G. HUNAULT
Licence MASS

Angers, 2002

Mini-Projet en Algorithmique et Programmation :

Calculs de *Kaprekar*

1. Position du problème

Soit n un nombre entier positif ou nul. On appelle successeur de n au sens de *Kaprekar* le nombre entier noté $sk(n)$ défini par la différence entre les chiffres de n rangés par ordre décroissant et les chiffres de n rangés par ordre croissant. Par exemple $sk(4742) = 7442 - 2447 = 4995$. Un puits de *Kaprekar* est un point fixe de la fonction sk ainsi définie, c'est à dire un nombre entier x tel que $x = sk(x)$. La suite de *Kaprekar* de n est définie par l'application répétée de la fonction sk pour n. On pourra vérifier que les cinq premiers termes de la suite de *Kaprekar* de 4742 sont

$$\begin{array}{lll} u_0 & = & 4742 \\ u_1 & = & sk(4742) & = 4995 \\ u_2 & = & sk(4995) & = 5355 \\ u_3 & = & sk(5355) & = 1998 \\ u_4 & = & sk(1998) & = 8082 \end{array}$$

On appelle boucle *Kaprekar* de longueur p une suite d'entiers n_i pour i de 1 à p tels que $sk(n_p) = n_1$. Par exemple 74943 62964 71973 83952 est une boucle *Kaprekar* de longueur 4 puisque $sk(n_4) = sk(83952) = 74943$. On remaquera qu'on peut considérer un puits de *Kaprekar* comme une boucle *Kaprekar* de longueur 1.

Pour ce qui suit, on admettra que tout nombre entier positif ou nul inférieur à $10^6 - 1$ possède soit un puits soit une boucle dans sa suite de *Kaprekar*. Le but du mini-projet est de déterminer ces boucles et ces puits.

2. Travail demandé

2.1) Ecrire l'algorithme d'une fonction *pBoucle* qui admet un nombre n en paramètre et qui renvoie le premier terme de la boucle de *Kaprekar* de n. On admettra que la fonction sk est déja définie.

2.2) Ecrire enfin un algorithme principal qui affiche toutes les boucles de *Kaprekar* entre $ndeb$ et $nfin$, une valeur classique de ces entiers étant 1 et $10^6 - 1$. On utilisera la fonction *pBoucle*. Ces boucles seront bien sûr triées par ordre croissant de leur premier terme. Elles seront aussi numérotées ; on indiquera leur numéro et longueur selon le format :

```
Num.     Longueur    Termes de la suite
```

2.3) Ecrire enfin l'algorithme de la fonction sk ; on pourra penser à utiliser les restes sucessifs de la division de n par les puissances de 10. On notera $reste(a, b)$ le reste de la division entière de a par b : par exemple $reste(11, 3)$ est 2.

2.4) Trouver l'origine de ce problème, son ou ses auteurs et leur biographie succinte, les références et les théorèmes liés à ce problème. On pourra se contenter de références bibliograhiques et de pages *Web*.

On s'intéresse maintenant à la traduction des algorithmes en Rexx et en Apl.

2.5) Traduire tous les algorithmes en Rexx. Pour la fonction sk, si on ne veut pas utiliser les restes de division entière par les puissances de 10, on pourra utiliser les fonctions length et substr.

On ajoutera au programme principal la gestion de deux paramètres à savoir le nombre initial $ndeb$ et $nfin$. et le nombre final à parcourir. Ainsi pour les paramètres 320 et 15000 fournis dans cet ordre, le programme cherche toutes les boucles de *Kaprekar* entre 320 et 15000.

2.5) Traduire tous les algorithmes en Apl. Pour la fonction sk, si on ne veut pas utiliser les restes de division entière par les puissances de 10, on pourra utiliser les fonctions $\perp$ (encode) et $\top$ (decode). La fonction principale gérera aussi les deux paramètres $ndeb$ et $nfin$ que ce soit en mode monadique ou dyadique.

