

# BROUILLON - Quelques machines de Turing déterministes

Christophe BAL

7 Février 2020 – 20 Mars 2020

*Document, avec son source L<sup>A</sup>T<sub>E</sub>X, disponible sur la page  
<https://github.com/bc-writing/drafts>.*

---

## Mentions « légales »

Ce document est mis à disposition selon les termes de la licence Creative Commons “Attribution - Pas d’utilisation commerciale - Partage dans les mêmes conditions 4.0 International”.



---

## Table des matières

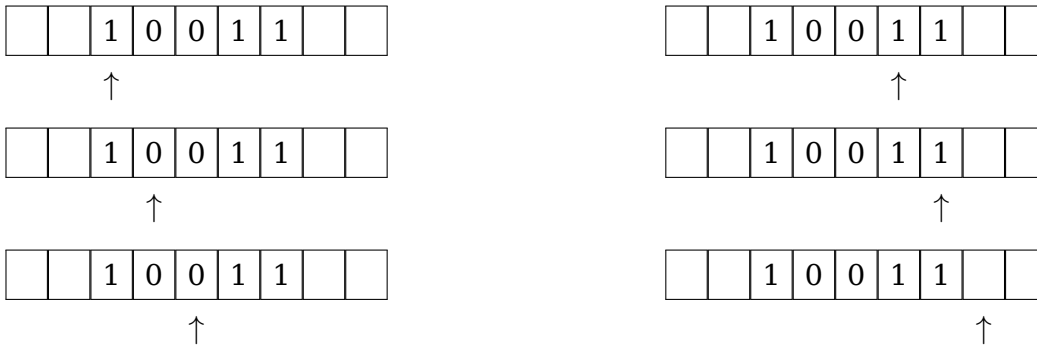
<b>1</b>	<b>Écriture binaire des naturels pairs</b>	<b>3</b>
1.1	La méthode . . . . .	3
1.2	Une table des transitions . . . . .	3
1.3	Coder pour tester . . . . .	4
<b>2</b>	<b>Écriture binaire des multiples de 3</b>	<b>5</b>
2.1	La méthode . . . . .	5
2.2	Une table des transitions . . . . .	6
2.3	Coder pour tester . . . . .	6
2.4	Généralisations . . . . .	6
<b>3</b>	<b>Écriture binaire des naturels impairs ou de ceux non multiples de 3</b>	<b>7</b>
3.1	Des tables des transitions . . . . .	7
3.2	Coder pour tester . . . . .	7
<b>4</b>	<b>Avec deux bandes</b>	<b>8</b>
4.1	Écriture binaire d’un multiple de 2 et 3 . . . . .	8
4.2	Écriture binaire d’un multiple de 2 ou/et de 3 . . . . .	11

4.3	Écriture binaire d'un multiple soit de 2, soit de 3 mais pas des deux en même temps . . . . .	13
<b>5</b>	<b>Détecter les palindromes avec une seule bande</b>	<b>14</b>
5.1	Un exemple avec le mot abbca . . . . .	14
5.2	Un exemple avec le mot abbcbbba . . . . .	14
5.3	Une table des transitions . . . . .	15
5.4	Coder pour tester . . . . .	16
<b>6</b>	<b>Détecter les palindromes avec deux bandes</b>	<b>17</b>
6.1	Un exemple avec le mot abbca . . . . .	17
6.2	Un exemple avec le mot abbcbbba . . . . .	17
6.3	Une table des transitions . . . . .	18
6.4	Coder pour tester . . . . .	18
<b>7</b>	<b>Deux fois plus de b que de a</b>	<b>19</b>
7.1	Un exemple avec le mot babbbbabab . . . . .	19
7.2	Une table des transitions . . . . .	21
7.3	Avec une seule bande? . . . . .	21
7.4	Coder pour tester . . . . .	22

# 1 Écriture binaire des naturels pairs

## 1.1 La méthode

Commençons par un exemple simple qui va nous permettre de fixer les notations que nous allons utiliser. Voyons comment repérer un entier naturel pair à partir de son écriture binaire. La réponse est évidente : un naturel est pair si et seulement si son écriture binaire se finit par un zéro. Il suffit donc de parcourir cette écriture binaire et d'analyser le chiffre le plus à droite. Ceci se schématise comme suit où  $\uparrow$  indique la tête de lecture.



Pourquoi s'arrêter à la première case vide ? L'idée va être de garder en mémoire la valeur de la dernière case visitée. Dans notre exemple, lors du dernier mouvement, on sait que la case précédente était un 1 et donc que l'écriture binaire n'est pas celle d'un entier naturel pair.

## 1.2 Une table des transitions

Pour formaliser les étapes à suivre, on va écrire une table des transitions. Nous travaillons avec des mots sur l'alphabet  $\{0; 1\}$  et nous allons utiliser les états de transition suivants.

- $q_0$  est l'état initial avant tout déplacement.
- $\ell_0$  et  $\ell_1$  sont les états indiquant que la dernière<sup>1</sup> case visitée contenait un 0 ou un 1 respectivement.
- L'état  $f$  sera notre état final indiquant qu'un mot est bien l'écriture binaire d'un naturel pair.

Voici la table des transitions de notre machine de Turing où  $B$  indique une case vide, tandis que  $D$  et  $G$  demandent de déplacer la tête de lecture  $\uparrow$  vers la droite et la gauche respectivement, tandis que  $I$  indique de ne pas faire bouger cette tête.

$\delta$	0	1	$B$
$q_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
$\ell_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(f, B, I)$
$\ell_1$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	

1.  $\ell$  pour « last ».

### 1.3 Coder pour tester

Sur le site de téléchargement de ce document se trouve le fichier `divisible-by-2.txt` dans le sous-dossier `turing/divisible-by-2`. Il contient le code suivant utilisable pour des tests manuels sur le site <https://turingmachinesimulator.com>. Nous expliquons comme cela fonctionne juste après.

```
// -- SPÉCIFIER LA MACHINE -- //
name: Écriture binaire d'un pair
init: q0
accept: f

// -- DÉPLACEMENT AVEC MÉMOIRE COURTE POUR LES 0 -- //
q0,0
l0,0,>

l0,0
l0,0,>

l1,0
l0,0,>

// -- DÉPLACEMENT AVEC MÉMOIRE COURTE POUR LES 1 -- //
q0,1
l1,1,>

l0,1
l1,1,>

l1,1
l1,1,>

// -- FIN GAGNANTE -- //
l0,-
f,-,-
```

Le langage utilisé est simple à comprendre.

1. Les lignes commençant par `//` sont des commentaires.
2. `name: Écriture binaire d'un pair` définit le nom de la machine. C'est utile pour stocker ses machines sur le site.
3. `init: q0` et `accept: f` définissent les états initial et final.
4. Enfin des lignes successives du type `q,s` et `q',s',>` indiquent que si l'état courant est `q` lorsque la tête de lecture est sur le symbole `s` alors l'état devient `q'` et le symbole `s'`, la tête de lecture se déplaçant vers la droite. Il est possible de rester immobile via `-` ou d'aller vers la gauche via `<` et de plus le symbole `_` sert pour une case vide.

