BROUILLON - Quelques machines de Turing déterministes

Christophe BAL

7 Février 2020 – 3 Mars 2020

Document, avec son source LaTeX, disponible sur la page `https://github.com/bc-writing/drafts`.

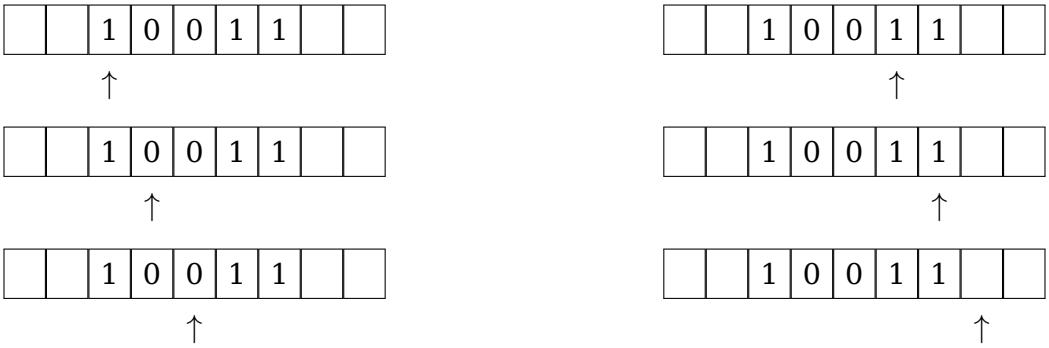
Mentions « légales »

Table des matières

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 La 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.
- ℓ_0 et ℓ_1 sont les états indiquant que la dernière[1] 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.

δ	0	1	B
q_0	$(\ell_0\,,0\,,D)$	$(\ell_1\,,1\,,D)$	
ℓ_0	$(\ell_0\,,0\,,D)$	$(\ell_1\,,1\,,D)$	$(f\,,B\,,I)$
ℓ_1	$(\ell_0\,,0\,,D)$	$(\ell_1\,,1\,,D)$	

1. ℓ 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

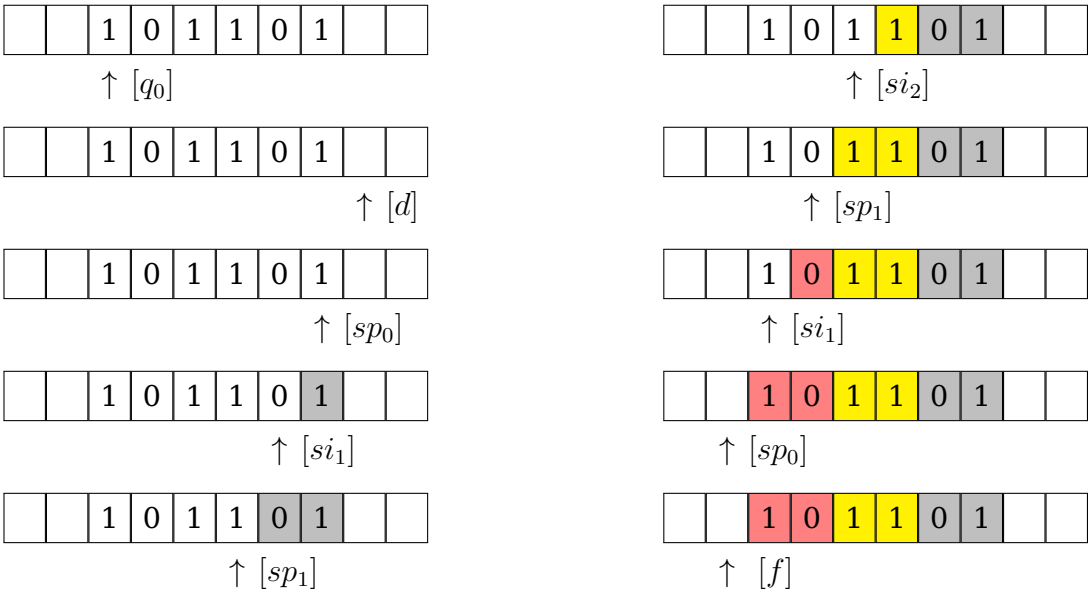
2.1 La méthode

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 [2].

