« Défaire » un nombre entier pour en faire une liste dont les éléments sont les chiffres de ce nombre.

Deux applications.

En fait, ce sont les applications en question qui posent le problème du titre :

Première application :

Dans le manuel « Sesamath » des élèves de seconde, dans le chapitre « Généralités sur les fonctions » est posée la question suivante : parmi tous les entiers de 0 à 10000 combien y en a-t-il dont la somme des chiffres est égale à 3 ?

J'ai abordé cet exercice de deux façons :

1) J'ai laissé l'initiative aux élèves en posant la question telle qu'énoncée ci-dessus et j'ai été surpris qu'assez rapidement des réponses me soient proposées, réponses qui étaient « la bonne » ou bien avoisinaient « la bonne » ; manifestement les réponses ont été motivées par des « classements » des entiers de 0 à 10000 qui excluaient la plupart de ceux qui n'étaient pas concernés.
2) Concevoir un algorithme qui, traduit en un programme sur « les machines » des élèves, envoie la réponse à la question.

Cette première application va consister, en plusieurs étapes, en la conception de l'algorithme cité ci-dessus et, ensuite, en sa traduction en un programme sur « TI 82-83-84 ».

Algorithme 1 : il détermine le nombre de chiffre d'un entier donné X.

- k est un entier initialisé à 0
- Tant que $10^{-k}X \geq 10$, k augmente de 1
- N=k+1 est le nombre de chiffres de X

A noter : un résultat mathématique dont la démonstration peut être le sujet d'un « bel » exercice pour les terminales scientifiques :

Le nombre N de chiffres d'un entier naturel X non nul écrit en base 10 est :

$N = E(\log(X)) + 1$ où $\log(X)$ désigne le logarithme décimal de X : $\log(X) = \frac{\ln(X)}{\ln(10)}$ et E( ) désigne la fonction partie entière.

Preuve

Considérons l'écriture en base 10 de l'entier naturel X :

$X = 10^{n} a_n + 10^{n-1} a_{n-1} + \ldots + 10\, a_1 + a_0$, $a_i \in \{0;1;2\ldots\ldots;9\}$, $i \in \{0;1;2\ldots\ldots;.n\}$ et $a_n \neq 0$.

$N = n + 1$ est alors le nombre de chiffres de X.

En prenant $a_i = 0$ et $a_n = 1$ pour $i \in \{0;1;2\ldots\ldots;n-1\}$ on obtient : $10^{n} \leq X$ d'une part et en prenant $a_i = 9$ pour $i \in \{0;1;2\ldots\ldots;.n\}$ on obtient :

$X \leq 9 \cdot 10^{n} + 9 \cdot 10^{n-1} + \ldots + 9 \cdot 10 + 9 = 9 \times \frac{10^{n+1} - 1}{10 - 1} = 10^{n+1} - 1 < 10^{n+1}$ d'autre part.

On a donc : $10^{n} \leq X < 10^{n+1} \Rightarrow n \leq \log(X) < n+1 \Rightarrow N = n + 1 = E(\log(X)) + 1$.

L'algorithme précédent se réduit alors à l'unique instruction « ***N = E(log(X)) + 1*** »

Algorithme 2 **:** il extrait les chiffres de l'entier X pour en faire une « liste » au sens des « listes » de la calculatrice ; ceci dans le but d'utiliser le catalogue des fonctions des listes de la machine, fonctions comme « min, max, tri croissant, tri décroissant, somme, somme cumulée …… »

Avant d'aborder l'algorithme je tiens à traiter un exemple :

