Réponses aux questions (phase 3)

Avant de répondre aux nombreuses questions que vous nous avez posées, nous voudrions vous remercier d'avoir pris le temps de réfléchir à notre vidéo, merci également pour toutes les questions très intéressantes qui nous ont vraiment fait avancer sur la résolution du problème. Nous avons pu nous rendre compte de certaines erreurs mais également approfondir la recherche et c'est ce qui nous a beaucoup plus. En raison du peu de temps accordé à la réponse aux questions, nous avons décidé d'en sélectionner certaines et d'en éliminer d'autres, notamment celles qui demandaient des preuves que vous aviez déjà apportées dans votre vidéo. Voilà donc ce que nous vous proposons comme réponses.

1. Pour la question 3 du problème, pourriez-vous nous dire ce que représente Uk ? Quel est le lien avec la question 3 ?

Pour cette question, nous avons remarqué que tous les états à k chiffres qui ne s'écrivent qu'avec des 0 et des 1 peuvent être obtenus en partant de 00…01. Nous avons simplement cherché à aller plus loin en calculant le nombre d'états vérifiant cette propriété. La suite (U_k) représente donc le nombre total d'états à k chiffres qui ne s'écrivent qu'avec des 0 et des 1, pouvant être obtenus à partir de 00…01.

Petit problème : la suite donnée ne correspondait pas à ce que l'on cherchait. Le nombre total d'états vérifiant cette propriété peut d'écrire sous la forme $U_k=2^k-1$ ou $U_k=2^k-2$ (en comptant le nombre de départ 00…01 ou non), avec k entier naturel non nul représentant le nombre de chiffres composant l'état.

2. Pour la même question, pourriez-vous expliquer votre utilisation du triangle de Pascal ?

Concernant le triangle de Pascal, nous avons simplement arrangé les états composés de 0 et de 1 pouvant être obtenus en 00…01 dans un tableau. Dans chaque cellule de ce tableau nous avons classé le nombre total de possibilités d'états à k chiffres contenant n chiffres 1. Par exemple pour k=3, en partant de 001 nous pouvons obtenir 010, 100, 011, 101, 110, 111. 6 possibilités au total (7 en comptant le nombre de départ 001) réparties en 3 catégories : 2 (ou 3) états comprenant un seul 1, 3 états comprenant 2 chiffres 1 et un seul état comprenant 3 chiffres 1.

En réalisant ce classement pour différents k, nous avons alors constaté la formation d'un triangle de Pascal, ce qui nous a permis de déterminer la suite U_k. Toutefois, nous n'avons pas eu le temps de généraliser la présence de ce triangle de Pascal pour tout k.

3. Pour la question 4 du problème: Pour un nombre à k=2 puis k=3 chiffres, obtient-on toujours un cycle ? de quelle longueur ? y a-t-il des nombres qui n'appartiennent pas à des cycles ? Lesquels ? Que peut-on alors dire des nombres appartenant à un cycle et de ceux qui n'y appartiennent pas ?

En utilisant uniquement le bouton B, il est toujours possible de retomber sur l'état initial n_0 (même si certains cycles ont une longueur de 1).

Pour k=2, lorsque n_0 est un nombre pair, on obtient un cycle de 5. Lorsqu'il est impair on obtient un cycle de 10. Petites exceptions : si le chiffre des unités est nul, la longueur du cycle est de 1, on peut donc dire que ces états n'appartiennent pas à des cycles. Si le chiffre des unités et 5, alors le cycle aura une longueur de 2.

Pour k=3, la longueur d'un cycle est principalement de 50. Mais il existe également des cas particuliers :

- Si n_0 est un multiple de 4, on obtient un cycle de 25
- Si n_0 est un multiple de 5, on obtient un cycle de 10
- Si n_0 est un multiple de 20, on obtient un cycle de 5
- Si n_0 est un multiple de 25, on obtient un cycle de 2
- Si n_0 est un multiple de 100, on obtient un cycle de 1 (ces états n'appartiennent donc pas à des cycles)

Toutefois, la longueur du cycle ou l'appartenance ou non à un cycle ne modifie pas l'accessibilité des états lorsque k>=3. En effet, il suffit d'utiliser le bouton A pour les transformer en états accessibles appartenant à d'autres cycles. Les états qui n'appartiennent à aucun cycle auront seulement une complexité plus importante (généralement égale à 2).

4. Pour la question 5 de l'exercice: pourriez-vous nous expliquer ce que font les programmes ? Pourriez-vous donner un programme qui détermine les nombres de complexité 1, i.e qui nécessite l'utilisation d'une fois exactement l'opération A pour k=2,3 ?

