Dans la plus grande partie de ce document, nous étudierons l'algorithme de Kaprekar pour un nombre de départ inférieur à 1000, dans le but que ces quelques pages puissent en partie être exploitées dès la seconde.
Néanmoins, nous verrons que certains algorithmes choisis sont facilement adaptables à un entier quelconque.

1. Dissociation des chiffres composant le nombre.

Ce travail à lui seul constitue un exercice intéressant pour des élèves de seconde.

L'utilisateur saisit un entier strictement inférieur à 1000.

On peut considérer qu'un tel nombre est formé de trois chiffres : par exemple, 7 peut être codé par 007.

Il nous appartient ici de retrouver les trois chiffres composant ce nombre.

On commencera par retrouver le chiffre des unités, puis le chiffre des dizaines et enfin celui des centaines.

Un nombre D étant saisi, son chiffre des unités est égal au reste dans la division de D par 10. Si l'outil utilisé possède la fonction « mod », c'est donc D mod 10. Si cette fonction n'est pas disponible, on utilisera la fonction partie entière : $D - 10\times \text{ int}\left(\dfrac{D}{10}\right)$.

Il suffit alors de remplacer D par $\text{int}\left(\dfrac{D}{10}\right)$ et de recommencer le processus pour trouver le chiffre des dizaines, puis celui des centaines.

Ces trois chiffres seront stockés dans une liste.

ALGORITHME : *Dissociation des 3 chiffres de D.*

Début
  Entrer la valeur de D
  Effacer la liste L_1
  Pour i **variant**De 1 à 3 **Faire**
    $L_1(i) \leftarrow D - 10\times \text{ int}\left(\dfrac{D}{10}\right)$
    $D \leftarrow \text{int}\left(\dfrac{D}{10}\right)$
  FinPour
Fin

Cette technique peut facilement être adaptée à un nombre de n chiffres, pour n entier non nul quelconque : il suffira de faire varier I de 1 à n. Si on veut un algorithme utilisable pour n'importe quel entier naturel, on utilisera $n = \text{int}\left(\log(D) + 1\right)$ ou on fera saisir à l'utilisateur le nombre de chiffres de l'entier G, plus grand nombre formé en arrangeant les chiffres de D.

ALGORITHME : *Dissociation des N chiffres de D.*

Début
  Entrer la valeur de D
  Effacer la liste L_1
  $N \leftarrow \texttt{int}(\log(D)+1)$
  Pour i variant**De** 1 **à** N **Faire**
    $L_1(i) \leftarrow D - 10\times \texttt{int}\left(\frac{D}{10}\right)$
    $D \leftarrow \texttt{int}\left(\frac{D}{10}\right)$
  FinPour
Fin

2. On détermine G et P.

On appelle ici P le plus petit entier que l'on peut former avec les chiffres de D, et G le plus grand.

A ce niveau, plusieurs possibilités s'offrent à nous. Tout dépend des objectifs fixés : si cet exercice est donné à des élèves de terminale qui suivent la spécialité mathématique, on peut aller rapidement à l'essentiel : l'algorithme a pour but de conjecturer un résultat qu'on cherchera à démontrer ensuite. Il n'est peut être pas nécessaire alors de s'attarder sur la manière d'ordonner les trois chiffres pour créer P et G. On peut utiliser très efficacement un tri croissant ou décroissant de liste, disponible sur les calculatrices : fonctions « SortA » ou « SortD » par exemple sur TI.

Nous avons plutôt choisi de proposer la réalisation de cet algorithme à des classes d'un niveau inférieur, pour lesquelles il nous paraît utile de construire un « algorithme de tri ».

Dans les pages suivantes, on propose plusieurs possibilités, sans toutefois aborder des algorithmes de tris classiques, puisqu'il y a très peu de nombres à trier, et que ces nombres sont des chiffres.

a) ***Première méthode : utilisation des fonctions « max » et « min »***

Cette méthode est très efficace lorsque D est formé de 3 chiffres A, B, C stockés dans la liste L_1.