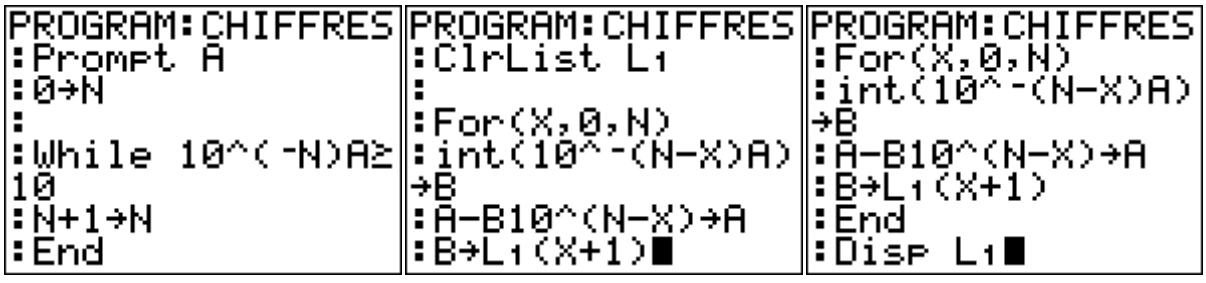
- X=2345
- N=4
- $10^{-3}X=2,345$
- $E(10^{-3}X)=2$ qui est stocké dans le premier de la liste L1
- $X-2*10^{3}=345$ devient X et on recommence
- $E(10^{-2}X)=3$ est stocké dans le deuxième de L1
- $X-3*10^{2}=45$ devient X et on recommence
- $E(10^{-1}X)=4$ est stocké dans le troisième de L1
- $X-4*10^{1}=5$ devient X et on recommence
- $E(10^{-0}X)=5$ est stocké dans le quatrième de L1
- L'algorithme s'arrête là et affiche L1={2 ;3 ;4 ;5}

Les élèves sont maintenant en mesure de mettre en place l'algorithme suivant :

- On saisit l'entier X
- L'algorithme 1 détermine le nombre N de chiffres de X
- Pour k variant de 1 à N , $E(10^{-(N-k)}X)$ est le terme d'indice k de la liste L1
- $X- E(10^{-(N-k)}X)\times10^{(N-k)}$ devient X
- On affiche L1

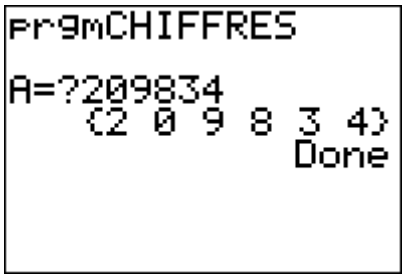
Le programme qui suit exécute les deux algorithmes précédents ; les noms des variables ne sont pas respectés mais les algorithmes sont les mêmes.

Les copies d'écran viennent du logiciel TI-SmartView et la TI 84 a pour langage l'Anglais.

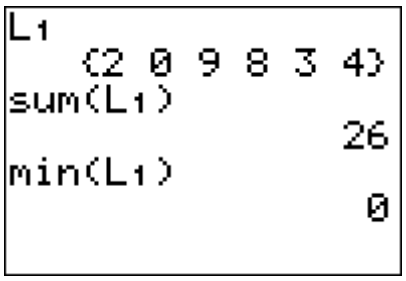


Attention, dans le troisième écran les quatre premières instructions apparaissent deux fois ; ne prendre en compte qu'une seule de ces instructions.

Dans la copie d'écran qui suit le programme précédent a été appliqué à l'entier 209834



Que faire de ceci ?
On peut demander quel est le plus petit chiffre, quelle est la somme des chiffres….
Ce qui est fait dans l'écran suivant :



On est, maintenant, en mesure de faire un programme, sur la « TI84 », qui réponde à la question posée :
« Parmi tous les entiers de 0 à 10000 combien y en a-t-il dont la somme des chiffres est égale à 3 ? »

L'algorithme est le suivant :

- On saisit les entiers p et k, où p=10000 et k=3 dans l'exercice, mais comme on n'a aucune raison de se limiter à 10000 et 3 on se réserve la possibilité de changer ces deux valeurs.
- Un compteur F est initialisé à 0 et a pour rôle de dénombrer les entiers de 0 à p dont la somme des chiffres est égale à k
- A étant un entier variant de 1 à p, A devient C et on défait l'entier C selon l'algorithme du programme précédent pour en faire la liste L1
- Si la somme de L1 est égale à k, le compteur F augmente de 1
- On affiche F qui est la solution du problème