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

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 $[10]_2=2$, $[11]_2=3$ et $[01]_2=1$. Voici alors comment analyser ce nombre via une machine de Turing. On a jouter 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.



Voici les points clés dans les étapes ci-dessus.

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

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 La table des transitions

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

δ	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 $k \in \mathbb{N}^*$. Ceci étant dit, les tables des transitions vont croître de façon exponentielle !

On peut en fait construire un automate d'état déterministe fini qui reconnait les multiples de 3, et plus généralement de $d \in \mathbb{N} - \{0\,;1\}$, à partir de l'écriture d'un nombre dans une base quelconque $b \in \mathbb{N} - \{0\,;1\}$. 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 Les 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[3]. **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.

δ	0	1	B
q_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
ℓ_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
ℓ_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.

δ	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 automatisables mais non nécessairement optimales. Voir à ce sujet la section 4.1.2.

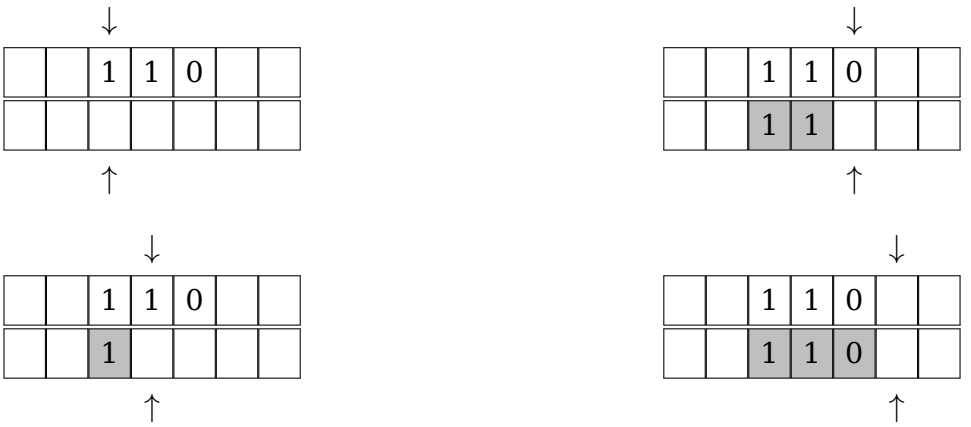
4.1 Écriture binaire d'un multiple de 2 et 3

4.1.1 Les tables des transitions

Une 1re 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 [4]. Cette démarche est valable car les machines que nous avons présentées ne modifient pas la donnée en entrée [5]. Ceci nous conduit à la table des transitions ci-après où le nouvel état m indique qu'un multiple de 3 a été repéré.

δ	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)
ℓ_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	(f, B, I)
ℓ_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 2e bande. Voici ce que cela peut donner schématiquement *(nous mettons la 1re 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 $\begin{smallmatrix}1\\B\end{smallmatrix}$ 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. De plus pour alléger la présentation, nous utilisons $\begin{smallmatrix}1\\\bullet\end{smallmatrix}$ pour indiquer un 1 au-dessus de n'importe quelle case *(lorsque ce raccourci est utilisé pour une règle d'écriture, ceci indique que le caractère correspondant à $\bullet$ est inchangé)*.

Phase 1 : copie de l'entrée.

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

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

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

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

δ	$\overset{\bullet}{0}$	$\overset{\bullet}{1}$	$\overset{\bullet}{B}$
t_p	$\left(\ell_0, \overset{\bullet}{0}, \overset{I}{D}\right)$	$\left(\ell_1, \overset{\bullet}{1}, \overset{I}{D}\right)$	
ℓ_0	$\left(\ell_0, \overset{\bullet}{0}, \overset{I}{D}\right)$	$\left(\ell_1, \overset{\bullet}{1}, \overset{I}{D}\right)$	$\left(f, \overset{\bullet}{B}, \overset{I}{I}\right)$
ℓ_1	$\left(\ell_0, \overset{\bullet}{0}, \overset{I}{D}\right)$	$\left(\ell_1, \overset{\bullet}{1}, \overset{I}{D}\right)$	

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.

