Automates cellulaires unidimensionnels

Année 2016 – 2017

Loïc TAIEB, élève de 1ère S

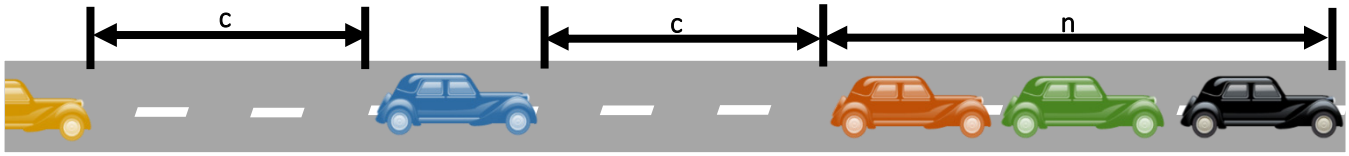
Encadré par Alice ORDINES

Établissement : Ecole de Provence, Marseille (13)

Chercheur : Pascal HUBERTboucle, Institut de Mathématiques de Marseille.

PRÉSENTATION DU PROBLÈME

On considère une route avec plusieurs voitures. Toutes les voitures avancent à la même vitesse et dans la même direction. Une voiture peut avancer si et seulement si aucune autre voiture ne se trouve devant elle.



Quelles configurations amènent à un embouteillage qui ne pourra pas se débloquer ?

On peut répondre à cette question en simulant cette situation dans un automate cellulaire(1).

UTILISATION DU LOGICIEL

Pour résoudre ce problème, j'ai créé un automate cellulaire qui simule une route finie de longueur choisie et en fonction d'une configuration(2) déterminée par l'utilisateur. Ce dernier est invité à choisir une entrée périodique de voitures afin de simuler une route infinie.

Une fois ces paramètres enregistrés, l'algorithme affiche la route qui évolue au cours du temps. Afin de trouver les dispositions qui amèneraient à un embouteillage, on teste l'algorithme avec différentes configurations de base, et une entrée périodique de voitures différente.

Ce scénario simple peut être interprété comme un automate cellulaire, programmable sous forme d'un algorithme.

CODE DE L'ALGORITHME (EN TI-BASIC)

Bien que le fonctionnement de l'algorithme semble d'une simplicité apparente, celui-ci est en réalité plutôt complexe.

Il est composé de 117 lignes de code en TI-Basic, le langage de programmation natif des calculatrices TI 80.

Chaque partie du programme sera détaillée dans la partie Fonctionnement de l'algorithme dans le paragraphe de la couleur correspondante. Celui-ci fonctionne parfaitement sur les calculatrices TI-83 Premium CE et devrait fonctionner sur la plupart des calculatrices graphiques TI.

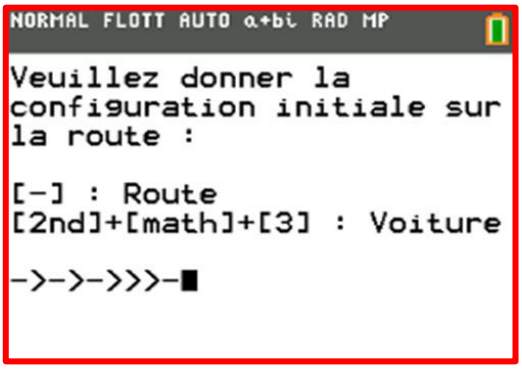
ATTENTION

Le code présent ci-dessous n'est pas utilisable directement sur la calculatrice. Des aménagements ont été apportés pour en améliorer la lisibilité :

- Ajout avant certaines lignes pour mettre en évidence les blocs de condition et les boucles(3).
- Le caractère [[ ], utilisé dans TI-Basic pour faire référence à des noms de listes, a été remplacé par des [L] en indice, l'apparence qu'il a sur l'éditeur de code de la calculatrice.

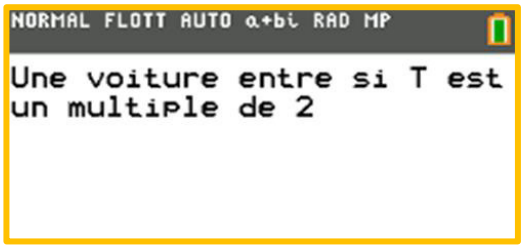
Des versions compatibles de l'algorithme seront jointes avec cet article