Pour cette question, nous avons travaillé sur les états qui possédaient une complexité de 0*A. Seuls les états ayant pour chiffre des unités 1 pouvaient être susceptibles de posséder cette complexité minimale. Nous sommes donc partis de 00…01 et nous avons listé tous les états appartenant au cycle de 00…01, cela nous donnait donc un certain nombre d'états accessibles possédant une complexité minimale. C'est le principe du premier programme : lorsque l'on rentre le nombre de chiffres (k), il nous liste les états appartenant au cycle de 00…01.

Mais nous n'étions pas sûrs d'avoir ainsi tous les états possédant une complexité de 0*A. C'est pourquoi nous avons créé un deuxième programme qui liste tous les états (avec 1 pour chiffre des unités) qui n'ont pas une complexité minimale. Pour cela, nous avons pris 00…021 comme point de départ en sachant que ce nombre ne permet pas de retomber sur 00…01 sans utiliser le bouton A. En rentrant le nombre de chiffres (k), ce programme nous liste donc tous les états appartenant au cycle de 00…021.

Après avoir fait tourner les deux programmes, il suffisait ensuite d'additionner le nombre d'états obtenus afin de regarder s'ils y sont tous.

Vous nous avez demandé un autre programme qui détermine les nombres de complexité 1. Dans le peu de temps imparti, nous vous proposons donc un brouillon de programme qui aurait pu être étoffé et amélioré en plus de temps. Ce programme permet d'afficher les états appartenant au cycle du nombre entré initialement n_0 (pour k=3) si sa complexité est de 1*A (puisque les états appartenant au cycle de n_0 possèdent aussi une complexité de 1*A). S'il possède par contre une complexité différente, le programme s'arrête et n'affiche rien.

Avant de faire tourner ce programme, il faut stocker les états appartenant au cycle de 001, donnés par le programme suivant (proposé dans la vidéo) :

```
Sub machine()
   ActiveSheet.Columns("A:B").ClearContents
   a = 11
   i = 1
   d = InputBox("diviseur ?")
   Cells(1, 1) = a
```

```
    Do While a > 1
        i = i + 1
        Cells(i, 1).FormulaR1C1 = "=MOD(11*" & a & "," & 10 ^ d & ")"
        a = Cells(i, 1)
    Loop

End Sub
```