δ	0	1	B
q_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
ℓ_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	(sp_0, B, G)
ℓ_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 Les 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 2e traitement. On va donc rendre tous ces états finaux. Ceci ne suffit pas car il faut aussi que les états bloquants[6] de la 1re 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 »**

δ	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)
ℓ_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	(f, B, I)
ℓ_1	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	

***Table* « optimisée »**

δ	0	1	B
q_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	
ℓ_0	$(\ell_0, 0, D)$	$(\ell_1, 1, D)$	(f, B, G)
ℓ_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.

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

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

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

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

δ	$\begin{smallmatrix}\bullet\\0\end{smallmatrix}$	$\begin{smallmatrix}\bullet\\1\end{smallmatrix}$	$\begin{smallmatrix}\bullet\\B\end{smallmatrix}$
t_p	$\left(\ell_0, \begin{smallmatrix}\bullet & I\\0 & D\end{smallmatrix}\right)$	$\left(\ell_1, \begin{smallmatrix}\bullet & I\\1 & D\end{smallmatrix}\right)$	
ℓ_0	$\left(\ell_0, \begin{smallmatrix}\bullet & I\\0 & D\end{smallmatrix}\right)$	$\left(\ell_1, \begin{smallmatrix}\bullet & I\\1 & D\end{smallmatrix}\right)$	$\left(f, \begin{smallmatrix}\bullet & I\\B & I\end{smallmatrix}\right)$
ℓ_1	$\left(\ell_0, \begin{smallmatrix}\bullet & I\\0 & D\end{smallmatrix}\right)$	$\left(\ell_1, \begin{smallmatrix}\bullet & I\\1 & D\end{smallmatrix}\right)$	

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 Les tables des transitions

On cherche ici à faire une table pour un ou exclusif. En utilisant l'identité booléenne A **OUEX** $B \rightleftharpoons$ (**NON** A **ET** B) **OU** (A **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 1er cas final en passant via l'état $presp_0$ avant d'aller à l'état sp_0 qui sert à passer la main à la 2e machine. Ce court-circuit nous sert à garder la trace du succès du 1er test.
3. Il reste à gérer les états bloquants de la 2e machine lorsque la 1re a fonctionné. Dans l'optique d'utilisation séquentielle de machines, nous remettons à zéro la bande supplémentaire.

***Table* « optimisée »**

δ	$\begin{smallmatrix}\bullet\\0\end{smallmatrix}$	$\begin{smallmatrix}\bullet\\1\end{smallmatrix}$	$\begin{smallmatrix}B\\B\end{smallmatrix}$	$\begin{smallmatrix}1\\B\end{smallmatrix}$
q_0	$\left(\ell_0, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\D\end{smallmatrix}\right)$	$\left(\ell_1, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\D\end{smallmatrix}\right)$		
ℓ_0	$\left(\ell_0, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\D\end{smallmatrix}\right)$	$\left(\ell_1, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\D\end{smallmatrix}\right)$	$\left(presp_0, \begin{smallmatrix}1\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$	
$presp_0$				$\left(sp_0, \begin{smallmatrix}1\\B\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$
ℓ_1	$\left(\ell_0, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\D\end{smallmatrix}\right)$	$\left(\ell_1, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\D\end{smallmatrix}\right)$	$\left(sp_0, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	
sp_0	$\left(si_0, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(si_1, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(f, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$	
sp_1	$\left(si_1, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(si_2, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$		$\left(f, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$
sp_2	$\left(si_2, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(si_0, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$		$\left(f, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$
si_0	$\left(sp_0, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(sp_2, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(f, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$	
si_1	$\left(sp_1, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(sp_0, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$		$\left(f, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$
si_2	$\left(sp_2, \begin{smallmatrix}\bullet\\0\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$	$\left(sp_1, \begin{smallmatrix}\bullet\\1\end{smallmatrix}, \begin{smallmatrix}I\\G\end{smallmatrix}\right)$		$\left(f, \begin{smallmatrix}B\\B\end{smallmatrix}, \begin{smallmatrix}I\\I\end{smallmatrix}\right)$

