Concours blanc Informatique 2019

L'année 2019 a été marquée par une annonce phénomènale le mercredi 10 avril : des scientifiques ont dévoilé la première image de l'horizon d'un trou noir en utilisant les données collectées par huit grands télescopes distribués à la surface du globe et grâce à plus d'un pétaoctet de données (un million de gigaoctets...). Cette avancée est déjà un exploit en soi (les trous noirs sont des objets qui ont été prédits depuis le XVIII^{e} siècle mais n'avaient encore jamais été observés « directement »), mais elle montre en outre qu'une telle prouesse n'aurait jamais pu être possible sans le concours de l'informatique.

Le présent sujet va s'attarder sur quelques thèmes inspirés de préoccupations astrophysiques pour illustrer l'ensemble des techniques et algorithmes que vous avez pu croiser cette année. **Les parties sont entièrement indépendantes** et constituées pour la plupart de **questions globalement indépendantes** les unes des autres.

À chaque question, vous avez le droit d'utiliser toute fonction dont la définition est demandée dans les questions précédentes, même si vous n'avez pas réussi à l'exprimer par vous-même.

On prêtera un soin particulier à produire des codes informatiques lisibles et bien structurés où les indentations seront clairement indiquées par des traits verticaux comme dans le code suivant

```
1  def premier_essai(x,n):
2      s = 0
3      for i in range(n):
4          if i != 0:
5              s = s + x**i
6      return s
```

Partie I

Analyse spectrale (23pts)

La lumière est la seule information dont nous disposons pour étudier les objets astronomiques comme les étoiles. Mais si la quantité totale de lumière reçue est déjà une information en soi, la *répartition* de cette lumière selon les différentes longueurs d'onde (c'est-à-dire le spectre de l'étoile) est une notion fondamentale en astronomie car on peut en déduire de multiples choses depuis la température de surface de l'étoile jusqu'à sa composition en terme d'atomes présents.

La figure 1 montre de tels spectres représentant le flux lumineux $\mathcal{F}$ en fonction de la longueur d'onde λ.

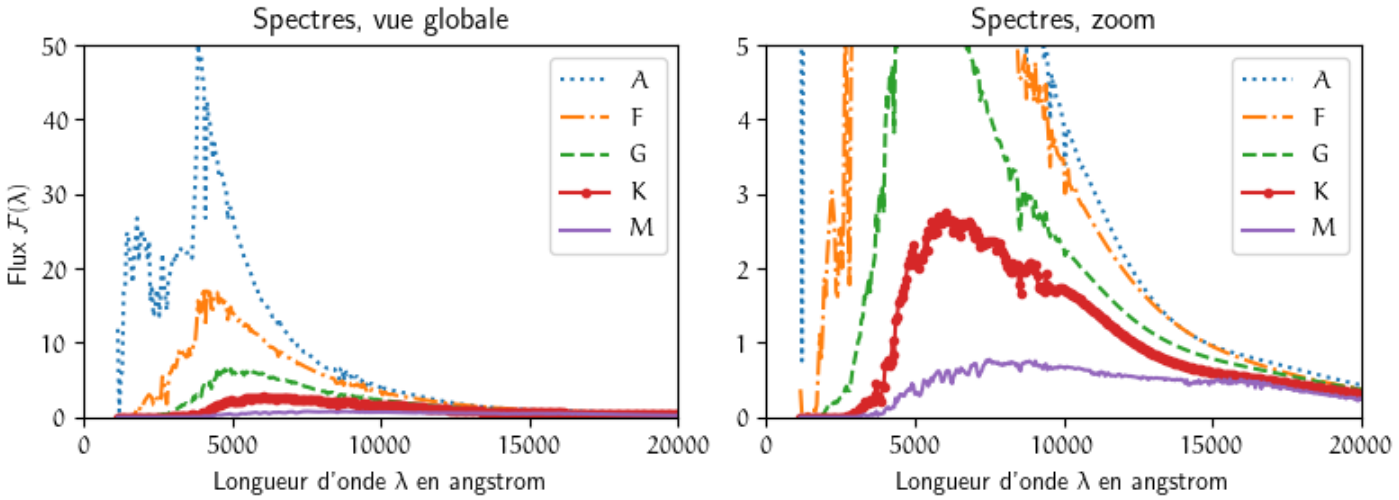

Figure 1 – Quelques spectres d'étoiles de différents types spectraux (A, F, G, K et M). La partie droite est un zoom de la partie gauche pour que l'on puisse avoir une idée de l'évolution du spectre d'une étoile de type M, bien moins lumineuse qu'une étoile de type A.

Les différents spectres sont stockés dans des fichers textes du type suivant :

```
0.120500E+04  0.151689E-03
0.125500E+04  0.180296E-02
0.130500E+04  0.210274E-01
0.141500E+04  0.434747E+00
```

Q1 Écrire une fonction `lecture(fichier)` qui prenne le nom du fichier en argument et renvoie deux listes de flottants contenant respectivement toutes les longueurs d'onde disponibles ainsi que les flux correspondants.

