TFJM²

Problème 4 : Musique déformée

Avant-propos : compréhension du problème

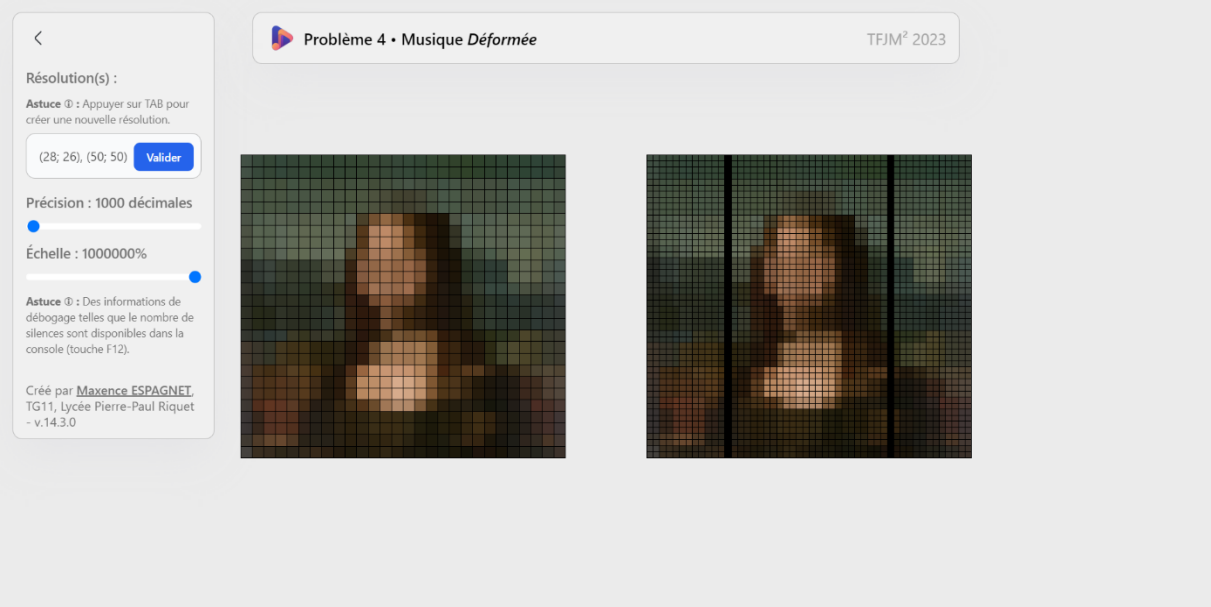
Dans le cadre de notre recherche sur le problème nous nous sommes placés sur un axe de longueur 1 et d'origine 0 le long duquel on place les n notes du fichier initial. Chaque note aura une longueur de 1/n (à mettre en forme comme le 1/m) et l'extrémité de la q-ième note du fichier sera $\frac{q}{n}$; $q \in [\![1; n]\!]$. On fait de même pour le fichier suivant de résolution m, chaque note aura cette fois une longueur de $\frac{1}{m}$ et l'extrémité de la k-ième note du fichier sera $\frac{k}{m}$; $k \in [\![1; m]\!]$. De cette définition il en découle que le milieu de la note de rang k sera $\frac{\frac{k-1}{m}+\frac{k}{m}}{2} = \frac{2k-1}{2m}$.

Pour la suite on appellera *conversion*, tout changement d'une résolution n vers une résolution m, cette dernière sera appelée résolution de conversion. Pour plusieurs conversions à la suite, la résolution de conversion au tour i sera notée m_i.

Avant-propos : visualisation du problème à l'aide d'un programme informatique

L'identification des propriétés démontrées dans ce document a été rendue possible par un algorithme spécialement développé et conçu pour ce problème. Il est accessible en libre-service via ce lien : https://cutt.ly/R7bEZTq

Capture d'écran de l'interface du programme informatique pour la manipulation de bandes son. Valeurs prises pour l'exemple : $n = 4$ et $m = 7$.



Capture d'écran de l'interface du programme informatique pour la manipulation d'images. Valeurs prises pour l'exemple : $n = 28$; $n' = 26$ et $m = 50$; $m' = 50$.

Le cœur du programme informatique réside dans les quelques lignes de code qui vont suivre :

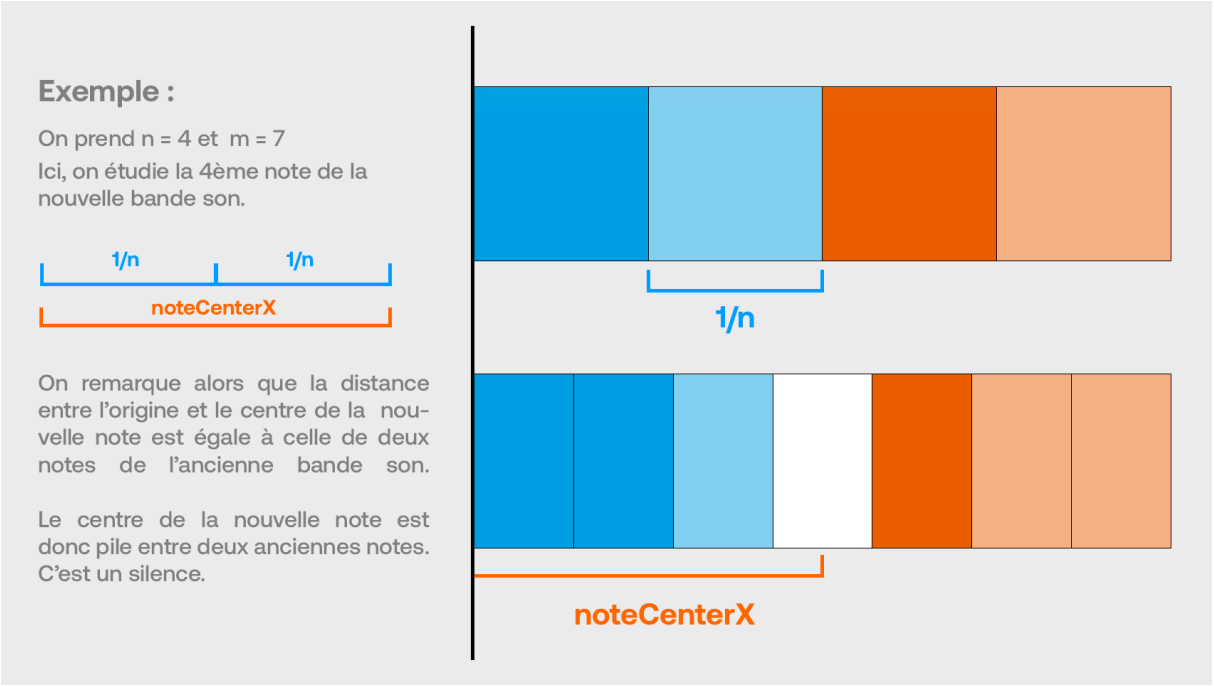
a) *Déterminer l'apparition d'un silence*

Problème 4 - Questions 1 à 5

```
// The position of the note
const noteX = noteWidth.times(i);

const halfNoteWidth = noteWidth.dividedBy(2);
const noteCenterX = noteX.add(halfNoteWidth);

// Determine if the note falls between two previous notes
const modulo = noteCenterX.mod(previousNoteWidth);
const isModuloNull =
  modulo.toNumber() === 0 ||
  // Fix an issue where the rest of the modulo should be 0
  // It is caused by float numbers inaccuracy
  modulo.toNumber() === previousNoteWidth.toNumber();

// Determine the index of the note color
const colorIndex = noteCenterX
  .dividedToIntegerBy(previousNoteWidth)
  .toNumber();
```

Ici, on utilise des coordonnées pour déterminer les silences. Pour chaque note de chaque bande son, le programme détermine d'abord si la note est un silence à l'aide d'un modulo. Ainsi, on note $\frac{1}{n}$ ou $previousNoteWidth$ la taille d'une note de la bande son précédente et $noteCenterX$ la distance entre l'origine et le centre de la note à générer :

De fait, si la note est un silence, on a $noteCenterX \equiv 0\ [previousNoteWidth]$, avec n le nombre de notes de la bande son précédente.



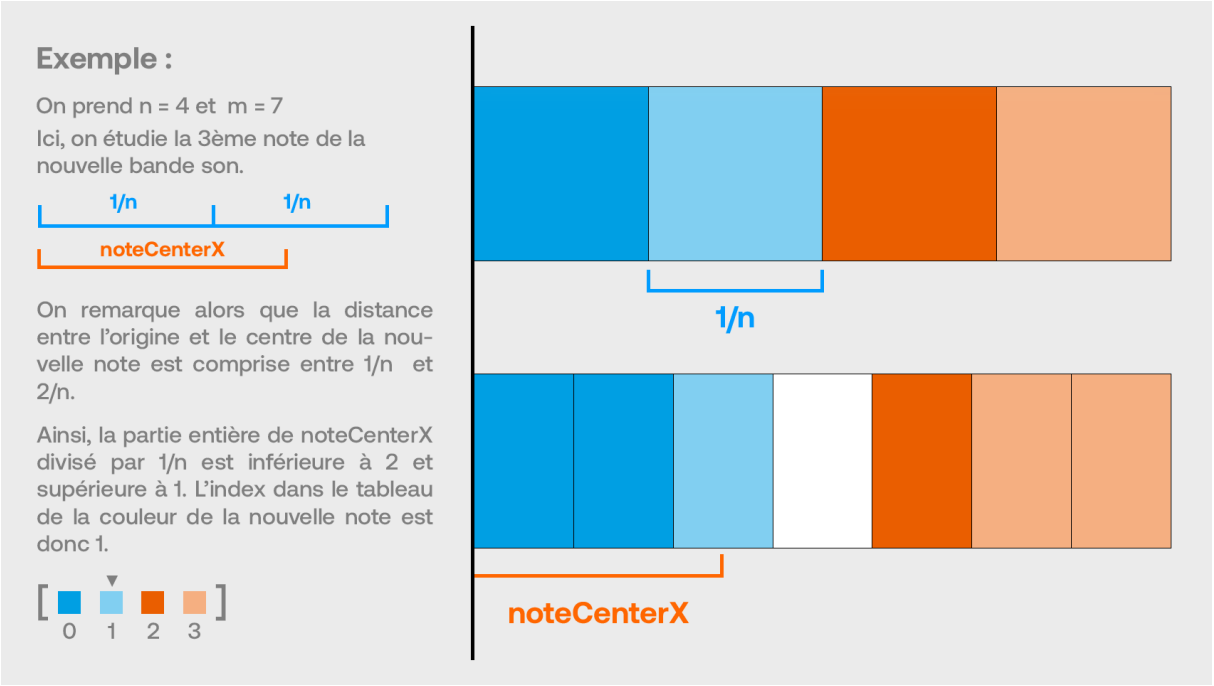
b) Déterminer la couleur d'une note

Problème 4 - Questions 1 à 5

```
// If the note center falls between two previous notes, it is deleted
and colored white.
// Otherwise, its color is based on the color of the previous note
that it falls within.
const noteColor = isModuloNull
  ? new Color(0xffffff)
  : colors[colorIndex];
```