La dernière mise à jour système de la calculatrice est nécessaire(4) pour utiliser cet algorithme, à cause de la fonction ***versChaîne***, utilisée pour convertir des nombres en chaînes de caractères, absente dans les anciennes versions du système de la calculatrice.(5)




```
EffListe ₗROUTE
0→T
Lbl 1
EffÉcran
Disp "Veuillez donner la"
Disp "configuration initiale sur"
Disp "la route :"
Disp " "
Disp "[-] : Route"
Disp "[2nd]+[math]+[3] : Voiture"
Disp " "
Input "",Chn1
EffÉcran
Disp "Chargement ..."
For(I,1,longueur(Chn1))
  If sousch(Chn1,I,1)="-"
  Then
    0→ₗROUTE(I)
  Else
    If sousch(Chn1,I,1)=">"
    Then
      1→ₗROUTE(I)
    Else
```

```
        EffÉcran
        Menu("Erreur : Symbole invalide","Retour",1,"Quitter",0)
      End
    End
End
Lbl 8
Menu("Arrêt entre étapes ?","Oui",2,"Non",3,"Retour",1)
Lbl 2
1→P
Goto 4
Lbl 3
0→P
Goto 4
Lbl 4
Menu("Voitures périodiques ?","Oui",5,"Non",6,"Retour",8)
Lbl 5
EffÉcran
Disp "Une voiture si T est un"
Input "multiple de ",N
1→M
If N<1 ou N≠ent(N)
Then
  Menu("Erreur : Valeur invalide","Retour",5,"Quitter",0)
End
Goto 7
Lbl 6
0→M
1→N
Goto 7
Lbl 7
EffÉcran
Disp "Chargement ..."
While 1
  If T/N=ent(T/N) et T≠0 et M≠0
  Then
    1→ₗROUTE(1)
  End
  " "→Chn1
  For(I,1,dim(ₗROUTE))
    If ₗROUTE(I)=0
    Then
      Chn1+"-"→Chn1
    Else
      Chn1+">"→Chn1
    End
  End
  sousch(Chn1,2,longueur(Chn1)-1)→Chn1
  EffÉcran
  If longueur(Chn1)>26
  Then
    For(I,1,abs(ent(-(longueur(Chn1)/26))))
      If I≠26
      Then
        Disp sousch(Chn1,26*I+1,26)
      Else
```

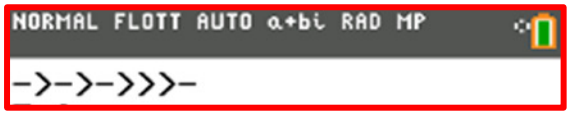


```
            Disp sousch(Chn1,26*I+1,Longueur(Chn1)-(26*I))
          End
        End
      Else
        Disp Chn1
      End
      Disp "T="+versChaîne(T)
      0→K
      If P=1
      Then
        Pause
      End
      For(I,1,dim(ʟROUTE))
        If I≠dim(ʟROUTE)
        Then
          If ʟROUTE(I)=1 et ʟROUTE(I+1)=0
          Then
            0→ʟROUTE(I)
            2→ʟROUTE(I+1)
          End
          If ʟROUTE(I)=2
          Then
            1→ʟROUTE(I)
          End
        End
      End
      If ʟROUTE(dim(ʟROUTE))=2
      Then
        1→ʟROUTE(dim(ʟROUTE))
      Else
        If ʟROUTE(dim(ʟROUTE))=1
        Then
          0→ʟROUTE(dim(ʟROUTE))
        End
      End
      T+1→T
End
Lbl 0
Stop
```

FONCTIONNEMENT DE L'ALGORITHME

Lors du lancement de l'algorithme, il réinitialise les variables. Puis, on demande à l'utilisateur d'entrer la configuration initiale sur la route. La route est symbolisée par des signes [-] et les voitures par des signes [>]

Par exemple : **--->-->---->>---**



Ensuite, l'algorithme analyse la saisie de l'utilisateur et, en fonction de ces caractères, il la convertit en une suite binaire de 0 et de 1. Les 0 correspondent à des signes [-] et les 1 correspondent à des signes [>]. Sinon, s'il détecte un caractère invalide, il affiche un message d'erreur invitant l'utilisateur à ressaisir sa configuration. On note cette suite $(\mathcal{R}_{n,t})_{n\in\mathbb{N}^*,t\in\mathbb{N}}$, avec n le rang de la suite et t l'étape de la suite.(6)

Exemple de suite binaire : **0001001000011000**

Une fois la configuration correcte, l'algorithme demande à l'utilisateur s'il souhaite avoir des pauses entre les différentes étapes du programme principal, et s'il souhaite une entrée périodique de voitures pour simuler une route infinie. Si l'utilisateur souhaite une entrée périodique de voitures, il devra entrer la fréquence d'apparition des voitures, notée N. Il lui propose également de revenir en arrière s'il s'est trompé quelque part.(7)

