r \geq m + r + 1$  egyenlőtlenségnek kell teljesülnie.
- Az egyenlőtlenséget teljesítő minimális egész  $r$  bittel tényleg létrehozható egy bithibát javító kódolás: a Hamming-kód.



# A Hamming(7, 4)-kód



# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.

# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.

# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.
- Az egyes paritásbitek nem az összes bitet „ellenőrzik”.

# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.
- Az egyes paritásbitek nem az összes bitet „ellenőrzik”.
- A paritásbit ellenőrzi a teljes kulcsszó  $i$ . bitjét, ha az  $i$  kettes számrendszerbeli alakjában szerepel a  $2^i$  helyiérték.

# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.
- Az egyes paritásbitek nem az összes bitet „ellenőrzik”.
- A paritásbit ellenőrzi a teljes kulcsszó  $i$ . bitjét, ha az  $i$  kettes számrendszerbeli alakjában szerepel a  $2^i$  helyiérték.
- Például a 6. bitet ellenőrzi a  paritásbit, a többi nem.

# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.
- Az egyes paritásbitek nem az összes bitet „ellenőrzik”.
- A paritásbit ellenőrzi a teljes kulcsszó  $i$ . bitjét, ha az  $i$  kettes számrendszerbeli alakjában szerepel a  $2^i$  helyiérték.
- Például a 6. bitet ellenőrzi a 4-es és 2-es paritásbit, a többi nem.

# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.
- Az egyes paritásbitek nem az összes bitet „ellenőrzik”.
- A paritásbit ellenőrzi a teljes kulcsszó  $i$ . bitjét, ha az  $i$  kettes számrendszerbeli alakjában szerepel a  $2^i$  helyiérték.
- Például a 6. bitet ellenőrzi a 4-es és 2-es paritásbit, a többi nem.  $6=4+2=110_2$



# A Hamming-kód

- A biteket 1-től sorszámozzuk balról jobbra.
- Minden kettő-hatvány ( $2^i$ ) helyen paritásbit van.
- Az egyes paritásbitek nem az összes bitet „ellenőrzik”.
- A paritásbit ellenőrzi a teljes kulcsszó  $i$ . bitjét, ha az  $i$  kettes számrendszerbeli alakjában szerepel a  $2^i$  helyiérték.
- Például a 6. bitet ellenőrzi a 4-es és 2-es paritásbit, a többi nem.  $6=4+2=110_2$

bit	1.	2.	3.	4.	5.	6.	7.
	$P_1$	$P_2$	$2 + 1$	$P_4$	$4 + 1$	$4 + 2$	$4 + 2 + 1$
	1	10	11	100	101	110	111
$P_1$	+		+		+		+
$P_2$		+	+			+	+
$P_4$				+	+	+	+

## Példa

Az 100111 üzenet kódolása, és visszaállítása a kód 7. bitjének meghibásodása esetén.

bit	1. $P_1$	2. $P_2$	3. 11	4. $P_4$	5. 101	6. 110	7. 111	8. $P_8$	9. 1001	10. 1010
$P_1$	?		1		0		1		1	
$P_2$		?	1			0	1			1
$P_4$				?	0	0	1			
$P_8$								?	1	1
			1		0	0	1		1	1

A keresett Hamming-kódszó 1111001011, a  $P_i$  jelű oszlopok a paritásbiteket, a többi az üzenet (message) bitjek helye.

## Példa

Az 100111 üzenet kódolása, és visszaállítása a kód 7. bitjének meghibásodása esetén.

bit	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
	$P_1$	$P_2$	11	$P_4$	101	110	111	$P_8$	1001	1010
$P_1$	?		1		0		1		1	
$P_2$		?	1			0	1			1
$P_4$				?	0	0	1			
$P_8$								?	1	1
	1		1		0	0	1		1	1

A keresett Hamming-kódszó 1111001011, a  $P_i$  jelű oszlopok a paritásbiteket, a többi az üzenet (message) bitjek helye.

## Példa

Az 100111 üzenet kódolása, és visszaállítása a kód 7. bitjének meghibásodása esetén.

bit	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
	$P_1$	$P_2$	11	$P_4$	101	110	111	$P_8$	1001	1010
$P_1$	?		1		0		1		1	
$P_2$		?	1			0	1			1
$P_4$				?	0	0	1			
$P_8$								?	1	1
	1	1	1		0	0	1		1	1

A keresett Hamming-kódszó 1111001011, a  $P_i$  jelű oszlopok a paritásbiteket, a többi az üzenet (message) bitjek helye.