La classification des étoiles fait intervenir des lettres (O, B, A, F, G, K et M) qui sont un reste historique des premières tentatives de classifications avant que l'on ne s'aperçoive que l'on pouvait les classer selon la température de surface de l'étoile. Cette température est directement reliée à la position $\lambda^{(\max)}$ du maximum du spectre que l'on va à présent déterminer.

Q2 Écrire une fonction `lambda_max(LAMBDA,flux)` qui renvoie la valeur de longueur d'onde de la liste `LAMBDA` qui correspond au maximum de la liste `flux`.

La température est reliée à $\lambda^{(\max)}$ par la relation de Wien

$$T = \frac{\sigma_w}{\lambda^{(\max)}} \qquad \text{avec} \qquad \sigma_w \approx 2{,}9.10^{-3} \text{ m.K}$$

Q3 Écrire une fonction `temperature(fichier)` qui, étant donné le nom du fichier texte où sont stockées les données, utilise les fonctions précédentes pour renvoyer la température (approchée) de l'étoile considérée. On n'oubliera pas que les longueurs d'ondes sont stockées en angstroms et que $1 \text{ Å} = 10^{-10}$ m.

Pour obtenir la luminosité Λ_{tot} d'une étoile donnée, il suffit d'intégrer le spectre obtenu précédemment pour toutes les valeurs de longueur d'onde disponibles.

$$\Lambda_{\text{tot}} = \int_{\min(\lambda)}^{\max(\lambda)} \mathcal{F}(\lambda)\,\mathrm{d}\lambda$$

Il va donc falloir calculer cette intégrale à partir d'un échantillonnage (pas forcément régulier) du spectre réel.

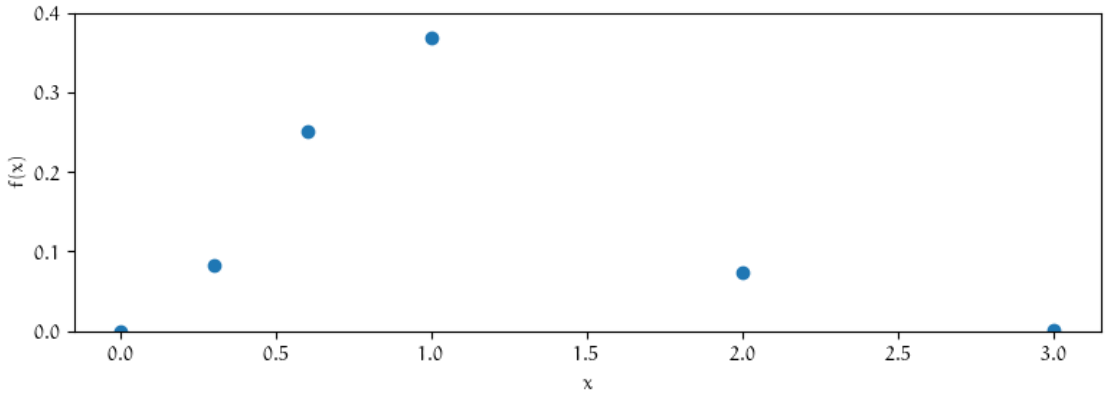

FIGURE 2 – Figure à reproduire (assez) soigneusement sur votre copie pour répondre aux questions 4 et 5 (donc en double exemplaire).

Q4 Après avoir rappelé son principe en recopiant et en complétant la figure 2 sur votre copie, implémenter une fonction dénommée `integration_rectangle(X,FX)` qui, étant donné deux listes `X` et `FX` respectivement de valeurs x_i des abscisses (données dans l'ordre croissant, mais pas forcément régulièrement espacées) et des ordonnées $f(x_i)$ correspondantes, calcule une approximation de l'intégrale $\int_{X[0]}^{X[-1]} f(x)\,\mathrm{d}x$ par méthode des rectangles. On prendra un soin tout particulier à ce que l'algorithme proposé corresponde effectivement au schéma que vous aurez proposé.

Q5 Après avoir rappelé son principe en recopiant et en complétant la figure 2 sur votre copie, implémenter une fonction dénommée `integration_trapeze(X,FX)` qui, étant donné deux listes `X` et `FX` respectivement de valeurs x_i des abscisses (données dans l'ordre croissant, mais pas forcément régulièrement espacées) et des ordonnées $f(x_i)$ correspondantes, calcule une approximation de l'intégrale $\int_{X[0]}^{X[-1]} f(x)\,dx$ par méthode des trapèzes. On prendra un soin tout particulier à ce que l'algorithme proposé corresponde effectivement au schéma que vous aurez proposé.