## 2 Écriture binaire des multiples de 3

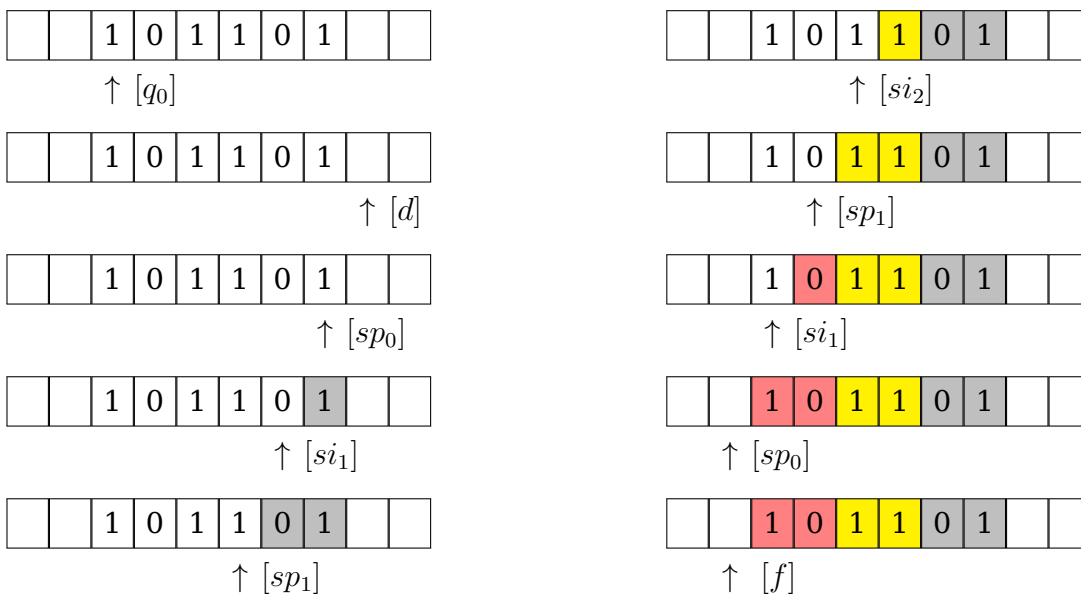
### 2.1 La méthode

Le problème est ici plus intéressant que celui de l'écriture binaire d'un naturel pair. Tout le monde connaît le critère de divisibilité par 3 d'un entier écrit en base 10. Par exemple, 1234567 n'est pas divisible par 3 car  $1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$  ne l'est pas. Ce critère fonctionne car  $10 = 3^2 + 1 = 3k + 1$  nous donne ce qui suit où les  $k_i$  sont des naturels<sup>2</sup>.

$$\begin{aligned}
 1234567 &= 123456 \times 10 + 7 \\
 &= 123456 (3k_1 + 1) + 7 \\
 &= 3k_1 + 123456 + 7 \\
 &= 3k_1 + 12345 \times 10 + 6 + 7 \\
 &= 3k_2 + 12345 + 6 + 7 \\
 &\vdots \\
 &= 3k_6 + 1 + 2 + 3 + 4 + 5 + 6 + 7
 \end{aligned}$$

Comme  $4 = 3 + 1$ , on peut de même à partir d'une écriture en base 4 comme  $[1203]_4$  déterminer si un nombre est divisible par 3 en faisant ici  $[1]_4 + [2]_4 + [0]_4 + [3]_4 = [12]_4$  puis  $[1]_4 + [2]_4 = [3]_4$  pour conclure que  $[1203]_4 = 64 + 2 \times 16 + 3 = 99$  est un multiple de 3.

Le passage aux écritures binaires devient maintenant assez simple. Notons pour commencer que par exemple  $[101101]_2 = [231]_4$  s'obtient sans effort via  $[10]_2 = [2]_4$ ,  $[11]_2 = [3]_4$  et  $[01]_2 = [1]_4$ . Voici alors comment analyser ce nombre via une machine de Turing. On a ajouté à droite de la tête de lecture un état permettant de calculer le cumul en base 4 des chiffres modulo 3 en tenant compte de la parité de l'indice de position du chiffre qui vient d'être lu.



2. Tout ceci n'est en fait que du calcul modulo 3.

Voici les points clés dans les étapes ci-dessus.

1. Si la tête de lecture est sous un chiffre d'indice de position  $(2k + 1)$ , il suffit d'effectuer modulo 3 l'ajout en base 4 de la valeur de la case. Les valeurs possibles seront donc 0, 1 et 2 mais pas 3 qu'on fait passer à 0. Ce calcul se justifie par le fait que l'on passe à un nouveau chiffre de poids plus fort en base 4.
2. Si la tête de lecture est sous un chiffre d'indice de position  $2k$ , il suffit d'effectuer modulo 3 l'ajout en base 4 du double de la valeur de la case.

## 2.2 Une table des transitions

Les explications de la section précédente permettent de comprendre la table des transitions suivantes.

$\delta$	0	1	$B$
$q_0$	$(d, 0, D)$	$(d, 1, D)$	
$d$	$(d, 0, D)$	$(d, 1, D)$	$(sp_0, B, G)$
$sp_0$	$(si_0, 0, G)$	$(si_1, 1, G)$	$(f, B, I)$
$sp_1$	$(si_1, 0, G)$	$(si_2, 1, G)$	
$sp_2$	$(si_2, 0, G)$	$(si_0, 1, G)$	
$si_0$	$(sp_0, 0, G)$	$(sp_2, 1, G)$	$(f, B, I)$
$si_1$	$(sp_1, 0, G)$	$(sp_0, 1, G)$	
$si_2$	$(sp_2, 0, G)$	$(sp_1, 1, G)$	

## 2.3 Coder pour tester

Sur le site de téléchargement de ce document se trouve le fichier `divisible-by-3.txt` dans le sous-dossier `turing/divisible-by-3`. Il contient un code utilisable pour des tests manuels sur le site <https://turingmachinesimulator.com>.

## 2.4 Généralisations

Le raisonnement précédent s'est appuyé sur  $4 = 3 + 1$  soit  $3 = 2^2 - 1$ . Il est en fait facile de généraliser ce qui a été fait pour repérer les nombres divisibles par  $2^k - 1$  où  $k \in \mathbb{N}_{\geq 2}$ . Ceci étant dit, les tables des transitions vont croître de façon exponentielle !

**Remarque :** on peut en fait construire un automate d'état déterministe fini qui reconnaît les multiples de 3, et plus généralement de  $d \in \mathbb{N}_{\geq 2}$ , à partir de l'écriture d'un nombre dans une base quelconque  $b \in \mathbb{N}_{\geq 2}$ . Ceci prouve que l'on peut construire des expressions rationnelles pour savoir si une écriture correspond à un multiple.

### 3 Écriture binaire des naturels impairs ou de ceux non multiples de 3

#### 3.1 Des tables des transitions

Une méthode simple et généraliste consiste à modifier les tables des transitions précédentes mécaniquement en respectant les deux règles suivantes.

1. Toutes les cellules qui contiennent l'état final sont vidées.
2. Toutes les cases vides, **sauf celle correspondant à la case vide et l'état initial**, sont remplies pour indiquer un passage à l'état final<sup>3</sup>. **La cellule non traitée l'est afin d'éviter d'accepter le mot vide.**

Ceci nous donne la table des transitions ci-après pour les naturels impairs.

$\delta$	0	1	$B$
$q_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
$\ell_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
$\ell_1$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(f, B, I)$

Pour les non multiples de 3, on obtient la table des transitions suivante.

$\delta$	0	1	$B$
$q_0$	$(d, 0, D)$	$(d, 1, D)$	
$d$	$(d, 0, D)$	$(d, 1, D)$	$(sp_0, B, G)$
$sp_0$	$(si_0, 0, G)$	$(si_1, 1, G)$	
$sp_1$	$(si_1, 0, G)$	$(si_2, 1, G)$	$(f, B, I)$
$sp_2$	$(si_2, 0, G)$	$(si_0, 1, G)$	$(f, B, I)$
$si_0$	$(sp_0, 0, G)$	$(sp_2, 1, G)$	
$si_1$	$(sp_1, 0, G)$	$(sp_0, 1, G)$	$(f, B, I)$
$si_2$	$(sp_2, 0, G)$	$(sp_1, 1, G)$	$(f, B, I)$

#### 3.2 Coder pour tester

Sur le site de téléchargement de ce document se trouvent not-divisible-by-2.txt et not-divisible-by-3.txt dans le sous-dossier turing/not-divisible-by-2-or-3 qui contiennent des codes utilisables sur le site <https://turingmachinesimulator.com>.

---

3. Ne pas oublier de traiter les états bloquants non indiqués sur la table initiale !

## 4 Avec deux bandes

Dans cette section, **nous cherchons à proposer des méthodes généralistes automatiques mais non nécessairement optimales** (voir à ce sujet la section 4.1.2).