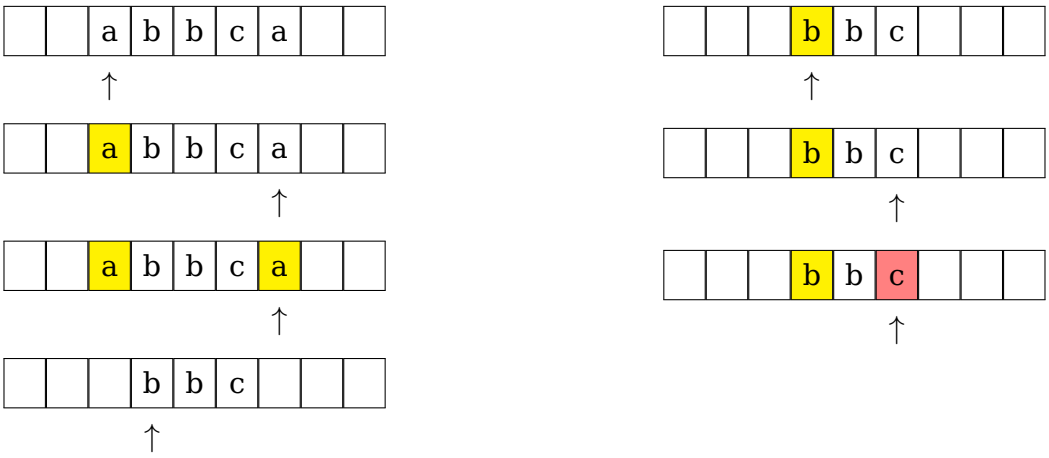
4.3.2 Coder pour tester

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.



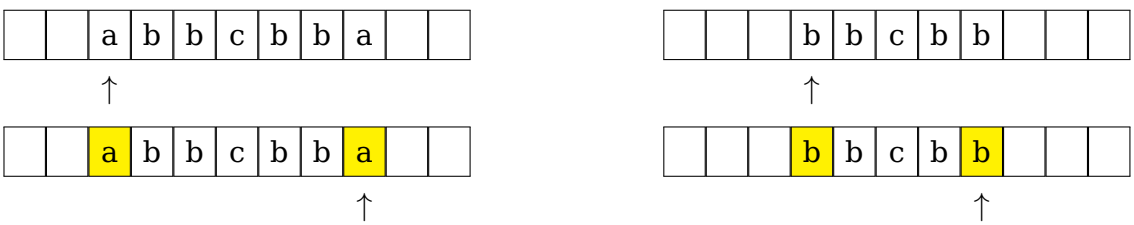
Qu'a-t-on fait ?

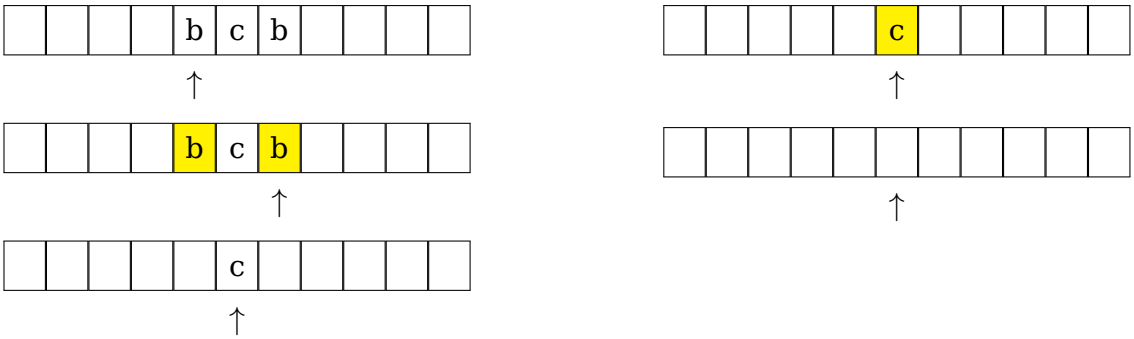
1. On note la lettre pointée par la tête de lecture. Commence alors une phase de recherche d'une lettre connue.
2. On avance tant que l'on ne rencontre par une case vide. Une fois celle-ci repérée on revient d'une case en arrière.
3. On compare alors la lettre pointée par la tête de lecture avec celle 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 abbcbba