Le plus grand d'entre eux est obtenu par « $\max(L_1)$ », le plus petit par « $\min(L_1)$ » et le terme intermédiaire, chiffre des dizaines de P et G est $\text{Somme}(L_1) - (\max(L_1) + \min(L_1))$.

on stocke A, B, C dans l'ordre croissant dans la liste L2.

On a alors :

$$100L_2(3) + 10L_2(2) + L_2(1) = G$$

et

$$100L_2(1) + 10L_2(2) + L_2(3) = P$$

ALGORITHME : *Première méthode de recherche de G et P.*

Début
  Entrer la valeur de D
  Effacer la liste L_1
  Effacer la liste L_2
  Pour i **variantDe** 1 **à** 3 **Faire**
    $L_1(i) \leftarrow D - 10\times \texttt{int}\left(\frac{D}{10}\right)$
    $D \leftarrow \texttt{int}\left(\frac{D}{10}\right)$
  FinPour
  $L_2(1) \leftarrow \texttt{Min}(L_1)$
  $L_2(2) \leftarrow \texttt{Somme}(L_1)\texttt{-(Min}(L_1)\texttt{+Max}(L_1)\texttt{)}$
  $L_2(3) \leftarrow \texttt{Max}(L_1)$
  $G \leftarrow 100L_2(3) + 10L_2(2) + L_2(1))$
  $P \leftarrow 100L_2(1) + 10L_2(2) + L_2(3)$
Fin

Très simple, cette méthode ne s'applique cependant qu'au cas où l'entier de départ est composé de seulement trois chiffres.

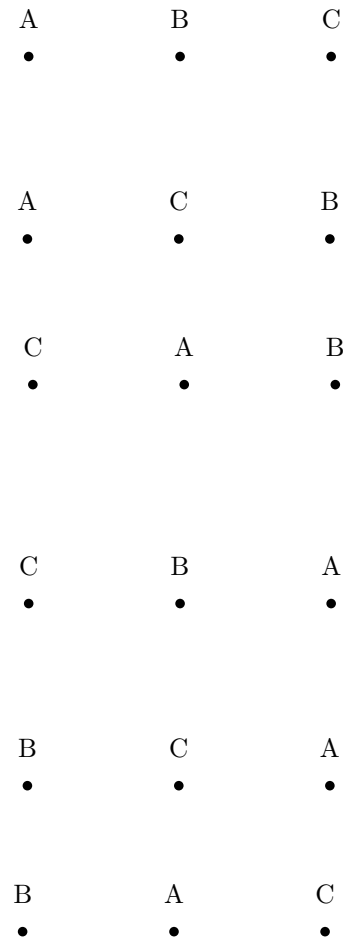
b) ***Deuxième méthode : traitement conditionnel.***

Pour faciliter la compréhension, les trois chiffres à partir desquels on détermine P et G sont notés ici A, B et C.

On compare successivement ces entiers deux à deux de façon à les ordonner, afin de déterminer les chiffres des unités, dizaines et centaines de P et G.

En langage naturel :

Si $A \leqslant B$
Alors
    Si $B \leqslant C$
    Alors
        à G attribuer $100C + 10B + A$
        à P attribuer $100A + 10B + C$
    Sinon
        Si $A \leqslant C$
        Alors
            à G attribuer $100B + 10C + A$
            à P attribuer $100A + 10C + B$
        Sinon
            à G attribuer $100B + 10A + C$
            à P attribuer $100C + 10A + B$
        Fin
    Fin
Sinon
    Si $C \leqslant B$
    Alors
        à G attribuer $100A + 10B + C$
        à P attribuer $100C + 10B + A$
    Sinon
        Si $C \leqslant A$
        Alors
            à G attribuer $100A + 10C + B$
            à P attribuer $100B + 10C + A$
        Sinon
            à G attribuer $100C + 10A + B$
            à P attribuer $100B + 10A + C$
        Fin
    Fin
Fin



Un petit graphique, à mener en parallèle à l'algorithme, aide beaucoup à la compréhension de celui-ci. On obtient 6 cas possibles, comme l'ensemble des permutations des éléments A, B, C, considérés ici comme distincts, même si ce n'est pas nécessaire.