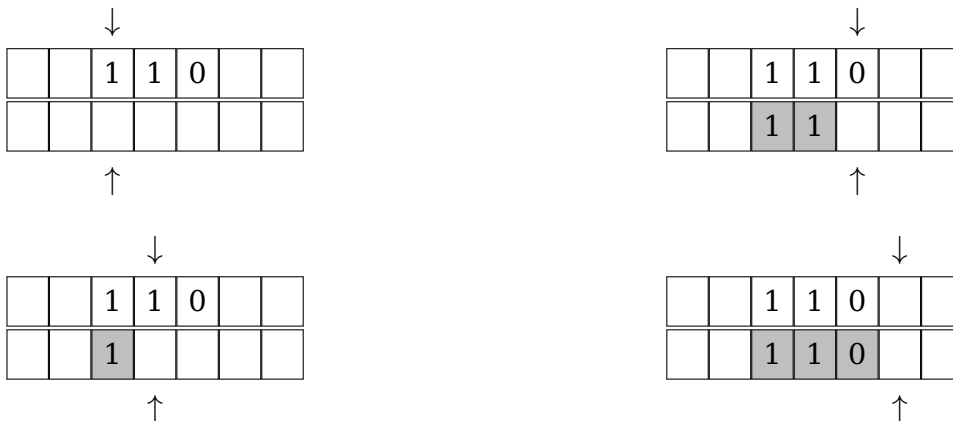
### 4.1 Écriture binaire d'un multiple de 2 et 3

#### 4.1.1 Des tables des transitions

Une 1<sup>re</sup> idée consiste à utiliser la machine qui repère les multiples de 3 et si elle finit de passer à l'analyse avec la machine repérant les pairs<sup>4</sup>. Cette démarche est valable car les machines que nous avons présentées ne modifient pas la donnée en entrée<sup>5</sup>. Ceci nous conduit à la table des transitions ci-après où le nouvel état  $m$  indique qu'un multiple de 3 a été repéré.

$\delta$	0	1	$B$
$q_0$	$(d, 0, D)$	$(d, 1, D)$	
$d$	$(d, 0, D)$	$(d, 1, D)$	$(sp_0, B, G)$
$sp_0$	$(si_0, 0, G)$	$(si_1, 1, G)$	$(m, B, I)$
$sp_1$	$(si_1, 0, G)$	$(si_2, 1, G)$	
$sp_2$	$(si_2, 0, G)$	$(si_0, 1, G)$	
$si_0$	$(sp_0, 0, G)$	$(sp_2, 1, G)$	$(m, B, I)$
$si_1$	$(sp_1, 0, G)$	$(sp_0, 1, G)$	
$si_2$	$(sp_2, 0, G)$	$(sp_1, 1, G)$	
$m$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(m, B, D)$
$\ell_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(f, B, I)$
$\ell_1$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	

Mais que faire si nous avons des machines qui agissent sur la donnée en entrée ? Une réponse simple est de travailler avec deux bandes au lieu d'une seule en commençant toujours par dupliquer la donnée sur la 2<sup>e</sup> bande. Voici ce que cela peut donner schématiquement (*nous mettons la 1<sup>re</sup> bande au-dessus*).



4. On commence par les multiples de 3 car notre machine en cas de succès positionne la tête de lecture juste à gauche du premier chiffre. Ceci va nous simplifier un peu la tâche.

5. Ceci montre aussi comment composer séquentiellement des machines.



Une fois ceci fait, on repositionne les deux têtes à gauche au début des données. Il suffit alors de faire agir une sous-machine sur une bande puis si elle finit de passer la main à la seconde sous-machine sur l'autre bande. Ceci nous amène à la table des transitions suivantes où  $\overset{1}{B}$  indique la lecture d'une case du haut avec un 1 et d'une case du bas vide, et on utilise la même convention pour les déplacements. Les nouveaux états utilisés sont  $c$  pour « copier »,  $g$  pour « retourner à gauche »,  $t_m$  pour « tester un multiple de 3 » et  $t_p$  pour « tester un pair ». Nous avons éclaté la table en plusieurs parties car le procédé choisi, bien qu'automatisable, fait exploser le nombre de cas à traiter.

*Phase 1 : copie de l'entrée.*

$\delta$	0 $B$	1 $B$	$B$ $B$	0 0	1 1
$q_0$	$\left( c, \overset{0}{B}, D \right)$	$\left( c, \overset{1}{B}, D \right)$			
$c$	$\left( c, \overset{0}{B}, D \right)$	$\left( c, \overset{1}{B}, D \right)$	$\left( g, \overset{B}{B}, \overset{G}{G} \right)$		
$g$			$\left( t_m, \overset{B}{B}, \overset{D}{I} \right)$	$\left( g, \overset{0}{B}, \overset{G}{G} \right)$	$\left( g, \overset{1}{B}, \overset{G}{G} \right)$

*Phase 2 : a-t-on un multiple de 3 via la bande du haut ?*

$\delta$	0 $B$	1 $B$	$B$ $B$
$t_m$	$\left( d, \overset{0}{B}, \overset{D}{I} \right)$	$\left( d, \overset{1}{B}, \overset{D}{I} \right)$	
$d$	$\left( d, \overset{0}{B}, \overset{D}{I} \right)$	$\left( d, \overset{1}{B}, \overset{D}{I} \right)$	$\left( sp_0, \overset{B}{B}, \overset{G}{I} \right)$
$sp_0$	$\left( si_0, \overset{0}{B}, \overset{G}{I} \right)$	$\left( si_1, \overset{1}{B}, \overset{G}{I} \right)$	$\left( t_p, \overset{B}{B}, \overset{I}{D} \right)$
$sp_1$	$\left( si_1, \overset{0}{B}, \overset{G}{I} \right)$	$\left( si_2, \overset{1}{B}, \overset{G}{I} \right)$	
$sp_2$	$\left( si_2, \overset{0}{B}, \overset{G}{I} \right)$	$\left( si_0, \overset{1}{B}, \overset{G}{I} \right)$	
$si_0$	$\left( sp_0, \overset{0}{B}, \overset{G}{I} \right)$	$\left( sp_2, \overset{1}{B}, \overset{G}{I} \right)$	$\left( t_p, \overset{B}{B}, \overset{I}{D} \right)$
$si_1$	$\left( sp_1, \overset{0}{B}, \overset{G}{I} \right)$	$\left( sp_0, \overset{1}{B}, \overset{G}{I} \right)$	
$si_2$	$\left( sp_2, \overset{0}{B}, \overset{G}{I} \right)$	$\left( sp_1, \overset{1}{B}, \overset{G}{I} \right)$	

*Phase 3 : a-t-on un multiple de 3 qui est aussi pair via la bande du bas ?*

$\delta$	$B$ 0	$B$ 1	$B$ $B$
$t_p$	$\left( \ell_0, \overset{B}{B}, \overset{I}{D} \right)$	$\left( \ell_1, \overset{B}{B}, \overset{I}{D} \right)$	
$\ell_0$	$\left( \ell_0, \overset{B}{B}, \overset{I}{D} \right)$	$\left( \ell_1, \overset{B}{B}, \overset{I}{D} \right)$	$\left( f, \overset{B}{B}, \overset{I}{I} \right)$
$\ell_1$	$\left( \ell_0, \overset{B}{B}, \overset{I}{D} \right)$	$\left( \ell_1, \overset{B}{B}, \overset{I}{D} \right)$	

### 4.1.2 Une table efficace

Si l'on sort des méthodes généralistes et automatisables, on peut faire une table plus simple. En effet, pour tester si l'on a un multiple de 3, nous partons de la droite. Or nous savons aussi que le dernier chiffre lu à droite nous permet de savoir si l'on a ou non un nombre pair. Il suffit donc lors du déplacement à droite de garder la trace de ce dernier chiffre, puis de continuer le travail si l'on a bien un zéro final. Ceci nous donne ci-après une table des transitions plus courte que les précédentes.

$\delta$	0	1	$B$
$q_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
$\ell_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(sp_0, B, G)$
$\ell_1$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
$sp_0$	$(si_0, 0, G)$	$(si_1, 1, G)$	$(f, B, I)$
$sp_1$	$(si_1, 0, G)$	$(si_2, 1, G)$	
$sp_2$	$(si_2, 0, G)$	$(si_0, 1, G)$	
$si_0$	$(sp_0, 0, G)$	$(sp_2, 1, G)$	$(f, B, I)$
$si_1$	$(sp_1, 0, G)$	$(sp_0, 1, G)$	
$si_2$	$(sp_2, 0, G)$	$(sp_1, 1, G)$	

### 4.1.3 Coder pour tester

Sur le site de téléchargement de ce document se trouve `divisible-by-2-and-3.txt` qui contient un code utilisable sur le site <https://turingmachinesimulator.com>. Voir le sous-dossier `turing/two-tapes/divisible-by-2-and-3`. Vous y trouverez aussi le fichier `divisible-by-2-and-3-single-tape.txt` pour la version à une seule bande, et `divisible-by-2-and-3-human-trick.txt` pour la version améliorée à une seule bande.

