

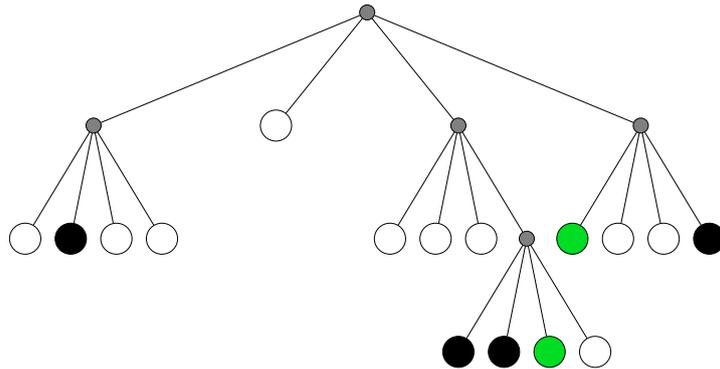
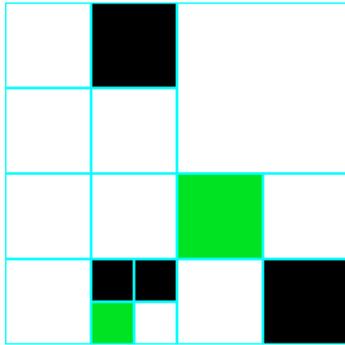
---

## PROJET « CODAGE D'UNE IMAGE PAR UN ARBRE QUATERNAIRE »

---

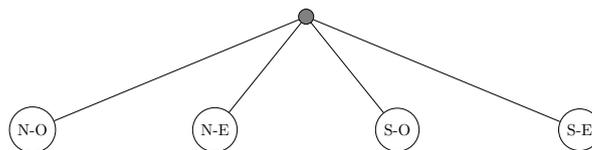
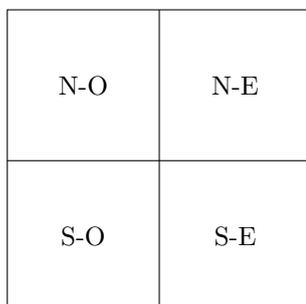
### 0 Présentation

**Principe du codage** Une image peut être décrite récursivement par des zones en la découpant systématiquement par quatre jusqu'à obtenir une couleur homogène dans chacune des zones. L'image peut ainsi être représentée par un arbre dont chaque noeud peut à son tour avoir quatre noeuds, appelé **arbre quaternaire** ou « quadtree » en anglais. Dans ce cas, chaque noeud interne possède quatre sous-arbres et les feuilles contiennent la couleur.



On représente ci-dessus les découpages successifs d'une image ainsi que l'arbre quaternaire correspondant. On obtient seize zones de différentes tailles associées à une unique couleur, toutes représentées par une feuille dans l'arbre.

**Implémentation** Comme pour les arbres binaires, on choisit une structure de données chaînée pour représenter un arbre quaternaire ; chaque noeud est lui-même un arbre possédant quatre sous-arbres, chacun identifié par le nom d'un point cardinal parmi les suivants ; N-O, N-E, S-O et S-E.



En programmation objet, cette structure peut être implémentée par la classe suivante :

```

1 class ArbreQuaternaire:
2     def __init__(self, px, x, y, dim):
3         self.no = None
4         self.ne = None
5         self.so = None
6         self.se = None
7         self.px = px
8         self.x = x
9         self.y = y
10        self.dim = dim
11        self.couleur = None

```

Les attributs `x`, `y` stockent les coordonnées de l'origine d'une zone de forme carrée de côté `dim` pixels de l'image codée sous la forme d'une matrice de pixels, dont la référence est stockée dans l'attribut `px`. La couleur du pixel de coordonnées `(0,0)` (le coin supérieur-droit, par convention) pourra être stockée dans l'attribut de même nom de la manière suivante : `self.couleur = px[0][0]`.

**Format d'encodage** Pour sauvegarder les images construites sous forme d'arbre quaternaire, on définit le type de format `.aq`. Les fichiers de ce format seront à ouvrir, lire ou écrire en format texte `utf-8`.

On placera dans ces fichiers un **parcours préfixe** de l'arbre quaternaire avec les règles suivantes :

- La première valeur doit être la dimension de l'image de forme carrée.
- Les nœuds internes sont codés par un 0.
- Les feuilles sont codées par un 1 suivis des trois composantes R, V, B de la couleur de la zone.

Par exemple, l'encodage de l'arbre donné en introduction serait :

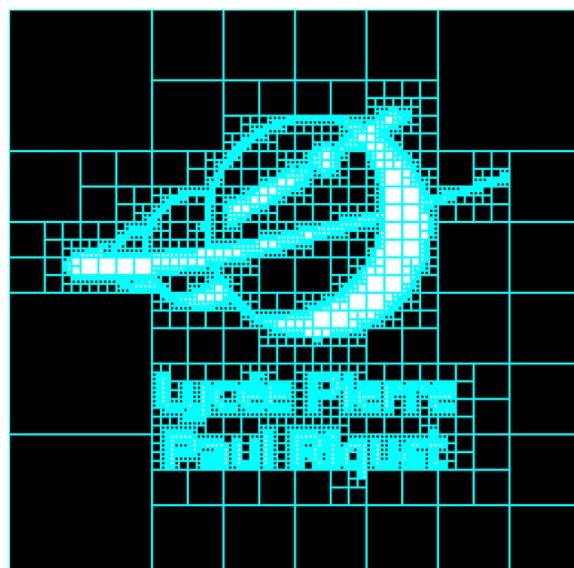
```
256 0 0 1 255 255 255 1 0 0 0 1 255 255 255 1 255 255 255 1 255 255 255 0 1 255 255 255 1 255 255 255
1 255 255 255 0 1 0 0 0 1 0 0 0 1 0 227 35 1 255 255 255 0 1 0 227 35 1 255 255 255 1 255 255 255 1
0 0 0.
```

## 1 Cahier des charges

Le programme développé permet d'encoder et de décoder des images en utilisant des arbres quaternaires. Il est distribué sous la forme d'un script Python, utilisable en mode console, ou sous la forme d'une bibliothèque et doit posséder les fonctionnalités suivantes, il peut :

- ouvrir des fichiers images de forme carrée ;
- générer un arbre quaternaire représentant l'image chargée ;
- afficher l'image en faisant apparaître les bordures de chaque zone à partir d'un arbre quaternaire ;
- sauvegarder l'arbre dans un fichier au format `.aq` ;
- ouvrir l'arbre dans un fichier au format `.aq` ;

**Bonus** Le programme peut être intégré dans une application graphique, permettant la sélection des différents fichiers (`.png`, `.aq`, etc) à partir de boutons et l'affichage de l'image en activant l'ajout des bordures de zone à partir d'une case à cocher.



Exemple d'affichage des bordures de zone de l'arbre quaternaire encodant l'image du logo du lycée.