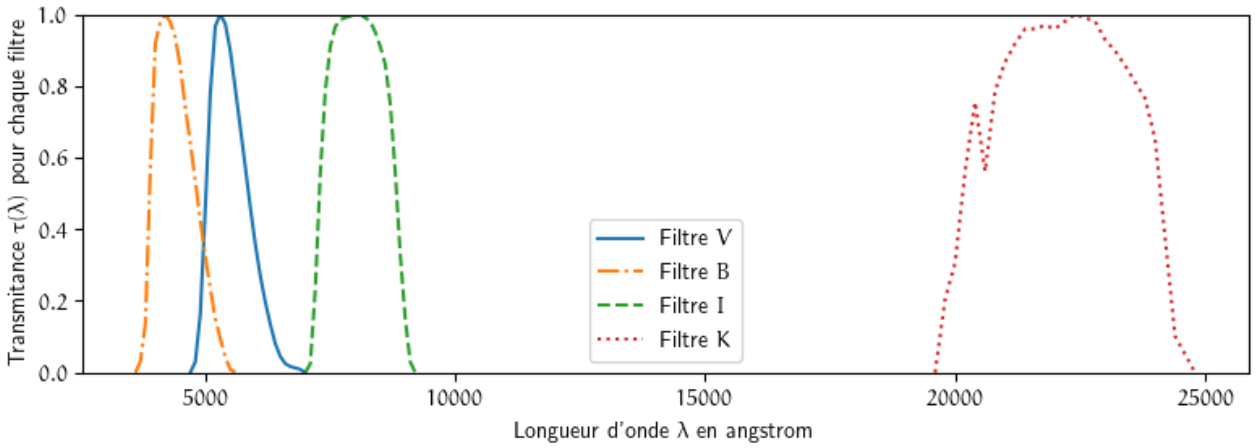

FIGURE 3 – Représentation des fonctions de transmissions pour les différents filtres disponibles

Aucun instrument astronomique n'est capable d'observer une étoile dans toutes les bandes d'observations [1]. Les observateurs ont donc pris l'habitude de travailler avec des filtres qui sélectionnent uniquement une certaine bande de longueur d'onde. La figure 3 donne les fonctions $\tau_f(\lambda)$ de transmission pour chaque filtre f. Une observation dans un filtre f donné va alors renvoyer une valeur Λ_f de luminosité correspondant à l'intégration du flux lumineux de l'étoile modulé par l'application du filtre, soit

$$\Lambda_f = \int_{\min(\lambda)}^{\max(\lambda)} \mathcal{F}(\lambda) \times \tau_f(\lambda)\,d\lambda$$

Tous commes les spectres, les fonctions de transmission sont définies à partir d'un échantillonnage de valeurs de longueurs d'onde λ_i avec les valeurs $\tau_f(\lambda_i)$ correspondantes. En dehors des λ_i d'échantillonnage, la fonction τ_f est considérée comme identiquement nulle. Et, bien évidemment, les personnes qui ont mesuré les spectres et celles qui ont conçu les filtres ne se sont pas mises d'accord sur les valeurs de longueurs d'onde à prendre de sorte que les deux échantillonnages ne coïncident pas...

Q6 Étant donné deux points de mesures de coordonnées (x_i, y_i) et (x_{i+1}, y_{i+1}), écrire l'équation de la droite qui passe par ces deux points et permet de connaître la valeur $y(x)$ de l'ordonnée y pour un x appartenant à l'intervalle $[x_i\,;x_{i+1}]$. On prendra garde de vérifier (succinctement) que $y(x_i) = y_i$ et $y(x_{i+1}) = y_{i+1}$.

Q7 Proposer une procédure `flux_fois_filtre(LAMBDAflux,flux,LAMBDAfiltre,filtre)` qui, étant données les listes `LAMBDAflux` et `flux` échantillonnant le spectre de l'étoile ainsi que les listes `LAMBDAfiltre` et `filtre` échantillonnant la fonction de transmission τ_f du filtre permet de renvoyer une liste contenant le produit $\mathcal{F} \times \tau_f$ pour toutes les valeurs de longueurs d'ondes contenues dans la liste `LAMBDAflux`. Vous pouvez supposer que `LAMBDAflux` est mieux échantillonnée que `LAMBDAfiltre` de sorte qu'entre deux valeurs de longueurs d'ondes de la liste `LAMBDAfiltre`, il y aura toujours au moins une valeur dans la liste `LAMBDAflux`. On fera une interpolation affine (ou, à défaut, l'approximation d'une fonction constante par morceaux) pour évaluer la valeur $\tau_f(\lambda)$ pour tous les λ de l'échantillonnage du flux.

Q8 Proposer alors une procédure `couleur(LAMBDAflux,flux,LAMBDAfiltre,filtre)` qui, étant données les mêmes listes qu'à la question précédente permet de renvoyer l'intégrale Λ_f avec une bonne approximation.

1. Les spectres proposés s'étendent depuis des longueurs d'onde de 100 nm, soit des rayons X, jusqu'à 2 µm, soit de l'infra-rouge lointain.