La couleur de chaque note de la bande son précédente est stockée dans un tableau nommé $colors$. Pour déterminer la couleur de la nouvelle note, il suffit donc de calculer la partie entière de la division euclidienne de $noteCenterX$ par $previousNoteWidth$, soit $\left\lfloor \frac{noteCenterX}{previousNoteWidth} \right\rfloor$, ce qui va nous donner l'index dans le tableau de la couleur recherchée.

On répète l'opération pour chaque note de la nouvelle bande son, soit m fois. Il est cependant impératif de stocker les couleurs ainsi générées dans un nouveau tableau afin de pouvoir répéter l'opération.



Démonstrations préalables :

En traduisant l'énoncé, "Pour trouver la k-ième note jouée, le logiciel regarde l'intervalle de temps pendant lequel elle doit être jouée dans le nouveau fichier, prend son milieu et choisit la note qui était jouée à cet instant dans l'ancien fichier. Si ce milieu tombe pile entre deux notes dans l'ancien fichier, le logiciel met un silence", il vient la propriété :

Un silence apparait si et seulement si $\frac{2k-1}{2m} = \frac{q}{n} \Leftrightarrow \frac{m}{n} = \frac{2k-1}{2q}$

De cette égalité découlent de nombreuses propriétés différentes, que voici :

1er cas, n est un entier impair :

$\frac{2k-1}{2m} = \frac{q}{n} \Leftrightarrow (2k-1)n = 2mq$ Or cette égalité n'est jamais vérifiée, car $\forall\ (q;m) \in \mathbb{N}^2, 2qm$ sera un entier pair tandis que $\forall k \in \mathbb{N}\ et\ \forall n\ \in 2\mathbb{N}-1,\ \ (2k-1)n$ sera un entier impair. Nous pouvons en conclure que

(1) :« Si la résolution initiale n, est un entier impair, alors, aucun silence n'apparaîtra lors de la conversion pour une résolution m, quelle qu'elle soit »

On note cette propriété (1)

2ème cas, quand n est un entier pair :

- Soit m est un multiple de n, alors la fraction $\frac{m}{n}$ est un entier naturel, or $\forall (q;k) \in [\![1;n]\!] \times [\![1;m]\!]$; $\frac{2k-1}{2q}$ est irréductible et différent d'un entier. On en déduit que $\frac{m}{n} \neq \frac{2k-1}{2q}$, d'où la propriété qui suit :

(2): « Si la résolution de conversion m est un multiple de la résolution initiale n, alors aucun silence n'apparaît lors de la conversion.»

On note cette propriété (2)

- Soit $m \wedge n = 1$, alors $\frac{m}{n}$ est irréductible et m est un entier impair. Et nécessairement m = 2k-1 et n=2q. On souhaite démontrer qu'il n'existe qu'un couple $(k; q) \in [\![1; m]\!] \times [\![1; n]\!]$ qui vérifie les égalités précédentes.
  On pose les fonctions f(x)=2x-1 et g(x)= 2x.
  La fonction f(x) est une fonction affine strictement croissante et dérivable sur $\mathbb{R}$ donc elle est continue et monotone sur $[1; m]$, f(1)=1 et f(m)=2m-1 avec $m \geq 1$ donc f([1;m])= $[1; 2m-1]$. D'après le TVIC il existe une unique solution sur $[1; m]$ telle que f(x) = m, on note cette solution α
  $f(\alpha) = m \iff 2\alpha - 1 = m \iff \alpha = \frac{m+1}{2}$, m+1 est un entier pair puisque m est impair donc α est un entier.
  Il n'existe qu'une solution sur $[1; m]$ telle que f(x) = m notée α et α est un entier, donc il n'existe qu'une solution sur $[\![1; m]\!]$ telle que f(x)=m. Donc une seule valeur de k vérifiant l'égalité m = 2k-1.

  Même raisonnement pour g(x). Donc il n'existe qu'une seule valeur de q vérifiant l'égalité n=2q.

  Enfin nous pouvons conclure qu'il n'existe qu'un couple $(k; q) \in [\![1; m]\!] \times [\![1; n]\!]$ qui vérifie les égalités m=2k-1 et n =2q.

  De plus, le milieu de la note centrale du fichier post-conversion pourra s'écrire $\frac{\left(\frac{\lfloor \frac{m}{2} \rfloor}{2}\right)+\left(\frac{\lceil \frac{m}{2} \rceil}{2}\right)}{m} = \frac{\lfloor \frac{m}{2} \rfloor + \lceil \frac{m}{2} \rceil}{2m} = \frac{\frac{m-1}{2}+\frac{m+1}{2}}{2m} = \frac{m}{2m} = \frac{1}{2}$, or, nous avons défini n comme étant pair donc il existe nécessairement une note de rang q dans le fichier initial, telle que $\frac{q}{n} = \frac{1}{2}$ donc la note centrale sera toujours un silence si n est pair et m impair.
  De fait, on obtient la propriété suivante

(3): « Si la résolution initiale n et la résolution suivante m sont premières entre elles, avec n pair, alors un seul silence apparaît. Ce silence est la note centrale du fichier post-conversion, soit la note de rang $\left\lceil \frac{m}{2} \right\rceil$ »

On note cette propriété (3).

- Soit m est un diviseur de n tel que $m = \frac{n}{2^p}; p \in \mathbb{N}^*$, dans ce cas $\frac{m}{n} = \frac{1}{2^p}$ donc d'après (3) il apparaît un silence. Mais, $n = m \times 2^p$ donc il y a dans le fichier initial m partitions de tailles 2^p, ce qui simule m conversions réduites et identiques. Il y a donc $m \times 1$ silences non confondus qui apparaissent lors de la conversion (voir figure 1). Il vient la propriété suivante :

(4) : « Si la résolution de conversion m, est de la forme $\frac{n}{2^p}$; $p \in \mathbb{N}^*$, le fichier post-conversion sera uniquement composé de silences »

On note cette propriété (4)

- Soit dans ce cas, la fraction $\frac{m}{n}$ est réductible par PGCD(m ; n)

  Deux cas apparaissent alors :

  1. Soit $\frac{n}{PGCD(m;n)}$ est impair. Lorsqu'on simplifie $\frac{m}{n}$ par $PGCD(m;n)$, cela revient à décomposer les deux fichiers en $PGCD(m;n)$ parties égales, ce qui simule $PGCD(m;n)$ conversions réduites et identiques. On retrouve pour chacune de ces conversions (comme $\frac{n}{PGCD(m;n)}$ est impair) les conditions pour appliquer la propriété (1). Donc pour chacune de ces conversions réduites aucun silence n'apparaît, ce qui signifie qu'aucun silence n'apparaît dans le fichier final.

  2. Soit $\frac{n}{PGCD(m;n)}$ est pair. Lorsqu'on simplifie $\frac{m}{n}$ par $PGCD(m;n)$, cela revient à décomposer le fichier en $PGCD(m;n)$ parties égales, ce qui simule $PGCD(m;n)$ conversions réduites et identiques dans ce cas, d'après (3), il y aura 1 silence qui va apparaître pour chacune des conversions réduites. Il y a alors $PGCD(m;n) \times 1$ silences qui vont apparaître lors de la conversion, chacun espacé de $\frac{m}{PGCD(m;n)}$ notes avec le suivant/précédent (voir figure 3). Il vient la propriété suivante :

(5) : « Lorsque m et n ne sont pas premiers entre eux et m n'est pas un multiple de n, alors

pour $\frac{n}{PGCD(m;n)}$ impair, aucun silence n'apparaît lors de la conversion et

pour $\frac{n}{PGCD(m;n)}$ pair, il y aura PGCD (m ;n) silences qui apparaîtront lors de la conversion. Le rang du j-ième silence, quand il existe, sera alors

$\left[\frac{m}{2PGCD(m;n)}\right] + \frac{(j-1)m}{PGCD(m;n)}$; $j \in [\![1; PGCD(m;n)]\!]$. *»*

On note cette propriété (5).

Figures

Figure 1.

