Erreur détectée !

Comment les ordinateurs réparent toutes nos erreurs (ou presque !)

Christophe Grandmont Pierre Vandenhove

Département d'Informatique, Faculté des Sciences, UMONS

Journées Math-Sciences – Mars 2025

Mise en contexte : situation de la vie quotidienne

- Vous êtes à la **friterie**.
- Après avoir fini votre hamburger, vous souhaitez commander un dessert.
- On est en 2025 : les commandes se font via un **QR code**...
- ***Oh non*** ! Le QR code est partiellement recouvert de **ketchup** !
- Déçu, vous envoyez une photo du QR code à un·e ami·e pour lui raconter votre mésaventure...
- ... mais votre téléphone parvient à décoder le QR code !

Comment le QR code peut-il être lisible, peu importe où une quantité (raisonnable) de ketchup se trouve ?




Pour répondre à cette question, nous allons faire un détour qui va nous amener à nous intéresser à la façon dont les ordinateurs représentent l'**information**...

Tour de magie

- 25 carrés, gris ou rouges.
- Christophe sort de la pièce.
- Un volontaire choisit une disposition de 5×5 carrés.
- Un autre volontaire choisit **un** des carrés à retourner.
- Christophe revient et nous **É-P-A-T-E** !

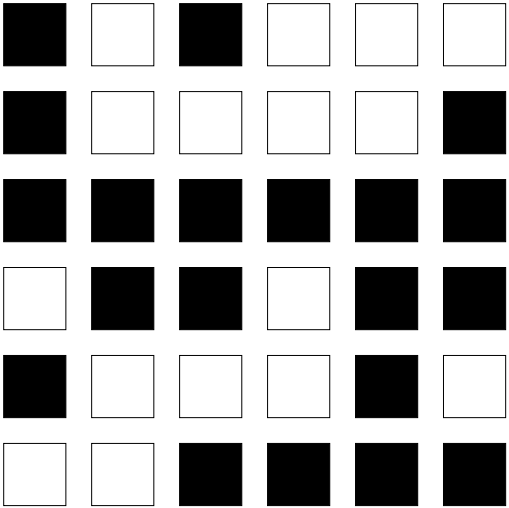
Comment avons-nous fait ?

Explication : parité

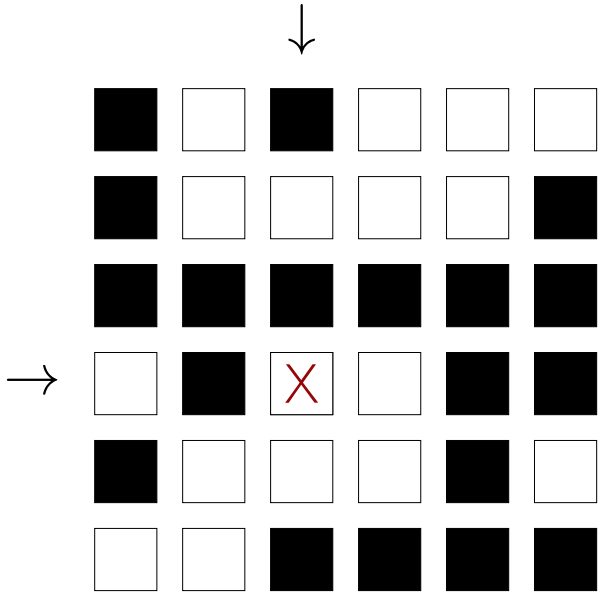
Le tour repose sur la **parité**.

- **Astuce** : en ajoutant une colonne et une ligne, Pierre a fait en sorte que **chaque ligne** et **chaque colonne** contienne un nombre **pair** de carrés gris.
- C'est toujours possible : pour chaque ligne/colonne, s'il y a un nombre impair de carrés gris, on ajoute un carré gris ; sinon, on ajoute un carré rouge.
- Si on modifie **un** carré, la ligne et la colonne dans lesquelles il se trouve contiendront un nombre **impair** de carrés gris !
- $\rightsquigarrow$ Christophe n'a qu'à identifier le carré à l'intersection de la ligne impaire et de la colonne impaire.