Partie II

HiPS au CDS (25pts)

Le CDS (Centre de Données astronomiques de Strasbourg) est connu dans le monde entier [2] et s'occupe de rassembler et partager l'ensemble des observations astronomiques disponibles, notamment au travers des grands survey visant à couvrir une grande partie du ciel nocturne. Ils ont développé un logiciel de visualisation appelé Aladin [3] qui, une fois installé sur votre ordinateur, vous permet de voyager sur la carte du ciel au travers de toutes les bases d'images astronomiques disponibles en zoomant plus ou moins selon le degré de précision qui vous intéresse. La technologie derrière cette prouesse technique est basée sur les HiPS [4] (**Hi**erarchical **P**rogressive **S**urveys) qui permettent de découper/réassembler les images disponibles à diverses résolutions de sorte que le chargement des images nécessaires au champ de vue demandé soit raisonnablement rapide. L'idée est similaire à celle des logiciel cartographiques bien connus dont la résolution change à mesure que l'on zoome sur un point donné.

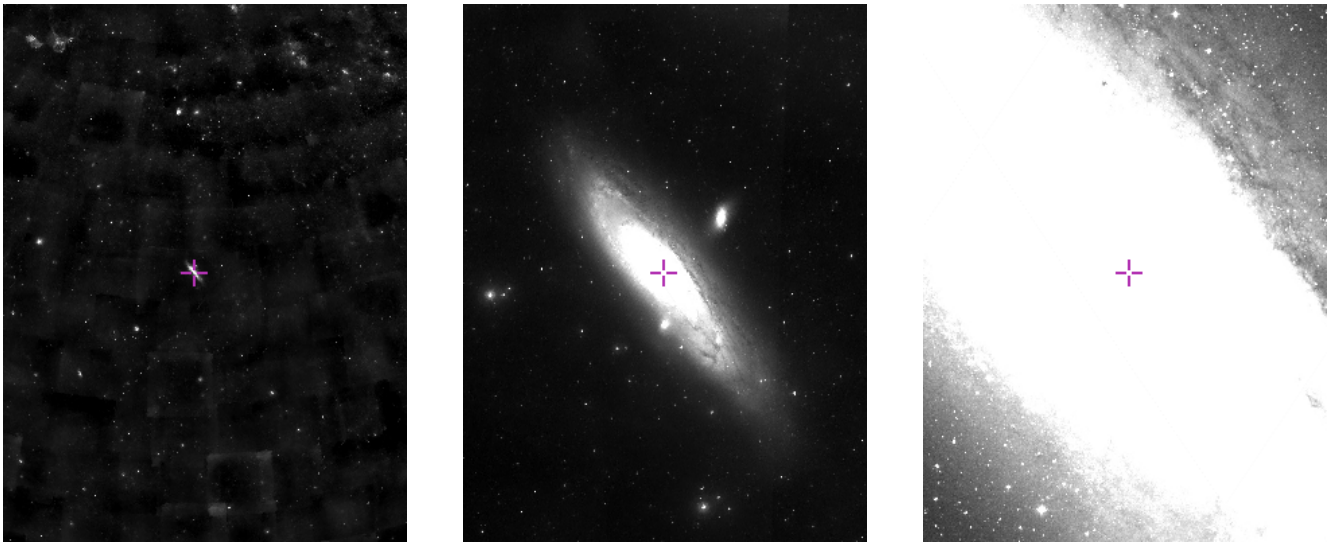

FIGURE 4 – La galaxie M31 vue avec un zoom de plus en plus poussé.

Nous allons étudier ici une version simplifiée du système HEALPix [5] qui permet de découper récursivement la sphère céleste en pixels de taille de plus en plus petite comme indiquée sur les figures 5 et 6