Le programme qui suit, appelé « somchifr » traduit cet algorithme :

```
PROGRAM:SOMCHIFR
:Prompt P,K
:0→F
:For(A,0,P)
:ClrList L1
:0→N
:A→C
:While 10^(-N)C≥
```

```
PROGRAM:SOMCHIFR
10
:N+1→N
:End
:
:For(X,0,N)
:int(10^(X-N)C)→
B
```

```
PROGRAM:SOMCHIFR
:B→L1(X+1)
:C-B10^(N-X)→C
:End
:If sum(L1)=K
:Then
:F+1→F
:End
```

```
PROGRAM:SOMCHIFR
:End
:If sum(L1)=K
:Then
:F+1→F
:End
:End
:Disp F
```

Attention, dans le dernier écran, seules les deux dernières instructions sont à considérer ; les autres sont des répétitions de l'écran d'avant.

Dans les écrans qui suivent on exécute plusieurs fois ce programme pour le tester :

```
prgmSOMCHIFR

P=?20
K=?3
                2
             Done
```

Pour p=20 et k=3 les entiers en question sont : 3 et 12

```
prgmSOMCHIFR

P=?100
K=?3
                4
             Done
```

Pour p=100 et k=3 les entiers en question sont : 3 ; 12 ;21 ;30

```
prgmSOMCHIFR

P=?1000
K=?3
               10
             Done
```

Pour p=1000 et k=3 les entiers en question sont : 3 ; 12 ;21 ;30 ;102 ;111 ;120 ;201 ;210 ;300

Le programme fonctionne mais les élèves protestent : ils sont beaucoup plus performants que la machine car, dans le dernier écran, plusieurs minutes ont été nécessaires pour que la machine affiche le résultat.

On demande maintenant à la machine de répondre à la question initiale :

La machine a mis 37mn14s pour envoyer l'écran précédent !

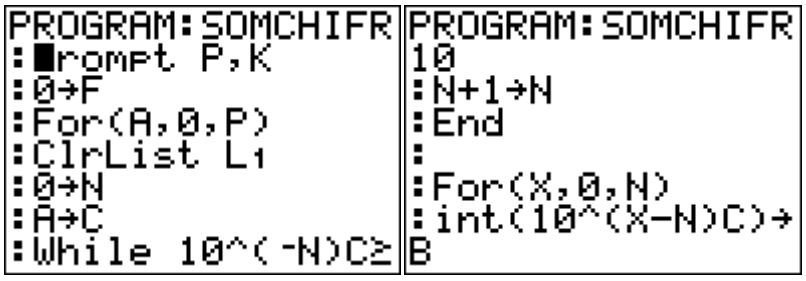
Les élèves se gaussent : ils ont été plus efficaces que la machine en travaillant plus intelligemment.

Il se pose alors, naturellement, le problème de l'amélioration de ce programme.

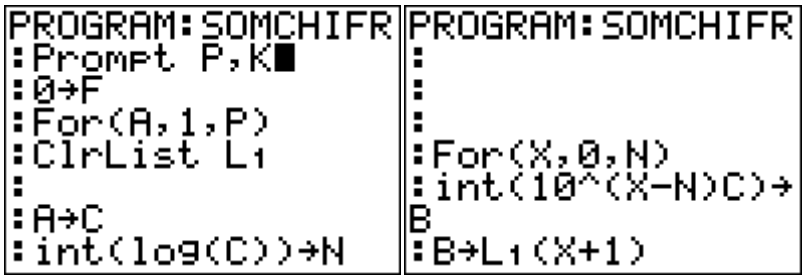
Dans ce qui suit j'explore donc plusieurs possibilités sensées permettre à ce programme d'accomplir sa tâche en un temps « raisonnable ».

Première tentative

Dans le programme ***« SOMCHIFR »***, dont je garde le nom, je supprime la partie qui calcule le nombre de chiffres de l'entier C pour la remplacer par l'instruction « ***N = E(log(C)) + 1 » ; les deux premiers écrans du programme « SOMCHIFR », soit***



sont alors remplacés par ceux-ci



Les lignes vides correspondent à ce qui a été effacé.