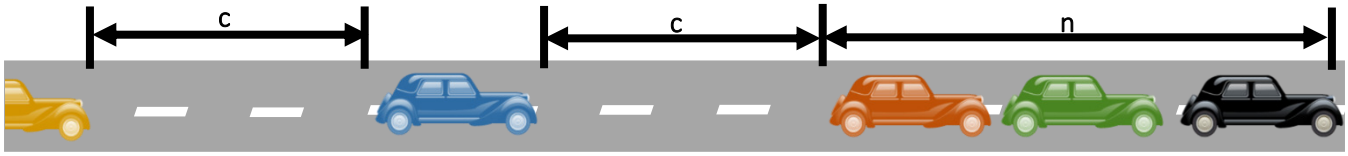
Après un temps de chargement, le programme principal se lance : il consiste en une interface graphique affichant l'état de la route symbolisée par le même code défini précédemment à une étape T donnée. En arrière-plan (ce que l'utilisateur ne peut pas voir), l'algorithme détermine la position des voitures à l'étape suivante : une voiture avance d'une case si la case devant elle est libre(8).

Ce système fonctionne de la manière suivante :

- S'il y a une entrée périodique de voitures, et que T est un multiple de la fréquence d'apparition N (c.-à-d. si $\frac{T}{N} \in \mathbb{N}$), l'algorithme place une voiture en première position sur la route, en remplaçant le premier chiffre de la suite binaire par un 1 ($\mathcal{R}_{1,T} = 1$), qui symbolise une voiture dans le programme.
- Ensuite, l'algorithme met à jour l'état de la route en affichant la suite $(\mathcal{R}_{n,T})_{n\in\mathbb{N}^*}$, en remplaçant les 0 par des [-] et les 1 par des [>]. Il affiche la route sur plusieurs lignes si celle-ci est trop grande pour l'écran de la calculatrice. Il met le programme sur pause si l'utilisateur l'a demandé.
- L'algorithme parcourt la suite binaire(9) et lorsqu'il détecte un 1 à un rang I (c.-à-d. si $\mathcal{R}_{I,T} = 1$), il le mémorise et analyse également le rang suivant c'est-à-dire le rang $I + 1$.
- Si le chiffre au rang $I + 1$ est 0 (c.-à-d. si $\mathcal{R}_{I+1,T} = 0$), alors il le remplace par 2, qui symbolise une voiture venant d'avancer d'une case ($\mathcal{R}_{I+1,T} = 2$). Il affecte également 0 au rang I puisque la voiture n'y est plus ($\mathcal{R}_{I,T} = 0$). (10)
- Il effectue ces deux actions en boucle jusqu'à ce qu'il atteigne la fin de la liste
- Il remplace enfin tous les 2 de la suite par des 1 et passe à l'étape suivante.

OBSERVATIONS

On considère une route de longueur infinie (simulable grâce à l'algorithme). Des voitures séparées de c cases se suivent, avec c naturel non nul (en effet, si $c = 0$, les voitures seraient collées et formeraient un embouteillage infini, qui ne pourrait pas se débloquer), devancées d'un embouteillage de n voitures, avec n naturel strictement supérieur à 1 (en effet, si $n = 1$, il n'y aurait pas d'embouteillage).



À chaque étape, la première voiture de l'embouteillage avance puisqu'il y a une place libre devant elle. Toutes les c étapes, si l'embouteillage ne s'est pas débloqué, c'est-à-dire si $c < n$, la voiture derrière l'embouteillage s'ajoute à celui-ci, qui recule d'une case.

On remarque que, lorsque $c = 1$, que l'embouteillage recule indéfiniment sans se réduire.

