1 Instructions pour le Projet

- Ce projet est à réaliser **en binôme**, c'est-à-dire par groupe de 2. Toutes similitudes suspectes entre les rapports ou codes de groupes différents seront fortement sanctionnées.
- Le langage à utiliser est Prolog (comme en TP). Les commandes vues en cours, TD et TP sont largement suffisantes pour ce projet. Tout appel à des modules ou fonctions non utilisés en TP **est interdit**.
- Votre code doit s'exécuter sans erreur, c'est la base de tout travail en informatique. Un code qui ne s'exécute pas est noté sur la moitié des points.
- Le rendu de votre projet contiendra : un fichier de code `vos_noms.pl` et un rapport au format pdf. Le tout sera à mettre dans une archive (par exemple .zip) [1] portant vos deux noms.
- Trouvez le juste milieu pour le rapport : pas besoin de faire 20 pages, mais 2 pages ne sont pas suffisantes.
  Décrivez ce que vous avez fait, vos choix d'implémentations, ce que vous avez compris ou non, ce que font vos clauses, comment utiliser votre programme et enfin quelques captures d'écrans.
- Le projet est à rendre sur l'espace de dépôt prévu à cet effet pour le **Samedi 23 Mai à 23h55 au plus tard**, tout retard sera sanctionné [2].
- Et enfin, pas de panique ! Le sujet peut sembler long et compliqué mais il est particulièrement facile cette année et surtout le dernier TP (le 6ème) qui se trouve ci-dessous vous permettra de commencer le projet et d'obtenir une correction des premières étapes du projet.
  Bonne chance à tous !

1. C'est à dire que votre rendu de projet doit se limiter à une seule archive contenant **uniquement** 2 fichiers : un programme et un rapport. Merci de supprimer les exécutables ou autres fichiers.
2. Malus de 2 points par tranche de 12h.

Dans ce projet, vous allez manipuler des matrices de n lignes et k colonnes contenant des valeurs booléennes : `v` pour vrai et `f` pour faux.

Le produit *à gauche* d'une telle matrice par un vecteur ligne `V` de longueur n se calcule comme suit :

- partez d'un vecteur ligne de longueur k ne contenant que des `f` (faux), appelons ce vecteur `R` pour "résultat".
- parcourez le vecteur `V` de gauche à droite, si vous rencontrez un `f` (faux), ne faites rien. Si vous rencontrez un `v` (vrai) en position i, alors ajoutez la i-ème ligne de la matrice à votre vecteur `R`.
- À la fin, le vecteur `R` de longueur k contient donc la somme de toutes les lignes de la matrice qui sont à des positions où il y a des `v` (vrais) dans le vecteur `V`.

La somme de deux vecteurs contenant des booléens se fait ici terme à terme en utilisant l'opérateur XOR, par exemple :

$$\begin{pmatrix}\texttt{f} & \texttt{f} & \texttt{v}\end{pmatrix} + \begin{pmatrix}\texttt{v} & \texttt{f} & \texttt{v}\end{pmatrix} = \begin{pmatrix}\texttt{v} & \texttt{f} & \texttt{f}\end{pmatrix}.$$

Pour que les choses soient claires, voici un exemple de produit vecteur-matrice suivant les règles énoncées ci-dessus :

$$\begin{pmatrix}\texttt{f} & \texttt{f} & \texttt{v} & \texttt{v}\end{pmatrix} \times \begin{pmatrix}\texttt{v} & \texttt{f}\\ \texttt{f} & \texttt{v}\\ \texttt{v} & \texttt{v}\\ \texttt{f} & \texttt{v}\end{pmatrix} = \begin{pmatrix}\texttt{v} & \texttt{f}\end{pmatrix}.$$

Il est très important que vous repreniez le petit exemple ci-dessus étape par étape pour bien comprendre comment se fait le produit.

Créez tout d'abord une clause `solution(M,V)` (`M` est une matrice écrite en dur dans votre programme) qui sera vraie quand `V` sera un vecteur tel que $\texttt{V} \times \texttt{M}$ sera égal au vecteur ne contenant que des faux (`f`).

Par exemple, si `M` est la petite matrice 4×2 donnée en exemple ci-dessus, la requête `solution(M,V)` donnera les réponses suivantes :

```
V = (v f v v) ;
V = (f v f v) ;
V = (v v v f) ;
V = (f f f f)
```

Dans un second temps, vous modifierez votre programme pour pouvoir spécifier le nombre de "vrais" que vous souhaitez avoir dans votre vecteur `V`.

Vous obtiendrez donc une clause de la forme `solution(M,V,nombre)` où nombre sera le nombre de coefficients valant vrai (c'est à dire `v`) dans `V`.