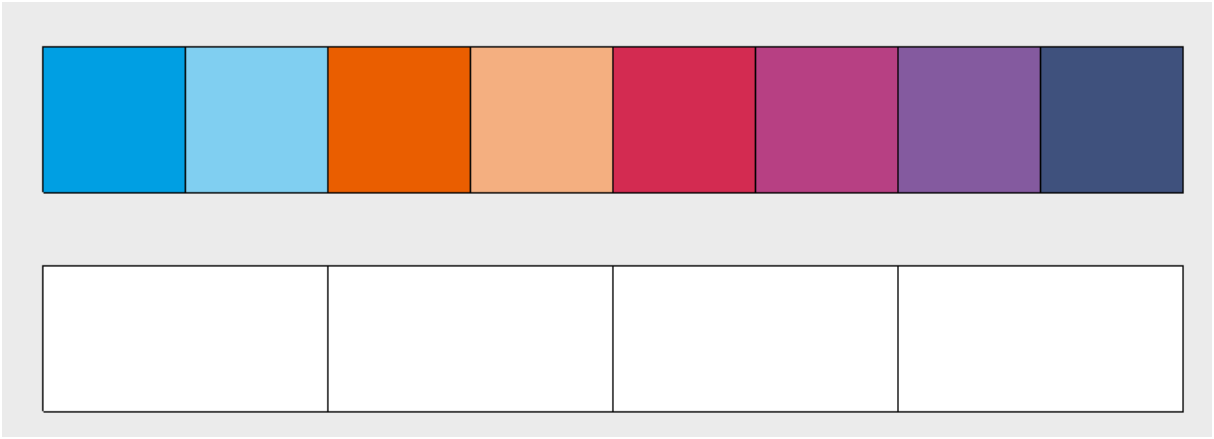
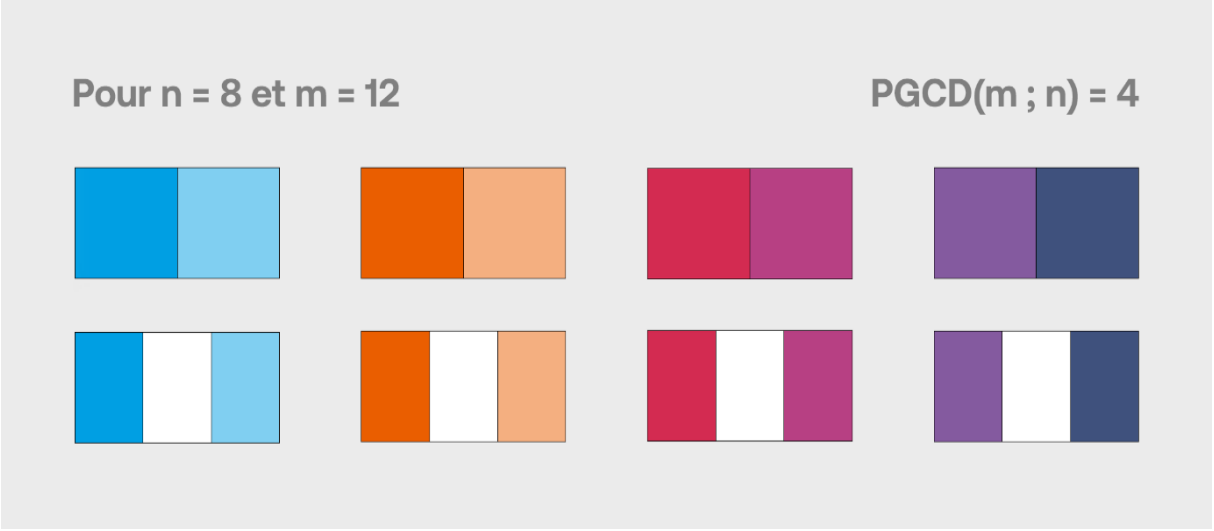


Figure 2.

Figure 3.

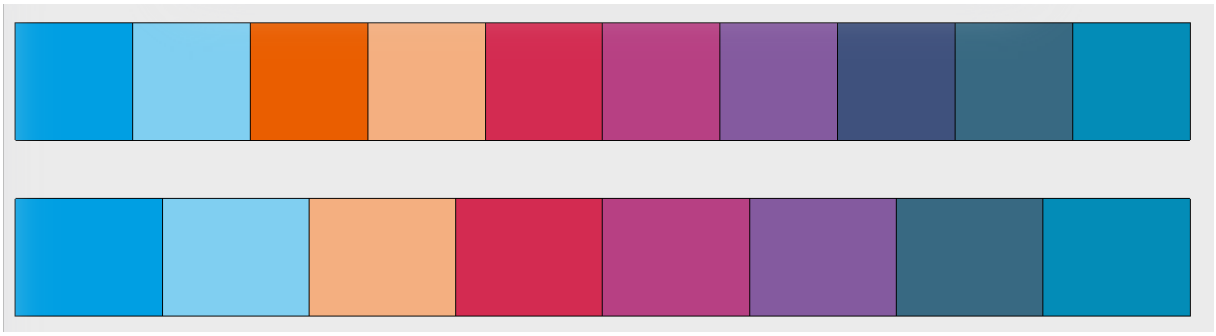


1) L'évènement peut-il arriver après un nombre fini d'opérations ?

a) Une des notes présentes dans le fichier initial disparait

Il est possible de faire disparaître une note, lorsque par exemple la résolution de conversion m est inférieure à la résolution initiale n.

Exemple : pour n = 10 et m = 8

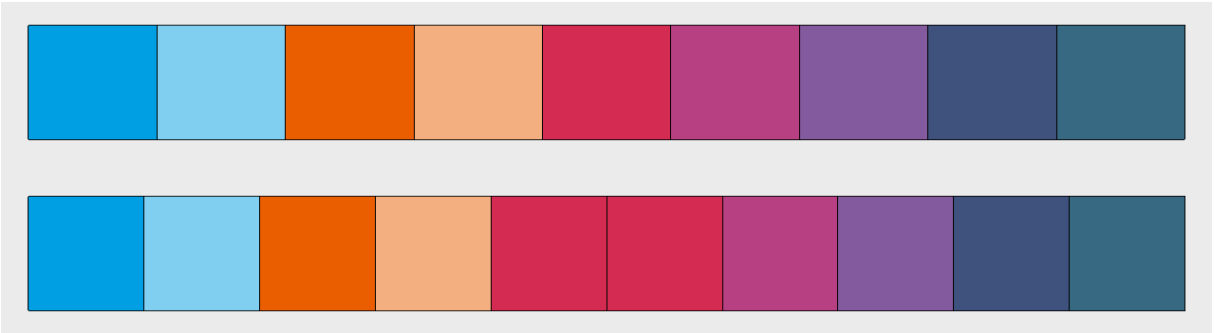


b) Les notes ne sont plus deux à deux distinctes

Plusieurs cas possibles peuvent se produire :

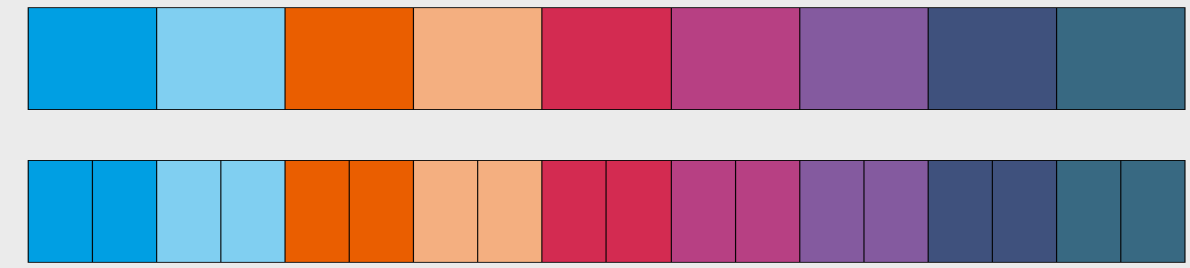
D'après (1), si la résolution initiale n est impaire, il suffit que m>n. En effet, dans ce cas de figure, aucun silence ne va apparaître lors de la conversion, il y aura au moins une même note qui apparaîtra plus d'une fois. Les notes ne seront alors plus deux à deux distinctes.

Exemple : pour n = 9 et m = 10



Autre cas possible, si la résolution de conversion m est un multiple de la résolution initiale n telle que m > n, alors d'après (1) et (2) (suivant la parité de n), aucun silence ne va apparaître et chaque note apparaîtra $\frac{m}{n}$ fois. Les notes ne seront alors plus deux à deux distinctes.

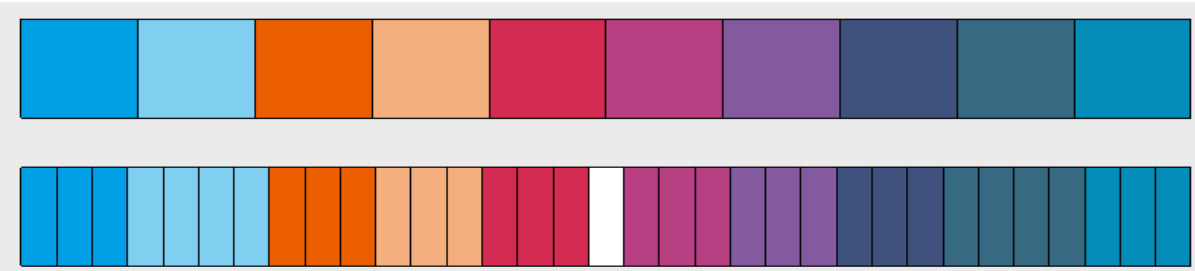
Exemple : pour n = 9 et m = 18



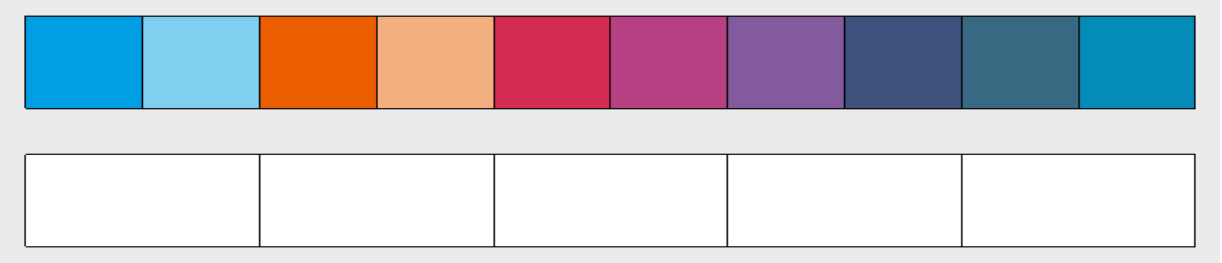
c) Un silence apparaît

Il est possible qu'un silence apparaisse. Les propriétés (3), (4) et (5) reprennent plusieurs cas réalisables.