## 4.2 Écriture binaire d'un multiple de 2 ou /et de 3

### 4.2.1 Des tables des transitions

La méthodologie est ici assez simple. On reprend l'une des tables vues dans la section précédente. Vous avez dû noter que des états passerelles sont utilisés pour passer la main au 2<sup>e</sup> traitement. On va donc rendre tous ces états finaux. Ceci ne suffit pas car il faut aussi que les états bloquants<sup>6</sup> de la 1<sup>re</sup> machine donne la main à la seconde en faisant tout de même attention pour les machines à une seule bande à bien replacer la tête de lecture (*dans notre cas, ce problème ne se pose pas, ce qui nous enlève un peu de travail*). Nous avons alors les trois tables des transitions suivantes.

**Table à une bande « automatique »**

$\delta$	0	1	$B$
$q_0$	$(d, 0, D)$	$(d, 1, D)$	
$d$	$(d, 0, D)$	$(d, 1, D)$	$(sp_0, B, G)$
$sp_0$	$(si_0, 0, G)$	$(si_1, 1, G)$	$(f, B, I)$
$sp_1$	$(si_1, 0, G)$	$(si_2, 1, G)$	$(m, B, I)$
$sp_2$	$(si_2, 0, G)$	$(si_0, 1, G)$	$(m, B, I)$
$si_0$	$(sp_0, 0, G)$	$(sp_2, 1, G)$	$(f, B, I)$
$si_1$	$(sp_1, 0, G)$	$(sp_0, 1, G)$	$(m, B, I)$
$si_2$	$(sp_2, 0, G)$	$(sp_1, 1, G)$	$(m, B, I)$
$m$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(m, B, D)$
$\ell_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(f, B, I)$
$\ell_1$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	

**Table « optimisée »**

$\delta$	0	1	$B$
$q_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
$\ell_0$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(f, B, G)$
$\ell_1$	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	$(sp_0, B, G)$
$sp_0$	$(si_0, 0, G)$	$(si_1, 1, G)$	$(f, B, I)$
$sp_1$	$(si_1, 0, G)$	$(si_2, 1, G)$	
$sp_2$	$(si_2, 0, G)$	$(si_0, 1, G)$	
$si_0$	$(sp_0, 0, G)$	$(sp_2, 1, G)$	$(f, B, I)$
$si_1$	$(sp_1, 0, G)$	$(sp_0, 1, G)$	
$si_2$	$(sp_2, 0, G)$	$(sp_1, 1, G)$	

6. Il faut bien entendu penser à traiter les états bloquants non indiqués sur la table initiale !

### Table à deux bandes

Phase 1 : copie de l'entrée.

$\delta$	0 $B$	1 $B$	$B$ $B$	0 0	1 1
$q_0$	$\left( c, \begin{smallmatrix} 0 & D \\ 0 & D \end{smallmatrix} \right)$	$\left( c, \begin{smallmatrix} 1 & D \\ 1 & D \end{smallmatrix} \right)$			
$c$	$\left( c, \begin{smallmatrix} 0 & D \\ 0 & D \end{smallmatrix} \right)$	$\left( c, \begin{smallmatrix} 1 & D \\ 1 & D \end{smallmatrix} \right)$	$\left( g, \begin{smallmatrix} B & G \\ B & G \end{smallmatrix} \right)$		
$g$			$\left( t_m, \begin{smallmatrix} B & D \\ B & I \end{smallmatrix} \right)$	$\left( g, \begin{smallmatrix} 0 & G \\ 0 & G \end{smallmatrix} \right)$	$\left( g, \begin{smallmatrix} 1 & G \\ 1 & G \end{smallmatrix} \right)$

Phase 2 : a-t-on un multiple de 3 via la bande du haut ?

$\delta$	0 $B$	1 $B$	$B$ $B$
$t_m$	$\left( d, \begin{smallmatrix} 0 & D \\ B & I \end{smallmatrix} \right)$	$\left( d, \begin{smallmatrix} 1 & D \\ B & I \end{smallmatrix} \right)$	
$d$	$\left( d, \begin{smallmatrix} 0 & D \\ B & I \end{smallmatrix} \right)$	$\left( d, \begin{smallmatrix} 1 & D \\ B & I \end{smallmatrix} \right)$	$\left( sp_0, \begin{smallmatrix} B & G \\ B & I \end{smallmatrix} \right)$
$sp_0$	$\left( si_0, \begin{smallmatrix} 0 & G \\ B & I \end{smallmatrix} \right)$	$\left( si_1, \begin{smallmatrix} 1 & G \\ B & I \end{smallmatrix} \right)$	$\left( f, \begin{smallmatrix} B & I \\ B & I \end{smallmatrix} \right)$
$sp_1$	$\left( si_1, \begin{smallmatrix} 0 & G \\ B & I \end{smallmatrix} \right)$	$\left( si_2, \begin{smallmatrix} 1 & G \\ B & I \end{smallmatrix} \right)$	$\left( t_p, \begin{smallmatrix} B & I \\ B & D \end{smallmatrix} \right)$
$sp_2$	$\left( si_2, \begin{smallmatrix} 0 & G \\ B & I \end{smallmatrix} \right)$	$\left( si_0, \begin{smallmatrix} 1 & G \\ B & I \end{smallmatrix} \right)$	$\left( t_p, \begin{smallmatrix} B & I \\ B & D \end{smallmatrix} \right)$
$si_0$	$\left( sp_0, \begin{smallmatrix} 0 & G \\ B & I \end{smallmatrix} \right)$	$\left( sp_2, \begin{smallmatrix} 1 & G \\ B & I \end{smallmatrix} \right)$	$\left( f, \begin{smallmatrix} B & I \\ B & I \end{smallmatrix} \right)$
$si_1$	$\left( sp_1, \begin{smallmatrix} 0 & G \\ B & I \end{smallmatrix} \right)$	$\left( sp_0, \begin{smallmatrix} 1 & G \\ B & I \end{smallmatrix} \right)$	$\left( t_p, \begin{smallmatrix} B & I \\ B & D \end{smallmatrix} \right)$
$si_2$	$\left( sp_2, \begin{smallmatrix} 0 & G \\ B & I \end{smallmatrix} \right)$	$\left( sp_1, \begin{smallmatrix} 1 & G \\ B & I \end{smallmatrix} \right)$	$\left( t_p, \begin{smallmatrix} B & I \\ B & D \end{smallmatrix} \right)$

Phase 3 : a-t-on un non multiple de 3 qui est pair via la bande du bas ?

$\delta$	$B$ 0	$B$ 1	$B$ $B$
$t_p$	$\left( \ell_0, \begin{smallmatrix} B & I \\ 0 & D \end{smallmatrix} \right)$	$\left( \ell_1, \begin{smallmatrix} B & I \\ 1 & D \end{smallmatrix} \right)$	
$\ell_0$	$\left( \ell_0, \begin{smallmatrix} B & I \\ 0 & D \end{smallmatrix} \right)$	$\left( \ell_1, \begin{smallmatrix} B & I \\ 1 & D \end{smallmatrix} \right)$	$\left( f, \begin{smallmatrix} B & I \\ B & I \end{smallmatrix} \right)$
$\ell_1$	$\left( \ell_0, \begin{smallmatrix} B & I \\ 0 & D \end{smallmatrix} \right)$	$\left( \ell_1, \begin{smallmatrix} B & I \\ 1 & D \end{smallmatrix} \right)$	

#### 4.2.2 Coder pour tester

Le sous-dossier `turing/two-tapes/divisible-by-2-or-3`, accessible depuis le site de téléchargement de ce document, donne accès aux fichiers `divisible-by-2-or-3.txt`, `divisible-by-2-or-3-single-tape.txt` et `divisible-by-2-or-3-human-trick.txt` qui contiennent des codes utilisables sur le site <https://turingmachinesimulator.com>.