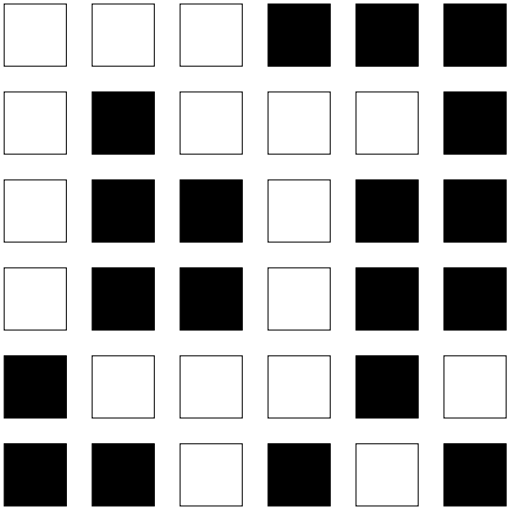
Exemple



Exemple



Exercice : quel carré a été remplacé ?



Erreurs dans des communications

Nos informations contiennent régulièrement des erreurs :

- Données mobiles et réseaux Wi-Fi sont sujets à des **interférences**.
- Informations stockées **de façon imparfaite** dans les disques durs des ordinateurs.
- Codes-barres ou QR codes **abîmés**.
- CDs **griffés**.
- **Erreurs humaines** lors du recopiage de numéros.
- ...

Pourquoi ne rencontrons-nous pas plus d'erreurs au quotidien ?

Idée centrale : **redondance**

- On souhaite communiquer une information, sujette à erreurs.
- **Principe central** : on **ajoute de l'information** au message (**redondance**) dans le seul but de permettre au destinataire de **détecter** ou **corriger** des erreurs.
- Dans le tour de magie $\rightsquigarrow$ on ajoute une rangée et une colonne de carrés.

Idée centrale : **redondance**

Exemple de la vie quotidienne

- Vous souhaitez communiquer votre date d'anniversaire (*28 mars 2025*) à un·e ami·e.
- Envoyer "*28 mars 2025*" contient **exactement** l'information suffisante. Toutefois, à la moindre erreur, l'information est **incorrecte** et l'erreur est **indétectable** !
- Redondance raisonnable à ajouter : "***vendredi** 28 mars 2025*".
- Si vous vous trompez dans la date (par exemple, *vendredi 27 mars 2025*), votre ami·e pourra **détecter** l'erreur et vous demander confirmation !
- En cas d'erreur, il n'y a qu'une chance sur 7 de tomber sur une date **erronée**, **mais valide** (par exemple, *vendredi 21 mars 2025*).

Même idée à la poste : une erreur dans "*7000 Mons*" est plus facile à corriger qu'une erreur dans "*7000*".

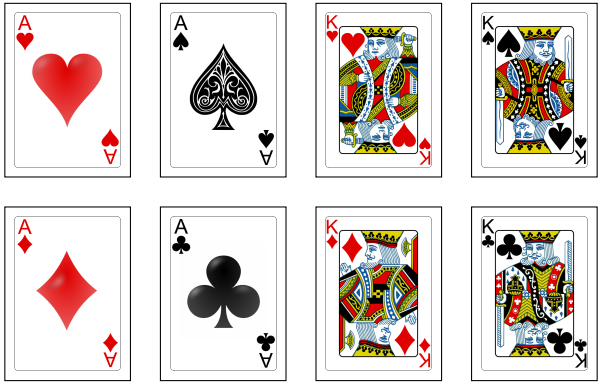
Contexte : théorie de l'information

- Ces questions sont liées au concept d'*information* : quelle **quantité d'information** faut-il envoyer pour transmettre un message ?
- L'**informatique**, étymologiquement, est la science de l'**information** !
  Abréviation **IT** en anglais : *information technology*.

Jouons à un jeu pour déterminer
la **quantité d'information** à utiliser !

Qui est-ce ?

Par groupes de **deux** : retrouver la carte à laquelle pense votre camarade parmi ces **8 cartes**, en posant le moins de questions (réponses *oui/non*) possible. **Jouez !**

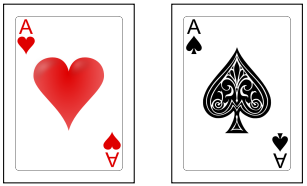


Quel est le nombre **minimal** de questions à poser pour garantir une victoire ?

Combien de questions ?

Partons d'un problème plus simple ! Combien de **questions devez-vous poser** s'il y a...

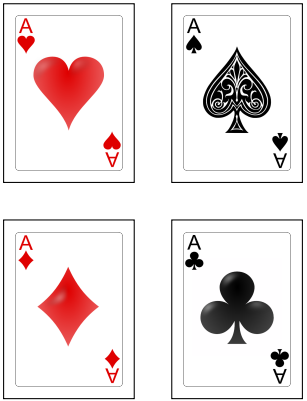
Deux cartes ?



Combien de questions ?

Partons d'un problème plus simple ! Combien de **questions devez-vous poser** s'il y a...