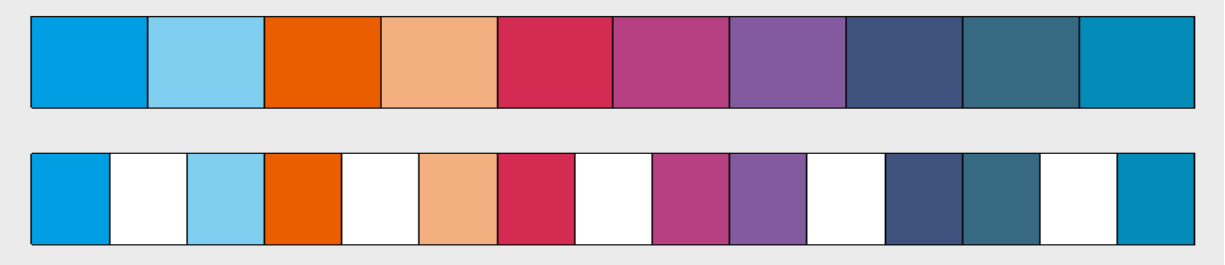
Exemple pour la (3) : avec n = 10 et m = 33



Exemple pour la (4) : avec n = 10 et m = 5



Exemple pour la (5) : avec n = 10 et m = 15

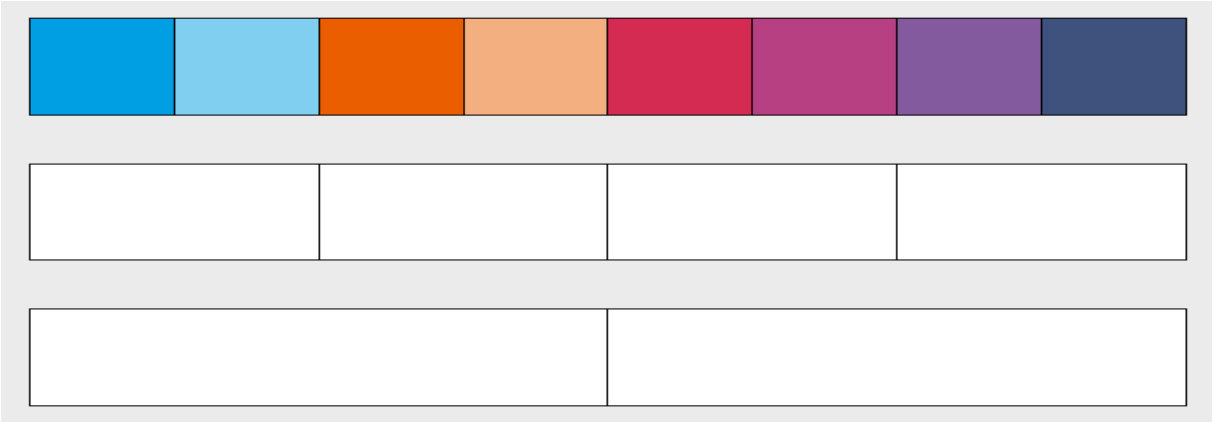


d) Un silence disparaît

On considère qu'un silence disparaît lorsqu'il y a moins de silences dans le fichier final que dans le fichier initial. Pour qu'un silence puisse disparaître il faut avant tout qu'il apparaisse car il ne peut y avoir de silences dans le fichier initial. Donc cela nécessite au moins deux tours. Or nous avons listé les différents cas pour qu'il apparaisse dans les propriétés.

Dans le cas de la propriété (4), toutes les notes du fichier à la suite de la première conversion seront des silences. Ainsi, comme à la question 1.a), pour faire disparaître un silence il faudra que $m_2 < m_1$.

Exemple : avec n = 8, $m_1 = 4$ et $m_2 = 2$



Dans les cas illustrés par les propriétés (3) et (5), il sera toujours possible de faire disparaître les silences lorsque pour $m_1 > n$ on a $m_2 = n$. (Pour bien pouvoir appliquer les propriétés (3) et (5) il va de soi que m_1 ne doit pas être un multiple de n). Lorsque l'on fait apparaître un silence, par définition, le milieu d'une note du fichier post-conversion tombe sur l'extrémité d'une note du fichier précédent. Ce qui signifie que pour tout silence de rang k (dans le deuxième fichier, donc celui de résolution m_1) il existe une note de rang q dans le fichier initial, et par extension dans le fichier final car m_2 = n, telle que $\frac{2k-1}{2m_1} = \frac{q}{n} = \frac{q}{m_2}$.

De plus, si $m_1 > n$ alors $m_1 > m_2 \Leftrightarrow \frac{1}{m_1} < \frac{1}{m_2}$ ce qui signifie que la longueur d'une note dans le fichier final sera plus grande que celle d'une note dans le fichier de résolution m_1. Ainsi, pour chaque silence dans le deuxième fichier, il y aura au mieux deux notes du fichier final qui pourraient devenir des silences. Ces notes sont celles de rang q et q+1 lorsque $\frac{2k-1}{2m} = \frac{q}{m_2}$ (ce qui est notre cas). Pour que les silences disparaissent il faut montrer que les milieux des notes de rang q et q+1 sont à l'extérieur de la longueur du silence, pour ne pas le reproduire. Ce qui se traduit par

$$\begin{cases} \frac{q}{m_2} - \frac{1}{2m_2} < \frac{2k-1}{2m_1} - \frac{1}{2m_1} \\ \frac{q}{m_2} + \frac{1}{2m_2} > \frac{2k-1}{2m_1} + \frac{1}{2m_1} \end{cases} \Leftrightarrow \begin{cases} -\frac{1}{2m_2} < -\frac{1}{2m_1} \\ \frac{1}{2m_2} > \frac{1}{2m_1} \end{cases} \quad \text{car } \frac{2k-1}{2m_1} = \frac{q}{n} = \frac{q}{m_2}$$

$$\Leftrightarrow \begin{cases} \frac{1}{m_2} > \frac{1}{m_1} \\ \frac{1}{m_2} > \frac{1}{m_1} \end{cases}$$ ce qui est vrai dans notre cas.

Donc les silences présents dans le fichier de résolution m_1, disparaissent dans le fichier de résolution m_2 lorsque $m_2 = n$ et $m_1 > n$.

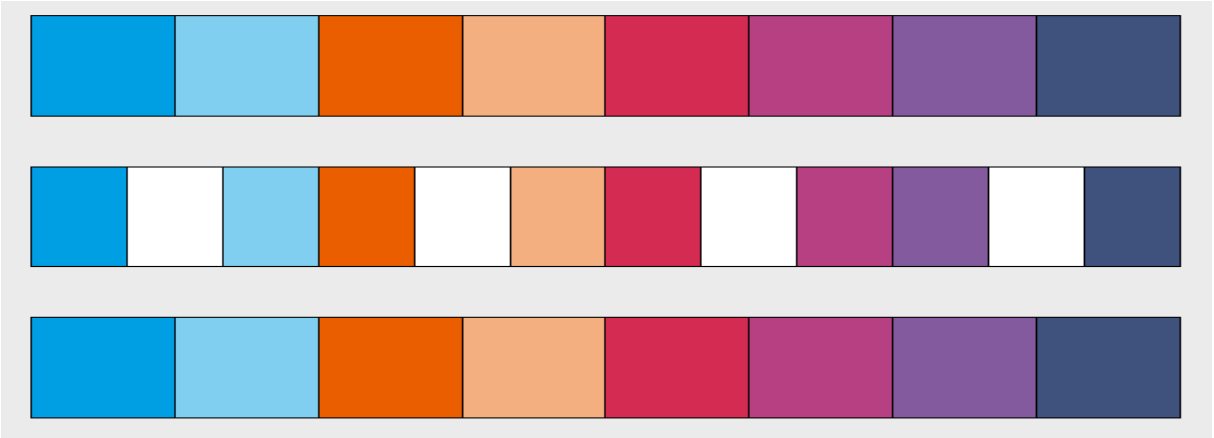
Maintenant il faut s'assurer qu'aucun nouveau silence n'est créé.

Pour remplir les conditions de (3), il faut que n soit pair, donc m_2 sera pair, et m_1 impair. Or, d'après (1), lors de la conversion de m_1 vers m_2 aucun silence ne sera créé

Pour remplir les conditions de (5), il faut que $\frac{n}{PGCD(m_1;n)}$ et, par conséquent, $\frac{m_2}{PGCD(m_1;m_2)}$ soient pairs, et $\frac{m_1}{PGCD(m_1;m_2)}$ soit impair. Ce qui, toujours d'après (5), ne crée aucun silence si l'on passe de la résolution m_1 à la résolution m_2.

Nous pouvons enfin conclure qu'il sera toujours possible de faire disparaître les silences en 2 conversions lorsque pour $m_1 > n$ on a $m_2 = n$

Exemple avec la propriété (5) : n=8, $m_1 = 10$ et $m_2 = 8 = n$



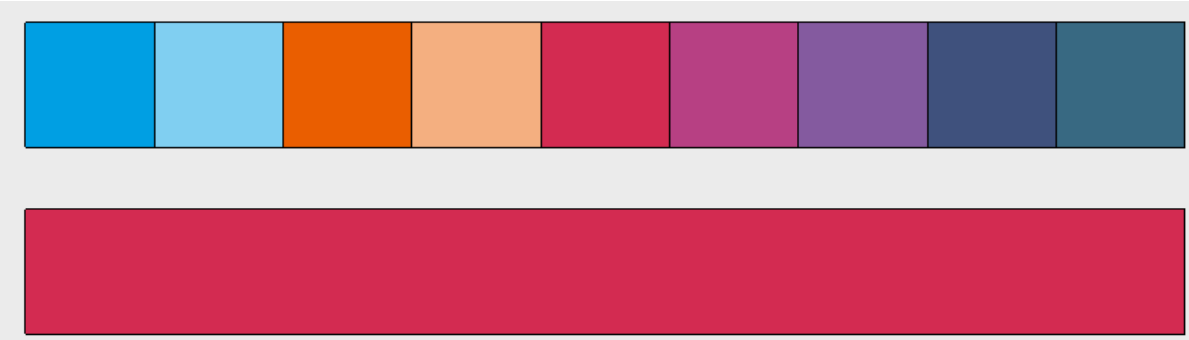
e) Toutes les notes sont les mêmes

Pour n'avoir que la même note dans le fichier « final » il faut impérativement passer par 1 car d'après l'énoncé, il y a n notes différentes dans le fichier initial.