## 4.3 Écriture binaire d'un multiple soit de 2, soit de 3 mais pas des deux en même temps

### 4.3.1 Une table des transitions

On cherche ici à faire une table pour un « *ou exclusif* ». En utilisant l'identité booléenne  $A \text{ OUEX } B \Leftrightarrow (\text{NON } A \text{ ET } B) \text{ OU } (A \text{ ET NON } B)$ , nous pouvons appliquer ce que nous avons utilisé précédemment pour traduire les opérateurs logiques **ET**, **OU** et **NON**. Bien que théoriquement correct, ce raisonnement automatique va nous conduire à une trop « *grosse* » table des transitions.

Pour obtenir une table de taille raisonnable, nous allons ajouter une nouvelle bande. Expliquons comment faire automatiquement via la table « *optimisée* » de la section 4.2.1 (la méthode présentée est généralisable aux tables à plusieurs bandes).

1. La bande supplémentaire va juste nous servir à « *compter* » les cas gagnants.
2. Nous allons court-circuiter le 1<sup>er</sup> cas final pour aller à l'état  $sp_0$  tout en gardant la trace du succès du 1<sup>er</sup> test.
3. Il reste à gérer les états bloquants de la 2<sup>e</sup> machine lorsque la 1<sup>re</sup> a fonctionné. Dans l'optique d'utilisation séquentielle de machines, nous remettons à zéro la bande supplémentaire.

Nous utilisons  $\bullet$  pour simplifier la table. Dans cette écriture,  $\bullet$  indique un caractère quelconque. Dans une transition, il indique le caractère initial qui est donc inchangé.

*Phase 1 : a-t-on un pair via la bande du bas ?*

$\delta$	$\begin{smallmatrix} 0 \\ B \end{smallmatrix}$	$\begin{smallmatrix} 1 \\ B \end{smallmatrix}$	$\begin{smallmatrix} B \\ B \end{smallmatrix}$
$q_0$	$(\ell_0, \begin{smallmatrix} 0 \\ B, I \end{smallmatrix}, \begin{smallmatrix} D \\ I \end{smallmatrix})$	$(\ell_1, \begin{smallmatrix} 1 \\ B, I \end{smallmatrix}, \begin{smallmatrix} D \\ I \end{smallmatrix})$	
$\ell_0$	$(\ell_0, \begin{smallmatrix} 0 \\ B, I \end{smallmatrix}, \begin{smallmatrix} D \\ I \end{smallmatrix})$	$(\ell_1, \begin{smallmatrix} 1 \\ B, I \end{smallmatrix}, \begin{smallmatrix} D \\ I \end{smallmatrix})$	$(sp_0, \begin{smallmatrix} B \\ 1, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$
$\ell_1$	$(\ell_0, \begin{smallmatrix} 0 \\ B, I \end{smallmatrix}, \begin{smallmatrix} D \\ I \end{smallmatrix})$	$(\ell_1, \begin{smallmatrix} 1 \\ B, I \end{smallmatrix}, \begin{smallmatrix} D \\ I \end{smallmatrix})$	$(sp_0, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$

*Phase 2 : a-t-on un multiple de 3 via la bande du bas ?*

$\delta$	$\begin{smallmatrix} 0 \\ \bullet \end{smallmatrix}$	$\begin{smallmatrix} 1 \\ \bullet \end{smallmatrix}$	$\begin{smallmatrix} B \\ B \end{smallmatrix}$	$\begin{smallmatrix} B \\ 1 \end{smallmatrix}$
$sp_0$	$(si_0, \begin{smallmatrix} 0 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(si_1, \begin{smallmatrix} 1 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(f, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} I \\ I \end{smallmatrix})$	
$sp_1$	$(si_1, \begin{smallmatrix} 0 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(si_2, \begin{smallmatrix} 1 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$		$(f, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} I \\ I \end{smallmatrix})$
$sp_2$	$(si_2, \begin{smallmatrix} 0 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(si_0, \begin{smallmatrix} 1 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$		$(f, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} I \\ I \end{smallmatrix})$
$si_0$	$(sp_0, \begin{smallmatrix} 0 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(sp_2, \begin{smallmatrix} 1 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(f, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} I \\ I \end{smallmatrix})$	
$si_1$	$(sp_1, \begin{smallmatrix} 0 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(sp_0, \begin{smallmatrix} 1 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$		$(f, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} I \\ I \end{smallmatrix})$
$si_2$	$(sp_2, \begin{smallmatrix} 0 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$	$(sp_1, \begin{smallmatrix} 1 \\ \bullet, I \end{smallmatrix}, \begin{smallmatrix} G \\ I \end{smallmatrix})$		$(f, \begin{smallmatrix} B \\ B, I \end{smallmatrix}, \begin{smallmatrix} I \\ I \end{smallmatrix})$

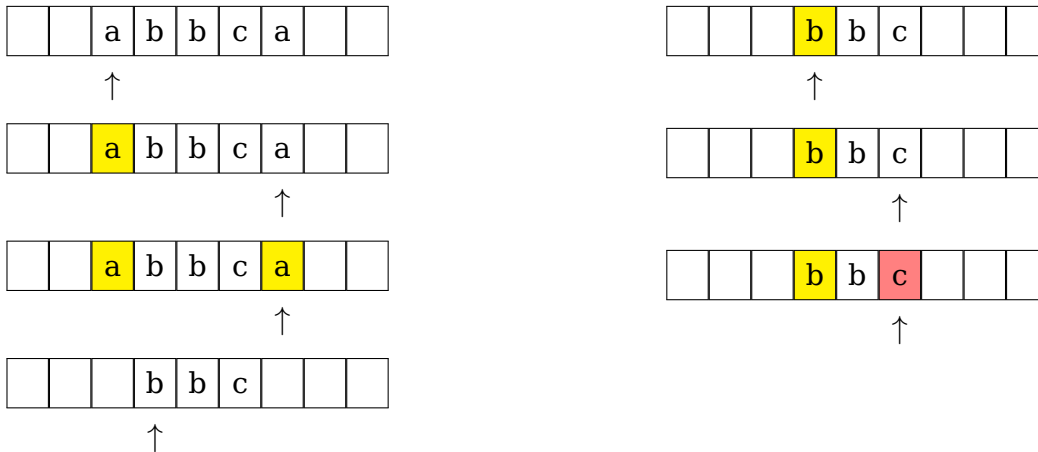
### 4.3.2 Coder pour tester

Sur le site de téléchargement de ce document se trouve divisible-by-2-xor-3.txt qui contient un code utilisable sur le site <https://turingmachinesimulator.com>. Voir le sous-dossier turing/two-tapes/divisible-by-2-xor-3.

## 5 Détecter les palindromes avec une seule bande

### 5.1 Un exemple avec le mot abbca

Voici les grandes étapes présentées sur deux colonnes.



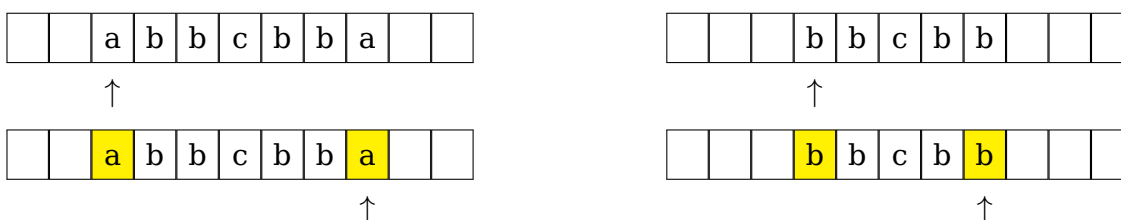
Qu'a-t-on fait ?

