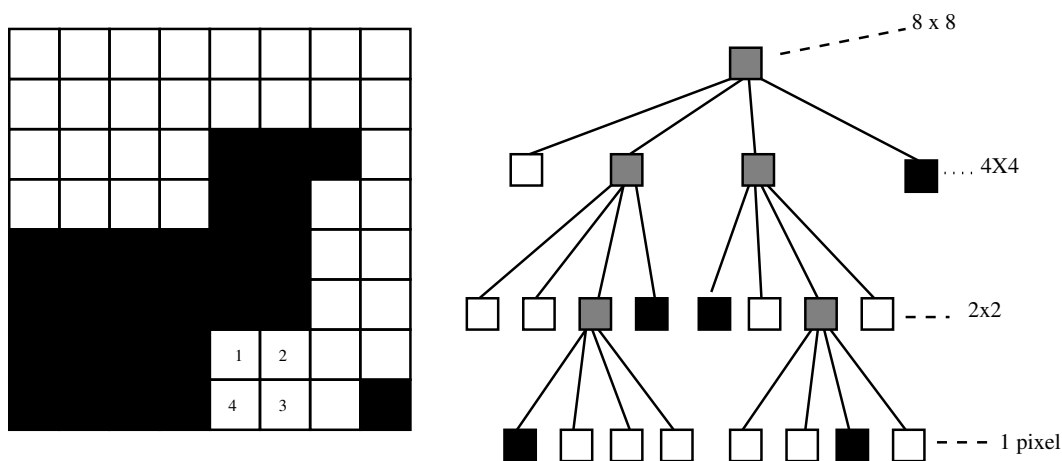


## Quadtrees et compression d'image

Pour simplifier, on suppose les images carrées, de côté  $2^n$ , et en noir et blanc. L'idée est la suivante : une image toute blanche ou toute noire se représente par sa couleur, tandis qu'une image composite se divise naturellement en quatre images carrées.



Nous considérons les types suivants :

```
1 type couleur = Blanc | Noir
2 type quadtree = Feuille of couleur
3                 | Noeud of quadtree * quadtree * quadtree * quadtree
```

- Le résultat de ces deux fonctions ne correspond pas au quadtree d'une image monochrome, qui lui est une feuille (soit noire pour `quadtree_full`, soit blanche pour `quadtree_empty`). Leur intérêt est de construire un quadtree dont la hauteur permet de déduire la taille de l'image. Le résultat de ces fonctions n'est pas unique (on peut construire plusieurs `quadtree` convenables. On vous laisse construire celui de votre choix.

- Écrire une fonction `inverse` qui prend un `quadtree` `a` représentant une image `i` et qui renvoie un `quadtree` représentant l'image `i'` obtenue à partir de `i` en échangeant `noir` et `blanc`.

3. Écrire une fonction `rotate` qui prend un `quadtree` `a` représentant une image `i` et qui renvoie un `quadtree` représentant l'image `i` tournée d'un quart de tour vers la gauche.
4. Écrire une fonction `union` qui prend deux `quadtree` `a` et `b` représentant chacun respectivement l'image `i` et l'image `i'` et qui renvoie un `quadtree` `c` représentant l'union des deux images.
5. Écrire une fonction `intersection` qui prend deux `quadtree` `a` et `b` représentant chacun respectivement l'image `i` et l'image `i'` et qui renvoie un `quadtree` `c` représentant l'intersection des deux images.

Pour l'union et l'intersection, on suppose que les `quadtree` `a` et `b` correspondent à des images de même taille. On se donne les règles suivantes :

union ( $\cup$ )	intersection ( $\cap$ )
<code>blanc <math>\cup</math> x = x <math>\cup</math> blanc = x</code>	<code>blanc <math>\cap</math> x = x <math>\cap</math> blanc = blanc</code>
<code>noir <math>\cup</math> noir = noir</code>	<code>noir <math>\cap</math> noir = noir</code>

6. Écrire une fonction `color` qui prend en arguments les coordonnées (`x`, `y`) d'un point dans l'image `i`, la taille de `i` et le `quadtree` qui lui correspond et qui retourne la couleur du point. Si les coordonnées du point sont incorrectes, la fonction retourne la valeur `None`.
7. Écrire une fonction `modify` qui prend un `quadtree` `a` représentant une image `i` et la taille de `i` et qui modifie `i` en appliquant à chacun de ses points une fonction de profil `int -> int -> couleur -> couleur`. L'argument de type `couleur` est l'ancienne couleur du point, et la fonction retourne sa nouvelle couleur.
8. Puisqu'on peut modifier une image, écrire une fonction `optimise` qui prend le `quadtree` `a` qui lui correspond et qui optimise sa représentation arborescente : un nœud dont les fils sont tous de la même couleur sera modifié en feuille.

## Codage de quadrees

On souhaite pouvoir transformer un `quadtree` en une liste de 0 et de 1, et réciproquement.

On note `code(a)` la liste de 0 et de 1 représentant un `quadtree` `a`. On choisit le codage suivant :

```
code(Feuille Blanc) = 00
code(Feuille Noir) = 01
code(Noeud (a1,a2,a3,a4)) = 1 code(a1) code(a2) code(a3) code(a4)
```

Considérons le type suivant :

```
type bit = Zero | Un
```

9. Écrire une fonction `quadtree_to_list` de type `quadtree -> bit list` qui transforme un `quadtree` en une liste de bits selon le codage.
10. Écrire une fonction de décodage `list_to_quadtree` de type `bit list -> quadtree` qui transforme une liste de bits en le `quadtree` correspondant. Le `quadtree` devra être optimal. Ne pas utiliser la fonction `optimise` pour écrire cette fonction.

## Travail demandé :

Le devoir est à programmer individuellement. Il sera accompagné d'un dossier contenant la description des fonctions. Pour chaque fonction, on donnera impérativement l'interface complète (dans le code en commentaire et dans le rapport pour les fonctions présentées).

Le dossier fournira également des cas de tests accompagnés des résultats attendus et retournés.

Le dossier doit également comporter un mode d'emploi de l'outil développé. Rapport et programme ".ml" (suffisamment commenté) devront être archivés dans un "(.zip)" et déposés sur Celene. Un dépôt sera créé pour ce devoir. Vous devrez impérativement indiquer dans votre dossier les numéros de votre groupe de TD et de TP.