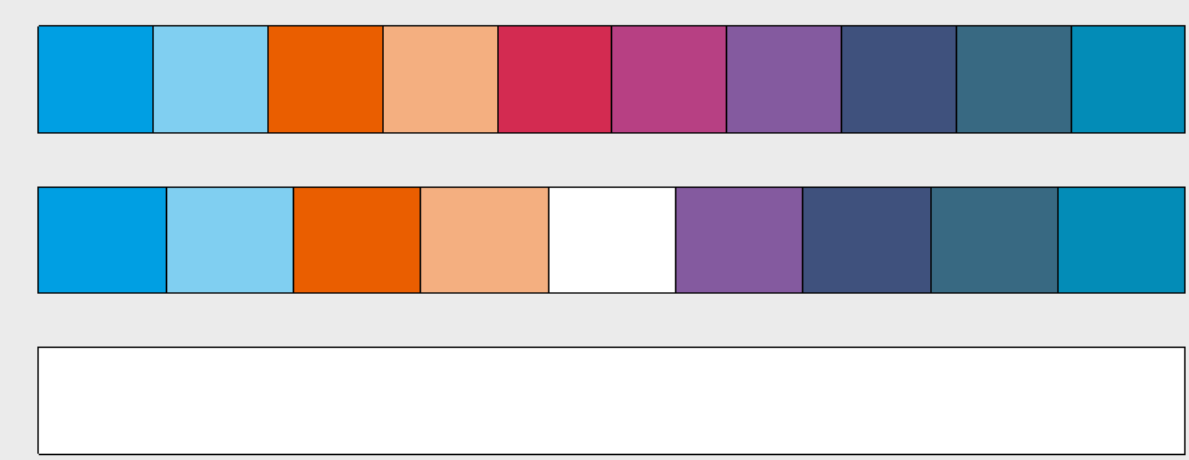
Soit, la résolution initiale vaut 1 (n = 1) dans ce cas aucune conversion n'est nécessaire pour remplir la condition.

Soit n est impair et m vaut 1, d'après (1) la note finale ne sera pas un silence, et comme il ne reste qu'une note identique, on peut dire que toutes les notes sont les mêmes.

Exemple : avec n = 9 et m = 1



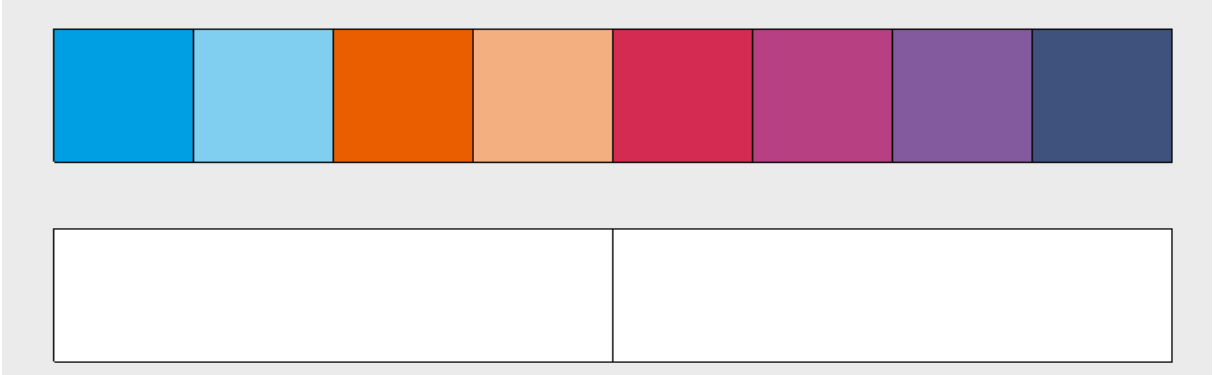
Exemple : avec n = 10 ; m_1 = 9 et m_2 = 1



f) La bande son ne contient que des silences

D'après (4), il est possible de faire apparaitre uniquement des silences.

Exemple : avec n = 8 et m = 2



2) Nombre minimal d'opérations qui donnent ce résultat

a) Une des notes présentes dans le fichier initial disparait

Comme nous l'avons vu à la question 1.a) pour qu'une note disparaisse il est suffisant que m<n. Donc, lorsque n>1, la condition est réalisable en un seul tour

Si n = 1, alors il faudra nécessairement deux tours au minimum, tels que $m_2 < m_1$, pour remplir la condition.

b) Les notes ne sont plus deux à deux distinctes

Comme vu à la question 1.b), si n est impair il suffit que m>n.

Lorsque n est pair, nous savons grâce à (5) que lors d'une conversion d'une résolution paire à une résolution plus grande, le nombre de silences qui apparaissent au maximum est $PGCD(m;n)$, or $|m-n| \geq PGCD(m;n)$, car $PGCD(m;n) | m-n$, donc lorsque m>n, il existe des résolutions pour lesquelles le nombre de nouvelles notes est supérieur au nombre de silences qui apparaissent, donc au moins une note va se répéter après conversion et la condition sera remplie. Toujours réalisable en un seul tour. Par exemple, lorsque m est un multiple de n, strictement supérieur à n. Dans les deux cas, un seul tour suffit pour remplir la condition demandée.

c) Un silence apparaît

Lorsque n est pair, les propriétés (3), (4) et (5) permettent de conclure que la condition est réalisable en un seul tour.

Lorsque n est impair, il faudra d'abord passer par une résolution paire pour faire apparaître un silence, car d'après (1), si n est impair, peu importe la résolution m, aucun silence ne va apparaître à la suite d'une conversion. Un silence apparait donc en minimum 2 tours. Pour cela, il faut que la m_1 soit pair, puis à l'aide des propriétés (3), (4) et (5) on pourra faire apparaître un silence.

d) Un silence disparaît

Lorsque n est pair, il faudra compter au minimum deux tours pour faire disparaître un silence. Grâce aux propriétés (4) et (5), il est possible de faire apparaître un silence en un tour, et grâce à la démonstration faîte en 1.d), si $m_1 > n$, on peut faire disparaître les silences en un tour supplémentaire si $m_2 = n$. Soit deux tours au minimum.

Lorsque n est impair, cela rajoute un tour car il en faudra désormais deux pour faire apparaître un silence d'après 1.c), et un pour le faire disparaître d'après la démonstration faîte en 1.d). Donc il faudra compter au minimum 3 tours.

e) Toutes les notes sont les mêmes

Pour n'avoir que la même note dans le fichier « final » il faut impérativement passer par 1 car d'après l'énoncé, il y a n notes différentes dans le fichier initial.

Soit, la résolution initiale vaut 1 (n = 1), dans ce cas aucune conversion n'est nécessaire pour remplir la condition. On peut dire que 0 tours suffisent.

Soit, la résolution initiale est un impair différent de 1. Alors un seul tour suffira pour que toutes les notes soient les mêmes si m=1, car d'après (1) cela ne crée pas de silence.

Si n est pair alors il sera impossible d'avoir plusieurs fois la même note car lorsque l'on passe d'une résolution paire à une résolution impaire on fait nécessairement apparaître un silence, d'après (3) (impossible donc de passer d'une résolution paire à 1, ou d'une résolution paire à impaire puis à 1 car le silence reste. En effet, le milieu de la note centrale d'un fichier avec une résolution impaire sera toujours $\frac{\left(\frac{\lfloor\frac{m}{2}\rfloor}{2}\right)+\left(\frac{\lceil\frac{m}{2}\rceil}{2}\right)}{m} = \frac{\lfloor\frac{m}{2}\rfloor+\lceil\frac{m}{2}\rceil}{2m} = \frac{\frac{m-1}{2}+\frac{m+1}{2}}{2m} = \frac{m}{2m} = \frac{1}{2}$.)

f) La bande son ne contient que des silences

D'après (4), il est possible de faire apparaitre uniquement des silences et ce un seul tour, lorsque n est pair.

Si n est impair, d'après (1) il sera impossible de faire apparaître des silences avec une seule conversion. Deux tours seront nécessaires au minimum. Par exemple en passant d'une résolution impaire à une résolution paire, puis faire apparaître que des silences avec la propriété (4).

3) Perrine s'interdit désormais de créer des fichiers avec moins de k notes ($1 \leq k \leq n$)

a) Une des notes présentes dans le fichier initial disparait

Si n = 1, alors k =1. Comme à la question 2.a) il faudra nécessairement deux tours au minimum, tels que $m_2 < m_1$, pour remplir la condition.

Si n est différent de 1, et k < n, alors comme à la question 1.a) il sera possible de faire disparaître une note présente dans le fichier initiale lorsque $k \leq m < n$. Dans ce cas, un tour peut suffire.

Si k = n, il faudra nécessairement qu'au premier tour, la résolution de conversion soit supérieure, autrement dit $m_1 > n$, car sinon le fichier final sera identique au fichier initial. Or d'après (5) on sait que le nombre de silences maximum qui apparaissent lors d'une conversion d'une résolution n à une résolution m supérieure vaut $PGCD(m;n)$, et $PGCD(m;n) \leq m - n$. Donc le nombre de silences qui apparaissent au maximum ne dépasse pas le nombre de nouvelles notes du fichier post-conversion, il est

donc impossible de faire disparaître des notes en un seul tour avec une résolution de conversion supérieure à n. Si m_1 est pair et suffisamment grand (de sorte à ce que $\frac{m}{2} > n$), on pourra créer un fichier avec seulement des silences, d'après (4), soit un fichier qui ne contient aucune note présente dans le fichier initial. On peut en conclure qu'il faudra compter au moins deux tours pour faire disparaître une note présente dans le fichier initial lorsque k=n.

b) Les notes ne sont plus deux à deux distinctes

Si n est impair alors il suffit que m>n, comme vu en 1.b). Ainsi peu importe la valeur de k, il sera toujours possible de remplir cette condition en un tour minimum.