## Példa

Az 100111 üzenet kódolása, és visszaállítása a kód 7. bitjének meghibásodása esetén.

bit	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
	$P_1$	$P_2$	11	$P_4$	101	110	111	$P_8$	1001	1010
$P_1$	?		1		0		1		1	
$P_2$		?	1			0	1			1
$P_4$				?	0	0	1			
$P_8$								?	1	1
	1	1	1	1	0	0	1		1	1

A keresett Hamming-kódszó 1111001011, a  $P_i$  jelű oszlopok a paritásbiteket, a többi az üzenet (message) bitjek helye.

## Példa

Az 100111 üzenet kódolása, és visszaállítása a kód 7. bitjének meghibásodása esetén.

bit	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
	$P_1$	$P_2$	11	$P_4$	101	110	111	$P_8$	1001	1010
$P_1$	?		1		0		1		1	
$P_2$		?	1			0	1			1
$P_4$				?	0	0	1			
$P_8$								?	1	1
	1	1	1	1	0	0	1	0	1	1

A keresett Hamming-kódszó 1111001011, a  $P_i$  jelű oszlopok a paritásbiteket, a többi az üzenet (message) bitjek helye.

## Példa folytatása

A keresett Hamming-kódszó 1111001011, a kódot a csatornán átküldve a 7. bitjének meghibásodása esetén a nyelőnél kapott kód 1111000011.

	1. $P_1$	2. $P_2$	3. 11	4. $P_4$	5. 101	6. 110	7. 111	8. $P_8$	9. 1001	10. 1010	Helyes? (egyesek száma)
$P_1$	1		1		0		0		1		hibás (3)
$P_2$		1	1			0	0			1	hibás (3)
$P_4$				1	0	0	0				hibás (1)
$P_8$								0	1	1	helyes (2)
	1	1	1	1	0	0	0	0	1	1	Hibás $1+2+4=7$ .

## Több bithiba javítása

Bonyolultabb algoritmussal olyan kód is előállítható, amely több bithibát is képes javítani.

A DVD-n található filmeket, és a digitális televíziós előadásokat (pl. DVB) MPEG kódolással kódolják, amelynek részét képezi a hibajavító kódolás. Az úgynevezett Reed–Solomon kódolást használják.

Az videó-műsorszórás európai szabványában (DVB) egy 188 bájtos adatcsomaghoz 16 bájtnyi redundáns információt fűznek, amely csomagonként 8 bithiba javítására alkalmas.



## Csoportos hibák elleni védekezés

- Soronként felírjuk az átküldendő Hamming-kódolt kódszavakat ( $k$  darabot).

## Csoportos hibák elleni védekezés

- Soronként felírjuk az átküldendő Hamming-kódolt kódszavakat ( $k$  darabot).
- Oszloponként küldjük át a jeleket.

## Csoportos hibák elleni védekezés

- Soronként felírjuk az átküldendő Hamming-kódolt kódszavakat ( $k$  darabot).
- Oszloponként küldjük át a jeleket.
- A nyelőnél visszaállítjuk a sorokat, és úgy ellenőrzzük.

## Csoportos hibák elleni védekezés

- Soronként felírjuk az átküldendő Hamming-kódolt kódszavakat ( $k$  darabot).
- Oszloponként küldjük át a jeleket.
- A nyelőnél visszaállítjuk a sorokat, és úgy ellenőrizzük.
- Ha egy  $k$  hosszúságú csoportos hiba lép fel, akkor az minden sorban egy bitet ront el, amit a Hamming-kód javítani képes.

## Csoportos hibák elleni védekezés

- Soronként felírjuk az átküldendő Hamming-kódolt kódszavakat ( $k$  darabot).
- Oszloponként küldjük át a jeleket.
- A nyelőnél visszaállítjuk a sorokat, és úgy ellenőrizzük.
- Ha egy  $k$  hosszúságú csoportos hiba lép fel, akkor az minden sorban egy bitet ront el, amit a Hamming-kód javítani képes.
- $km$  méretű adatblokkokhoz  $kr$  ellenőrző bit.