Cette modification permet au programme d'envoyer l'écran

En 30 minutes, approximativement.

Le gain, 7 minutes, n'est pas négligeable mais c'est encore beaucoup trop long ; de plus, dans la précédente version les instructions qui ont été effacées avaient un rôle clair pour tous les élèves, pour employer une formule en vogue « elles leurs parlaient », alors que la nouvelle instruction qui les remplace n'a de sens que pour les élèves de terminales.

Deuxième tentative

Plutôt que défaire l'entier « de la gauche vers la droite » où 537 donne {5 ;3 ;7} je vais le défaire « de la droite vers la gauche », 537 va donner {7 ;3 ;5}.

Cette démarche va me permettre d'employer des instructions « moins gourmandes » en temps : la division Euclidienne dans un premier temps puis les instructions « Ipart() » et « Fpart() » dans un deuxième temps**,** instructions implantées dans la machine et qui envoient, respectivement, la partie entière et la partie décimale d'un décimal positif .

1) *La division Euclidienne*

La machine n'est pas équipée de fonctions qui calculent le quotient q et le reste r de la division Euclidienne de a par b : a=bq+r.

Il est facile de palier à cette lacune puisque : q=E(a/b) et r=a-bq.

Pour « défaire » l'entier a « de droite à gauche » l'algorithme est le suivant :

- On divise a par 10, donc b=10, on place r dans le premier de la liste L1 et q devient a
- On recommence

Sur un exemple ça donne :

a=205

205=20×10+5

5 est le premier de la liste L1 et q=20 devient a

20=10×2+0

0 devient le deuxième terme de L1 et q=2 devient a.On met a dans le troisième de L1 et l'algorithme s'arrête là, en affichant L1={5 ;0 ;2} puisque a ≤ 9

Le programme « CHIFFRE2 » qui s'affiche dans les écrans suivants exécute cet algorithme :

```
PROGRAM:CHIFFRE2
:Prompt A
:ClrList L1
:0→F
:A→Q
:While Q≥10
:F+1→F
:int(A/10)→Q
```

```
PROGRAM:CHIFFRE2
:int(A/10)→Q
:A-10Q→R
:R→L1(F)
:Q→A
:End
:Q→L1(F+1)
:Disp L1
```

Attention, dans le deuxième écran la première instruction se répète deux fois

Dans l'écran qui suit, on exécute ce programme :

```
prgmCHIFFRE2

A=?908276
    {6 7 2 8 0 9}
            Done
```

On reprend maintenant le programme « SOMCHIFR », le premier, pour en faire le programme « SOMCHIF2 » où seul change l'algorithme qui « défait » le nombre, le nouvel algorithme est celui du programme « CHIFFRE2 », ce qui donne :

```
PROGRAM:SOMCHIF2
:Prompt P,K
:0→F
:For(C,1,P)
:0→G
:ClrList L1
:C→A
:A→Q
```

```
PROGRAM:SOMCHIF2
:While Q≥10
:G+1→G
:int(A/10)→Q
:A-10Q→R
:R→L1(G)
:Q→A
:End
```

```
PROGRAM:SOMCHIF2
:Q→L1(G+1)
:If sum(L1)=K
:Then
:F+1→F
:End
:End
:Disp F
```

Ce programme envoie l'écran

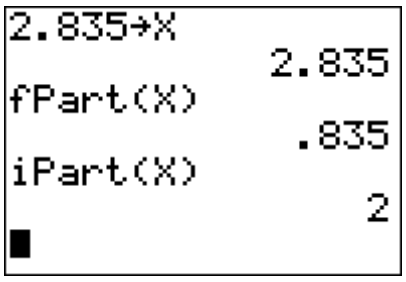
```
            Done
prgmSOMCHIF2
P=?10000
K=?3
              20
            Done
```

en 26 minutes, soit un gain d'environ 11 minutes par rapport au premier programme.