Si n est pair, il suffit que m soit un multiple de n strictement supérieur à n, comme vu en 1.b). Ainsi peu importe la valeur de k, il sera toujours possible de remplir cette condition en un tour minimum.

c) Un silence apparaît

Si n est pair, on sait d'après (3) que si la résolution de conversion est impaire alors au moins un silence apparaît. Si m est un impair strictement supérieur à n alors peu importe la valeur de k, la condition sera toujours réalisable en un tour minimum.

Si n est impair on sait d'après 2.c) qu'il faudra au moins deux tours pour faire apparaître un silence. Pour cela il est suffisant que m_1 soit pair et strictement supérieur à n, et m_2 soit un impair supérieur ou égal à n. Peu importe la valeur de k, si n est impair, la condition est réalisable en deux tours minimums.

d) Un silence disparaît

Lorsque n est pair, la même réponse qu'en 2.d) est possible lorsque m1 > n. Il faudra donc compter au minimum deux tours pour faire disparaître un silence. Grâce aux propriétés (4) et (5), il est possible de faire apparaître un silence en un tour, et grâce à la démonstration faîte en 1.d), si $m_1 > n$, on peut faire disparaître les silences en un tour supplémentaire avec $m_2 = n$. On en conclut que peu importe le k, si n est pair, on pourra toujours faire disparaître un silence en minimum deux tours.

Lorsque n est impair, on peut toujours faire disparaître un silence et ce peu importe la valeur de k, en passant par une résolution paire supérieure puis en réalisant les mêmes étapes qu'au-dessus. On en conclut qu'au minimum, cela rajoute un tour par rapport à n pair. Donc, peu importe la valeur de k, lorsque n est impair il faudra au minimum trois tours pour faire disparaître un silence.

e) Toutes les notes sont les mêmes

Dans le fichier initial, de résolution n, d'après l'énoncé, il y a n notes différentes.

Donc pour que toutes les notes soient les mêmes il faut passer par 1.

Si n est pair, il sera impossible d'avoir plusieurs fois la même note car en passant d'une résolution paire à impaire on fait nécessairement apparaître un silence, d'après (3).

Si n est impair :

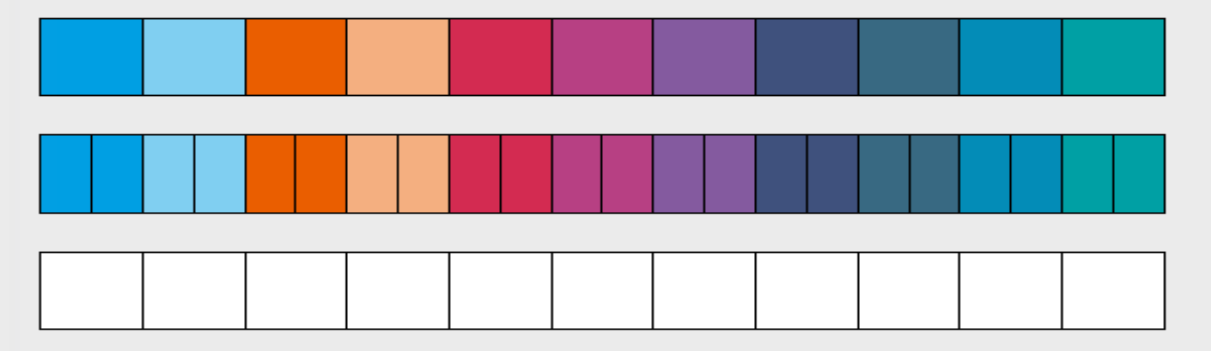
- pour k = 1 : Un tour suffit comme vu en réponse 2 e).

- pour k > 1 : Quel que soit n, il sera impossible d'avoir toutes les notes identiques.

f) La bande son ne contient que des silences

On cherche à appliquer la propriété (4), sans créer de fichier de résolution inférieure à k. Pour cela on multiplie la résolution par 2 avant de la diviser par 2. Ce qui nous permet d'obtenir une partition avec que des silences. On peut donc le faire en 2 tours pour tout n et pour tout k.

Exemple : *pour n= 11.*



On cherche n tel qu'on puisse réaliser l'objectif en moins de 2 tours.

Si n est impair, il n'est pas possible de le réaliser en moins de 2 tours (on ne peut pas diviser par2).

Si n est pair et $k \leq n/2$, on peut le réaliser en un tour en divisant la résolution par 2.

Si n est pair et $k > n/2$, on ne peut pas directement diviser la résolution par 2, on doit appliquer la méthode ci-dessus : 2 tours.

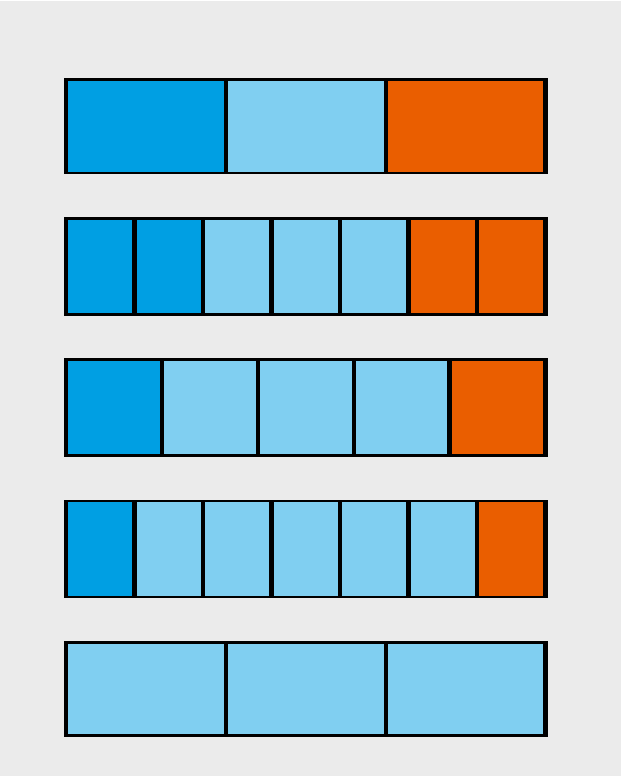
4) Le logiciel ne travaille plus qu'avec des résolutions impaires

a) Est-il possible qu'un silence apparaisse ?

Comme le logiciel ne travaille qu'avec des résolutions impaires, aucun silence ne pourra apparaître peu importe les conversions réalisées. En effet, toutes les résolutions de conversion seront impaires donc peu importe le tour, d'après (1), aucun silence ne pourra apparaître.

b) Est-il possible que le fichier final soit différent du fichier initial ?

Oui il est possible que le fichier final soit différent du fichier initial, comme dans cet exemple :



c) Quels sont les fichiers finaux possibles ?

Dans cette question nous n'avons pas réussi à démontrer nos résultats par le calcul, toutes nos pistes de recherche se montrant infructueuses. (Nous avons par exemple essayé de travailler avec des intervalles censés représenter une note du fichier initial et de comparer la position relative des centres des notes du fichier suivant pour en déduire la couleur de la note mais tout ceci en vain).

Une conjecture que l'on a retenue est qu'une note est capable de *s'étaler* au fil des tours jusqu'à une certaine limite (d'une part la note centrale lorsque ce n'est pas elle qui *s'étale* et d'autre part une limite arithmétique car plus la résolution augmente et moins l'étalement sera efficace car les longueurs des notes seront de plus en plus réduites). Nous n'avons cependant pas réussi à démontrer cette conjecture.

Si n=1, le fichier initial sera toujours le même que le fichier final car le milieu de la note centrale dans tout fichier impaire est $\frac{1}{2}$.

Pour n=3, on dénombre deux fichiers finaux différents atteignables par essais successifs.

Pour n=5, on en dénombre quatre fichiers finaux différents atteignables par essais successifs.

5) Reprendre les questions 4 sans se limiter aux résolutions impaires

a) Est-il possible qu'un silence apparaisse ?

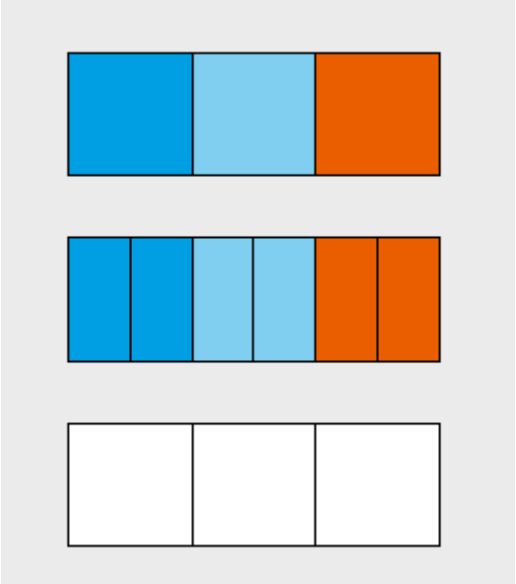
Oui il est possible qu'un silence apparaisse comme vu en question 3.c) (lorsque k=n). Plusieurs cas sont possibles, tous définis par les propriétés (3), (4) et (5).

b) Est-il possible que le fichier final soit différent du fichier initial ?

Oui il est possible que le fichier final soit différent du fichier initial avec n=3.

Lorsque la résolution de conversion m est de la forme $\frac{n}{2^p}$, $p \in \mathbb{N} \setminus \{0;1\}$, d'après (4), le fichier post-conversion sera composé uniquement de silences. Or, Perrine ne peut choisir des résolutions inférieures à n, elle devrait donc passer par une résolution plus grande. Ainsi, si m_1=2n et m_2=n, le fichier final sera composé uniquement de silences, il sera par définition différent du fichier initial.

Exemple : n=3, m_1=6 et m_2=3



c) Quels sont les fichiers finaux possibles ?

Nous n'avons pas réussi à répondre à cette question.

6) Reprendre les questions précédentes, mais en travaillant cette fois-ci avec des images.

