Question 1

Cas d'erreurs identifiés :

- fichier vide
- erreur sur le nombre ligne/colonnes
- nombre de lignes/colonnes négatif
- caractères non numériques
- Autre chose que 0 ou 1 dans le tiling

Question 2

a) Dans la solution 1 un rectangle est représenté par deux points : son coin supérieur gauche (x0,y0) et son coin inférieur droit (x1,y1).

L'algorithme commence donc par énumérer les coins supérieurs gauches possibles et pour chacun il énumère les coins inférieurs droits possibles (sachant que ∀(x0,y0) x1>x0, y1>y0 car le coin inférieur droit est toujours plus bas et plus à droite que le coin supérieur gauche).

Pour chaque rectangle ainsi formé, l'algorithme en examine chaque case et vérifie sa couleur. Si elles sont toutes blanche, sa taille est comparé par rapport à la taille du plus grand rectangle déjà trouvé.

b) Tests réalisés :

- tous les carrés sont noirs
- tous les carrés sont blancs

c) Coût en temps : $O(n^6)$.

Coût en mémoire : $O(1)$. C'est un algorithme en place

Question 3

a) Pour chaque case du dallage, on regarde tous les plus grand rectangles qu'on puisse faire en utilisant la case de dallage actuel comme coin supérieur gauche.

Ensuite pour éviter de tester des cas de rectangles impossibles on s'arrête dès qu'on trouve une case noire sur la ligne.

Si sa surface est plus grande que le pus grand rectangle précédent, on le retient comme candidat.

b) Tests réalisés :

- tous les carrés sont noirs
- tous les carrés sont blancs

- résultats comparés avec ceux de la solution 1

c) Coût en temps : $O(n^4)$

Coût en mémoire : $O(1)$. C'est un algorithme en place

Question 4

a) Après avoir calculé les hauteurs, pour trouver les rectangles valide il est possible de faire cette algorithme :

- Parcourir chaque colonne, récupérer sa hauteur
- Regarder jusqu'à quel colonnes de droites on peut aller sans réduire sa hauteur
- Regarder jusqu'à quel colonnes de gauche on peut aller sans réduire sa hauteur

Ce qui permet de tester correctement ce tableau de hauteur par exemple :

[33222] => le rectangle 33 est testé, et le 25 qui commence dès le premier 3 est testé aussi.

b) Tests réalisés :

- tous les carrés sont noirs.
- tous les carrés sont blancs
- un cas ou le tableau de hauteurs est souvent changé (diagonale de cases noires)

c) Coût en temps : $O(n^3)$.

En effet la boucle principale parcourt toutes les lignes, toutes les colonnes pour étudier chaque hauteur, et il faut enfin comparer ces hauteurs avec les autres sur la ligne, donc de nouveau au pire re-parcourir toutes les colonnes.

La complexité du calcul du tableau de hauteur est de $O(n^2)$ ce qui est inférieur à la complexité trouvé au dessus.

Donc on garde la complexité à $O(n^3)$.

Le coût est mémoire : O(n) à cause du tableau de hauteurs.

Question 5

- Encapsulation effectué en mettant la définition de la structure dans le .c au lieu du .h
- Ainsi, le main (ou autre) n'a pas accès à la structure car il n'inclut que le header. (Même si en C il est toujours possible de casser cette encapsulation avec un cast et la définition d'une structure semblable, cela demanderait à l'utilisateur d'en être bien conscient)
- tests réalisés :

- ajout d'éléments
- affichage
- test vide
- vidage
- nouveau test vide

Question 6

a/ Quand on tombe sur une case noire, il faut penser à tout dépiler.

Quand on dépile à cause d'une réduction de hauteur ou d'une case noir, il faut penser à considérer le dépilage une case avant, sinon la dernière case qui a posé problème va être compté dans le rectangle.

b/ Il faut stocker un couple d'entiers contenant respectivement la hauteur et la position du début du rectangle.

c/ tests réalisés :

- tous les carrés sont noirs.
- tous les carrés sont blancs
- un cas ou le tableau de hauteurs est souvent changé (diagonale de cases noires)

Question 7

a) Pour une ligne sur un dallage carré de côté n :

- calculer n hauteurs
- au pire empiler+depiler n rectangles

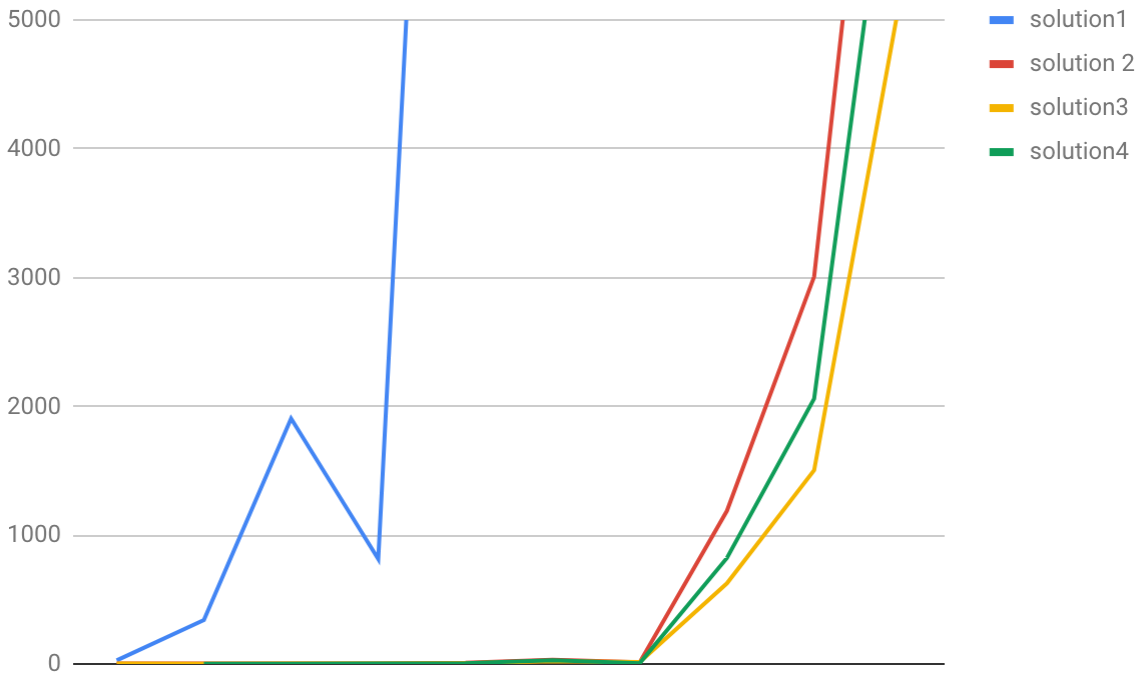
b) Coût en temps : $O(n^2)$: deux parcours imbriqués lignes x colonnes

Coût en mémoire :

- taille du tableau de hauteurs : n
- taille max de la pile : n (on n'empile pas plus d'un rectangle par colonne)
- ⇒ Coût en mémoire total : 2 x n

c) Cette solution est optimale car on ne parcourt chaque case qu'une fois

Question 9



Pour une raison qui nous est inconnue, en pratique, le coût en temps de la solution 4 est supérieur à celui de la solution 3. Nous avons investi beaucoup de temps pour en chercher la raison mais nous n'avons pas trouvé.