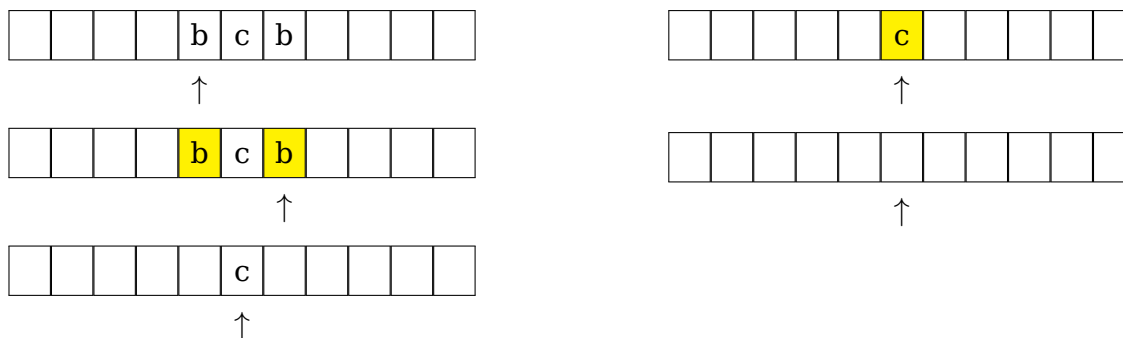
1. On note la lettre pointée par la tête de lecture. Commence alors une phase de recherche d'une lettre connue.
2. On avance tant que l'on ne rencontre pas une case vide. Une fois celle-ci repérée on revient d'une case en arrière.
3. On compare alors la lettre pointée par la tête de lecture avec celle de la phase de recherche en cours. Deux cas sont possibles.
  - a) Si les lettres sont différentes alors on ne fait plus rien. On a un état bloquant et le mot n'est pas validé.
  - b) Si les lettres sont identiques alors on efface la lettre en cours puis on va vers la gauche jusqu'à la prochaine case vide. Une fois celle-ci trouvée, on avance d'une case vers la droite pour effacer son contenu, puis on avance d'une autre case vers la droite pour recommencer les actions à partir du point 1.

La méthode ci-dessus est en fait incomplète comme nous allons le voir dans la section suivante avec un exemple de palindrome à repérer.

### 5.2 Un exemple avec le mot abbcbbba

On reprend la même méthodologie que dans la section précédente en la complétant sur la fin. Voici les étapes principales.





Quelle est la nouveauté ici ? Il faut juste ajouter après chaque nouvelle phase suite à un effacement la règle suivante : si la tête de lecture est sur une case vide alors on est dans un état final qui marque la validation d'un mot palindromique.

### 5.3 Une table des transitions

Formalisons les étapes utilisées dans les exemples précédents pour des mots sur l'alphabet  $\{a; b; c\}$  (la méthode est généralisable sans souci). Nous avons besoin des états suivants.

- $q_0$  est l'état initial.
- $q_1$  est l'état pour chaque début de nouvelle recherche de lettres identiques en début et en fin de mot à partir de la deuxième lettre.
- Il nous faut des états pour rechercher une lettre identique à droite, ce sera les états  $d_a$ ,  $d_b$  et  $d_c$ .
- Les états précédents vont en fait nous permettre d'arriver sur la case vide à droite de la dernière lettre. Nous ajoutons donc les états  $v_a$ ,  $v_b$  et  $v_c$  pour valider ou non la dernière lettre.
- Il nous faut aussi un état  $g$  pour retourner vers la gauche recommencer l'analyse avec la lettre suivante ainsi qu'un état  $e$  pour effacer la 1<sup>re</sup> lettre qui avait été repérée.
- Enfin l'état  $f$  sera notre état final indiquant qu'un mot est bien un palindrome.

Voici finalement la table des transitions de notre machine de Turing pour repérer les palindromes sur l'alphabet  $\{a; b; c\}$ .

$\delta$	a	b	c	B
$q_0$	$(d_a, a, D)$	$(d_b, b, D)$	$(d_c, c, D)$	
$q_1$	$(d_a, a, D)$	$(d_b, b, D)$	$(d_c, c, D)$	$(f, B, I)$
$d_a$	$(d_a, a, D)$	$(d_a, b, D)$	$(d_a, c, D)$	$(v_a, B, G)$
$d_b$	$(d_b, a, D)$	$(d_b, b, D)$	$(d_b, c, D)$	$(v_b, B, G)$
$d_c$	$(d_c, a, D)$	$(d_c, b, D)$	$(d_c, c, D)$	$(v_c, B, G)$
$v_a$	$(g, B, G)$			
$v_b$		$(g, B, G)$		
$v_c$			$(g, B, G)$	
$g$	$(g, a, G)$	$(g, b, G)$	$(g, c, G)$	$(e, B, D)$
$e$	$(q_1, B, D)$	$(q_1, B, D)$	$(q_1, B, D)$	$(q_1, B, D)$

## 5.4 Coder pour tester

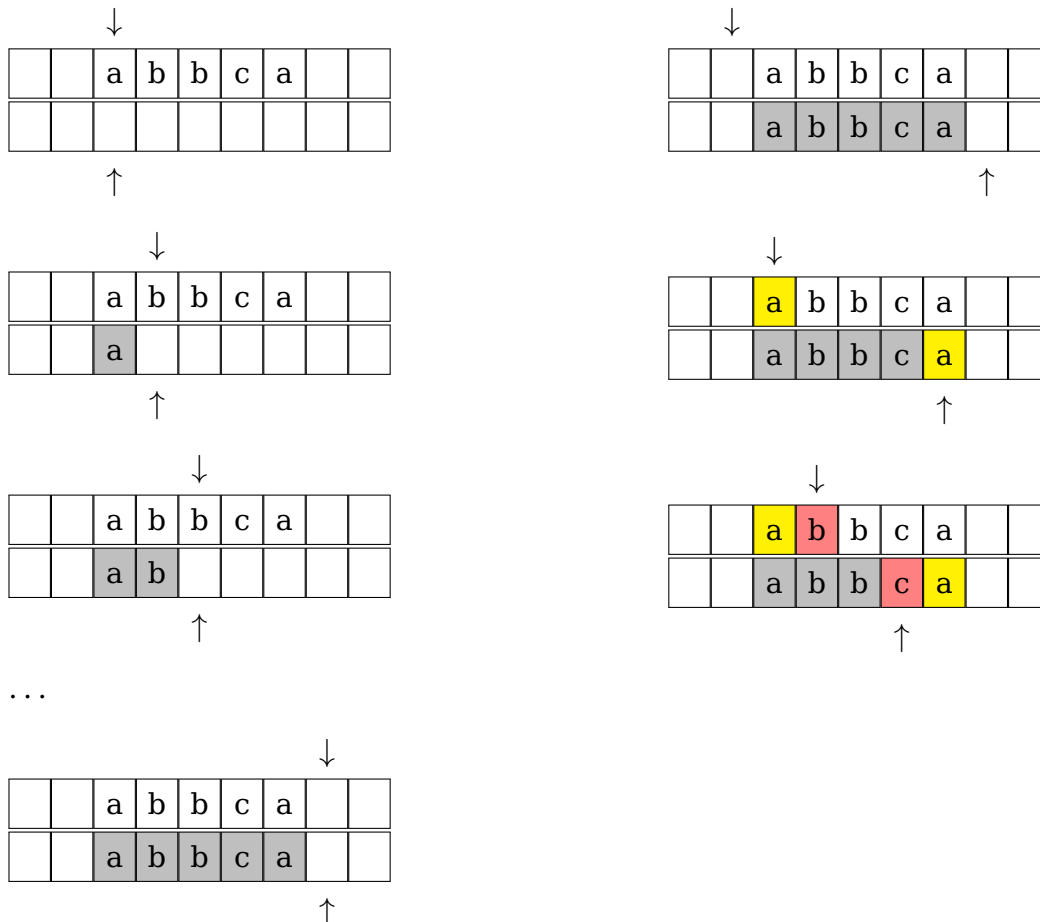
Sur le site de téléchargement de ce document, dans le sous-dossier `turing/palindrome`, se trouve le fichier `palindrome.txt` contenant un code utilisable pour des tests manuels sur le site <https://turingmachinesimulator.com>.



