Jean M. Turgeon
Université de Montréal

La TI-83+ et les invités difficiles

Exemple 1. On est en train d'organiser un banquet à l'occasion d'un grand mariage. Les deux familles sont riches et les invités sont nombreux. Pour favoriser certaines rencontres, et en éviter d'autres, les places à table sont fixées à l'avance. Il reste cinq places libres, à cinq tables différentes, et encore trois personnes à placer. Mais, ces personnes sont des invités difficiles, car des querelles anciennes ont laissé des rancunes. Appelons les trois personnes P1, P2 et P3 et les tables T1, ..., T5. La présence de P1 créerait un froid aux tables T1, T2 et T5. On sait que P2 refuserait tout net de s'asseoir aux tables T2, T3 ou T4. Quant à P3, il se sentirait à l'aise partout, sauf à la table T1. Comment placer ces trois invités difficiles de façon à n'offenser personne ?

Le tableau ci-contre représente notre problème sous la forme d'un petit échiquier auquel il manque les cases grises. Il s'agit de dresser la liste des façons de placer trois tours dans les cases blanches de telle manière qu'aucune d'entre elle ne puisse en « manger » une autre, c'est-à-dire qu'il n'y en ait pas deux sur la même ligne ou sur la même colonne. L'énoncé du problème indique où sont les cases grises, mais ce sont les cases blanches qui nous intéressent.

	P1	P2	P3
T1			
T2			
T3			
T4			
T5			

Pour introduire ces informations dans la TI-83+, on peut utiliser une matrice ayant une colonne par invité et, pour chaque invité, une ligne par case blanche dans sa colonne du tableau. Les colonnes de P1 et de P2 auront chacune deux lignes, celle de P3 aura quatre lignes. Comme la matrice doit être rectangulaire, on lui donnera quatre lignes, remplissant les cases non utilisées par le nombre 99. On verra ci-après que ce nombre 99 sera utile. Voici la représentation matricielle de notre problème à introduire, dans le TI-83+.

$$[A] = \begin{bmatrix} 3 & 1 & 2 \\ 4 & 5 & 3 \\ 99 & 99 & 4 \\ 99 & 99 & 5 \end{bmatrix}$$

Avec cette matrice [A], le programme suivant permet à la TI-83+ de donner le nombre des solutions de notre problème, et même leur liste.

```
PROGRAM:INVITE3
:Menu("3 INVITES","LE NOMBRE",A,"LA LISTE",B)
:Lbl A:0→R:Goto C
:Lbl B:1→R
:Lbl C

:ClrHome
:0→P
:For(H,1,2)
:For(I,1,2)
:For(J,1,4)
:[A](H,1)→A
:[A](I,2)→B
:[A](J,3)→C
:If A≠B and A≠C and B≠C
:Then
:P+1→P

:If R=1:Then:Disp P:Pause {A,B,C}:End

:End:End:End:End

:If R=0:Then:Disp P:End
```

Le programme présente d'abord un menu qui permet de choisir le résultat que l'on souhaite obtenir. Si on appuie sur la touche 1, on reçoit seulement le nombre des solutions. Avec la touche 2, on a la liste, numé-

rotée, de toutes les solutions. Après chaque solution, on appuie sur la touche **ENTER** pour obtenir la suivante, jusqu'à ce que le message **done** indique que la liste est complète. Vous devriez obtenir la liste donnée dans le tableau ci-dessous, qui donne les numéros des tables du banquet à assigner aux trois invités difficiles, selon les dix solutions acceptables. La solution 1, par exemple, assigne la table 3 à l'invité P1, la table 1 à l'invité P2 et la table 2 à l'invité P3.

Les parties du programme sont séparées par des espaces pour en rendre la lecture plus facile. Le début sert à faire apparaître le menu. La deuxième choisit, parmi les 16 cas à considérer (2 chacun pour P1 et P2, 4 pour P3), ceux qui assignent des tables distinctes aux trois invités. La troisième partie fait apparaître les solutions à l'écran, si on a choisi cette option. La quatrième partie ferme les diverses boucles de la deuxième partie. La dernière partie donne le nombre des solutions, si on a choisi cette option.

	P1	P2	P3
1	3	1	2
2	3	1	4
3	3	1	5
4	3	5	2
5	3	5	4
6	4	1	2
7	4	1	3
8	4	1	5
9	4	5	2
10	4	5	3

Quatre invités, six tables

Exemple 2. Considérons maintenant une modification du problème, où il y a sept tables libres et quatre invités difficiles à y placer.

