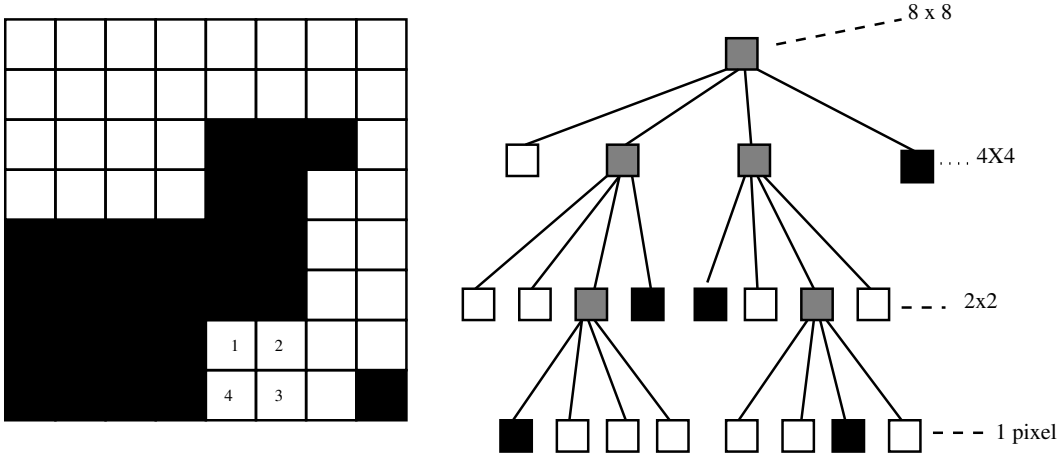
Lic. Info. 2ème année — Université d'Orléans
SLA4IF02 - Prog. fonctionnelle — Année 2019-2020

Devoir Maison de Programmation
À rendre le 10/4/2020

Quadtrees et compression d'image

On présente une représentation d'images sous forme d'arbres. Cette représentation donne une méthode de compression plus ou moins efficace et facilite certaines opérations sur les images.

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.



Le carré de taille k lorsqu'il est de deux couleurs différentes se subdivise en quatre carrés chacun de taille $k/2$ considérés dans l'ordre donné par les chiffres dans la figure, i.e. dans l'ordre des aiguilles d'une montre en commençant par le carré en haut à gauche.

Nous considérons les types suivants :

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

1. Écrire une fonction `quadtree_full` (respec. une fonction `quadtree_empty`) qui prend en argument un entier n et qui retourne un `quadtree` de taille 2^n totalement noir (respec. totalement blanc).
2. 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.

On se donne les règles suivantes :

union ($\cup$)	intersection ($\cap$)
`blanc` $\cup$ x = x $\cup$ `blanc` = x `noir` $\cup$ `noir` = `noir`	`blanc` $\cap$ x = x $\cap$ `blanc` = `blanc` `noir` $\cap$ `noir` = `noir`

6. Écrire une fonction `color` qui prend les coordonnées `(x, y)` d'un point dans le quadtree `a` et qui retourne sa couleur. 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 qui modifie `i` en appliquant à chaque point de `a` 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 optimise sa représentation arborescente : un nœud dont les fils sont tous de la même couleur sera modifié en feuille.

Codage de quadtrees

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.