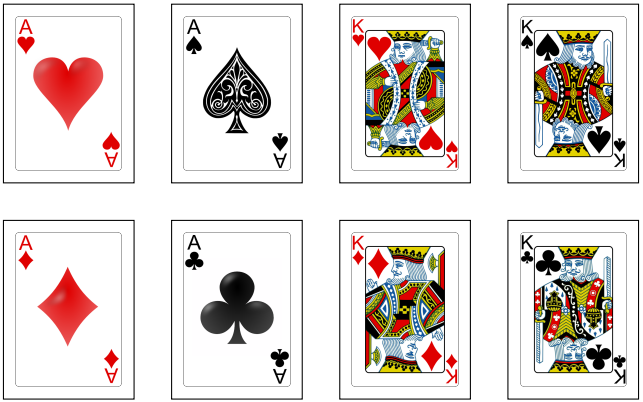
Quatre cartes ?



Combien de questions ?

Partons d'un problème plus simple ! Combien de **questions devez-vous poser** s'il y a...

Huit cartes ?



Représentation décimale

Dans la vie courante, nous utilisons dix chiffres différents (de 0 à 9).
Cela nous permet d'écrire les nombres en termes de **puissances de 10** :

$$\begin{aligned} 1952 &= 1000 + 900 + 50 + 2 \\ &= 1 \cdot 1000 + 9 \cdot 100 + 5 \cdot 10 + 2 \cdot 1 \\ &= 1 \cdot 10^3 + 9 \cdot 10^2 + 5 \cdot 10^1 + 2 \cdot 10^0. \end{aligned}$$

Représentation **décimale** (car 10 symboles).

Représentation binaire

On peut faire la même chose avec seulement **deux** chiffres différents (0 et 1).
On peut écrire les nombres en termes de **puissances de 2** :

$$\begin{aligned} 5 &= 4 + 1 \\ &= 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &\rightsquigarrow 101, \\ 139 &= 128 + 8 + 2 + 1 \\ &= 1 \cdot 2^7 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &\rightsquigarrow 10001011. \end{aligned}$$

Représentation **binaire** (car 2 symboles).

Information

Toute information peut se transmettre en utilisant des séquences plus ou moins longues de **deux symboles** ! Par exemple,

- **0** et **1** (ordinateurs),
- des carrés **noirs** et des carrés **blancs** (QR codes),
- réponses **oui**/**non** à des questions (devinettes),
- des signaux **longs** et **courts** (Morse),
- des points en **relief** ou **plats** (Braille),
- ...

Pour des **lettres**, il suffit de décider d'un **encodage** ($a \mapsto 00000$, $b \mapsto 00001\ldots$).

Pourquoi **deux** symboles...

... et pas plus ? **Raison pratique** : on peut envoyer un courant électrique ou non, un signal lumineux ou non, etc.
$\rightsquigarrow$ Il est physiquement **plus facile** de ne distinguer que deux états.

Retour à *Combien de questions ?*

Si on veut associer des nombres binaires à **8 cartes**, il faut **3 chiffres**.

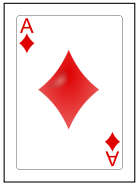
Exemple sur 3 chiffres :

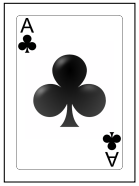

000


001


100


101


010


011


110

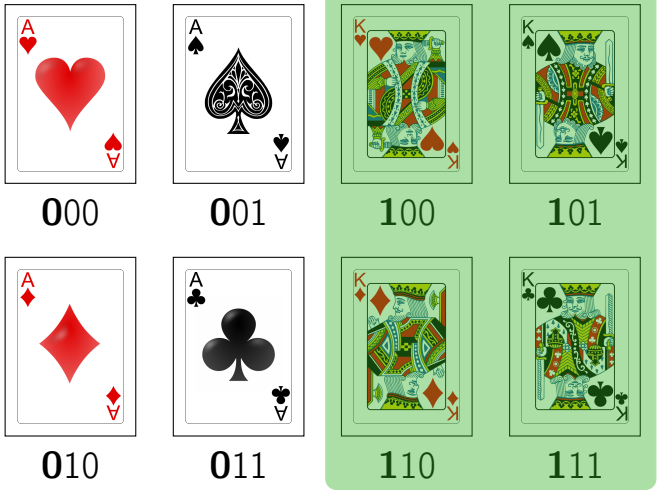

111

Retour à *Combien de questions ?*

Si on veut associer des nombres binaires à **8 cartes**, il faut **3 chiffres**.