Les élèves doivent maîtriser la notion de « négation » pour comprendre les instructions qui suivent les « Sinon ».

On trouvera page suivante cet algorithme dans une version plus lisible, mais sans le graphique.

ALGORITHME : *Deuxième méthode de recherche de G et P.*

```
Début
    Si A ⩽ B Alors
        Si B ⩽ C Alors
            G ←100C + 10B + A
            P ←100A + 10B + C
        Sinon
            Si A ⩽ C Alors
                G ←100B + 10C + A
                P ←100A + 10C + B
            Sinon
                G ←100B + 10A + C
                P ←100C + 10A + B
            FinSi
        FinSi
    Sinon
        Si C ⩽ B Alors
            G ←100A + 10B + C
            P ←100C + 10B + A
        Sinon
            Si C ⩽ A Alors
                G ←100A + 10C + B
                P ←100B + 10C + A
            Sinon
                G ←100C + 10A + B
                P ←100B + 10A + C
            FinSi
        FinSi
    FinSi
Fin
```

c) ***Troisième méthode : tri par comparaisons successives et classement dans une liste.***

Les trois chiffres à ordonner sont placés dans une liste.

Notre but est de les classer par ordre croissant dans une nouvelle liste.

On aura alors $100L_3(3) + 10L_3(2) + L_3(1) = G$ et $100L_3(1) + 10L_3(2) + L_3(3) = P$

Chaque élément de cette liste est successivement comparé à tous les autres, dans le but de repérer le plus petit.

L'idée de base est la suivante :

Soit i un entier entre 1 et 3
Pour j de 1 à 3 par pas de 1
    Si $L_1(i) \leqslant L_1(j)$
    Alors $L_2(j) \leftarrow 1$
    Sinon $L_2(j) \leftarrow 0$

$L_1(i)$ est le plus petit élément de la liste si et seulement si la liste L_2 ne contient que des 1, et donc aucun 0.

Pour tester cette condition, il suffit de comparer le produit des nombres placés en liste 2 avec 0 : si ce produit est non nul, l'élément $L_1(i)$ testé est le plus petit de la liste.

On le place en position 1 dans la liste L_3.

On pourra suggérer aux élèves de remplacer le symbole $\leqslant$ par $<$ et d'imaginer le résultat (il est essentiel d'utiliser un test avec une inégalité large, puisque chaque élément de la liste est comparé à lui même).

Un algorithme permettant de repérer le plus petit nombre d'une liste à n éléments est donc le suivant (la liste L_1 contenant les chiffres de D étant créée) :

ALGORITHME : *Plus petit élément d'une liste.*

```
Début
    Pour i variantDe 1 à n Faire
        Pour j variantDe 1 à n Faire
            Si L1(i) ⩽ L1(j) Alors
                L2(j) ←1
            Sinon
                L2(j) ←0
            FinSi
            Si Prod (L2) = 1 Alors
                L3(1) ← L1(i)
            FinSi
        FinPour
    FinPour
Fin
```

Pour le problème qui nous concerne, on peut balayer plusieurs fois la liste contenant les nombres à comparer avec l'algorithme précédent, à condition de remplacer chaque fois dans cette liste le plus petit nombre par un nombre supérieur à tous les autres, afin qu'il ne soit plus jamais repris. C'est tout à fait faisable dans le cas particulier qui est le nôtre puisque nous comparons des chiffres. On sait donc qu'ils sont tous inférieurs à 10. A chaque balayage, le plus petit nombre obtenu sera donc remplacé par 10 :
$10 \rightarrow L_1(I))$
Il ne reste plus qu'à itérer le processus. Intéressons nous ici au cas d'une liste de trois nombres à ordonner, et considérons l'algorithme suivant :