Les restrictions sont indiquées dans le tableau ci-contre. La matrice à introduire dans la TI-83+ est cette fois :

$$[A]=\begin{bmatrix} 3 & 2 & 1 & 1 \\ 6 & 4 & 3 & 6 \\ 99 & 5 & 99 & 99 \\ 99 & 6 & 99 & 99 \end{bmatrix}.$$

	P1	P2	P3	P4
T1				
T2				
T3				
T4				
T5				
T6				

Il faut, bien sûr, modifier aussi le programme. Voici sa nouvelle version.

```
PROGRAM:INVITE4
:Menu("4 INVITES","LE NOMBRE",A,"LA LISTE",B)
:Lbl A:0→R:Goto C
:Lbl B:1→R
:Lbl C

:0→P
:For(H,1,2)
:For(I,1,4)
:For(J,1,2)
:For(K,1,2)
:[A](H,1)→A
:[A](I,2)→B
:[A](J,3)→C
:[A](K,4)→D
:If A≠B and A≠C and A≠D
       and B≠C and B≠D and C≠D
:Then
:P+1→P

:If R=1:Then:Disp P:Pause {A,B,C,D}:End

:End:End:End:End:End

:If R=0:Then:Disp P:End
```

Le nombre des conditions d'inégalité est passé de 3 à 6, le nombre des boucles « For » est passé de 3 à 4, et ainsi de suite, mais le programme est essentiellement le même. Ce second problème admet six solutions.

	P1	P2	P3	P4
1	3	2	1	6
2	3	4	1	6
3	3	5	1	6
4	6	2	3	1
5	6	4	3	1
6	6	5	3	1

N invités, M tables
Un programme universel

Avec N invités et M tables, la matrice [A] aura et N colonnes et jusqu'à M lignes et la deuxième partie du programme se terminera par N(N+1)/2 inégalités. On risque de manquer de variables, puisque la TI-83+ n'admet comme variables, pour représenter les nom bres réels, que les lettres de l'alphabet. Notre approche a aussi l'inconvénient d'exiger des modifications des programmes pour les adapter à chaque matrice [A].

Heureusement, la calculatrice TI-83+ offre la possibilité d'utiliser aussi des listes, qui ont l'avantage de

multiplier considérablement le nombre des variables. Voici un troisième programme. Il utilise les listes et tire les informations directement de la matrice [A] pour le nombre des invités et des tables. Il utilise les nombres 99 pour trouver le nombre des tables acceptables à chaque invité. Il suffit d'introduire la matrice [A] dans la calculatrice, le programme s'occupe du reste.

```
PROGRAM:INVITES
:Menu("N INVITES","LE NOMBRE",A,"LA LISTE",B)
:Lbl A:0→R:Goto C
:Lbl B:1→R
:Lbl C

:ClrHome
:Disp "UN INSTANT SVP"
:dim([A])→L4
:L4(1)→M
:L4(2)→N
:N→dim(L3)
:N→dim(L5)
:N→dim(L6)
:Fill(1,L5):0→L5(N)

:For(J,1,N)
:For(I,1,M)
:If [A](I,J)=99
:Then
:I-1→L6(J)
:M→I
:Else
:If I=M
:Then
:M→L6(J)
:End:End:End:End

:N→K:0→P
:While K>0
:If L5(K)=L6(K)
:Then
:K-1→K
:Else
:L5(K)+1→L5(K)
:For(I,K+1,N)
:1→L5(I)
:End
:For(J,1,N)
:[A](L5(J),J)→L3(J)
:End
:1→Q
:For(I,1,N-1)
:For(J,I+1,N)
:If L3(I)=L3(J)
:Then
:0→Q
:N+1→I
:N→J
:End
:End
:End
:If Q=1
:Then
:P+1→P

:If R=1:Then:Disp P::Pause L3:End

:End
:N→K
:End
:End

:If R=0:Then:Disp P:End
```

La première partie est la même que dans les programmes précédents. La deuxième calcule les dimensions de la matrice [A] ; elles servent à délimiter les listes L_3, L_5 et L_6 et les boucles de la suite du programme. La troisième partie met dans la liste L_6, pour chaque invité, le nombre des tables qui lui sont acceptables. C'est, dans chaque colonne de [A], le nombre des éléments qui précèdent le premier 99 ou, en l'absence de 99, le nombre des lignes de [A]. Ces trois premières parties sont effectuées une seule fois.

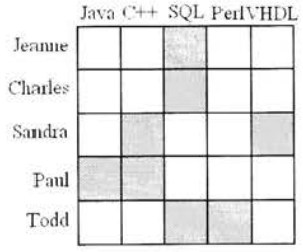
La quatrième partie énumère dans L_5 toutes les façons de prendre dans chaque colonne de [A] une case ne contenant pas 99. Pour passer d'une liste L_5 à la suivante, elle ajoute 1 à la dernière case de L_5 qui n'est pas déjà à son maximum et remplace toutes celles qui suivent par des 1. Elle se sert ensuite de L_5 pour mettre dans L_3 les contenus de ces cases de [A]. Il faut ensuite éliminer les listes L_3 qui placent plus d'un invité à une des tables, c'est-à-dire, qui comportent des répétitions. En l'absence de répétitions, on ajoute 1 à P.