Exemple sur 3 chiffres :

- le **premier** correspond à as/roi,

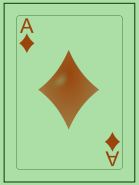


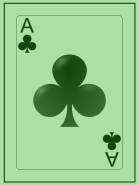
Retour à *Combien de questions ?*

Si on veut associer des nombres binaires à **8 cartes**, il faut **3 chiffres**.

Exemple sur 3 chiffres :

- le **premier** correspond à as/roi,
- le **deuxième** correspond à la rangée,





Retour à *Combien de questions ?*

Si on veut associer des nombres binaires à **8 cartes**, il faut **3 chiffres**.

Exemple sur 3 chiffres :

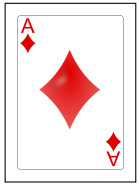
- le **premier** correspond à as/roi,
- le **deuxième** correspond à la rangée,
- le **troisième** correspond à la couleur.

00**0**

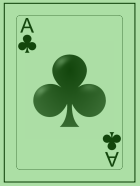
00**1**

10**0**

10**1**



01**0**



01**1**

11**0**

11**1**

Retour à *Combien de questions ?*

Si on veut associer des nombres binaires à **8 cartes**, il faut **3 chiffres**.

Exemple sur 3 chiffres :

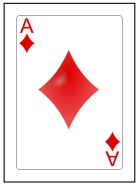
- le **premier** correspond à as/roi,
- le **deuxième** correspond à la rangée,
- le **troisième** correspond à la couleur.
- Chaque combinaison possible de trois réponses encode **exactement** une carte !

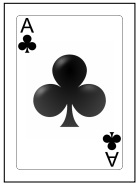

000


001


100


101


010


011


110


111

Combien de possibilités pour combien de chiffres ?

- Pour distinguer 2 possibilités, il faut 1 chiffre binaire (0 et 1).
- Pour distinguer $4 = 2^2$ possibilités, il faut 2 chiffres binaires (00, 01, 10 et 11).
- Pour distinguer $8 = 2^3$ possibilités, il faut 3 chiffres binaires (transparent précédent).
- ...
- Pour distinguer 2^k possibilités, il faut... k chiffres binaires !
- Dit autrement, pour distinguer n possibilités, il faut $\log_2(n)$ chiffres binaires.
- Quand on augmente le nombre de chiffres binaires utilisés, le nombre de possibilités qu'on peut représenter augmente **exponentiellement**.
- Dans un QR code moyen, il y a environ 1000 carrés.
  Cela permet de distinguer 2^{1000} **messages différents**...
- C'est ***beaucoup*** plus que le nombre d'**atomes dans l'univers observable** (2^{270}) !

Qui est-ce ?

- Stratégie **optimale** pour jouer au *Qui est-ce ?*
- Diviser le nombre de possibilités **par deux** à chaque question !
- Ici, on suppose qu'*optimal* signifie qui "minimise le nombre de questions **dans le pire cas**".
- Si $n = 2^k$ personnages, il faut $\log_2(n) = k$ questions pour trouver le personnage à coup sûr.
- Pour le *Qui est-ce ?* classique (24 personnages), 5 questions suffisent !

Autres jeux

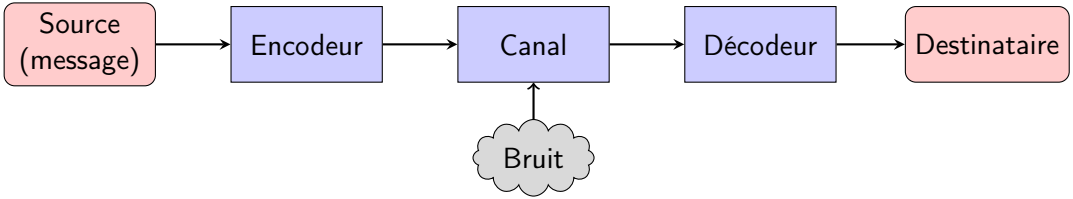
Des idées similaires peuvent être utilisées pour trouver une stratégie optimale pour **Motus**, **Wordle**, **Mastermind**... mais la stratégie est moins facile à calculer et décrire !

G	R	A	I	L
T	R	A	C	K
C	R	A	M	P
C	R	A	B	S
C	R	A	Z	Y
C	R	A	Z	E

La situation est encore plus compliquée quand des **erreurs** peuvent se glisser dans les informations qu'on reçoit...

Erreurs possibles

À chaque fois qu'on veut envoyer un message, on est confronté au schéma suivant :



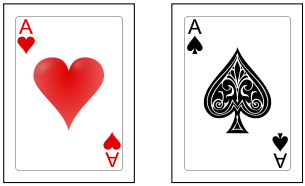
Aucun canal de communication n'est épargné par le "bruit" !
Le décodeur **doit** prendre en compte des erreurs potentielles. Deux tâches possibles :

DÉTECTION et CORRECTION (plus ambitieux !).