## 6 Détecter les palindromes avec deux bandes

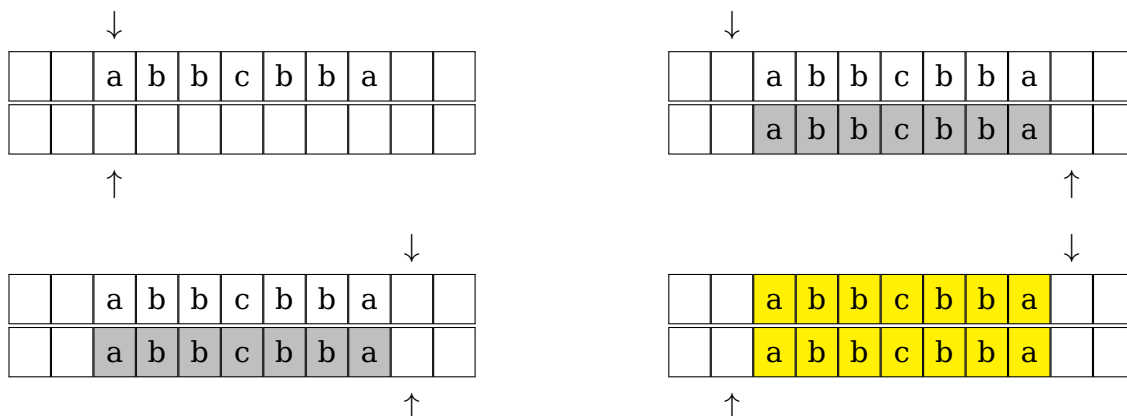
### 6.1 Un exemple avec le mot abbca

Voici les grandes étapes présentées sur deux colonnes avec une 1<sup>re</sup> phase de duplication du mot puis une 2<sup>e</sup> testant la présence ou non d'un palindrome.



### 6.2 Un exemple avec le mot abbcbbba

On fait comme précédemment en devant tout parcourir ! Voici les grandes étapes.



## 6.3 Une table des transitions

Nous donnons directement la table des transitions scindée en morceaux pour en faciliter la compréhension. Notez au passage que le nombre d'états ne dépend plus des lettres utilisées contrairement à la table présentée dans la section 5.3, cette dernière ayant des dimensions dépendant du nombre de lettres (*on voit donc ici toute la puissance de l'ajout d'une 2<sup>e</sup> bande*).

*Phase 1 : copie du mot suivi du retour vers la gauche en haut.*

$\delta$	a B	b B	c B	B B
$q_0$	$\left( d, \begin{smallmatrix} a & D \\ a & D \end{smallmatrix} \right)$	$\left( d, \begin{smallmatrix} b & D \\ b & D \end{smallmatrix} \right)$	$\left( d, \begin{smallmatrix} c & D \\ c & D \end{smallmatrix} \right)$	
$d$	$\left( d, \begin{smallmatrix} a & D \\ a & D \end{smallmatrix} \right)$	$\left( d, \begin{smallmatrix} b & D \\ b & D \end{smallmatrix} \right)$	$\left( d, \begin{smallmatrix} c & D \\ c & D \end{smallmatrix} \right)$	$\left( g, \begin{smallmatrix} B & G \\ B & I \end{smallmatrix} \right)$
$g$	$\left( g, \begin{smallmatrix} a & G \\ B & I \end{smallmatrix} \right)$	$\left( g, \begin{smallmatrix} b & G \\ b & I \end{smallmatrix} \right)$	$\left( g, \begin{smallmatrix} c & G \\ c & I \end{smallmatrix} \right)$	$\left( t, \begin{smallmatrix} B & D \\ B & G \end{smallmatrix} \right)$

*Phase 2 : comparaison « inversée » du mot et de sa copie.*

$\delta$	a a	b b	c c	B B
$t$	$\left( t, \begin{smallmatrix} a & D \\ a & G \end{smallmatrix} \right)$	$\left( t, \begin{smallmatrix} b & D \\ b & G \end{smallmatrix} \right)$	$\left( t, \begin{smallmatrix} c & D \\ c & G \end{smallmatrix} \right)$	$\left( f, \begin{smallmatrix} B & I \\ B & I \end{smallmatrix} \right)$

## 6.4 Coder pour tester

Sur le site de téléchargement de ce document se trouve `palindrome-2-tapes.txt` qui contient un code utilisable sur le site <https://turingmachinesimulator.com>. Voir le sous-dossier `turing/palindrome-2-tapes`.

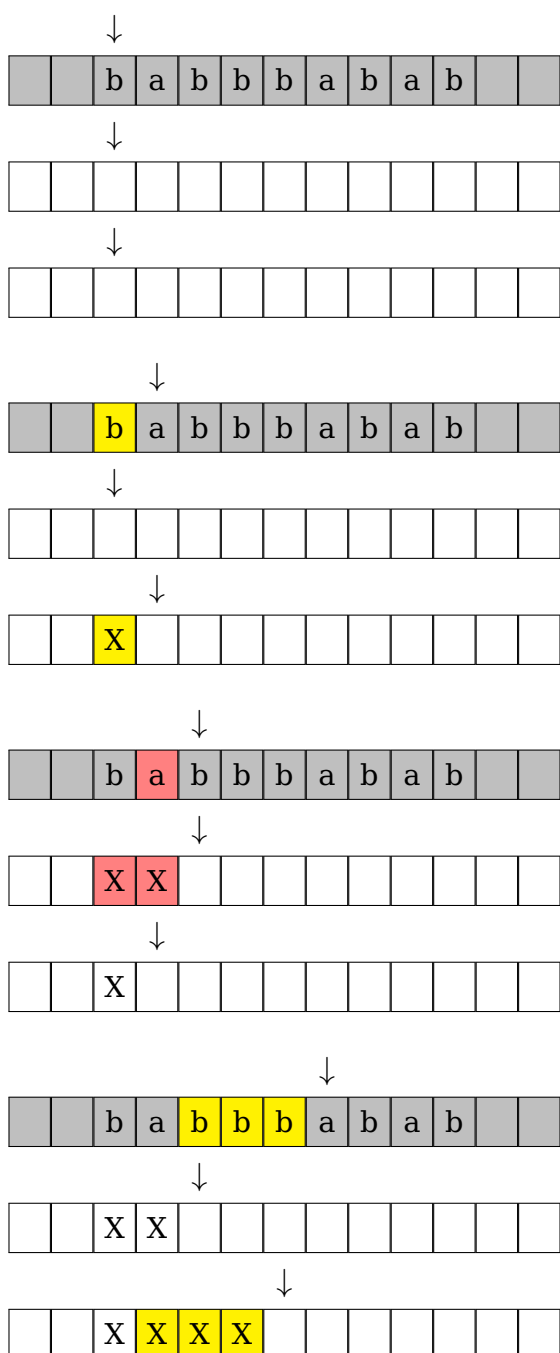
## 7 Deux fois plus de b que de a

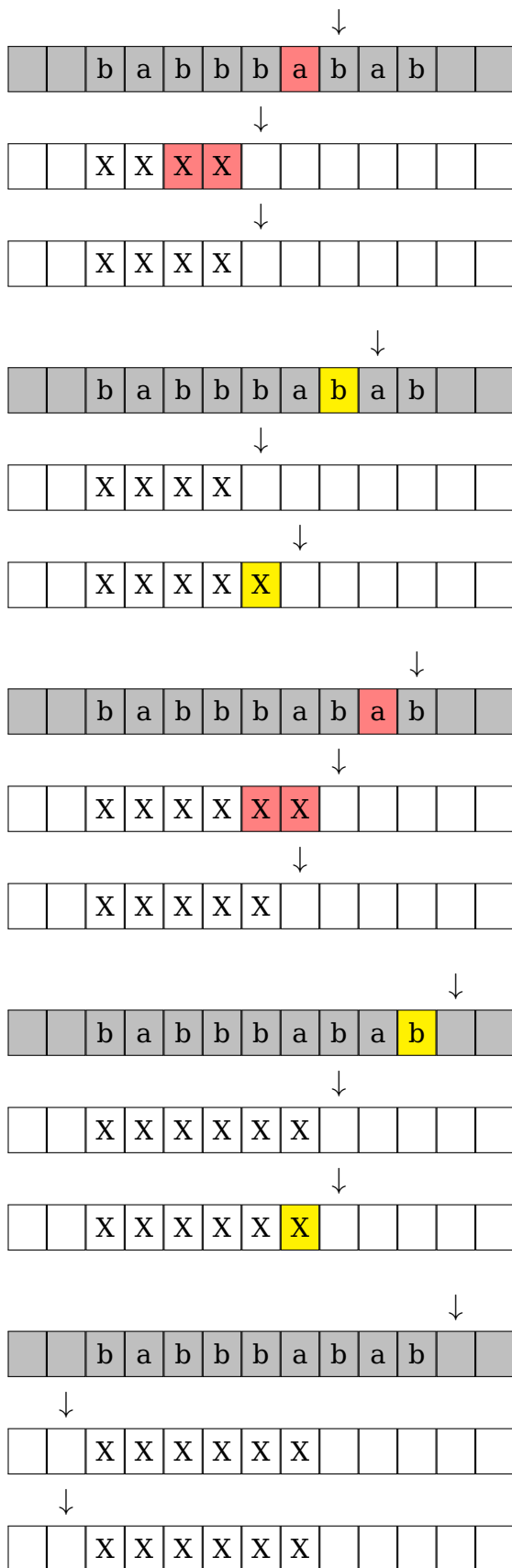
### 7.1 Un exemple avec le mot babbbabab

On souhaite savoir si un mot contient deux fois plus de b que de a. Pour faire ceci, nous allons utiliser trois bandes comme suit.