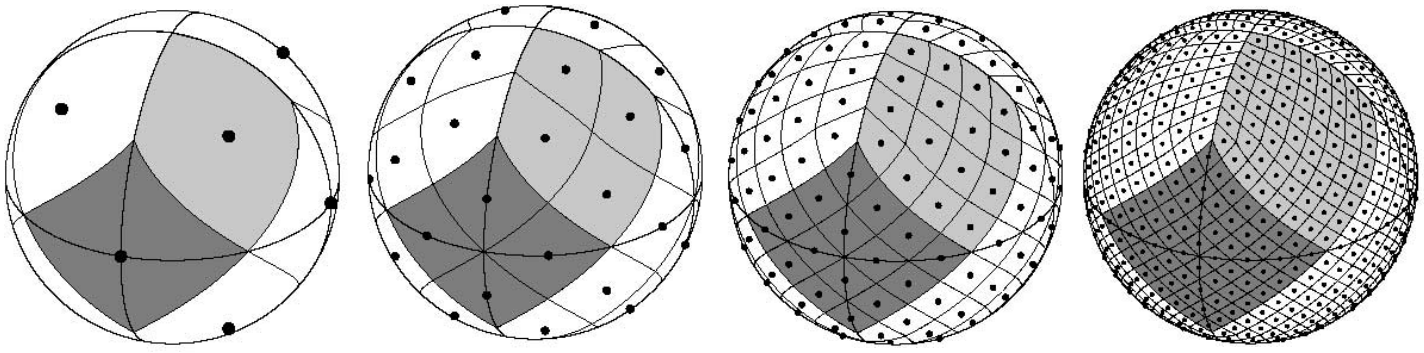

FIGURE 5 – Exemple du découpage de la sphère en 12 zones principales qui sont, à chaque niveau de résolution supplémentaire, subdivisées récursivement en 4 pixels jusqu'à atteindre le niveau de résolution requis. Pour le niveau 0 de résolution (à gauche), on dispose de 12 pixels de base. Pour le niveau 1 (milieu gauche), on a $12 \times 4 = 48$ pixels. Pour le niveau 2 (milieu droite), on a $48 \times 4 = 192$ pixels. Quant au niveau 3 (à droite), on a $192 \times 4 = 768$ pixels. Bien entendu, on peut ontinuer à découper autant que l'on veut en continuant de diviser chaque pixel en 4.

2. Même à Hawaii !
3. Voir `https://aladin.u-strasbg.fr/aladin.gml`
4. Voir `http://aladin.u-strasbg.fr/hips/`
5. Hierarchical Equal Area isoLatitude Pixelisation

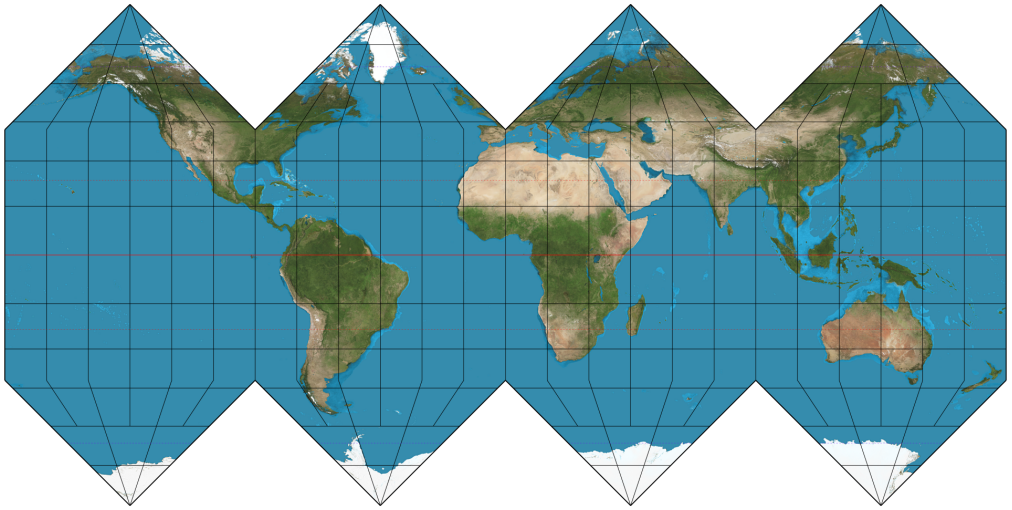

FIGURE 6 – Découpage de la sphère terrestre selon HEALPix en projection sur un plan.

Q9 En supposant que l'on dispose d'images couvrant tout le ciel et dont un pixel (réel) correspond au niveau 20 de subdivision du système HEALPix, estimer la place mémoire occupée par l'image totale du ciel (on supposera que l'information correspondant à un pixel de niveau 20 est codée sur 8 bits). Est-il raisonnable de charger dès le départ l'image totale pour que l'utilisateur puisse zoomer au maximum où bon lui semble ? (on donnera des ordres de grandeur de taille d'image dont l'affichage serait quasi-instantané en consultation d'un site internet)

Pour éviter toute complication due à la géométrie sphérique intrinsèque à tous ces problèmes astronomiques, on va considérer un système voisin de HEALPix permettant de se repérer dans un carré de côté 1, l'axe des x étant horizontal orienté vers la droite et l'axe des y vertical orienté vers le haut (voir figure 7). Le but de la manœuvre va être de trouver rapidement le numéro du pixel correspondant à une position donnée pour un niveau N donné.