Voilà le programme pour la complexité de 1*A :

```
Sub machine3()
    ActiveSheet.Columns("B:L").ClearContents
    a = CInt(InputBox("a ?"))
    b = CInt(InputBox("b ?"))
    c = CInt(InputBox("c ?"))
    n = 100 * a + 10 * b + c
    For i = 1 To 50
        If n = Cells(i, 1) Then
            Exit Sub
        End If
    Next i
    If a = 1 Then
        k = a
        a = c
        c = k
        m = 100 * a + 10 * b + c
        For i = 1 To 50
            If m = Cells(i, 1) Then
                Cells(1, 2) = n
                Cells(2, 2).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
                m = Cells(2, 2)
                j = 3
                Do While m <> n
                    Cells(j, 2).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
                    m = Cells(j, 2)
                    j = j + 1
                Loop
                Exit Sub
            End If
        Next i
        '1*A
        k = a
        a = b
        b = k
        m = 100 * a + 10 * b + c
        For i = 1 To 50
            If m = Cells(i, 1) Then
                Cells(1, 3) = n
```

```
            Cells(2, 3).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
            m = Cells(2, 3)
            j = 3
            Do While m <> n
               Cells(j, 3).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
               m = Cells(j, 3)
               j = j + 1
            Loop
            Exit Sub
         End If
      Next i
      '1*A
   End If
   If b = 1 Then
      k = b
      b = c
      c = k
      m = 100 * a + 10 * b + c
      For i = 1 To 50
         If m = Cells(i, 1) Then
            Cells(1, 4) = n
            Cells(2, 4).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
            m = Cells(2, 4)
            j = 3
            Do While m <> n
               Cells(j, 4).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
               m = Cells(j, 4)
               j = j + 1
            Loop
            Exit Sub
         End If
         '1*A
      Next i
      k = a
      a = b
      b = k
      m = 100 * a + 10 * b + c
      For i = 1 To 50
         If m = Cells(i, 1) Then
            Cells(1, 5) = n
            Cells(2, 5).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
            m = Cells(2, 5)
            j = 3
            Do While m <> n
               Cells(j, 5).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
               m = Cells(j, 5)
               j = j + 1
            Loop
```

```
            Exit Sub
        End If
        '1*A
    Next i
End If
If b + c < 10 Then
    a = a + b - 10 * Application.WorksheetFunction.Quotient(a + b, 10)
    b = b + c - 10 * Application.WorksheetFunction.Quotient(b + c, 10)
    'c=c
End If
If b + c >= 10 Then
    a = 1 + a + b - 10 * Application.WorksheetFunction.Quotient(a + b, 10)
    b = b + c - 10 * Application.WorksheetFunction.Quotient(b + c, 10)
    'c=c
End If
m = 100 * a + 10 * b + c
Do While m <> n
    If a = 1 Then
        k = a
        a = c
        c = k
        'c=1
        m = 100 * a + 10 * b + c
        For i = 1 To 50
            If m = Cells(i, 1) Then
                Cells(1, 2) = n
                Cells(2, 2).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
                m = Cells(2, 2)
                j = 3
                Do While m <> n
                    Cells(j, 2).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
                    m = Cells(j, 2)
                    j = j + 1
                Loop
                Exit Sub
            End If
            '1*A
        Next i
        k = a
        a = b
        b = k
        m = 100 * a + 10 * b + c
        For i = 1 To 50
            If m = Cells(i, 1) Then
                Cells(1, 3) = n
                Cells(2, 3).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
                m = Cells(2, 3)
                j = 3
```

```
        Do While m <> n
          Cells(j, 3).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
          m = Cells(j, 3)
          j = j + 1
        Loop
        Exit Sub
      End If
      '1*A
    Next i
  End If
  If b = 1 Then
    k = b
    b = c
    c = k
    m = 100 * a + 10 * b + c
    For i = 1 To 50
      If m = Cells(i, 1) Then
        Cells(1, 4) = n
        Cells(2, 4).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
        m = Cells(2, 4)
        j = 3
        Do While m <> n
          Cells(j, 4).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
          m = Cells(j, 4)
          j = j + 1
        Loop
        Exit Sub
      End If
      '1*A
    Next i
    k = a
    a = b
    b = k
    m = 100 * a + 10 * b + c
    For i = 1 To 50
      If m = Cells(i, 1) Then
        Cells(1, 5) = n
        Cells(2, 5).FormulaR1C1 = "=MOD(11*" & n & "," & 1000 & ")"
        m = Cells(2, 5)
        j = 3
        Do While m <> n
          Cells(j, 5).FormulaR1C1 = "=MOD(11*" & m & "," & 1000 & ")"
          m = Cells(j, 5)
          j = j + 1
        Loop
        Exit Sub
      End If
      '1*A
```

```
            Next i
        End If
        If b + c < 10 Then
            a = a + b - 10 * Application.WorksheetFunction.Quotient(a + b, 10)
            b = b + c - 10 * Application.WorksheetFunction.Quotient(b + c, 10)
        End If
        If b + c >= 10 Then
            a = a + b - 10 * Application.WorksheetFunction.Quotient(a + b, 10) + 1
            b = b + c - 10 * Application.WorksheetFunction.Quotient(b + c, 10)
        End If
        m = 100 * a + 10 * b + c
    Loop
End Sub
```

Voilà pour le programme, désolé pour la taille ;)

5. Pourriez-vous donner un nombre qui a une complexité de 2, puis de 3 pour k=2 et k=3 ? Quelle est la complexité de 021 par exemple ? Quelle peut être la complexité maximale pour k=2 et k=3? Sauriez-vous généraliser pour un k quelconque?

Pour k=2, 25 possède une complexité de 2*A mais il n'existe pas d'états possédant une complexité de 3*A (sans l'avoir prouvé rigoureusement, nous avons quand même étudié toutes les possibilités). 25, 52 (A), 72 (B), 92(B), 12(B), 21 (A), … (B), 01 (B).

Pour k=3, 400 possède une complexité de 2*A mais nous n'avons pas trouvé d'états possédant une complexité de 3*A (sans l'avoir prouvé rigoureusement nous pensons tout de même qu'elle n'existe pas pour k=3). 400, 004 (A), 044 (B), … (B), 164 (B), 641 (A), 051 (B), … (B), 091 (B), 001 (B). 021 possède une complexité de 1*A : 021, 201 (A), 211 (B), … (B), 091 (B), 001 (B).

Nous n'avons pas trouvé de complexité supérieure à 2*A, c'est pourquoi nous pensons que la complexité maximale pour un k quelconque est de 2*A.

Remarque : Pour cette question 5 nous nous sommes trompés en pensant que la complexité maximale représentait le nombre « le plus minimum possible » de fois qu'il fallait utiliser le bouton A pour trouver 00…01. C'est pourquoi ce que nous avons décrit dans notre vidéo ne correspondait pas à la question posée dans le problème.

Voilà pour les réponses aux questions que vous nous avez posées. Merci beaucoup parce que nous avons vraiment eu du plaisir à les travailler même si nous avons trouvé le délai très court. Nous espérons que nos explications ont été assez claires ;)

A bientôt !

Les drôles de matheux