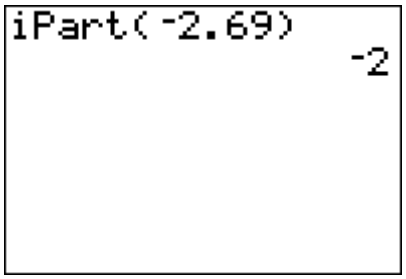
On avance, le gain est conséquent mais, pour être critique, on ne peut pas dire que la division Euclidienne « parle beaucoup » à des élèves de seconde ; peut-être y a-t-il ici l'occasion de s'attarder avec eux sur cette notion qu'ils ont vaguement abordée au collège ?

2) *En utilisant les fonctions « Ipart() » et « Fpart() » implantées dans la machine*

Exemple :



La copie d'écran qui suit est uniquement donnée pour montrer que « Ipart() » n'envoie pas la partie entière quand le décimal est négatif.



Le nouveau programme s'appelle « SOMCHIF 3» et il utilise le même algorithme que le précédent sans utiliser la division Euclidienne.

Sur un exemple, l'algorithme qui « défait » a est :

.a=205

.10fPart(a /10)=5 qui est placé dans le premier de L1

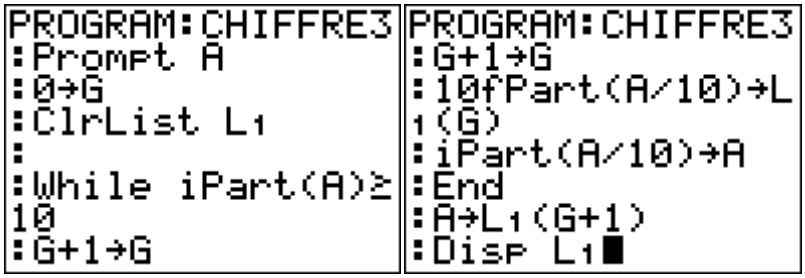
.iPar(a/10)=20 devient a

. 10fPart(a /10)=0 est le deuxième de L1

. iPar(a/10)=2 devient a qui devient le troisième de L1 car a est plus petit que 10

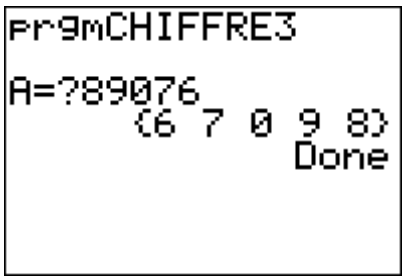
L'algorithme s'arrête là, en affichant L1={5 ;0 ;2}

Le programme qui « défait » l'entier en utilisant les fonctions ci-dessus s'appelle « CHIFFR3 » et il est donné dans les écrans suivants :

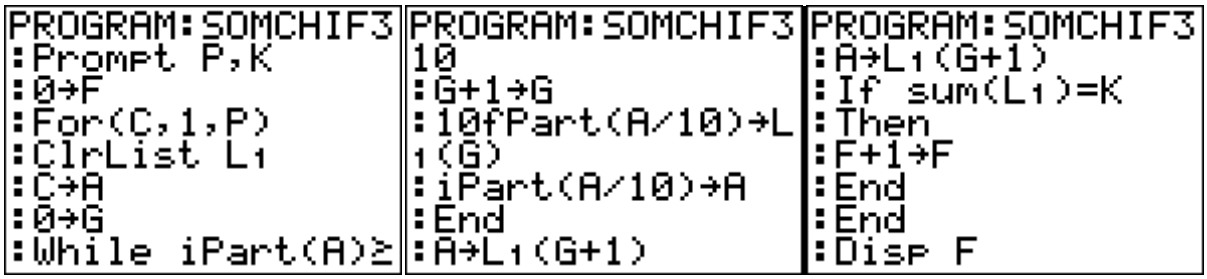


Attention, dans le deuxième écran, la première instruction est sortie deux fois.