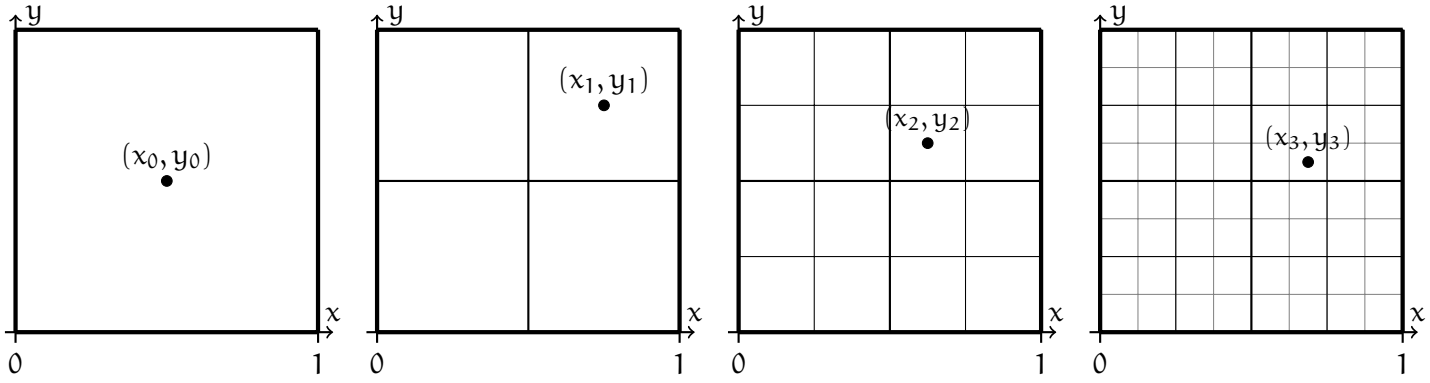

FIGURE 7 – Version simplifiée étudiée dans le reste du devoir sous forme d'un simple carré de côté 1. Les premiers niveaux (de $N = 0$ à gauche à $N = 3$ à droite) ont été représentés. On a aussi mis une suite de centres de pixels consécutifs (voir question Q11 et suivantes)

Q10 Combien y a-t-il de pixels au total au niveau N ?
De combien de bits a-t-on besoin pour numéroter tous ces pixels en base deux ?

Plaçons nous à un niveau de précision $N > 0$ et voyons comment on peut numéroter les pixels en passant au niveau $N+1$ pour que la détection des numéros des pixels voisins soit assez facile. Le procédé est décrit en figure 8.

Q11 Prenons $N > 0$. Notons (x_{N-1}, y_{N-1}) les coordonnées du centre d'un pixel donné au niveau $N-1$ (de côté $1/2^{N-1}$). Notons (a_N, b_N) les deux bits (de valeurs 0 ou 1) définissant la position d'un sous-pixel du pixel précédent au niveau N (le bit a étant associée à la coordonnée x, le bit b à y), comme indiqué en figure 8. Comment peut-on déterminer la position (x_N, y_N) du centre du pixel défini par les bits (a_N, b_N) en fonction du centre du pixel (x_{N-1}, y_{N-1}) parent ? Un exemple de positionnement est fourni en figure 8.

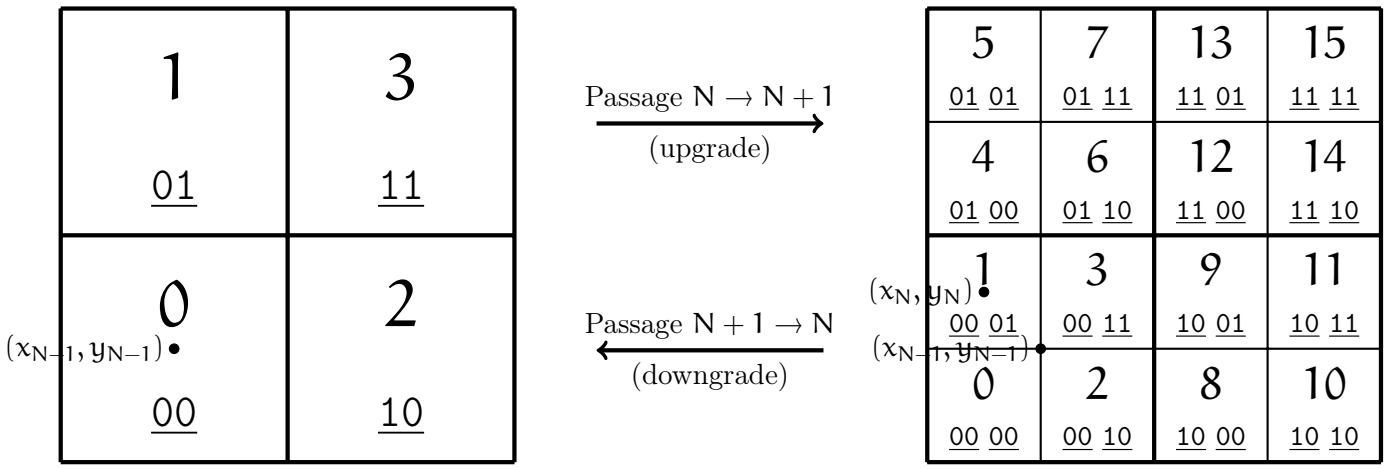

FIGURE 8 – Passage du niveau N au niveau N + 1 avec propagation des identifiants des pixels en fonction de l'identifiant du niveau supérieur. Les écritures en binaire sont soulignées. On remarque que les pixels « enfants » héritent systématiquement leurs (ici deux) premiers bits du pixel parent. Bien entendu, les parents ont eux-aussi des bits hérités de leurs propres parents et grands-parents qui n'ont pas été représentés pour simplifier.