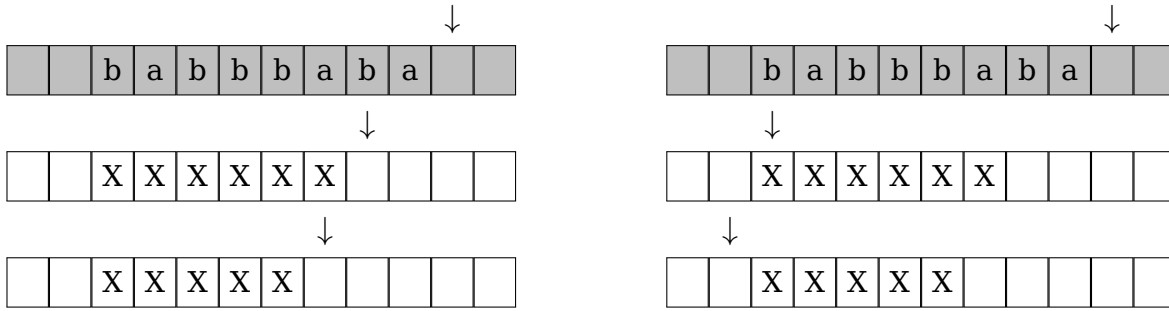
1. La bande du haut contient le mot initial qui sera juste lu de gauche à droite.
2. Lors de la lecture si l'on tombe sur un a, on écrit deux symboles X sur la 2<sup>e</sup> bande, et si c'est un b on écrit un unique symbole X sur la 3<sup>e</sup>.
3. Une fois le mot lu, il suffit de lire en simultanée les bandes 2 et 3 de droite à gauche pour arriver à un état vide et vide sur ses deux bandes pour valider le mot.

Voici l'application de cette méthode au mot babbbabab.





Avec le mot babbbaba qui a un b de moins à droite, on arrête car on arrive sur X non au-dessus d'un autre comme le montre le schéma résumé ci-après.



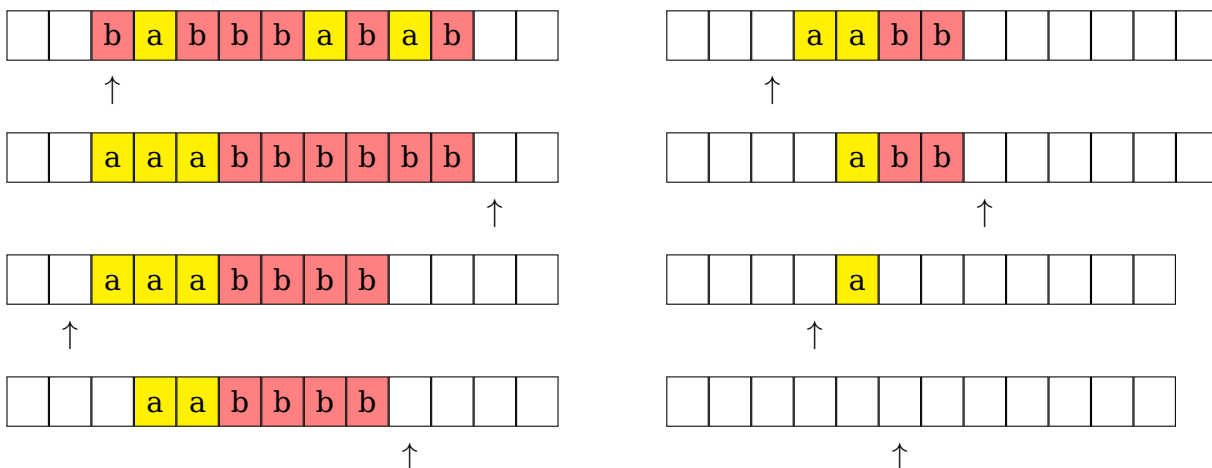
## 7.2 Une table des transitions

La table des transitions de la méthode proposée est facile à établir. Là voici.

$\delta$	a <i>B</i> <i>B</i>	b <i>B</i> <i>B</i>	<i>B</i> <i>B</i> <i>B</i>	<i>B</i> <i>X</i> <i>X</i>
$q_0$	$\begin{pmatrix} a & I \\ q'_a, X, D \\ B & I \end{pmatrix}$	$\begin{pmatrix} b & D \\ d, B, I \\ X & D \end{pmatrix}$		
$q'_a$	$\begin{pmatrix} a & D \\ d, X, D \\ B & I \end{pmatrix}$			
$d$	$\begin{pmatrix} a & I \\ q'_a, X, D \\ B & I \end{pmatrix}$	$\begin{pmatrix} b & D \\ d, B, I \\ X & D \end{pmatrix}$	$\begin{pmatrix} B & I \\ g, B, G \\ B & G \end{pmatrix}$	
$g$			$\begin{pmatrix} B & I \\ f, B, I \\ B & I \end{pmatrix}$	$\begin{pmatrix} B & I \\ g, X, G \\ X & G \end{pmatrix}$

## 7.3 Avec une seule bande ?

Il a été facile de résoudre la problème avec trois bandes. Essayons de voir si l'on peut se limiter à une seule bande<sup>7</sup> en n'utilisant pas la lettre supplémentaire X<sup>8</sup>. Voici les grandes lignes de la méthode.



7. Théoriquement on sait que toute machine à  $n$  bandes peut être traduite en une machine à une seule bande. Malheureusement le procédé ne produit pas forcément des machines qu'un humain aurait conçu tout seul donc nous allons laisser de côté ce procédé.

8. Là est le mini-défi.

On aboutit alors à la table des transitions suivante où le plus délicat est la réorganisation des a et des b (*on effectue un tri à bulles*). L'état  $g$  et ceux de type  $s$  sont chargés de cette réorganisation, tandis que les états de type  $e$  s'occupent de la procédure d'effacement.

*Phase 1 : réorganisation (tri à bulles).*

$\delta$	a	b	$B$
$q_0$	$(s_a, a, D)$	$(s_b, b, D)$	
$s_a$	$(s_a, a, D)$	$(s_b, b, D)$	$(e_b, B, G)$
$s_b$	$(s'_b, b, G)$	$(s_b, b, D)$	$(e_b, B, G)$
$s'_b$		$(s''_b, a, D)$	
$s''_b$	$(s'_b, b, G)$	$(s''_b, b, D)$	$(g, b, G)$
$g$	$(g, a, G)$	$(g, b, G)$	$(q_0, B, D)$

*Phase 2 : effacement.*

$\delta$	a	b	$B$
$e_b$		$(e'_b, B, G)$	$(f, B, I)$
$e'_b$		$(g_a, B, G)$	
$g_a$	$(g_a, a, G)$	$(g_a, b, G)$	$(e_a, B, D)$
$e_a$	$(d_b, B, D)$		
$d_b$	$(d_b, a, D)$	$(d_b, b, D)$	$(e_b, B, G)$

## 7.4 Coder pour tester

Sur le site de téléchargement de ce document se trouvent des codes utilisables sur le site <https://turingmachinesimulator.com>. Voir les fichiers suivants dans le sous-dossier `turing/twice-more-b-than-a`.

- `twice-more-b-than-a-3-tapes.txt`
- `twice-more-b-than-a-1-tape.txt`