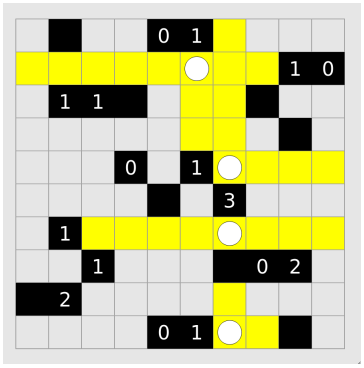
# Complexité et Calculabilité : projet 2022/2023
# Lightup!

## 1 Présentation du problème



Lightup est un jeu de logique, qui se joue sur une grille rectangulaire. Sur certaines cases se trouvent des murs, et sur certains murs se trouvent des nombres de 0 à 4.

On peut placer sur les cases vides de la grille (celles où il n'y a pas de mur) des lampes, qui éclairent toutes les cases sur leur ligne et leur colonne jusqu'au prochain mur.

Le but est de placer des lampes sur certaines cases vides de la grille de manière que

— toutes les cases vides soient éclairées,
— aucune lampe n'éclaire une autre lampe
— si un mur a un nombre $k$ écrit sur lui, alors il y a exactement $k$ lampes sur les cases adjacentes à ce mur.

Une interface pour jouer à ce jeu et vous familiariser avec les règles est disponible ici, sur la page de Simon Tatham.

Dans ce projet nous représenterons les grilles comme des matrices : Une grille avec $n$ lignes et $m$ colonnes sera représentée par une matrice des mêmes dimensions, avec comme coefficients des étiquettes pouvant être BLANK

(case vide), WALLU (mur sans nombre), WALL0, WALL1, WALL2, WALL3, WALL4 (murs numérotés) dont l'ensemble est noté square.

L'ensemble de ces matrices est noté $\mathcal{M}^{n\times m}(\text{square})$.

Une solution à une grille sera également représentée comme une matrice, mais avec des coefficients booléens : il y a un $\top$ aux coordonnées $(i, j)$ si il y a une lampe à cet endroit.

L'ensemble de ces matrices est noté $\mathcal{M}^{n\times m}(\{\top, \bot\})$.

Notez que dans l'implémentation qui vous sera demandée, les deux types de matrices seront confondues en une seule, square contenant alors un élément supplémentaire LIGHTBULB, représentant une lampe (celles-ci ne pouvant être placées que sur des cases vides, c'est une représentation correcte). Les séparer dans la partie théorique nous permet de distinguer clairement l'instance du certificat.

# 2 Vérificateur

Dans un premier temps, nous allons programmer un vérificateur, c'est-à-dire un algorithme qui prend en entrée une instance de Lightup (une grille avec des murs) et une solution (un ensemble de cases), et indique si cet ensemble de cases est bien une solution à la grille.

1. Écrivez un algorithme en pseudo-code de haut niveau qui vérifie qu'un ensemble de lampes est une solution. Votre algorithme doit prendre en entrée deux matrices de même taille $n \times m$, l'une de $\mathcal{M}^{n\times m}(\{\text{square}\})$ (représentant la grille) et l'autre de $\mathcal{M}^{n\times m}(\{\top, \bot\})$ (représentant un ensemble de lampes).

   Il doit renvoyer un booléen indiquant si la deuxième matrice représente une solution au problème représenté par la première.

   Votre algorithme doit traiter clairement tous les cas précisés dans les règles. N'hésitez pas à le découper en plusieurs sous-algorithmes pour plus de lisibilité.
2. Quelle est la complexité de votre algorithme ?
3. Qu'en déduisez-en sur la complexité du problème Lightup ? Justifiez votre réponse.
4. Utiliser le vérificateur de la question précédente pour écrire un algorithme qui, étant donné une grille (sous la forme d'une matrices de $\mathcal{M}^{n\times m}(\{\text{square}\})$), indique si il existe une solution à cette instance du problème. Votre programme doit énumérer tous les ensembles de lampes possibles et les vérifier un à un.
5. Quelle est la complexité de cet algorithme ?

6. Afin d'accélérer le programme précédent, présentez des idées pour diminuer le nombre d'ensemble de lampes à vérifier, en évitant d'explorer des ensembles de lampes clairement mauvais. Écrivez un algorithme qui utilise ces idées pour déterminer si une grille a une solution plus rapidement que celui donné en question 2.
7. Quelle est la complexité de ce nouvel algorithme ?
8. Dans le fichier `brute_force.c`, vous implémenterez le vérificateur et la dernière version de l'algorithme de résolution. Attention, dans le code à réaliser, les deux matrices sont regroupées en une seule. Votre implémentation devra permettre de résoudre une instance où certaines lampes sont déjà placées en ne modifiant pas les lampes déjà placées.

   Incluez dans la partie écrite un rapport d'implémentation de cette fonction (cf Consignes de rendu).

# 3 Réduction à SAT

Nous allons écrire une réduction du problème Lightup à SAT, c'est-à-dire un programme transformant (en temps polynomial) une grille $G$ en une formule booléenne $\phi$ telle que la première a une solution si et seulement si la deuxième est satisfaisable.