Q12 En déduire que la donnée du numéro du pixel sous forme d'un nombre binaire $a_1b_1a_2b_2\cdots a_{N-1}b_{N-1}a_Nb_N$ est suffisante pour déterminer le centre du pixel correspondant à ce numéro et qu'on a

$$x_N = 0{,}5 - \sum_{i=1}^{N} \frac{(-1)^{a_i}}{2^{i+1}} \qquad \text{et} \qquad y_N = 0{,}5 - \sum_{i=1}^{N} \frac{(-1)^{b_i}}{2^{i+1}}$$

Pour placer correctement la fenêtre de visualisation de l'image, on a plutôt besoin de savoir dans quel zone de niveau N (centrée sur (x_N, y_N) et de côté $1/2^N$) un point (x, y) est inclus.

Q13 Estimer la complexité d'une fonction[6] `cherche_pixel_naif(x,y,N)` qui passe en revue tous les pixels du niveau N et vérifie pour chacun si x est compris dans l'intervalle $[x_N - 1/2^{N+1}; x_N + 1/2^{N+1}]$ et y dans l'intervalle $[y_N - 1/2^{N+1}; y_N + 1/2^{N+1}]$ dans l'idée de renvoyer le numéro (binaire) du pixel gagnant. On précisera bien la complexité de chaque étape nécessaire au calcul[7].

L'approche précédente n'est pas très économe en ressources de calcul et il vaut mieux lui préférer une descente dans l'arbre des niveaux réalisée dans la fonction suivante.

```
1  def cherche_pixel(x,y,N):
2      """ Recherche de la liste des bits du pixel contenant (x,y) au niveau N. """
3      numero = []
4      xi,yi = 0.5,0.5  # Initialisation du centre du pixel courant
5      for i in range(1,N+1):
6          # Calcul des bits pour le prochain niveau
7          ai = int(x > xi)
8          bi = int(y > yi)
9          # Stockage du résultats
10         numero.append(ai)              # D'abord pour x
11         numero.append(bi)              # ensuite pour y
12         # Calcul du nouveau centre
13         xi = xi - (-1)**ai / 2**(i+1)
14         yi = yi - (-1)**ai / 2**(i+1)
15     return numero
```

Q14 Expliquer le principe caché derrière les lignes 7 et 8.
(pour ce faire, il peut être utile de savoir que `int(True)` renvoie `1` alors que `int(False)` renvoie `0`)

6. On ne demande pas de l'écrire, juste de décrire ce qu'il est nécessaire qu'elle calcule.
7. On pourra supposer que les prises de puissances sont des opérations élémentaires.

Q15 Estimer (en la justifiant) la complexité de cet algorithme. Est-ce bien meilleur que l'algorithme précédent ?

Q16 Une erreur de copier/coller [8] s'est glissée dans l'algorithme de la page précédente, saurez-vous la trouver ?

Pour éviter l'erreur de copier/coller précédente, il est plus sage d'écrire une fonction qui évite au maximum les redites. Cela pourrait prendre la forme suivante.

```
def liste_bits(z,N):
    Z = 0.5
    L = []
    for i in range(1,N+1):
        b = int(z > Z)
        L.append(b)
        Z = Z - (-1)**b / 2**(i+1)
    return L

def cherche_pixel(x,y,N):
    aListe = liste_bits(x,N)
    bListe = liste_bits(y,N)
    numero = []
    for i in range(len(aListe)):
        numero.append(aListe[i])
        numero.append(bListe[i])
    return numero
```

Q17 Prouver que la fonction `liste_bits(z,N)` renvoie une listes de bits $(b_k)_{k\in[\![1;N]\!]}$ telle que

$$z \in \left[Z_N - \frac{1}{2^{N+1}} ; Z_N + \frac{1}{2^{N+1}} \right] \qquad \text{où} \qquad Z_N = \frac{1}{2} - \sum_{k=1}^{N} \frac{(-1)^{b_k}}{2^{k+1}}$$

Pour ce faire, on pourra utiliser l'invariant $\mathcal{P}(i)$ suivant :

$\mathcal{P}(i)$: « Après l'étape i, la liste des $(b_k)_{k\in[\![1;i]\!]}$ est telle que

$$z \in \left[Z_i - \frac{1}{2^{i+1}} ; Z_i + \frac{1}{2^{i+1}} \right] \qquad \text{où} \qquad Z_i = \frac{1}{2} - \sum_{k=1}^{i} \frac{(-1)^{b_k}}{2^{k+1}} \text{ »}$$

On fera clairement apparaître les différentes étapes de la preuve.

Q18 À quel algorithme classique étudié pendant l'année une telle recherche peut-elle s'apparenter ?

Pour compléter l'image correspondant à la position trouvée il est souvent utile de pouvoir rapidement accéder aux plus proches voisins. On va se concentrer ici sur les 4 voisins direct (Nord, Sud, Est et Ouest) d'un pixel donné dont on connaît la suite des $(a_k)_{k\in[\![1;N]\!]}$ et des $(b_k)_{k\in[\![1;N]\!]}$.