Dans l'écran qui suit, on exécute ce programme :

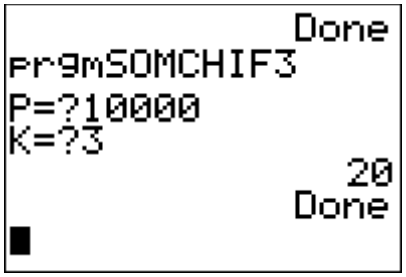


Le nouveau programme, qui « défait » les entiers de 1 à 10000 pour en calculer la somme….. s'appelle « SOMCHIF 3» et il utilise le même algorithme que le précédent sans utiliser la division Euclidienne, il est donné dans les écrans suivants :



Attention, dans le troisième écran la première instruction est sortie deux fois.

Ce programme exécute sa tâche dans l'écran suivant :



Ceci, en 20 minutes, soit un gain de 17 minutes par rapport au programme initial.

Ce sont des gains substantiels mais, à mon avis, c'est toujours beaucoup trop long et ce n'est pas là qu'il faut chercher à gagner du temps mais plutôt, comme les élèves l'ont bien vu, dans le fait que le programme analyse tous les entiers de 0 à 10000 alors que très peu sont concernés.

Dans cette optique, j'ai refait un nouveau programme nommé « SOMCHIF 4» qui reprend les fonctions du (2) et dont l'algorithme élimine de l'analyse les entiers de 4 à 9 puis ceux de 31 à 99 puis ceux de 301 à 999 puis ceux de 3001 à 10000 ; soit 7000+699+69+6=7774 entiers éliminés.

Les écrans qui suivent donnent le programme en question :

```
PROGRAM:SOMCHIF4
:{0,3,10,30,100,
300,1000,3000}→L
₂
:0→F
:For(M,1,7,2)
:For(C,L₂(M),L₂(
M+1))
```

```
PROGRAM:SOMCHIF4
:C→A
:0→G
:ClrList L₁
:While iPart(A)≥
10
:G+1→G
:10fPart(A/10)→L
```

```
PROGRAM:SOMCHIF4
₁(G)
:iPart(A/10)→A
:End
:A→L₁(G+1)
:
:If sum(L₁)=3
:hen
```

```
PROGRAM:SOMCHIF4
:If sum(L₁)=3
:Then
:F+1→F
:End
:End
:End
:Disp F
```

Attention, dans le quatrième écran les deux premières instructions sont sorties deux fois.

On exécute ce programme :

```
prgmSOMCHIF4
                    20
                  Done
```

Le nouveau programme exécute sa tache en 4 minutes, soit un gain de 33 minutes.

Cette durée peut légèrement variée, exécutée une deuxième fois la durée à été de 5 minutes ; je suppose que cela est dû au fait qu'entre les deux fois j'avais surchargé la mémoire de la machine avec d'autres fichiers ?

Très bien ! Les élèves peuvent exécuter ce programme en classe et attendre 4 minutes ; ce programme a un inconvénient majeur pourtant : il se limite à p=10000 et k=3, il perd donc en capacité de généralisation.

Deuxième application :

Une liste de chiffres peut-elle prétendre être une « liste de chiffres aléatoires » ?

Pour pouvoir répondre par l'affirmative à cette question cette liste doit passer avec succès un certain nombre de « tests de qualité » ; le « test du poker » est l'un de ceux-ci.

Citons Arthur ENGEL dans « Les certitudes du hasard » page 147 §11.2 :

« Aucun procédé de fabrication de chiffres aléatoires n'est entièrement fiable. Il est donc nécessaire de « mesurer » leur caractère aléatoire. En particulier il ne suffit pas de mesurer la fréquence de chaque chiffre il faut aussi vérifier la fréquence de différents blocs.

Un des tests les plus sûrs est le test dit du poker ».Fin de citation.

Description du « test du poker » :

Les chiffres sont regroupés par blocs de 5.

La probabilité d'un tel bloc est 10^{-5}. Ces blocs sont regroupés en 7 types dont les probabilités sont calculées. On compare alors ces probabilités avec les fréquences observées.