# Csoportos hibák elleni védekezés

jel	7 bites ASCII-kód	Hamming-kódszó
O	1001111	0011001001011
E	1000101	1110000101101
-	0101101	0100101101101
A	1000001	1110000010101
R	1010010	1011010100110
E	1000101	1110000101101
K	1001011	0011001110011

Függőlegesen küldjük el 01011100111010...1111011.

## Egy bithiba javítása (SEC–single error correction)

$m$  hosszúságú üzenetnél hány darab ( $r$ ) többletbit, hány % redundancia kell?

$n$	$m$	$r$	redundancia (%)
7	4	3	75
15	11	4	36
31	26	5	19
63	57	6	10
127	120	7	6
255	247	8	3
511	502	9	2

## Egy bithiba javítása (SEC–single error correction)

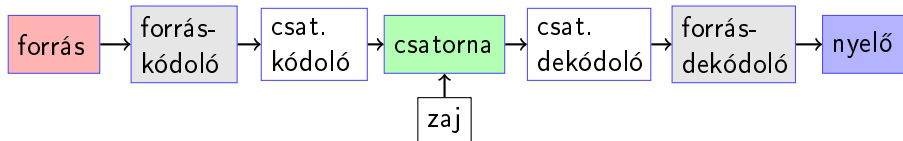
$m$  hosszúságú üzenetnél hány darab ( $r$ ) többletbit, hány % redundancia kell?

$n$	$m$	$r$	redundancia (%)
7	4	3	75
15	11	4	36
31	26	5	19
63	57	6	10
127	120	7	6
255	247	8	3
511	502	9	2

Miért nem lehet a gyakorlatban akármennyire növelni  $m$ -et?



# Információátvitel diagrammja



# A generátormátrix és a paritásellenőrző mátrix

Minden művelet úgy értelmezendő, hogy utána kettes maradékot veszünk, tehát 0 és 1 lesz mindenhol az eredményben.

A generátormátrix következőképp állítható elő az üzenethez tartozó kódszó:

Ha  $\mathbf{x}$  az üzenet bitjeit tartalmazó sorvektor, akkor  $\mathbf{y} = \mathbf{xG}$  a kapott Hammig-kódszó.

A  $\mathbf{H}$  paritásellenőrző mátrixsszal tudom ellenőrizni, hogy helyes kódszót kaptam-e.

Ha  $\mathbf{y}$  helyes kódszó, akkor  $\mathbf{Hy}^T$  elemei nullák. Ha nem nullák, akkor annyiadik bit romlott el, ahányadik oszlopa megjelenik a  $\mathbf{H}$ -nak a szorzatban.

# A generátormátrix és a paritásellenőrző mátrix kapcsolata

Az általánosan használt esetben a paritásbiteket a kód végére szokták rakni. Ebben az esetben

a generátormátrix mindig:

$$\mathbf{G} = [\mathbf{A} \mid \mathbf{E}_m]$$

alakú, a paritásellenőrző mátrix, pedig

$$\mathbf{H} = [\mathbf{E}_r \mid \mathbf{A}^T]$$

alakú, egymásból előállíthatóak.  $\mathbf{E}_k$  a  $k \times k$ -s egységmátrix.

Hamming-kód esetén:

r: paritásbitek (redundáns bitek) száma; m: üzenetbitek száma

# A generátormátrix és a paritásellenőrző mátrix a MATLAB-ban

A két mátrixot a következőképpen lehet megkapni:

```
[H, G] = hamngen(3);
```

ahol  $r = 3$  az ellenőrzőbitek száma.

Egy példa kódolásra:

```
>> x = [1 0 0 1];
```

```
>> y = mod(x*G, 2)
```

```
y =
```

```
    0    1    1    1    0    0    1
```

A  $\text{mod}(A, 2)$  függvény a kettes maradékát adja az eredménynek (mátrixnál minden elemnek).

# Kódszó ellenőrzése

```
>> y1 = [0 1 0 1 0 0 1];
```

```
>> mod(H*y1', 2)
```

```
ans =
```

```
0
```

```
0
```

```
1
```

```
>> H
```

```
H =
```

```
1      0      0      1      0      1      1
```

```
0      1      0      1      1      1      0
```

```
0      0      1      0      1      1      1
```

Mivel az eredménnyel a 3. oszlop egyezik, ezért a 3. bit sérült.

Összehasonlítható az előző oldallal.