Combien de questions pour **détecter** une erreur ?

- On souhaite que notre encodage puisse **détecter une** erreur.
- Combien de questions pour trouver une carte (*avec détection d'une erreur*) parmi

deux cartes ?

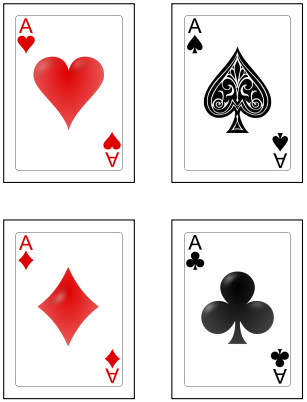


- Une question ne suffit plus, car la réponse peut être erronée ! **Deux** questions suffisent : on répète la question (“Est-ce la carte rouge ? Est-ce la carte rouge ?”).
- Des réponses 00 ou 11 indiquent qu'il n'y a probablement pas eu d'erreur.
  Des réponses 01 ou 10 indiquent qu'une erreur s'est produite $\rightsquigarrow$ il faut recommencer l'interaction.

Combien de questions pour **détecter** une erreur ?

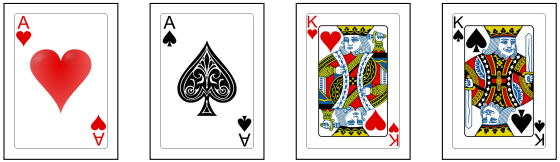
- On souhaite que notre encodage puisse **détecter une** erreur.
- Combien de questions pour trouver une carte (*avec détection d'une erreur*) parmi

quatre cartes ?



Idée simple : répétition

- Idée fréquente dans la vie quotidienne : *en cas d'incompréhension, on* ***répète*** !
- Si on **répète chaque symbole**, le nombre de symboles à transmettre **double**.
- Pour huit cartes, on utiliserait 6 chiffres au lieu de 3 :



00 00 00 | 00 00 11 | 11 00 00 | 11 00 11

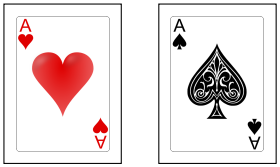
00 11 00 | 00 11 11 | 11 11 00 | 11 11 11

Peut-on faire **mieux** que **répéter** chaque symbole pour **détecter une erreur** ?

(Pensez à notre tour de magie du début...)

Idée plus maligne : **parité**

- Encodage plus court : on ajoute **un seul** chiffre pour que le nombre de 1 soit **pair**.
- Par exemple, pour **quatre cartes**, on passe de **deux**… à **trois** symboles.



00**0**

01**1**

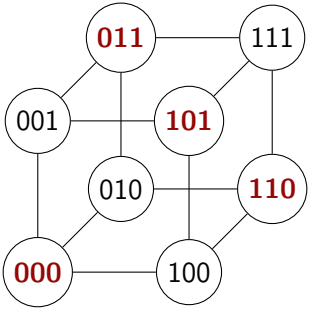
10**1**

11**0**

- Si **une** erreur se produit, le nombre de 1 sera **impair**…
- Suffisant pour détecter **une** erreur !

Pourquoi ça marche ?

- On peut représenter tous les mots de trois symboles comme un **cube**.
- Que constatez-vous sur **les mots utilisés** pour notre encodage ?
- Les mots utilisés sont "**à distance** 2" ! Il faut modifier **au moins deux chiffres** pour passer de l'un à l'autre.
- Si **une** erreur se produit, on tombe sur un mot **à distance** 1 d'un mot de l'encodage. Ça **ne peut donc pas** être un mot valide !



Distance de Hamming

Distance de Hamming

La **distance de Hamming** de deux mots est le nombre de positions où les deux mots diffèrent.

Exemple : "Frite" et "Fruit" ont une distance de Hamming de... 3 !

"Frite" $\rightarrow$ "Frute" $\rightarrow$ "Fruie" $\rightarrow$ "Fruit"

Pour nous, les "mots" sont des nombres **binaires**, utilisant deux symboles.
Par exemple : 0**0**101 et 0**1**101 ont une distance de Hamming de 1.

Forces et faiblesses

Une seule question supplémentaire permet de **détecter une** (*et une seule*) erreur, peu importe où elle se trouve !

Une seule erreur... est-ce suffisant ?