Pour traiter cette deuxième partie du problème nous avons développé un second programme accessible via ce lien : https://cutt.ly/J7bTr5o

La plupart des propriétés précédentes et des résultats précédents restent les mêmes car lorsqu'on convertit la résolution d'une image, c'est comme si on faisait un changement de résolution d'une bande son vers une autre sur toute la hauteur, et une seconde conversion d'une bande son vers une autre sur toute la largeur (parfois en croisant les propriétés). De plus lorsqu'on ne modifie que la résolution de la longueur n (donc lorsque m' = n') cela revient à étudier la conversion d'une bande son d'une résolution n vers une résolution m que l'on réalise n' fois. Même idée si on ne modifie que la largeur m' (donc m = n) lors d'une conversion. Dans ce cas particulier, les propriétés vues en début de document sont encore valables, seulement, il faut multiplier les résultats par n' ou n suivant le cas traité. On parlera alors de *dimension géométrique modifiée* pour désigner soit la longueur soit la largeur lorsqu'une seule des deux est modifiée, afin de faciliter l'usage des formules et des expressions.

Dans la suite du problème la résolution initiale d'une image $n \times n'$ sera notée $(n; n')$ et la résolution de conversion $m \times m'$ sera notée $(m; m')$. Lorsqu'on réalise plusieurs conversions à la suite, la résolution de conversion au tour i sera notée $(m_i; m_i')$.

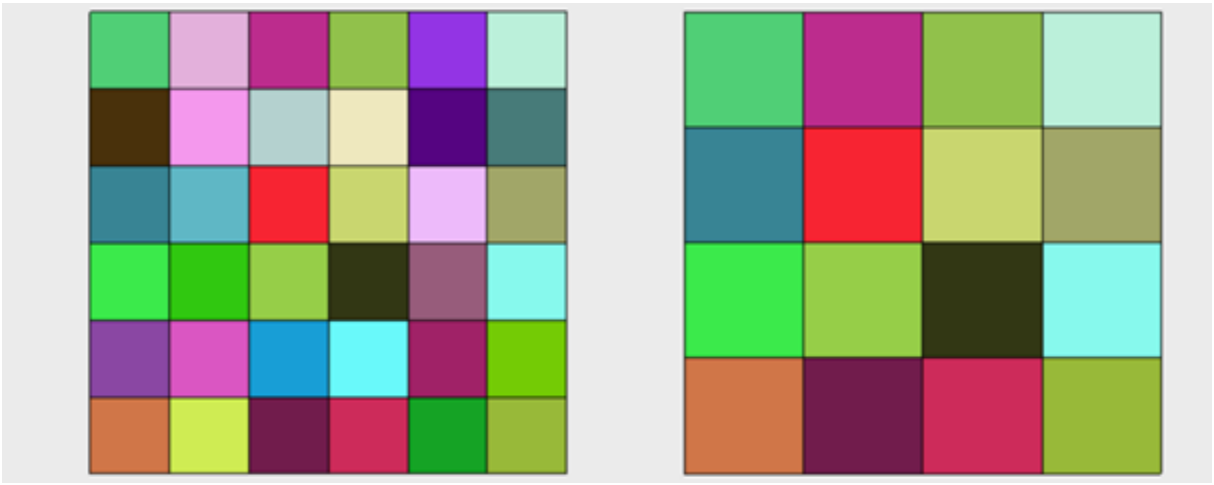
On appelle donnée de résolution les composantes n et n' du fichier initial et par extension, les *données de résolution de conversion* seront les composantes m et m'.

6.1/2) L'évènement peut-il arriver après un nombre fini d'opérations ?

a) Un des rectangles présents dans le fichier initial disparait (+nombre minimal d'opérations nécessaires)

Oui il est possible qu'un rectangle présent dans le fichier initial disparaisse notamment si Perrine choisit une résolution de conversion inférieure à la résolution initiale. C'est donc réalisable en un tour minimum.

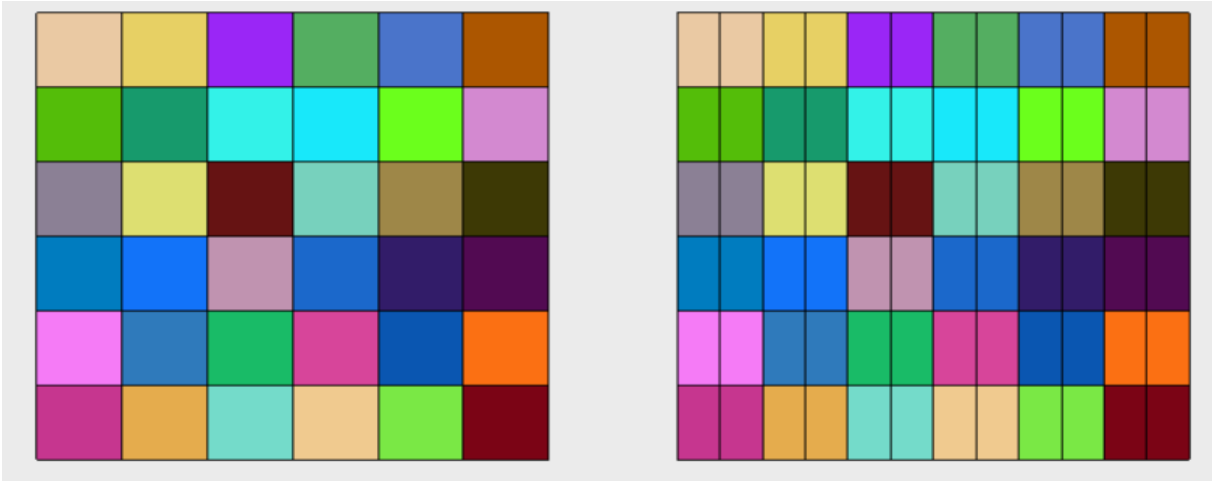
Comme dans cet exemple ou on passe d'une résolution (6;6) à une résolution (4;4) :



b) Les rectangles ne sont plus deux à deux distincts (+nombre minimal d'opérations nécessaires)

Une fois de plus, il est possible que les rectangles ne soient plus deux à deux distincts. Lorsque la résolution de conversion $(m; m')$ peut s'écrire $(p \cdot n; p \cdot n')$ avec $p \in \mathbb{N} \setminus \{0; 1\}$. Car dans ce cas, tous les rectangles apparaîtront p fois. Réalisable en 1 tour minimum.

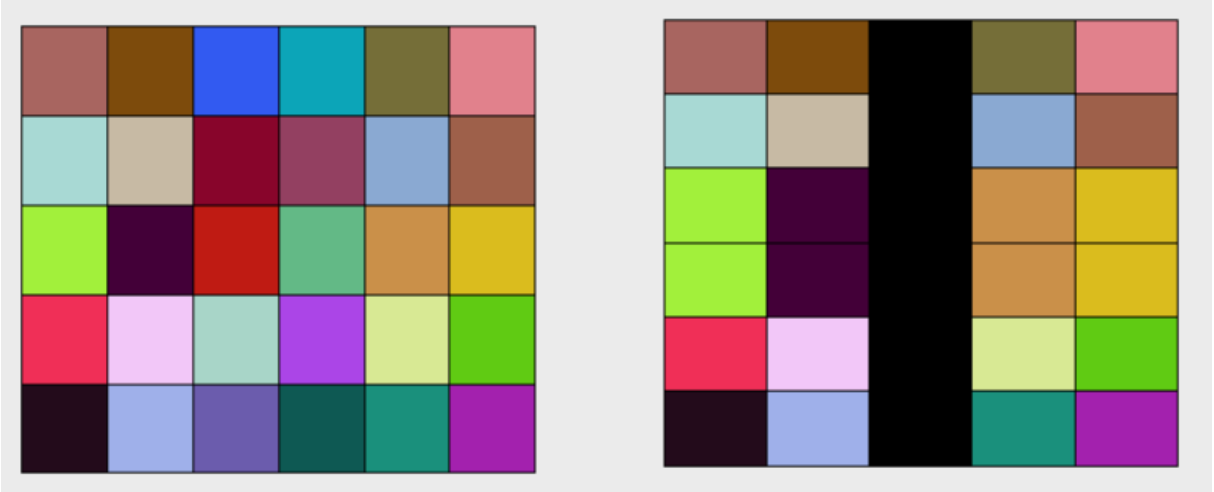
Comme dans cet exemple lorsqu'on passe d'une résolution de (6;6) à une résolution (12;12).



c) Un rectangle noir apparaît (+nombre minimal d'opérations nécessaires)

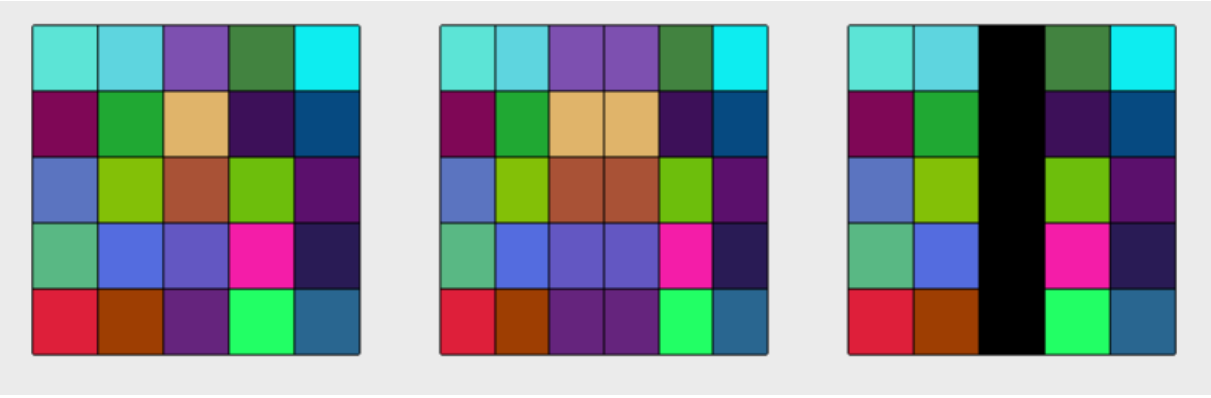
Comme précisé dans l'introduction à la question 6, si on ne modifie que la longueur m lors d'une conversion on peut utiliser les propriétés vues au début du document. Dans ce cas il est possible de faire apparaître des silences en utilisant (3), (4) ou (5) lorsque la dimension géométrique modifiée est paire, pour faire apparaître un silence en seulement un tour.