# Feladat

Az előző oldal paritásellenőrző mátrixa alapján határozzuk meg az A mátrixot és a generátormátrixot!

## Kódszó ellenőrzése

 $G =$ 

1	1	0	1	0	0	0
0	1	1	0	1	0	0
1	1	1	0	0	1	0
1	0	1	0	0	0	1

## CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.



## CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.

## CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.
- $10011 \Rightarrow 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = x^4 + x + 1$

# CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.
- $10011 \Rightarrow 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = x^4 + x + 1$
- A CRC kód (cyclic redundancy code) esetén az üzenet bitjeit egy polinom együtthatóinak tekintjük,

# CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.
- $10011 \Rightarrow 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = x^4 + x + 1$
- A CRC kód (cyclic redundancy code) esetén az üzenet bitjeit egy polinom együtthatóinak tekintjük,
- előre meghatározott polinommal, az úgynevezett generátor-polinommal, végezzük el az osztást,

# CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.
- $10011 \Rightarrow 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = x^4 + x + 1$
- A CRC kód (cyclic redundancy code) esetén az üzenet bitjeit egy polinom együtthatóinak tekintjük,
- előre meghatározott polinommal, az úgynevezett generátor-polinommal, végezzük el az osztást,
- a maradék együtthatóit az üzenethez fűzzük.

# CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.
- $10011 \Rightarrow 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = x^4 + x + 1$
- A CRC kód (cyclic redundancy code) esetén az üzenet bitjeit egy polinom együtthatóinak tekintjük,
- előre meghatározott polinommal, az úgynevezett generátor-polinommal, végezzük el az osztást,
- a maradék együtthatóit az üzenethez fűzzük.
- A nyelőnél (ismerve a generátor-polinomot) ugyanígy ellenőrizni tudjuk a redundáns részt,

# CRC-kód

- Hasonlat: Ha egy számot elosztunk  $r + 1$ -gyel, akkor a maradék legfeljebb  $r$  lesz.
- Ha egy polinomot elosztunk egy  $(r + 1)$ -ed fokszámú polinommal, akkor a maradék legfeljebb  $r$ -ed fokszámú lesz.
- $10011 \Rightarrow 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 = x^4 + x + 1$
- A CRC kód (cyclic redundancy code) esetén az üzenet bitjeit egy polinom együtthatóinak tekintjük,
- előre meghatározott polinommal, az úgynevezett generátor-polinommal, végezzük el az osztást,
- a maradék együtthatóit az üzenethez fűzzük.
- A nyelőnél (ismerve a generátor-polinomot) ugyanígy ellenőrizni tudjuk a redundáns részt,
- és ha az hibás, akkor újraküldjük az üzenetet.

# Polinomosztás elvégzése

Legyen a generátorpolinom

$$G(x) = x^3 + x + 1 = 1x^3 + 0x^2 + 1x^1 + 1x^0$$

azaz  $r = 3$ , az együtthatók 1011.

```

11010011100    <--- eredeti üzenet
11010011100000 <--- r darab 0-val kiegészítem
1011           <--- generátor polinommal osztok
01100011100000 <--- eredmény
  1011         <--- osztás ...
00111011100000
(...)
00000000000010
  
```

Az utolsó  $r$  bitet fűzöm az eredeti üzenethez:

11010011100010 bitsorozatot küldöm el.



## Ellenőrzés a nyelőnél

```
11010011100010 <--- kódolt üzenet
1011             <--- generátor polinommal osztok
01100011100010 <--- eredmény
  1011          <--- osztás ...
00111011100010
  1011
00010111100010
  1011
00000001100010
  (...)
00000000010110
  1011
00000000000000
Minden bit 0 lett, nem sérült az üzenet.
```

# CRC-kód tulajdonságai

- Hardveresen hatékonyan elvégezhető.

## CRC-kód tulajdonságai

- Hardveresen hatékonyan elvégezhető.
- Rézvezetéken, optikai szálakon és merevlemezeken, a bithiba-arány kicsi ( $< 10^{-6}$ ), ott alkalmazzuk.

## CRC-kód tulajdonságai

- Hardveresen hatékonyan elvégezhető.
- Rézvezetéken, optikai szálakon és merevlemezeken, a bithiba-arány kicsi ( $< 10^{-6}$ ), ott alkalmazzuk.
- A csoportos hibákat, amikor  $r$  egymás utáni bit sérül, hatékonyan kimutatja.