Q19 En supposant que le pixel ne se trouve pas sur un bord [9], donner des instructions relativement simples (en langage courant) permettant de déterminer les suites des $(a_k)_{k\in[\![1;N]\!]}$ et des $(b_k)_{k\in[\![1;N]\!]}$ des points voisins.

8. C'est mal de faire des copier/coller !
9. Ce qui revient à dire que tous les a_k sont égaux entre eux ou tous les b_k sont égaux entre eux.

Partie III

Astronomical Data Query Language (13pts)

Les données astronomiques occupant de plus en plus de place, les astronomes se sont naturellement tournés vers les bases de données et les langages associés pour les gérer. Néanmoins, comme le SQL de base ne semblait pas leur suffire, ils l'ont étendu en un langage nommé ADQL (Astronomical Data Query Language) qui n'est rien d'autre que du SQL auquel on a rajouté quelques fonctions géométriques souvent utiles dans le domaine [10]. Par exemple,

— Un nouveau type `point` est défini pour repérer un point dans le ciel. On peut créer un tel objet en appellant la fonction `POINT('',ra,dec)` et en lui fournissant l'ascension droite (`ra` pour « Right Ascension ») et la déclinaison (`dec`) qui correspondent aux coordonnées polaires (exprimées en degrés) pour repérer une position sur le ciel. Le premier argument doit rester une chaîne de caractères vide pour prendre le système de coordonnées par défaut.

— On dispose d'une fonction `DISTANCE(p1,p2)` qui calcule la distance (en degrés) entre les deux points `p1` et `p2` (tous les deux du type `point` défini au-dessus).

On va s'intéresser ici à la base de données de SIMBAD [11] dont la structure (simplifiée [12] !) est la suivante :

```
BASIC      (OID:int, MAIN_ID:string, COORD:point, OTYPE_TXT:string)
HAS_REF    (OIDBIBREF:int, OIDREF:int)
REF        (OIDBIB:int, TITLE: string, YEAR:int, JOURNAL:string)
ALLFLUXES  (OIDREF:int, B:float, V:float)
```

Quelques remarques utiles :

— Les attributs `OID` et `OIDBIB` représentent les numéros d'identifications (uniques) respectivement de l'objet considéré et de la référence bibliographique considérée. `OIDREF` et `OIDBIBREF` permettent d'y faire référence depuis une autre table.

— `MAIN_ID` contient le code d'identification « usuel » de l'objet (comme `'M31'` ou `'NGC4039'` par exemple).

— `OTYPE_TXT` permet de faire la différence entre les étoiles (représentées par `'*'`) et les galaxies (`'G'`).

— `B` et `V` contiennent les magnitudes [13] $\mathcal{M}_B$ et $\mathcal{M}_V$ pour l'objet considéré.

— La table `REF` contient les articles scientifiques qui font référence à un objet astronomique, liaison faite à l'aide de la table `HAS_REF`.

Q20 Écrire une requête ADQL qui renvoie les coordonnées de la galaxie M31.

Q21 Écrire une requête ADQL qui récupère la magnitude en V ainsi que la couleur $B-V$ (soit $\mathcal{M}_B-\mathcal{M}_V$) pour les étoiles [14] de la base.

La requête précédente permet de faire un diagramme de Hertzsprung-Russel, mais cela n'a de sens que si les étoiles considérées sont toutes à peu près au même endroit dans l'espace.

Q22 Écrire une requête ADQL récupère la magnitude en V ainsi que la couleur $B-V$ (soit $\mathcal{M}_B-\mathcal{M}_V$) pour les étoiles situées à moins d'un dixième de degré de distance de l'amas M22 (d'ascension droite 279,10° et de déclinaison −23,905°)

On s'intéresse à présent aux articles scientifiques qui concernent des objets en particulier pour savoir lesquels étaient « à la mode » par exemple en l'an 2000

Q23 Classer en ordre décroissant les objets astronomiques qui cumulent le plus d'articles de la revue MNRAS (Monthly Notice of the Royal Astronomical Society) pour l'année 2000. On donnera le nom de l'objet et le nombre d'articles associés.

Q24 Et pour finir la question bonus culturel pour ceux qui ont lu jusque là : quel est le nom usuel [15] de la galaxie M31 qui est présentée en figure 4 ?

10. Voir Osuna, Pedro, et al. "IVOA Astronomical Data Query Language Version 2.00." IVOA Recommendation 30 oct 2008.
11. Voir `http://simbad.u-strasbg.fr/simbad/sim-tap` pour essayer les requêtes.
12. Voir `http://simbad.u-strasbg.fr/simbad/tap/tapsearch.html` pour la structure complète.
13. La magnitude $\mathcal{M}$ dans une bande donnée est reliée à la luminosité Λ dans cette bande par la formule $\mathcal{M} = -2{,}5\log(\Lambda)$.
14. Rappel : tous les objets de la base ne sont pas des étoiles mais celles-ci peuvent être identifiées grâce à l'attribut `OTYPE_TXT`.
15. Indice : il s'agit de la plus grande des plus proches voisines de la Voie Lactée.