Exemple : conversion d'une image de résolution (6;5) à une résolution (5;6).



Cependant lorsque la dimension géométrique modifiée est impaire, d'après (1) aucun rectangle noir ne va apparaître lors de la conversion, il faudra compter au minimum deux tours pour faire apparaître des rectangles noirs. Une première conversion pour avoir les conditions nécessaires aux propriétés (3), (4) ou (5), puis une deuxième conversion pour faire apparaître un rectangle noir comme au-dessus.

Exemple : conversion d'une image de résolution (5;5) à une résolution (6;5) puis (5;5) à nouveau



On en conclut que, lorsque la dimension géométrique modifiée est paire, des rectangles noirs apparaissent en minimum 1 tour et lorsque la dimension géométrique modifiée est impaire, des rectangles noirs apparaissent en minimum deux tours.

d) Un rectangle noir disparaît (+nombre minimal d'opérations nécessaires)

Comme précisé en début de question, si on ne fait varier qu'une seule dimension géométrique on peut appliquer les propriétés vues au départ. Deuxièmement, pour faire disparaître un rectangle noir, il faut avant tout le faire apparaître.

Deux cas apparaissent, similaires à ceux traités en 2.d)

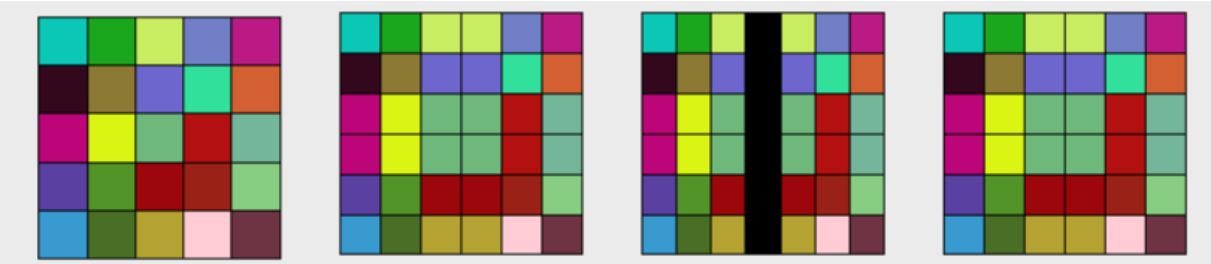
Soit la dimension géométrique modifiée est paire, dans ce cas on peut faire apparaître un rectangle noir en choisissant une résolution impaire supérieure comme vu avec (4) puis le faire disparaître comme vu en 1.d) en revenant à la résolution initiale.

Exemple : (6;6) -> (7;6) -> (6;6)

Soit la dimension géométrique modifiée est impaire, dans ce cas on doit passer par une première conversion en une résolution paire pour pouvoir faire apparaître un rectangle puis le faire disparaître, car d'après (1) on ne peut faire apparaître de rectangles noirs à partir d'une résolution initiale paire.

Dans ce cas, on rajoute un tour au procédé susmentionné pour remplir la condition.

Exemple : (5;5) -> (6;6) -> (7;6) -> (6;6)



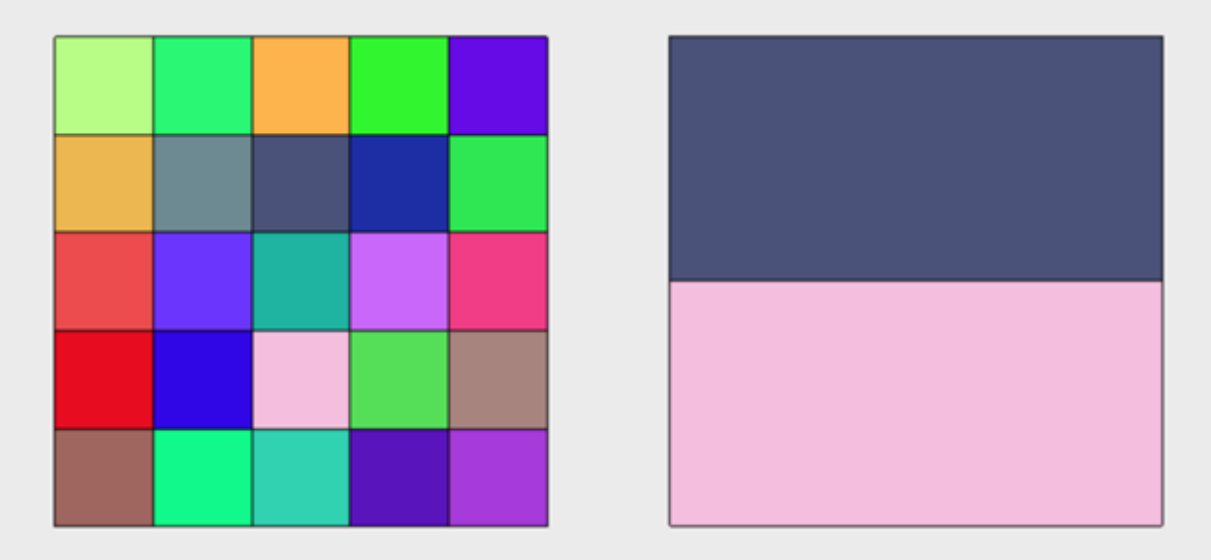
On en conclut que si la dimension géométrique modifiée est paire, alors c'est réalisable en au minimum 2 tours.

Si la dimension géométrique modifiée est impaire, c'est réalisable en minimum 3 tours.

e) Tous les rectangles sont les mêmes

Comme à la question 1.e) et 2.e), pour que tous les rectangles soient identiques il faut impérativement passer par la résolution (1;1). Si m ou m' est différent de 1, alors plusieurs rectangles vont apparaître dans le fichier final.

Exemple : (5;5) -> (1;2)



Cela s'explique par le fait que l'une des deux données de résolutions de conversion (m et m') est responsable du choix de ligne à convertir et l'autre de la colonne. Par définition il n'y a pas deux fois le même rectangle dans le fichier initial donc on aura différentes notes dans le fichier post-conversion.

Maintenant que l'on sait qu'il faut passer par une résolution (1;1) on s'intéresse à quand cela est possible.

Si une donnée de résolution du fichier initial est paire, lors de la conversion, on aura nécessairement au moins un silence qui va apparaître et ce sera toujours le rectangle central (d'après la propriété (3)). Donc il sera impossible d'obtenir tous les rectangles identiques, si au moins une des données de résolution du fichier initial est paire (comme en 2.e)).

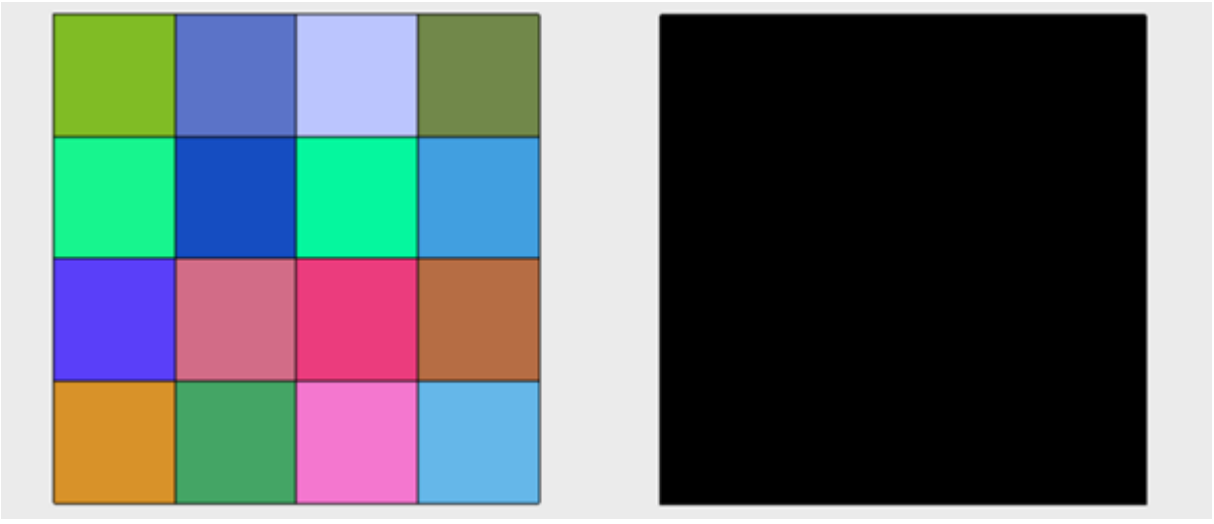
Si la résolution initiale est (1;1) on peut considérer que la condition est déjà remplie. 0 tours sont nécessaires.

Si les deux données de résolution sont impaires, avec au moins une différente de 1, alors c'est réalisable en au minimum 1 tour, d'après la propriété (1) car aucun silence n'apparaîtra, que ce soit à cause des rectangles en ligne ou en colonne. 1 tour minimum est donc nécessaire.

f) L'image n'est composée que de rectangles noirs

Si les deux composantes de résolution sont paires, il est possible de faire apparaître uniquement des rectangles noirs en un seul tour d'après (4).

Exemple : (4 ;4) -> (2 ;2)



Si au moins une composante est paire on pourra également n'avoir que des rectangles noirs en utilisant la propriété (4) et ce en 1 seul tour. En effet si on remplit les conditions de la (4) sur une ligne ou une colonne (suivant la donnée de résolution qui reprend la propriété) alors il y aura une ligne (ou une colonne du coup) qui sera composée que de rectangles noirs et qui apparaîtra alors n ou n' fois. Donc tout le fichier sera composé de rectangles noirs.

Exemple : (4;5) -> (2;2)

Si les deux données de résolution sont impaires alors il faudra compter un tour de plus car d'après (1) on ne peut créer de rectangle noir directement. Il faudra convertir vers une résolution ayant au moins une donnée paire puis reprendre les étapes au-dessus. Soit au moins 2 tours.