La fin du programme est semblable à celles des programmes précédents.

Le lecteur curieux peut appliquer le programme universel aux matrices des exemples 1 et 2, puis s'attaquer aux exemples suivants.

Exemple 3. Le Professeur Ruth a cinq correcteurs à son service pour évaluer les devoirs dans ses cours de langages Java, C++, SQL, Perl et VHDL. Les correcteurs Jeanne et Charles exècrent le SQL. Sandra cher-che à éviter le C++ et le VHDL. Paul déteste le Java et la C++ et Todd refuse de travailler en SQL et en Perl. De combien de façons Ruth peut-elle affecter chaque correcteur aux devoirs dans un seul langage de programmation, faire corriger tous les devoirs et ne contrarier personne (Grimaldi, 2004, p. 401, exercice 7) ?

Cet exemple est représenté par l'échiquier incomplet ci-dessous.



La matrice des données est

$$[A]=\begin{bmatrix}1 & 1 & 3 & 1 & 1\\ 2 & 2 & 4 & 2 & 2\\ 3 & 5 & 99 & 3 & 4\\ 5 & 99 & 99 & 4 & 5\end{bmatrix}$$

et la TI-83+ met plusieurs minutes à calculer le nombre des solutions : 20.

Exemple 4. Une agence de rencontres s'efforce de trouver, pour chacune des cinq clientes D_1, ..., D_5 un partenaire masculin qui corresponde à ses préférences. On doit choisir parmi sept hommes, H_1, ..., H_7. Chacune de ces personnes a subi des tests psychologiques et manifesté sa personnalité dans des entrevues avec des experts de l'agence.

L'étude des dossiers a révélé que les dames D_2 et D_3 ont une incompatibilité de caractère avec H_1, H_2, H_3, H_6 et H_7. De même, D_1 est incompatible avec H_6 et H_7 et D_4 et D_5 avec H_4 et H_5.

De combien de façons l'agence peut-elle proposer aux cinq clientes des partenaires convenables ?

Si on applique les colonnes aux dames et les lignes aux hommes, les données produisent la matrice suivante :

$$[A]=\begin{bmatrix}1 & 4 & 4 & 1 & 1\\ 2 & 5 & 5 & 2 & 2\\ 3 & 99 & 99 & 3 & 3\\ 4 & 99 & 99 & 6 & 6\\ 5 & 99 & 99 & 7 & 7\end{bmatrix}$$

Après environ trois minutes et quarante secondes, la TI-83+ donne la réponse : 72.

Exemple 5. On doit affecter sept nains, N_1, N_2, N_3, N_4, N_5, N_6, N_7, à sept tâches T_1, T_2, T_3, T_4, T_5, T_6, T_7, dans une mine. Mais, N_1 ne peut accomplir les tâches T_2 et T_3 ; N_2 est impropre à T_1 et à T_5 ; N_4 ne peut effectuer T_3, ni T_6 ; N_5 ne peut accomplir T_2 et T_7 ; N_7 ne peut accomplir T_4 ; N_3 et N_6 sont tous deux propres à accomplir toutes les tâches. De combien de façons différentes peut-on affecter les sept nains à des tâches que chacun peut effectuer (Tucker, 2002, p. 338, exercice 8.3.5) ?

On a :

$$[A]=\begin{bmatrix}1 & 2 & 1 & 1 & 1 & 1 & 1\\ 4 & 3 & 2 & 2 & 3 & 2 & 2\\ 5 & 4 & 3 & 4 & 4 & 3 & 3\\ 6 & 6 & 4 & 5 & 5 & 4 & 5\\ 7 & 7 & 5 & 7 & 6 & 5 & 6\\ 99 & 99 & 6 & 99 & 99 & 6 & 7\\ 99 & 99 & 7 & 99 & 99 & 7 & 99\end{bmatrix}$$

La réponse, trouvée par une autre méthode, est 1 232. Ce problème est trop gros pour la TI-83+. En six heures et vingt minutes de calcul, elle n'a trouvé que 348 des 1 232 solutions.■

Références bibliographiques

Grimaldi, R. P. (2004). *Discrete and Combinatorial Mathematics*, 5th edition. New York (NY), Pearson, Addison Wesley.

Tucker, A. (2002). *Applied Combinatorics*. New York (NY), John Wiley & Sons.

Veuillez adresser toute correspondance concernant cette rubrique à :

Jean M. Turgeon
Département de mathématiques
Université de Montréal
Case postale 6128, Succursale Centre-Ville
Montréal (Québec) H3C 3J7

turgeon@dms.umontreal.ca