Essai : *Quatrième méthode : tri des chiffres de l'entier de départ.*

```
C←1
Pour i variantDe 1 à 3 Faire
    Pour j variantDe 1 à 3 Faire
        Si L1(i) ⩽ L1(j) Alors
            L2(j)←1
        Sinon
            L2(j)←0
        FinSi
    FinPour
    Si Prod(L2 = 1) Alors
        L3(C)←L1(i)
        C←C + 1
        L1(i)←10
    FinSi
FinPour
```

On pourra demander aux élèves s'ils pensent que cet algorithme permet d'ordonner les nombres de la liste 1, en leur proposant d'imaginer les résultats pour une liste contenant au départ les nombres 1 2 3 dans cet ordre... puis dans un autre ordre.
Dans le premier cas, l'exécution de l'algorithme bascule 1 2 3 dans cet ordre en liste 3, et on retrouve bien 10 10 10 en liste 1. Par contre, si la liste de départ est par exemple 1 3 2, l'exécution de l'algorithme envoie seulement 1 et 2 en liste 3, et en fin de programme, la liste 1 contient respectivement 10 3 10. Si la liste de départ est 9 7 5, c'est encore pire, il faudra 7 balayages avant de retrouver la liste 1 ordonnée dans la liste 3.

L'algorithme précédent n'est donc pas valable, et on conçoit qu'il faut déterminer un critère d'arrêt avec la condition « tant que ».

Le processus sera terminé lorsque chaque nombre de la liste 1 aura été remplacé par 10, donc, par exemple, lorsque le produit des nombres de cette liste sera égale à 1000 (ou sa somme à 30).

L'entier comparé à tous les autres est successivement le premier, puis le deuxième, puis le troisième, et retour au premier, et ainsi de suite. On utilisera donc pour déterminer cet entier « pivot » une congruence modulo 3 augmentée de 1 puisque les termes d'une liste sont indicés à partir de 1.

L'entier comparé aux autres est celui de rang $I = \underbrace{N - 3\text{int}\left(\frac{N}{3}\right)}_{\text{reste dans la division de N par 3}} +1$, où N désigne le nombre de balayages.

Chiffres de départ 9 7 5

Rang du balayage	Entier comparé aux autres	Position dans la liste	Est il le plus petit ?	Liste 1	Liste 2
1	9	1	non	9	
				7	
				5	
2	7	2	non	9	
				7	
				5	
3	5	3	oui	9	5
				7	
				10	
4	9	1	non	9	5
				7	
				10	
5	7	2	oui	9	5
				10	7
				10	
6	10	3	non	9	5
				10	7
				10	
7	9	1	oui	10	5
				10	7
				10	9

En langage naturel, la liste L_1 étant créée :

ALGORITHME : *Troisième méthode de recherche de G et P.*

$C \leftarrow 1$
$N \leftarrow 0$
TantQue $\texttt{Prod}(L_1) \neq 1000$ **Faire**
  $i \leftarrow N - 3\text{int}\left(\frac{N}{3}\right) + 1$
  $N \leftarrow N + 1$
  Pour j **variantDe** 1 **à** 3 **Faire**
    Si $L_1(i) \leqslant L_1(j)$ **Alors**
      $L_2(j) \leftarrow 1$
    Sinon
      $L_2(j) \leftarrow 0$
    FinSi
  FinPour
  Si $\texttt{Prod}\ (L_2) = 1$ **Alors**
    $L_3(C) \leftarrow L_1(i)$
    $C \leftarrow C + 1$
    $L_1(i) \leftarrow 10$
  FinSi
FinTantQue
$G \leftarrow 100L_3(3) + 10L_3(2) + L_3(1)$
$P \leftarrow 100L_3(1) + 10L_3(2) + L_3(3)$

L'avantage de cette méthode par rapport à la méthode par comparaison étudiée précédemment est qu'elle est facilement généralisable au tri d'une liste de n chiffres.

d) ***Quatrième méthode : position ordinale dans une liste.***

La méthode qui va suivre est nettement moins coûteuse en termes d'opérations pour déterminer les entiers G et P.