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 La table des transitions

Formalisons les étapes utilisées dans les exemples précédents pour des mots sur l'alphabet $\{\mathsf{a}\,;\mathsf{b}\,;\mathsf{c}\}$ *(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 1re 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 $\{\mathsf{a}\,;\mathsf{b}\,;\mathsf{c}\}$.

δ	a	b	c	B
q_0	$(\,d_{\mathsf{a}}\,,\mathsf{a}\,,D\,)$	$(\,d_{\mathsf{b}}\,,\mathsf{b}\,,D\,)$	$(\,d_{\mathsf{c}}\,,\mathsf{c}\,,D\,)$	
q_1	$(\,d_{\mathsf{a}}\,,\mathsf{a}\,,D\,)$	$(\,d_{\mathsf{b}}\,,\mathsf{b}\,,D\,)$	$(\,d_{\mathsf{c}}\,,\mathsf{c}\,,D\,)$	$(\,f\,,B\,,I\,)$
d_a	$(\,d_{\mathsf{a}}\,,\mathsf{a}\,,D\,)$	$(\,d_{\mathsf{a}}\,,\mathsf{b}\,,D\,)$	$(\,d_{\mathsf{a}}\,,\mathsf{c}\,,D\,)$	$(\,v_{\mathsf{a}}\,,B\,,G\,)$
d_b	$(\,d_{\mathsf{b}}\,,\mathsf{a}\,,D\,)$	$(\,d_{\mathsf{b}}\,,\mathsf{b}\,,D\,)$	$(\,d_{\mathsf{b}}\,,\mathsf{c}\,,D\,)$	$(\,v_{\mathsf{b}}\,,B\,,G\,)$
d_c	$(\,d_{\mathsf{c}}\,,\mathsf{a}\,,D\,)$	$(\,d_{\mathsf{c}}\,,\mathsf{b}\,,D\,)$	$(\,d_{\mathsf{c}}\,,\mathsf{c}\,,D\,)$	$(\,v_{\mathsf{c}}\,,B\,,G\,)$
v_a	$(\,g\,,B\,,G\,)$			
v_b		$(\,g\,,B\,,G\,)$		
v_c			$(\,g\,,B\,,G\,)$	
g	$(\,g\,,\mathsf{a}\,,G\,)$	$(\,g\,,\mathsf{b}\,,G\,)$	$(\,g\,,\mathsf{c}\,,G\,)$	$(\,e\,,B\,,D\,)$
e	$(\,q_1\,,B\,,D\,)$	$(\,q_1\,,B\,,D\,)$	$(\,q_1\,,B\,,D\,)$	$(\,q_1\,,B\,,D\,)$

