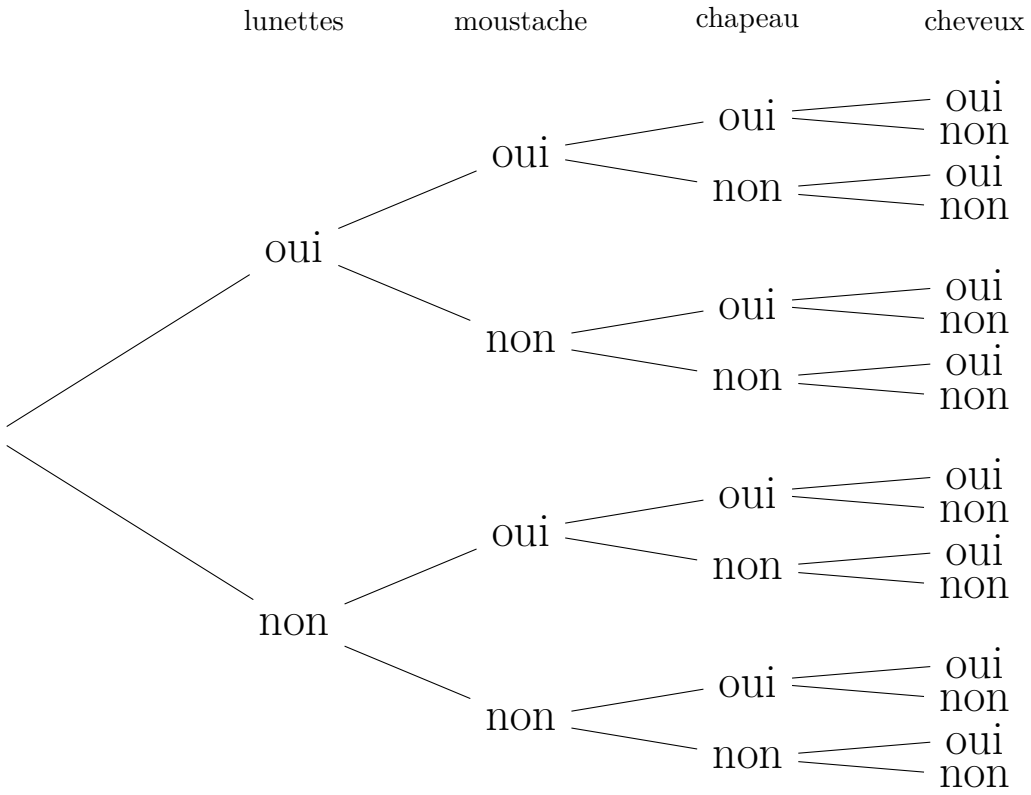
Explications de l'activité
Qui est-ce ?

1. Il y a $2^4 = 16$ personnages différents (ou $2 \times 2 \times 2 \times 2$ pour les élèves qui n'ont pas encore vu les puissances). Pour l'expliquer, on peut faire un arbre présentant toutes les possibilités.



2. On pose autant de questions qu'il y a de critères, soit 4 questions.

3. On rajoute maintenant des éléments sur certains de ces personnages : boucle d'oreille, barbe, nœud papillon. Cela ne change rien... on pose toujours les 4 mêmes questions !

On joue à ce jeu avec les éléments supplémentaires et en rajoutant la règle suivante : le joueur qui a choisi le personnage secret a le droit de mentir au plus une fois en répondant aux questions. Ces questions sont, dans l'ordre :

(1) A-t-il des lunettes ? (2) A-t-il une moustache ? (3) A-t-il un chapeau ?
(4) A-t-il des cheveux ? (5) A-t-il des boucles d'oreille ? (6) A-t-il une barbe ?
(7) A-t-il un nœud papillon ?

On peut lancer une partie sur l'ordinateur (jeu.html, à ouvrir de préférence avec Firefox, éventuellement Chrome, éviter Internet explorer et Safari). Utiliser le jeu de cartes (plus grandes) si les dessins sur l'écran sont trop petits ou pour jouer sans ordinateur.

Remarque pour les pro de l'algèbre linéaire sur le corps $\mathbf{F}_2 = \mathbf{Z}/2\mathbf{Z}$ *: on représente un personnage par un élément de* ${\mathbf{F}_2}^7$ *:* $(x_1, \ldots, x_7)$ *avec* $x_i = 1$ *si la réponse à la question* i *est « oui », 0 sinon. Les personnages représentés dans le jeu sont les 16 éléments du* $\mathbf{F}_2$*-espace vectoriel engendré par* $(1,0,0,0,1,1,0)$*,* $(0,1,0,0,1,0,1)$*,* $(0,0,1,0,0,1,1)$ *et* $(0,0,0,1,1,1,1)$*.*

4. Il y en a 3. On ne va pas demander de vérifier tous les couples de personnages mais l'admettre. Ce qui est important c'est plutôt de comprendre la question suivante.

5. Quand on ment à une question, on décrit un personnage (fictif) qui a une seul différence avec le personnage qu'on avait choisi. Et ce personnage fictif a au moins 2 différences avec les 15 autres personnages du jeu. Remarque : si deux personnages du jeu ne présentaient que 2 différences, on pourrait en mentant une fois décrire un personnage qui n'aurait qu'une différence avec ces deux personnages, et on n'aurait aucun moyen de savoir quel était le bon.