- On pourrait détecter **davantage d'erreurs** en ajoutant plus de redondance.
- Toutefois, aucun encodage ne peut détecter **toutes** les erreurs : si on n'a pas de chance, les erreurs nous font passer d'un mot utilisé pour l'encodage à un autre...
  $\rightsquigarrow$ Aucune façon de détecter d'erreur !
- En ajoutant plus de redondance, on **minimise la probabilité** qu'une telle situation se produise.

Numéro IBAN

BE**62** 5100 0754 7061

- La redondance est utilisée pour **détecter** des erreurs dans les numéros de compte !
- Les numéros belges utilisent 14 chiffres. Les deux premiers chiffres sont **dédiés à la redondance** : ils reprennent le **reste de la division par** 97 d'un calcul sur les autres caractères !
- Même concept que la **parité** ($\rightsquigarrow$ reste de la division par 2).
- Cela garantit que si on se trompe d'**un chiffre**, il est ***impossible*** de tomber sur un autre numéro de compte valide !
- Tous les numéros de compte valides sont **au moins à distance 2**.
- Si on se trompe de **deux chiffres**, la probabilité de tomber sur un numéro de compte valide est **très petite**.
- Toutefois, cet encodage ne permet pas de **corriger** l'erreur...

Correction d'erreurs : comment tout est né

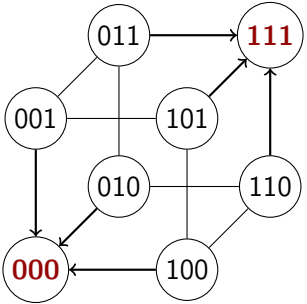
- Années 1940 : Richard Hamming écrit des programmes sur des **cartes perforées**.

- **Problème** : beaucoup d'erreurs dues aux machines et aux cartes ! Des erreurs impliquent qu'il faut attendre un opérateur pour redémarrer la machine... Mais Hamming travaille le week-end et la nuit, et n'a pas envie d'attendre !
- Il réfléchit à comment **corriger** automatiquement des erreurs...
- Il trouve des solutions très élégantes et reçoit pour ce travail le **prix Turing** en 1968, considéré comme équivalent au prix Nobel d'informatique !

Idée simple : trois copies !

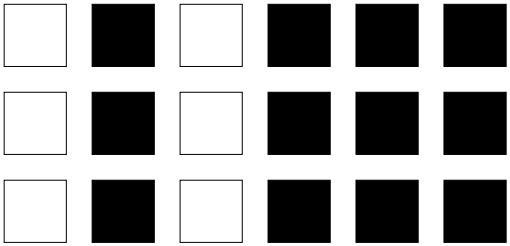
Idée : utiliser des codes qui sont à une "distance encore plus éloignée".

On envoie **trois** copies de chaque chiffre : donc **000** pour 0, et **111** pour 1.
Si on reçoit 010, on corrige vers **000**.
Il suffit que les mots soient à distance de Hamming 3 !



Idée simple : trois copies !

Seul **un tiers** du message envoyé contient vraiment l'information...



Peut-on trouver un encodage plus efficace (nous permettant quand même de corriger une erreur, mais avec moins de redondance) ?

Tour de magie : sept questions, un mensonge

Un volontaire pense à un nombre parmi 0, 1, ..., 15.
On pose 7 questions pour le deviner, mais vous avez droit à (au plus) **un mensonge** !

1. Est-ce que le nombre est plus grand ou égal à 8 ?
2. Est-ce que le nombre est parmi $4, 5, 6, 7, 12, 13, 14, 15$?
3. Est-ce que le nombre est parmi $2, 3, 6, 7, 10, 11, 14, 15$?
4. Est-ce que le nombre est impair ?
5. Est-ce que le nombre est parmi $1, 2, 4, 7, 9, 10, 12, 15$?
6. Est-ce que le nombre est parmi $1, 2, 5, 6, 8, 11, 12, 15$?
7. Est-ce que le nombre est parmi $1, 3, 4, 6, 8, 10, 13, 15$?

Comment avons-nous fait ?

Explication : tous les mots à distance 3 !

- Sans mensonge, il suffirait de 4 questions pour trouver le nombre (car $2^4 = 16$).
- Avec un mensonge, il faut de la **redondance**. Pour pouvoir corriger une erreur, il faut un encodage tel que les mots sont au moins à distance de Hamming 3 !
- C'est possible avec un encodage sur 7 chiffres binaires :

0	0000000	8	1000011
1	0001111	9	1001100
2	0010110	10	1010101
3	0011001	11	1011010
4	0100101	12	1100110
5	0101010	13	1101001
6	0110011	14	1110000
7	0111100	15	1111111

- Toutes les paires de mots sont à distance de Hamming ≥ 3.
- S'il y a **une** erreur (*un mensonge*), il y aura **un seul mot de l'encodage** à distance 1 du mot reçu !

