

PROJET ALGORITHMIQUE ET PROGRAMMATION ORIENTÉE OBJET

Le but de ce projet est d'écrire une bibliothèque de fonctions de manipulation d'arbres et de forêts, ainsi qu'une interface permettant d'utiliser ces fonctions, pour utiliser ces structures de données pour représenter des images. Il est à réaliser en C++ par groupe de trois étudiants au plus.

1 Arbres quadrants

On désire représenter des images, constituées de pixels disposés sur une surface carrée dont le côté est de longueur 2^n , où n est un entier positif. Chacun des pixels peut être colorié. Dans ce projet, on ne considérera que deux couleurs, blanc ou noir.

On représente ces images par un arbre *orienté* ayant les caractéristiques suivantes :

- chaque nœud interne possède quatre fils ;
- chaque feuille représente un pixel et porte une information qui est la couleur du pixel représenté ;
- l'arbre est complet, chaque feuille est à la même hauteur.

L'arbre est défini récursivement de la façon suivante :

- si le côté du carré est $1 = 2^0$, l'arbre est représenté par une feuille
- si le côté du carré est 2^n , la racine de l'arbre a 4 fils. Le premier fils représente la partie Nord-Ouest de l'image, le deuxième la partie Nord-Est, le troisième la partie Sud-Ouest, et enfin le quatrième la partie Sud-Est. Chacun des fils représente ainsi un carré de $2^{n-1} \times 2^{n-1}$ pixels, et l'arbre est de hauteur n . Dans l'exemple suivant $n = 3$. On a simplement marqué par des carrés noirs les feuilles correspondant à des pixels noirs. Les autres feuilles n'ont pas été dessinées et sont sous-entendues. Les nœuds internes sont marqués par des disques.

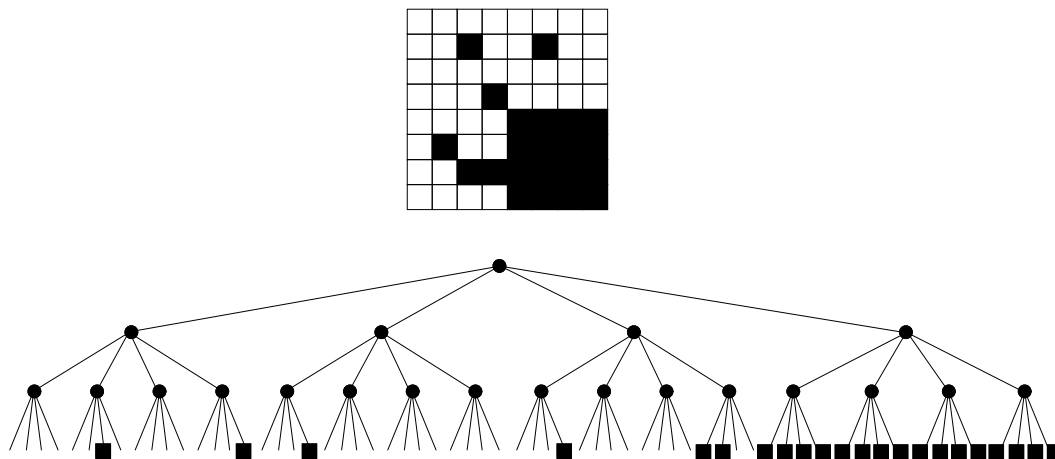


FIG. 1 – Un arbre quadrant et son image associée

À des fins de vérification, les figures pourront être représentées par des fichiers de caractères, un octet particulier représentant les pixels blancs (espace par exemple), un autre les pixels noirs. Si on

repréend l'exemple de la Figure 1, en représentant un pixel blanc par le caractère espace (␣ sur la figure ci-dessous) et un pixel noir par *, on obtiendrait (↵ représentant le caractère *newline*):

```

␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ↵
␣ ␣ * ␣ ␣ * ␣ ␣ ↵
␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ↵
␣ ␣ ␣ * ␣ ␣ ␣ ␣ ↵
␣ ␣ ␣ ␣ * * * * ↵
␣ * ␣ ␣ * * * * ↵
␣ ␣ * * * * * ↵
␣ ␣ ␣ ␣ * * * * ↵

```

FIG. 2 – Représentation des images dans un fichier

On demande d'abord d'écrire des fonctions permettant de

- sauver une image dans un fichier,
- lire un fichier représentant une image,

On demande ensuite d'écrire des fonctions de manipulation de ces images. La liste n'est pas exhaustive, mais on doit au moins pouvoir faire :