Nous utiliserons un outil efficace de résolution de formule booléennes, appelé Z3, pour déterminer si $\phi$ est satisfaisable, et donc si $G$ a une solution.

1. On considère un tableau de variables booléennes var de taille $n$, avec $n$ au plus 4. Pour chaque $k$ entre 0 et $n$, on souhaite déterminer une formule $exact_k(\text{var}, n)$ vraie si et seulement si exactement $k$ des variables sont évaluées à $\top$. Donnez une définition de cette formule. Vous pouvez donner ces formules explicitement, ou les définir de manière récursive (cela est plus que conseillé).
2. Soit $G$ une grille de $\mathcal{M}^{n\times m}$(square) Décrivez en détail une fonction red produisant une formule booléenne red($G$) satisfaisable si et seulement si $G$ a une solution.

   Il vous faudra définir des variables dont la valuation ($\top$ ou $\bot$) représente un ensemble de lampes, et exprimer le fait que cet ensemble soit une solution à $G$ comme une formule booléenne.

   Il vous est conseillé de découper votre fonction en sous-fonctions compréhensibles et courtes produisant des formules décrivant des propriétés simples que vous expliciterez (par exemple, une sous-fonction par règle du jeu).

   **Attention aux bords de la grille !**

3. Démontrez que la fonction que vous avez donnée est une réduction de Lightup vers SAT. Il vous faut démontrer que $G$ admet une solution si et seulement si red($G$) est une formule satisfaisable (n'oubliez pas de direction).
4. Quelle est la taille de red($G$) par rapport à la taille de $G$ ?
5. Qu'en déduisez-vous sur la complexité de Lightup ? Justifiez votre réponse.
6. Avons-nous, dans le cours de ce projet, démontré que Lightup est un problème NP-complet ? Si la réponse à cette question est non, que faudrait-il faire pour le démontrer ?
7. Complétez le fichier `reduction.c` pour y implémenter les fonctions décrites dans cette section. Il vous faudra également y implémenter une fonction qui étant donné une affectation des variables, place les lampes aux endroits correspondant dans $G$.

   Incluez dans la partie écrite un rapport d'implémentation de cette fonction (cf Consignes de rendu).

   **Il est plus que conseillé d'effectuer cette dernière question en parallèle des autres : l'implémentation de votre réduction vous permettra de déterminer la correction de celle-ci.**
8. (Bonus) Étant donné un jeu G, la formule obtenue par la réduction déterminée dans cette section red($G$) et une valuation la satisfaisant val, donnez un algorithme produisant une formule satisfaisable red2($G$) si et seulement si red($G$) a au moins deux valuations différentes la satisfaisant. Implémentez cette formule dans `reduction.c`.

# 4 Bonus : Grilles à une ligne

Nous allons étudier le problème sur $\mathcal{M}^{1\times m}$(square), c'est-à-dire les grilles à une seule ligne. Nous allons écrire un algorithme pour résoudre ces grilles. Nous allons marquer les cases de la grille au fur et à mesure par des $\top$ (indiquant qu'on est obligés de mettre une lampe là) et des $\bot$ (indiquant qu'on ne peut pas mettre de lampe là). Si on obtient une contradiction à un moment, on en conclut que la grille n'a pas de solution. Au départ tous les murs sont marqués par $\bot$, les autres cases ne sont pas marquées.

1. Pour chacune des situations suivantes, indiquez si vous pouvez ajouter des $\top$ ou $\bot$ à la grille, ou conclure immédiatement à une contradiction :
   — Un mur est marqué par 3 ou 4.
   — Un mur marqué par 2

— Un mur marqué par 1 avec un voisin marqué par $\bot$
— Deux murs séparés par une ou plusieurs cases vides, toutes marquées $\bot$
— Deux cases vides sans mur entre elles dont une est marquée $\top$

2. On considère à présent que les conditions précédentes ne permettent plus de conclure ou de marquer de nouvelles cases. Pour chaque intervalle de cases vides entre deux murs dont une case au moins est non marquée, on met une lampe dans la case non marquée la plus à gauche. Montrer que cela produit un remplissage valide de la grille.
3. Donnez un algorithme pour résoudre les grilles de Lightup à une ligne.

# 5 Consignes de rendu

Le projet est à effectuer en binômes, de préférence à l'intérieur de votre groupe de TD. Vous devez déclarer vos binômes sur le moodle du cours. En cas de binômes inter-groupe ou de nécessité à former un trinôme, vous en demanderez l'autorisation à votre chargé de TD qui, s'il en donne l'autorisation fera la manipulation nécessaire.

La date butoir de ce projet est le **19 octobre 2022**. Vous devez rendre sur le moodle du cours les documents suivants :

— Un fichier pdf contenant les réponses aux questions présentes dans ce sujet. Pour les dernières questions de chaque partie, vous inclurez un rapport d'implémentation décrivant l'état d'avancement de votre implémentation si celle-ci n'est pas correcte, une explication de ce qui fonctionne ou non, des difficultés rencontrées, ainsi qu'une liste des instances sur lesquelles vous avez testé votre programme et de si le résultat est correct ou non.
— Le dossier src du code dûment complété avec votre implémentation. Vous n'êtes pas autorisés à modifier les fichiers .h dans include – cela étant inutile.