Exercice

Vous êtes à notre place et vous recevez les réponses

$$1010010.$$

Quel est le nombre à deviner ? Pour rappel, voici l'encodage :

0	0000000	8	1000011
1	0001111	9	1001100
2	0010110	10	1010101
3	0011001	11	1011010
4	0100101	12	1100110
5	0101010	13	1101001
6	0110011	14	1110000
7	0111100	15	1111111

Exercice

Vous êtes à notre place et vous recevez les réponses

$$1010010.$$

Quel est le nombre à deviner ? Pour rappel, voici l'encodage :

0	0000000	8	1000011
1	0001111	9	1001100
2	0010110	10	1010101
3	0011001	11	1011010
4	0100101	12	1100110
5	0101010	13	1101001
6	0110011	14	1110000
7	0111100	15	1111111

La réponse est 11, car distance 1 :

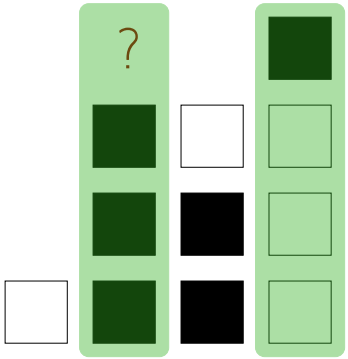
$$1010010$$
$$1011010$$

Codes de Hamming

- Cet encodage est parfois appelé **Code de Hamming** $(7, 4)$, car il permet d'envoyer 4 chiffres d'information, avec correction d'**une** erreur, en envoyant 7 chiffres.
- On peut le généraliser : pour envoyer k chiffres avec correction d'une erreur, il suffit d'ajouter $\approx \log_2(k)$ chiffres supplémentaires.
- Par exemple, si vous souhaitez envoyer 1000 chiffres, il suffit d'ajouter 10 chiffres pour permettre la correction d'une erreur à la réception !
- C'est beaucoup plus efficace que le code utilisé dans le premier tour de magie, qui demandait d'ajouter $2\sqrt{k} + 1$ carrés blancs ou noirs !
- On vous montre comment **construire** le code de Hamming $(15, 11)$.