RÉSOLUTION DU PROBLÈME

On définit la fonction $\lfloor\cdot\rfloor$ qui, à tout réel, associe un entier relatif qui correspond à la partie entière de ce réel.

$$\lfloor\cdot\rfloor : \mathbb{R} \longmapsto \mathbb{Z}$$

$$\lfloor\cdot\rfloor : x \longmapsto n$$

$$\text{Avec } n \in \mathbb{Z} \text{ et } n \leq x < n+1$$

La partie entière de x se note $\lfloor x \rfloor$

Le nombre de voitures dans l'embouteillage peut être représenté par la suite suivante :

$$u_t = n - t + \left\lfloor \frac{t}{c} \right\rfloor$$

$$\text{Avec } t \in \mathbb{N} \cap \{[0;x]/u_x = 0\}$$

Cette suite est composée de deux parties :

- $\mathrm{n} - \mathrm{t}$: Pour $\mathrm{t} = 0$, ce nombre est égal à n, qui est le nombre de voitures présentes initialement dans l'embouteillage. Il se décrémente de 1 à chaque étape. Ce nombre symbolise la propriété « À chaque étape, la première voiture de l'embouteillage avance puisqu'il y a une place libre devant elle. »
- $\left\lfloor \frac{\mathrm{t}}{\mathrm{c}} \right\rfloor$: Ce nombre, initialement égal à 0, s'incrémente de 1 toutes les c étapes. Il symbolise la propriété « Toutes les c étapes, la voiture derrière l'embouteillage s'ajoute à celui-ci. »

Pour vérifier notre théorie, on calcule u_t pour $c = 1$.

$$u_t = n - t + \left\lfloor \frac{t}{1} \right\rfloor = n - t + \lfloor t \rfloor$$

Or, puisque $t \in \mathbb{N} \cap \{[0;x]/u_x = 0\}, t \in \mathbb{N}$ donc $t = \lfloor t \rfloor$.(11)

$$u_t = n - t + t = n$$

On a donc, pour $c = 1, u_t = n$. $(u_t)_{t \in \mathbb{N} \cap \{[0;x]/u_x = 0\}}$ est une suite constante, donc le nombre de voitures dans l'embouteillage est constant, donc l'embouteillage ne se débloquera jamais.

PROPRIÉTÉ

On notera $t + \frac{1}{2}$ l'étape intermédiaire entre t et $t+1$.

L'étape $t + \frac{1}{2}$ n'est pas logique au sens mathématique puisqu'une suite numérique ne peut pas avoir de rang décimal. Elle n'est utilisée dans cette démonstration que pour simplifier le raisonnement.

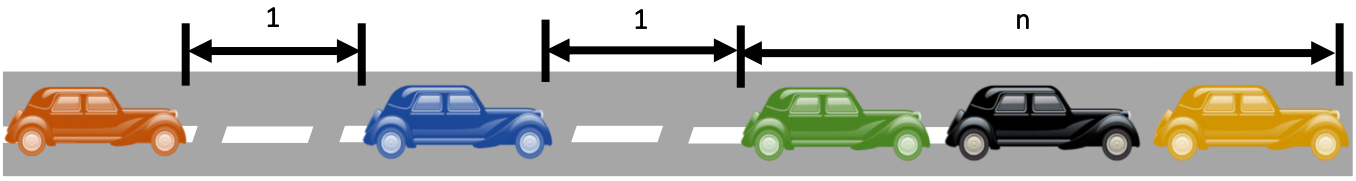
L'action se déroulant entre t et $t + \frac{1}{2}$ et l'action se déroulant entre $t + \frac{1}{2}$ et $t+1$ s'exécutent en réalité simultanément entre t et $t+1$.

L'embouteillage ne se débloquera jamais si et seulement si il est précédé d'une suite infinie de voitures séparées d'une seule case (c.-à-d. ssi $c = 1$).

Dans cette configuration, on remarque qu'entre chaque étape t et $t+1$:

- La voiture se trouvant en tête du bouchon sort de l'embouteillage ($u_{t+\frac{1}{2}} = u_t - 1$).
- Le bouchon recule d'une case car la voiture juste derrière le bouchon et séparée de celui-ci avance d'une case et le rejoint ($u_{t+1} = u_{t+\frac{1}{2}} + 1 = u_t - 1 + 1 = u_t \Rightarrow u_{t+1} = u_t$).

On peut donc conclure que dans cette situation, à chaque étape, l'embouteillage conservera le même nombre de voitures ($u_{t+1} = u_t$). Il ne se résorbera jamais.