Par exemple, si `M` est encore la petite matrice 4×2 donnée en exemple ci-dessus, la requête

```
solution(M,V,2)
```

donnera l'unique réponse suivante :

```
V = (f v f v)
```

À la fin de ce sujet, vous trouverez une grande matrice `M` ainsi que 3 niveaux de difficulté. Pour vous éviter de devoir la recopier "à la main", vous la trouverez dans un fichier .txt dans l'espace dédié au projet sur Moodle. Évidemment, pour l'intégrer dans votre fichier .pl, vous devrez la modifier pour quelle soit dans un format lisible par Prolog.

Le but de ce projet est de fournir un tableau indiquant vos temps de calculs pour obtenir au moins une solution pour chacun des niveaux de difficulté. Les temps de calculs sont indiqués par Prolog à la fin d'une requête.

Voici par exemple les temps (en millisecondes) que j'ai obtenus avec un programme naïf[3], puis optimisé :

Niveau	Naïf	Optimisé
FACILE	44 000 ms	3064 ms
MOYEN	180 100 ms	11 685 ms
DIFFICILE	520 363 ms	34 933 ms

Si vos temps se rapprochent de ceux de la dernière colonne, c'est déjà bien !

Mais vous pouvez obtenir encore mieux car je n'ai pas cherché à vraiment optimiser mon programme ☺.

N'hésitez pas à faire figurer vos anciens temps dans le rapport et à mettre les nouveaux après optimisation en expliquant bien pourquoi et comment vous avez procédé pour obtenir de meilleurs temps. Si vous avez d'autres idées pour améliorez votre code, décrivez-les, même si vous n'avez pas pu ou su les implémenter.

3. Par "naif" j'entends un programme que j'ai codé sans réfléchir et qui donc effectue énormément d'opérations inutiles qui le ralentissent.

Attention : je vous donne un exemple de matrice de taille 26×13 à la fin de ce projet pour vous entrainer ; néanmoins votre clause `solution(M,V,nombre)` doit pouvoir prendre n'importe quelle matrice `M` en entrée !

Il est donc important que votre clause analyse la matrice pour s'y adapter, interdiction de faire figurer les valeurs 26 et 13 en dur dans votre code !

2 La matrice et les niveaux de difficulté

Pour rappel, on dit qu'un vecteur `V` est solution du problème s'il contient le nombre désiré de "vrais" dans ses coefficients et si

$$\mathtt{V} \times \mathtt{M} = (\mathtt{f}, \mathtt{f}, \mathtt{f}, \ldots, \mathtt{f}).$$

La matrice `M` est donnée ci-dessous et voici les objectifs :

FACILE trouvez un vecteur solution `V` contenant 10 vrais.
MOYEN trouvez un vecteur solution `V` contenant 7 vrais.
DIFFICILE trouvez un vecteur solution `V` contenant 6 vrais.

$$\mathtt{M} = \begin{pmatrix}
\mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} \\
\mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} \\
\mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} \\
\mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} \\
\mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} \\
\mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} \\
\mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} \\
\mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} \\
\mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} \\
\mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} \\
\mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} \\
\mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} \\
\mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{v} & \mathtt{v} \\
\mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} \\
\mathtt{f} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{v} & \mathtt{f} & \mathtt{f} & \mathtt{v} & \mathtt{f}
\end{pmatrix}$$

3 TP 6

Pour commencer, vous devez déclarer les deux valeurs booléennes du projet avec les faits suivants :

```
booleen(v).
booleen(f).
```

Question 1 : Écrivez une clause pour obtenir le XOR entre deux valeurs booléennes.

Question 2 : Choisissez une manière pour représenter les vecteurs et matrices de booléens en Prolog.

Question 3 : Écrivez une clause permettant de récupérer tous les vecteurs possibles pour une longueur donnée. Par exemple, la clause

```
vecteur(X,2)
```

donnera les résultats suivants :

```
X = (v,v) ;
X = (f,v) ;
X = (v,f) ;
X = (f,f)
```

Question libre : Le but de ce TP est de vous permettre de comprendre le projet et de le commencer un peu. Si vous avez une question précise, envoyez moi votre code et votre question à `maxime.bros@unilim.fr`. J'y répondrai alors pendant la dernière séance de TP.

Point culture : Ce problème, tout simple en apparence, est très important en cryptographie car si vous arrivez à trouver des solutions rapidement, alors vous serez capable d'attaquer des cryptosystèmes modernes qui en dépendent.

La communauté des cryptographes tient donc à jour des tableaux de "records" sur ce problème, pour pouvoir jauger nos capacités d'attaque.

À l'heure actuelle, le record est detenu par un français et a été obtenu sur une matrice contenant 1280 lignes et 640 colonnes avec un vecteur `V` contenant 220 "vrais", le calcul de ce dernier a pris plus de 34 heures !

Vous l'aurez remarqué, plus le nombre de "vrais" est petit, plus c'est dur, et il y a naturellement une valeur minimale. Sur cette matrice, la théorie nous dit que le minimum, donc le plus dur, serait de trouver un vecteur avec 144 "vrais".