Le tableau suivant donne la description des 7 types, lointainement inspirés du poker (je cite) :

n°	description	type	exemple	probabilité
1	chiffres différents	abcde	34961	0,3024
2	une paire	aabcd	29512	0,5040
3	deux paires	aabbc	44533	0,1080
4	un triplet=brelan	aaabc	60366	0,0720
5	paire triplet=full	aaabb	23223	0,0090
6	Quadruplet=carré	aaaab	29222	0,0045
7	quintuplet	aaaaa	55555	0,0001

Remarque : le calcul des probabilités du tableau est un bon exercice pour les élèves de terminale scientifique.

Je calcule les deux premières probabilités :

Le modèle est : tirer au hasard et avec remise, successivement, cinq fois un chiffre parmi les dix

{0 ; 1 ;2 ;3 ;4 ;5 ;6 ;7 ;8 ;9}

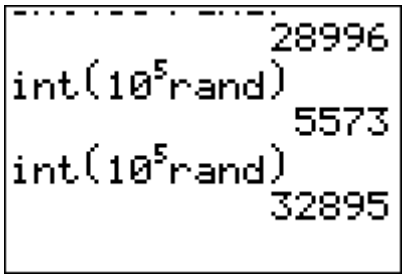
Il y a 10^5 tirages possibles et si p1 et p2 désignent, respectivement, les probabilités des types 1 et 2 on a :

- p1= $(A^5_{10}) \div 10^5 = 0.3024$, où A^5_{10} désigne le nombre d'arrangements de 5 éléments parmi 10
- p2 = $(C^2_5 \times 10 \times A^3_9) \div 10^5 = 0.504$, où C^2_5 désigne le nombre de parties à 2 éléments parmi 5

Je décide de choisir comme candidat à l'appellation « liste de chiffres aléatoires » la liste des chiffres engendrés par les cinq premières décimales des « nombres aléatoires » provenant des générateurs de « nombres aléatoires » implantés dans les machines : soit « Rand » pour les calculatrices ou « Alea » pour le tableur « Excel ».

Exemple :

L'instruction « Int(10^5Rand) » sur « TI83 » envoie les cinq premières décimales du nombre « Rand » sous la forme d'un entier ayant au plus cinq chiffres ; la copie d'écran qui suit illustre ceci :



Dans l'écran précédent on a ainsi fabriqué la liste des chiffres : 289960557332895 .

On va soumettre cette liste de chiffres au test du poker et, pour cela, on va « défaire » chacun des nombres d'au plus cinq chiffres pour en faire la liste de ses chiffres, liste de cinq chiffres ; par exemple 208 va donner la liste {0 ;0 ;2 ;0 ;8}. Le test va consister à classer la liste dans son type dans le tableau précédent et d'en déduire la fréquence de chaque type dans l'échantillon.

On peut choisir de faire un programme sur la calculatrice pour accomplir cette tache, ce que j'ai déjà fait, mais je dois avouer que la calculatrice n'est pas l'outil approprié, déterminer le type de chaque liste est fastidieux et le programme est trop compliqué.

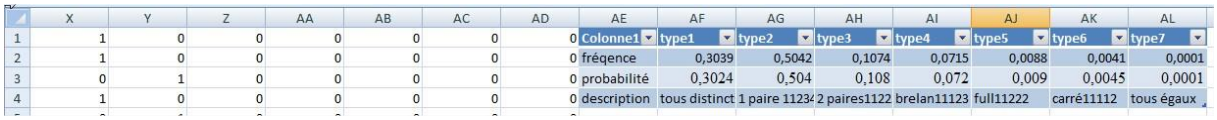
Je choisis le tableur « Excel » comme outil, les listes sont alors des « plages » et, surtout, la fonction « nb.si » qui calcule la fréquence d'une « valeur » dans une plage va nous permettre de classer facilement la plage dans son type ; la calculatrice n'est pas équipée d'une telle fonction.