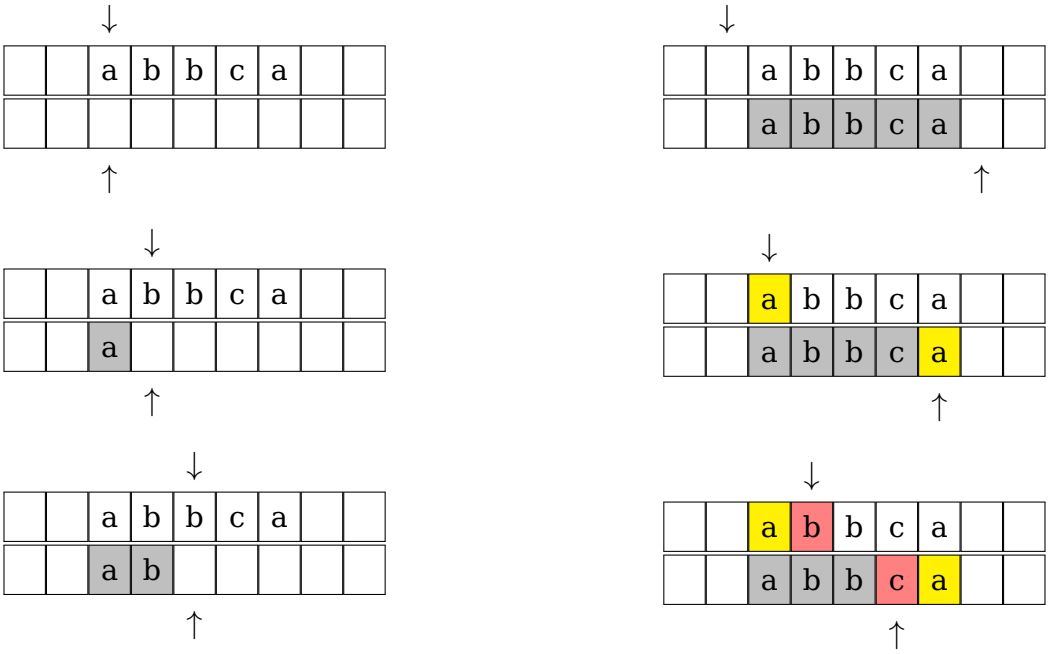
5.4 Coder pour tester

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`.

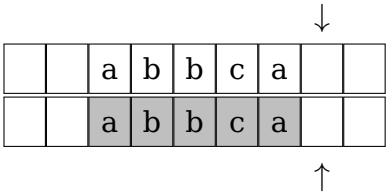
6 Détecter les palindromes avec deux bandes

6.1 Un exemple avec le mot abbca

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

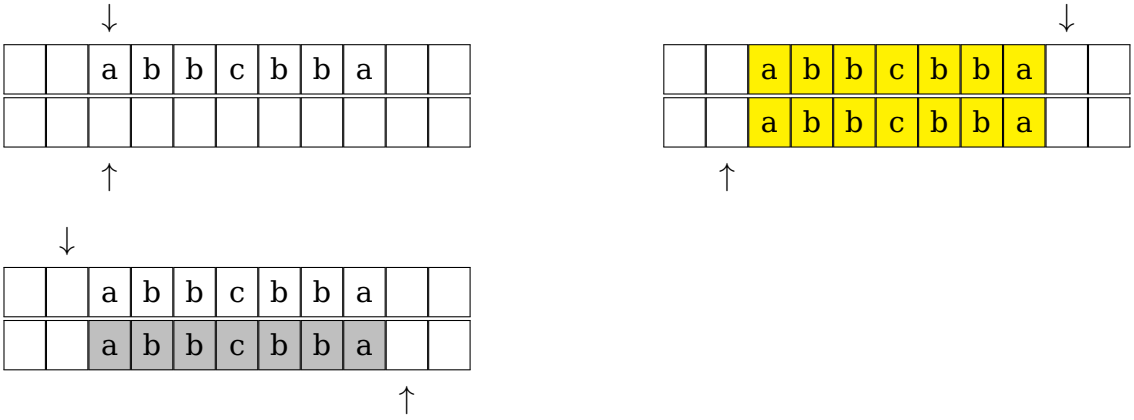


...



6.2 Un exemple avec le mot abbcbba

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



6.3 La 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 nombres de lettres *(on voit donc ici toute la puissance de l'ajout d'une 2^e bande)*.

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

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

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

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

6.4 Coder pour tester

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

7 Jouer à saute moutons en bicolorama

7.1 Les règles du jeu

7.2 Une résolution possible et sa table des transitions

7.3 Coder pour tester