3. Consignes pour le mini-projet

Ce miniprojet est à rédiger seul(e) même si vous travaillez à plusieurs pour la programmation ou les recherches bibliographiques. Vous devez envoyez en attaché par mail un fichier .zip ou un fichier .tar qui contient tout votre travail, à savoir :

- un ou plusieurs fichiers algorithmes .alg avec leur analyse par *Galg* (fichiers .lst et .lvm),
- une zone de travail Apl (fichier .ws = "*Workspace*" ou .apl ou .w3), avec les fonctions et si possible aucune variable sauf un texte d'explication,
- un fichier .rex texte du programme Rexx qui fonctionne sous *Dos/Windows* comme sous *Unix/Linux* avec l'interpréteur *regina*,
- un document .doc au sens de *Word* pour *Windows* de préférence au format RTF ou un fichier .tex pour LateX qui contiendra les la documentation des algorirhmes, les références du problème, les résultats des progammes et et vos commentaires concernant l'analyse, la programmation et la traduction.

Ces fichiers auront pour nom vos initiales : Jomes Band fournira donc JB.WS, JB.REX et JB.DOC le tout dans JB.ZIP ou dans JB.TAR ; par contre Nona Higane rendra NH.WS, NH.REX et NH.DOC, le tout dans NH.ZIP ou dans NH.TAR. Lors de l'archivage, vous ne mettrez aucun sous-répertoire.

Ce travail de recherche, programmation et rédaction est à rendre pour le lundi 9 décembre 2001 à 11h.

Aucun retard ne sera toléré.

Pour envoyer les fichiers, il faut impérativement utiliser la page *Web*

`http://www.info.univ-angers.fr/pub/gh/wals/envoiproj.php`

4. Remarques concernant la rédaction

Il vaut mieux rendre un travail incomplet que de ne rien rendre. La qualité des algorithmes, en particulier leur simplicité et leur lisibilité sera appréciée. Indiquez clairement vos choix : si un terme vous parait ambigu, précisez sa définition pour vous. Justifiez vos modes de programmation ; récursivité ou linéarité, appel d'opérateurs, de fonctions prédéfinies ou écriture de boucles explicites. Nommez distinctement vos variables (en particulier, vous utiliserez systématiquement au moins 4 lettres par identificateur). N'hésitez pas à commenter sans dépasser toutefois un commentaire toutes les 4 lignes en moyenne (sauf pour l'en-tête).

Vous n'êtes pas obligé(e) de tout tester dans tous les sens. Effectuez donc des tests élémentaires. L'affichage des résultats à l'écran devra être très propre, très soigné. Ainsi en cas d'affichage numérique multi-lignes, on mettra bien les unités sous les unités, les dizaines sous les dizaines etc. Soignez aussi la rédaction, l'orthographe (utilisez des correcteurs orthographiques et grammaticaux).

La note sera principalement fonction de la qualité de l'analyse du travail à effectuer, de l'exposé de la méthode et de la clarté de la rédaction. La valeur attribuée au programme ne représentera que 25 % de la note ; aucune interface n'est demandée.

Aucun langage autre que cité ne devra être utilisé, aucun traitement de textes autre que cité ne devra être utilisé. Tout fichier qui ne correspondra pas aux initiales de l'élève ne sera pas lu. En particulier, les fichiers nommés minip, algo, proj, proj, projet, doc ne seront pas lus.

Un programme même incomplet mais commenté et bien indenté sera au moins autant apprécié qu'un programme qui "marche" mais sans commentaire et illisible quand aux structures de flot (boucles, tests etc.).

5. Références

Le logiciel *APL* pour *Dos/Windows* version spéciale Université d'Angers peut être récupéré à l'adresse

`http://www.info.univ-angers.fr/pub/gh/wals/getapl.htm`

on y trouvera aussi comment récupérer un *APL* gratuit pour *Linux*.

Le logiciel *REGINA (REXX)* pour *Dos/Windows* et *Linux* peut être récupéré à l'adresse

`http://www.info.univ-angers.fr/pub/gh/wals/getrex.htm`

Plus généralement, tous les documents pour le cours Algorithmique, Langages et Styles de Programmation dont un manuel élémentaire d'*Algorithmiques raisonnées*, les consignes pour le projet et ce texte sont disponibles à l'adresse

`http://www.info.univ-angers.fr/pub/gh/wals/`

Pour être sur(e) que les algorithmes sont correctement écrits, on utilisera le programme galg et on rendra le listage des instructions (fichier `.lst` plutot que l'algorithme initial `.alg`) ainsi que le fichier `.lvm` des variables, tableaux et modules. Le programme galg est disponible à l'adresse

`http://www.info.univ-angers.fr/pub/gh/Galg/`

De plus, il est possible de tester si le projet a été correctement reçu grâce à la page

`http://www.info.univ-angers.fr/pub/gh/wstat/zipdirmpm.php3`