- l'intersection de deux images : les pixels noirs de la nouvelle image sont ceux qui étaient noirs dans les deux images de départ ;
- l'union de deux images : les pixels noirs de la nouvelle image sont ceux qui étaient noirs dans l'une *ou* l'autre des images de départ ;
- les symétries verticale et horizontale par rapport aux axes médians ;
- la rotation d'angle $\frac{\pi}{2}$ et de centre le milieu de la figure ;
- l'agrandissement (c'est-à-dire le *zoom*) d'un quart de la figure. On peut demander l'agrandissement de l'un des quatre quarts de plan correspondant aux quatre premiers fils.
- la recherche de la composante connexe d'un pixel de couleur noire. La composante connexe d'un pixel p est définie comme suit. C'est l'ensemble de tous les pixels de même couleur que p qui sont atteignables à partir de p en faisant uniquement des pas Nord, Sud, Ouest, ou Est, et en ne traversant que des pixels de même couleur que p . Par exemple, sur l'exemple de la Figure 1, il y a cinq composantes noires.
- un deuxième type de recherche de composante connexe, les pas autorisés étant cette fois Nord, Sud, Ouest, Est, Nord-Ouest, Nord-Est, Sud-Ouest, et Sud-Est. Sur l'exemple de la Figure 1, il n'y a que trois telles composantes noires.

Les fonctions de la bibliothèque devront pouvoir travailler

- directement au niveau des fichiers qui représentent les images.
- au niveau des arbres quadrants.

2 Autres représentations

On peut remarquer que la représentation interne des arbres est parfois inefficace en termes d'espace. Dans l'exemple de la Figure 1, de nombreux sous-arbres portent tous la même information. Par exemple, les quatre premières feuilles sont toutes blanches. Dans ce cas, on peut décider de faire porter l'information directement au niveau du nœud qui supporte ces feuilles. De même, le sous-arbre représentant tout le plan Sud-Est est de couleur uniforme. On le remplace donc par une

feuille. On obtient un arbre qui représente encore la même image, mais qui possède moins de nœuds (toujours avec la même convention de représentation des zones blanches) :

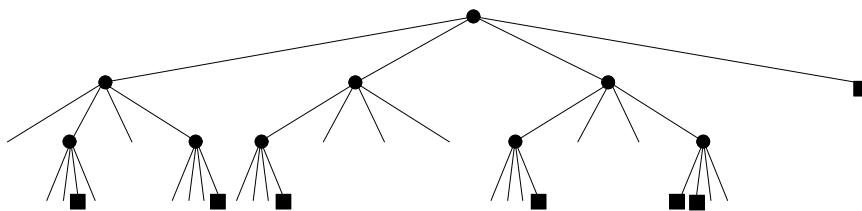


FIG. 3 – *L'arbre de la Figure 1 « allégé »*

Formellement, on remplace tout sous-arbre maximal de couleur uniforme par une feuille de cette couleur. On demande de réécrire la bibliothèque pour qu'elle puisse manipuler de tels arbres.

Enfin, on peut envisager une dernière représentation, que la bibliothèque devra également savoir gérer. On représente maintenant une image par un arbre binaire. Au lieu de la couper en quatre parties, on la coupe d'abord verticalement (premier niveau). Le fils gauche représente la demi-image gauche, le fils droit la demi-image droite. Puis on coupe chaque demi-image horizontalement (deuxième niveau). Le fils gauche représente la demi-image haute, le fils droit la demi-image basse. Puis on recommence à couper verticalement, et ainsi de suite. Par exemple, le quart d'image Sud-Ouest de l'image de la Figure 1 sera représenté par

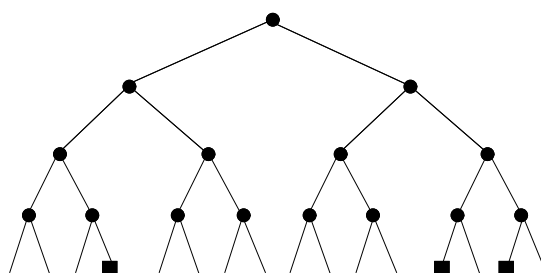


FIG. 4 – *L'arbre binaire associé à la partie Sud-Ouest de l'image de la Figure 1*

On demande de réécrire les fonctions de la bibliothèque pour qu'elles puissent manipuler ces arbres.

3 Remarques

- Développer de petits modules. Les fonctions sont d'autant plus difficiles à corriger qu'elles sont longues.
- Toute constante doit être définie via une directive `#define`. Le projet doit pouvoir être recompilé à la soutenance.
- Dans l'appréciation du projet, les algorithmes employés aussi bien que la programmation seront évalués.
- Le code doit être convenablement commenté. Par contre, le rapport ne doit pas être une paraphrase de ces commentaires. Il doit au contraire donner une vision synthétique du projet, illustrée de préférence par des exemples, des figures et expliquant les structures de données choisies et les algorithmes...

- La taille des images est fixée par l'utilisateur au début de la session de travail. Il ne doit évidemment pas être nécessaire de recompiler le projet pour changer cette taille !
- Il est possible de faire des extensions selon le temps et la motivation restante. Par exemple, on peut demander à agrandir une surface carrée d'aire quatre fois moins grande que l'image originale, mais qui n'est pas un des quatre quarts de plan Nord-Ouest, Nord-Est, Sud-Ouest ou Sud-Est. Une autre suggestion possible est de traiter les arbres binaires allégés.