   Remarque réservée aux pro de l'algèbre linéaire : notons H *la matrice suivante, à coeffcients dans* $\mathbf{F}_2$ *:*

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

   Si la réponse obtenue est $x = (x_1, \ldots, x_7)$*, les syndromes associés sont donnés par le produit matriciel* Hx*.*

6. Répartir la tâche dans la classe, chacun en calcule juste 1 ou 2 ! Dans tous les cas, on trouve $s_1 = s_2 = s_3 = 0$.

   En effet, le sous-espace vectoriel des (codes des) personnages du jeu est exactement le noyau de H (on a construit H pour cela).

7. Il va être important de réussir à faire comprendre que la réponse à cette question ne dépend pas du personnage choisi.

   Pour le calcul de s_1, on regarde □✕□✕□✕□. On a vu que si on ne mentait pas, s_1 valait 0. Si on ment à la première question, on change la valeur du premier bit (0 si c'était 1, 1 si c'était 0) et pas des autres, et s_1 vaut donc 1.

   Pour le calcul de s_2, ✕□□✕□□✕. Si on ne ment pas, s_2 vaut 0. Si on ment à la première question, on change la valeur du premier bit (0 si c'était 1, 1 si c'était 0) et pas des autres, ce qui ne change pas s_2 qui vaut toujours 0.

   Pour s_3, comme pour s_2, une modification du premier bit ne change rien, on a toujours $s_3 = 0$.

   Explication en termes d'algèbre linéaire : si le personnage choisi est $m \in \mathbf{F}_2^7$ et qu'on ment à la question i, le personnage fictif décrit est $m + e_i$ où e_i désigne le i-ième vecteur de la base canonique. Le syndrome obtenu est donc $H(m + e_i) = Hm + He_i = He_i$ qui est égal à la i-ième colonne de H. En particulier, si on ment à la première question, $s_1 = 1$, $s_2 = 0$, $s_3 = 0$.

8. On peut donner une explication du même type qu'à la question précédente, pour trouver $s_1 = 1$, $s_2 = 1$ et $s_3 = 1$.

9. Pour savoir à quelle question le joueur a menti, on calcule les syndromes. S'ils ne valent pas tous 0, on regarde le numéro de la colonne correspondant, il donne le numéro de la question à laquelle on a menti. Si les syndromes sont tous les trois nuls, le joueur n'a pas menti. On a construit une sorte de détecteur de mensonge !

	1	2	3	4	5	6	7
s_1	1	0	1	0	1	0	1
s_2	0	1	1	0	1	1	0
s_3	0	0	0	1	1	1	1

Le principe sur lequel repose ce jeu est en fait à la base d'applications très utiles, pour transmettre ou stocker des données. Disons que les données transmises sont des paquets de 0 et de 1. Lors de la transmission, que ce soit par ondes radio, par câble ou autre, des erreurs sont susceptibles de se produire et le message reçu n'est plus parfaitement conforme à celui qui a été envoyé (les erreurs jouent le rôle des mensonges de notre jeu). En envoyant, en plus de l'information de départ, une information supplémentaire redondante, on va être capable de dire s'il y a eu ou non des erreurs lors de la transmission et de les corriger s'il n'y en a pas eu trop. On appelle cela un code correcteur d'erreurs. C'est exactement le même principe que quand des pilotes d'avion disent « Alpha, Charlie, Tango » pour dire « A, C, T » mais avec plus d'algèbre !

10. Si le joueur ment deux fois.

    Regardons les syndromes qu'on peut obtenir, c'est-à-dire toutes les colonnes du tableau. Il y en a 7, soit tous les vecteurs non nul de $(\mathbf{F}_2)^3$. Par conséquent, quelle que soit la réponse que le joueur donne (avec autant de mensonges qu'il veut), si on croit qu'il a joué le jeu et menti au plus une fois, en faisant les calculs on tombe sur un syndrome qui est bien dans le tableau et on corrige en croyant qu'il n'y a eu qu'un mensonge. On ne pourra pas détecter s'il y a eu trop d'erreurs et on donnera un personnage qui ne pourra pas être le bon !

11. On peut introduire ce que l'on appelle un « bit de parité » pour détecteur un tricheur qui mentirait deux fois : chaque personnage était codé par

une suite de sept bits, on en rajoute un huitième qui est égal à 0 si le nombre de 1 dans le code du personnage était pair, 1 si le nombre de 1 était impair (ainsi, la somme des 8 bits d'un personnage est toujours paire). Dans le jeu, on met un piercing si ce bit vaut 1, pas de piercing si c'est 0.

La somme des 8 bits de la réponse vaut alors 0 si et seulement si le nombre de mensonges est pair.

On fait comme précédemment, on calcule le syndrome à l'aide des 7 premiers bits. S'il n'est pas nul, on fait la somme des bits : si elle vaut 1, il y a eu un nombre impair de mensonges (si c'est bien un seul, on sait corriger), si elle vaut 0 on sait qu'il y a eu au moins deux mensonges (qu'on n'est par contre pas en mesure de corriger).