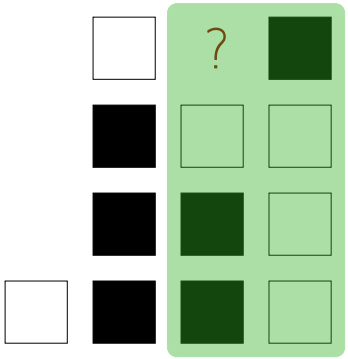
Code de Hamming $(15, 11)$

Exemple de 11 symboles à stocker : 00110011001, **(15, 11)-Hamming codes**



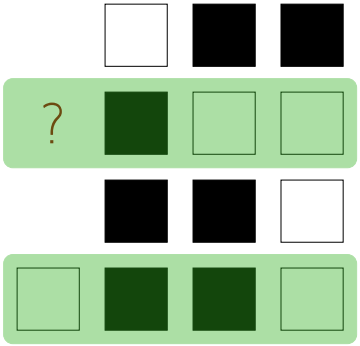
Code de Hamming $(15, 11)$

Exemple de 11 symboles à stocker : 00110011001, **(15, 11)-Hamming codes**



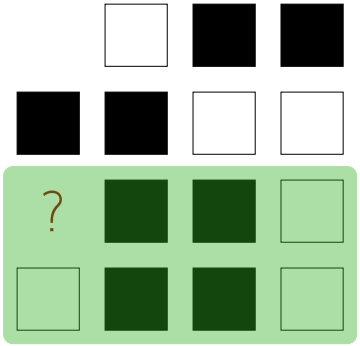
Code de Hamming $(15, 11)$

Exemple de 11 symboles à stocker : 00110011001, **(15, 11)-Hamming codes**



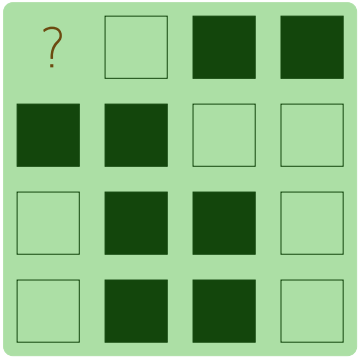
Code de Hamming $(15, 11)$

Exemple de 11 symboles à stocker : 00110011001, **(15, 11)-Hamming codes**



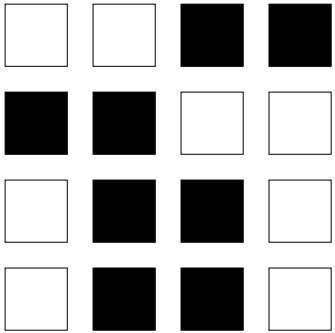
Code de Hamming $(15, 11)$

Exemple de 11 symboles à stocker : 00110011001, **(15, 11)-Hamming codes**

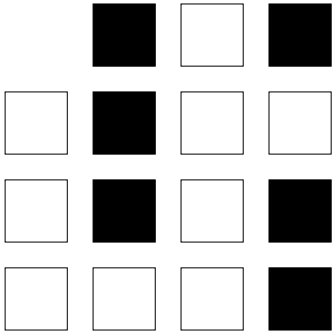


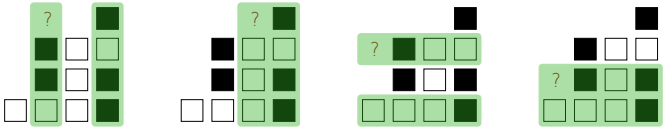
Code de Hamming $(15, 11)$

Exemple de 11 symboles à stocker : 00110011001, **(15, 11)-Hamming codes**



Exercice : quel carré contient une erreur ?





Langue française

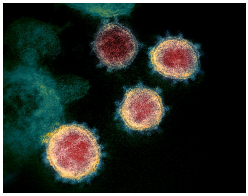
- Notre langage est **redondant** : on utilise davantage de caractères que strictement nécessaires pour transmettre l'information.
- Il permet très souvent de corriger quelques erreurs !
- Par exemple, si on écrit "Salut ! Comlent ça va ?", on est capables de corriger la lettre erronée, car il y a un seul message **valide** (ici, en français correct) à distance de Hamming 1 de notre message.

Codes plus avancés

- Nos codes peuvent corriger une seule erreur.
- Des codes plus avancés, avec plus de redondance, peuvent corriger **plusieurs** erreurs.
- Lorsque vous créez des QR codes, vous pouvez choisir le **niveau de correction d'erreur** : plus le niveau est élevé, plus le QR code est grand, mais plus il peut corriger d'erreurs. Les techniques utilisées pour les QR codes peuvent corriger jusqu'à 30 % d'erreurs, avec environ 50 % de redondance !

Dans la nature, ADN !

- À chaque division cellulaire, 6,4 *milliards* de bases de l'ADN sont copiées. Des erreurs se produisent fréquemment ; comment éviter les problèmes ?
- Beaucoup de **redondance** dans l'ADN : deux copies de chaque chromosome, chaque base de l'ADN est codée avec deux symboles (AT ou CG)...
- Des corrections ont lieu lors de la transcription de l'ADN en ARN messager.
- Le virus du Sars-COV-2 (COVID-19) possède une enzyme **corrigeant des erreurs**, rendant certains traitements antiviraux classiques obsolètes. [1]



1. https://www.medecinesciences.org/en/articles/medsci/full_html/2021/03/msc200459/msc200459.html.

Conclusion : de la contrainte naît l'innovation

- Dans les années 1940, Hamming était **contraint** par les machines de l'époque : compliquées à utiliser, peu résistantes aux erreurs...
- Ces contraintes l'ont poussé à **innover** : comment rendre l'utilisation de ces machines moins pénible ?
- Les techniques développées se sont révélées **essentielles** pour beaucoup de tâches : par exemple, pour communiquer avec des sondes spatiales très distantes !
- L'approche "**contrainte** $\rightarrow$ **innovation**" est un des **moteurs de la recherche scientifique**.
- La prochaine fois que vous utiliserez un QR code, que vous taperez un numéro de compte ou que vous téléphonerez avec Internet, souvenez-vous que derrière tout cela se cache un peu de **magie**... et beaucoup de **mathématiques** !

Merci !

Pour aller plus loin...

- *Solving Wordle using information theory*, https://www.youtube.com/watch?v=v68zYyaEmEA, consulté le 17/03/2025.
- *But what are Hamming codes ? The origin of error correction*, https://www.youtube.com/watch?v=X8jsijhllIA, consulté le 17/03/2025.
- *9.4. QR Codes*, https://www.csfieldguide.org.nz/en/chapters/coding-error-control/qr-codes/, consulté le 17/03/2025.
- *Qui est-ce ? (Avec un mensonge autorisé !!)*, https://www.youtube.com/watch?v=tqc0awqVeMA, consulté le 17/03/2025.