Elle consiste à placer les chiffres dans une liste dans leur position ordinale : par exemple : s'il s'agit respectivement de 7, 2 et 3, on envoie un 1 en huitième, troisième et quatrième position dans une deuxième liste, dont les dix premiers termes sont préalablement initialisés à 0 (la première position dans la liste est réservée au chiffre 0).

- Dans un premier temps, on considérera que les trois chiffres composant le nombre N sont distincts.Ils sont placés dans les trois premières positions de la liste L_1.

ALGORITHME : *Quatrième méthode : création de L_2 (chiffres distincts).*

Pour k **variantDe** 1 **à** 10 **Faire**
  $L_2(k) \leftarrow 0$
FinPour
Pour i **variantDe** 1 **à** 3 **Faire**
  $L_2(L_1(i) + 1) \leftarrow 1$
FinPour

On augmente $L_1(i)$ de 1 pour le cas où cette quantité serait nulle : le zéro est repéré en première position de la liste 2,..., le 9 en dixième.

Nombre de départ :			975
Rang dans la liste	Liste 1	Liste 2	Chiffre correspondant
K	L1(K)	L 2(K)	k-1
1	9		0
2	7		1
3	5		2
4			3
5			4
6		5	5
7			6
8		7	7
9			8
10		9	9

Il suffit maintenant de monter ou descendre la liste L_2 pour reconstituer G et P :

ALGORITHME : *Quatrième méthode : recherche de G et P (chiffres distincts).*

$P \leftarrow 0$
$G \leftarrow 0$
$J \leftarrow 0$
Pour K **variantDe** 1 **à** 10 **Faire**
  Si $L_2(k) \neq 0$ **Alors**
    $G \leftarrow (K-1) \times 10^J$
    $P \leftarrow (K-1) \times 10^{2-J}$
    $J \leftarrow J+1$
  FinSi
FinPour

- Intéressons nous maintenant aux nombres de trois chiffres avec deux éventuellement égaux.
  La liste L_2 contiendra donc une valeur égale à 2, et l'algorithme précédent est en défaut dès la construction de cette liste.
  Les dix premiers termes de la liste L_2 sont initialisés à 0. On écrit alors :

ALGORITHME : *Quatrième méthode : création de L_2 (chiffres pouvant se répéter).*

Pour k **variantDe** 1 **à** 10 **Faire**
  $L_2(k) \leftarrow 0$
FinPour
Pour i **variantDe** 1 **à** 3 **Faire**
  $L_2(L_1(i)+1) \leftarrow L_2(L_1(i)+1)+1$
FinPour

Le calcul de G et P se fait alors de la façon suivante :

ALGORITHME : *Quatrième méthode : recherche de G et P (chiffres pouvant se répéter).*

$P \leftarrow 0$
$G \leftarrow 0$
$J \leftarrow 0$
Pour k **variantDe** 1 **à** 10 **Faire**
  Si $L_2(k) \neq 0$ **Alors**
    $G \leftarrow (K-1) \times 10^J$
    $P \leftarrow (K-1) \times 10^{2-J}$
    $J \leftarrow J+1$
    Si $L_2(k) = 2$ **Alors**
      $G \leftarrow (K-1) \times 10^J$
      $P \leftarrow (K-1) \times 10^{2-J}$
      $J \leftarrow J+1$
    FinSi
  FinSi
FinPour

Si on veut utiliser cette technique pour un nombre de α chiffres, l'un d'entre eux pouvant se répéter jusqu'à $\alpha - 1$

fois, on n'écrira pa $\alpha - 1$ conditions. On pourra utiliser une boucle « Tant que » et une variable compteur qu'on décrémente :
La liste L_2 étant créée :