Dans le cas contraire, si les voitures précédant l'embouteillage sont séparées de plus d'une case (c.-à-d. ssi $c > 1$), l'embouteillage finira par se résorber au bout de x étapes, tel que $u_x = 0$.(12)

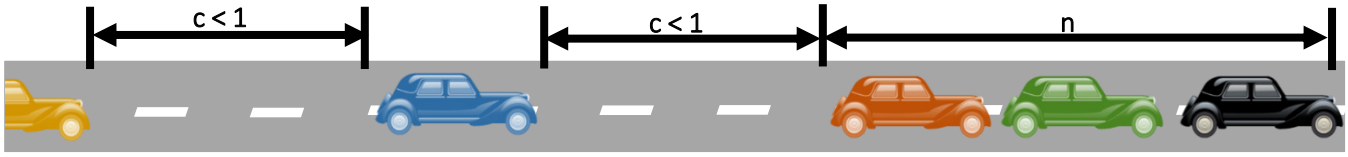
Dans cette configuration, on remarque qu'entre chaque étape t et $t+1$, si $t+1$ n'est pas un multiple de c :

- La voiture se trouvant en tête du bouchon sort de l'embouteillage ($u_{t+1} = u_t - 1$).
- La voiture précédant l'embouteillage avance d'une case, mais n'aura pas rejoint l'embouteillage.

On remarque également qu'entre chaque étape t et $t+1$, si $t+1$ est un multiple de c :

- La voiture se trouvant en tête du bouchon sort de l'embouteillage ($u_{t+\frac{1}{2}} = u_t - 1$).
- Le bouchon recule d'une case car la voiture précèdent le bouchon et séparée de celui-ci avance d'une case et le rejoint ($u_{t+1} = u_{t+\frac{1}{2}} + 1 = u_t - 1 + 1 = u_t \Rightarrow u_{t+1} = u_t$). Elle aura suffisamment avancé lors des étapes précédentes pour n'être séparée de l'embouteillage que d'une seule case à l'étape t.

Le nombre de voitures dans l'embouteillage u_t se décrémentera mais ne s'incrémentera jamais. La suite u sera donc décroissante et finira par atteindre 0. C'est à ce moment que l'embouteillage se résorbera.



(12)

Notes d'édition

(1) Un automate cellulaire unidimensionnel est une suite régulière de « cellules » contenant chacune un état (ici : place libre ou place occupée) qui peut évoluer au cours du temps. L'état d'une cellule au temps t+1 est fonction de l'état au temps t des cellules voisines. À chaque unité de temps, les mêmes règles sont appliquées simultanément à toutes les cellules, produisant une nouvelle « génération » de cellules dépendant entièrement de la génération précédente (source : Wikipedia).

(2) Une configuration = un état initial de la route + une règle précisant les instants d'arrivée de nouvelles voitures sur l'extrémité gauche de la route.

(3) Il s'agit d'ajouts d'espaces, appelés "indentations".

(4) Sinon, utiliser le code proposé sur ce site :

https://openclassrooms.com/courses/apprenez-a-programmer-en-ti-basic/les-chaines-de-caracteres-16 - /id/r-2244025

(5) Que le lecteur se reporte directement au paragraphe suivant s'il ne lit pas couramment le TI-basic... En général, on ne donne un code complet de la sorte qu'en annexe d'un document, et bien commenté ; on préfère les pseudo-codes qui s'extraient des contingences du langage utilisé.

(6) Pour l'instant, *t*=0. Les instants *t* prendront des valeurs entières.

(7) Il s'agit de définir les instants d'arrivée des voitures sur l'extrémité gauche de la route. *N* est plutôt la "période" d'apparition des voitures, inverse de la fréquence : une voiture arrive tous les *N* instant Selon la loi régissant un automate cellulaire, chaque voiture avance d'une case (de l'instant *T* à l'instant *T*+1) si la case devant elle est libre à l'instant *T*.

(8) Selon la loi régissant un automate cellulaire, chaque voiture avance d'une case (de l'instant *T* à l'instant *T*+1) si la case devant elle est libre à l'instant *T*.

(9) L'ordre de parcours est sans importance, c'est une propriété des automates cellulaires.

(10) ... et que la voiture qui est derrière n'a pu s'avancer jusque là. Sinon (si le chiffre au rang I+1 est 1), la voiture est bloquée et on ne change rien : R_{I,T}=1.

(11) Pour être mathématiquement correct, il faudrait écrire : "pour $t \in \mathbb{N} \cap [0, x]$ où x est le premier instant où $u_x = 0$".

(12) Il faut lire « c>1 » dans cette image

(13) On peut donner une valeur précise à ce x. En effet, x est le plus petit entier tel que n-x+[x/c]=0, donc tel que [x(-1+1/c)] = -n, donc tel que x(-1+1/c)<-n+1. Donc x est le plus petit entier strictement supérieur à (n-1)/(1-1/c). On voit alors que si n est grand (l'embouteillage initial est gros) alors il mettra plus de temps à se résorber, que si c est grand alors il sera rapide à résorber, et qu'il y a problème de division par zéro lorsque c=1 : l'embouteillage est de durée infinie !