Description de l'algorithme :

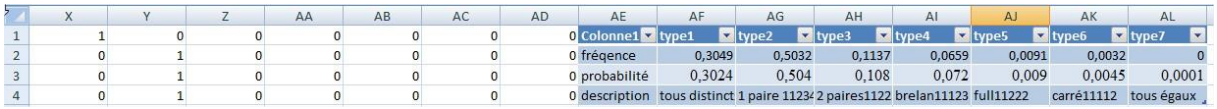
- On génère un entier d'au plus cinq chiffres comme décrit plus haut
- On fait de cet entier une plage de cinq cellules, comme décrit plus haut
- On demande la fréquence de chaque cellule de la plage précédente dans cette même plage, on obtient ainsi une nouvelle plage de cinq cellules
- Dans cette plage, on demande le « Min », le « Max » et la fréquence du « Max » dans la première plage ; on est maintenant en mesure de classer la première plage dans un des sept types du tableau
- Si le « Max » est 1 le type est le 1, si le « Max » est 2 et la fréquence du « Max » 2 le type est 2, si le « Max » est 2 et la fréquence du « Max » 4 le type est 3, si le « Max » est 3 et le

« Min » 1 le type est 4, si le « Max » est 3 et le « Min » 2 le type est 5, si le « Max » est 4 le type est 6 et si le « Max » est 5 le type est 7.

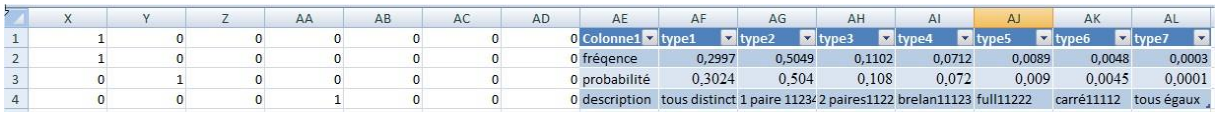
- On crée une plage de sept cellules, une cellule par type, chaque cellule affichera 1 si la plage de départ est dans ce type et 0 sinon.
- On crée un échantillon de taille 10000 et on calcule la fréquence de chaque type dans l'échantillon, fréquence qu'on compare avec la probabilité du type.



	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1	1	0	0	0	0	0	0	Colonne1	type1	type2	type3	type4	type5	type6	type7
2	1	0	0	0	0	0	0	fréqence	0,3039	0,5042	0,1074	0,0715	0,0088	0,0041	0,0001
3	0	1	0	0	0	0	0	probabilité	0,3024	0,504	0,108	0,072	0,009	0,0045	0,0001
4	1	0	0	0	0	0	0	description	tous distinct	1 paire 1123	2 paires1122	brelan11123	full11222	carré11112	tous égaux



	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1	1	0	0	0	0	0	0	Colonne1	type1	type2	type3	type4	type5	type6	type7
2	0	1	0	0	0	0	0	fréqence	0,3049	0,5032	0,1137	0,0659	0,0091	0,0032	0
3	0	1	0	0	0	0	0	probabilité	0,3024	0,504	0,108	0,072	0,009	0,0045	0,0001
4	0	1	0	0	0	0	0	description	tous distinct	1 paire 1123	2 paires1122	brelan11123	full11222	carré11112	tous égaux



	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1	1	0	0	0	0	0	0	Colonne1	type1	type2	type3	type4	type5	type6	type7
2	1	0	0	0	0	0	0	fréqence	0,2997	0,5049	0,1102	0,0712	0,0089	0,0048	0,0003
3	0	1	0	0	0	0	0	probabilité	0,3024	0,504	0,108	0,072	0,009	0,0045	0,0001
4	0	0	0	1	0	0	0	description	tous distinct	1 paire 1123	2 paires1122	brelan11123	full11222	carré11112	tous égaux

Les trois écrans ont été obtenus en appuyant sur F9, ce qui a pour effet de générer un nouvel échantillon.

La liste passe-t-elle avec succès le test du poker ? Le lecteur est laissé juge. Pour ma part la réponse est oui.

Les trois listes de 50000 chiffres des trois écrans précédents peuvent prétendre à être des « listes de chiffres aléatoires ».

Pour rendre la feuille de calcul interactive il est nécessaire de télécharger le fichier Excel « poker3 » et de cliquer sur le lien suivant : **poker3**