ALGORITHME : *Quatrième méthode : recherche de G et P (Nombre de départ à α chiffres, répétitions possibles).*

$P \leftarrow 0$
$G \leftarrow 0$
$J \leftarrow 0$
Pour k **variantDe** 1 **à** 10 **Faire**
  Si $L_2(k) \neq 0$ **Alors**
    $Z \leftarrow L_2(k)$
    TantQue $Z \neq 0$ **Faire**
      $G \leftarrow (K-1) \times 10^J$
      $P \leftarrow (K-1) \times 10^{\alpha-1-J}$
      $J \leftarrow J+1$
      $Z \leftarrow Z-1$
    FinTantQue
  FinSi
FinPour

Rappel : α désigne ici le nombre de chiffres dans l'écriture décimale de G (donc du nombre de départ si celui ci ne contient pas de 0). Ici, α est donc égal à la somme des termes de la liste L_2.

3. Itération du processus.

Nous sommes maintenant prêts pour répéter ce mécanisme. L'utilisateur de l'algorithme suivant est invité à donner le nombre d'itérations souhaitées en début de programme. Le nombre de départ, puis les entiers obtenus à chaque étape sont stockés en liste L_3.

Si l'entier de départ proposé est constitué de trois chiffres identiques, on renvoie un message d'erreur, et le programme se termine.

ALGORITHME : *Itérations à la demande*

Début
- Effacer les listes L_1, L_2, L_3
- Demander la valeur de D
- $D \leftarrow$ réponse
- **Si** $\text{Int}\left(\frac{D}{111}\right) = \frac{D}{111}$ **Alors**
  - Ecrire « Attention, chiffres identiques »
- **Sinon**
  - $L_3(1) \leftarrow$ D
  - Demander le nombre d'itérations
  - $F \leftarrow$ réponse
  - $C \leftarrow 1$
  - **TantQue** $C < F + 1$ **Faire**
    - **Pour** i **variantDe** 1 **à** 3 **Faire**
      - $L_1(i) \leftarrow D - 10 \times \text{int}\left(\frac{D}{10}\right)$
      - $D \leftarrow \text{int}\left(\frac{D}{10}\right)$
    - **FinPour**
    - **Pour** k **variantDe** 1 **à** 10 **Faire**
      - $L_2(k) \leftarrow 0$
    - **FinPour**
    - **Pour** i **variantDe** 1 **à** 3 **Faire**
      - $L_2(L_1(i)+1) \leftarrow L_2(L_1(i)+1) + 1$
    - **FinPour**
    - $P \leftarrow 0$
    - $G \leftarrow 0$
    - $J \leftarrow 0$
    - **Pour** k **variantDe** 1 **à** 10 **Faire**
      - **Si** $L_2(k) \neq 0$ **Alors**
        - $G \leftarrow G + (K-1) \times 10^J$
        - $P \leftarrow P + (K-1) \times 10^{2-J}$
        - $J \leftarrow J + 1$
        - **Si** $L_2(k) = 2$ **Alors**
          - $G \leftarrow G + (K-1) \times 10^J$
          - $P \leftarrow P + (K-1) \times 10^{2-J}$
          - $J \leftarrow J + 1$
        - **FinSi**
      - **FinSi**
    - **FinPour**
    - $C \leftarrow C + 1$
    - $D \leftarrow G - P$
    - $L_3(C) \leftarrow D$
  - **FinTantQue**
  - Afficher « Consulter la liste L_3 »
- **FinSi**

Fin

Pour un nombre de départ quelconque (deux chiffres ou plus) :

ALGORITHME : *Itérations à la demande. D a n chiffres*

Début
- Effacer les listes L_1, L_2, L_3
- Demander la valeur de D
- $D \leftarrow$réponse
- $n \leftarrow \texttt{Int}(\log(D)) + 1$
- $L_3(1) \leftarrow$D
- Demander le nombre d'itérations
- $F \leftarrow$réponse
- $C \leftarrow 1$
- **TantQue** $C < F + 1$ **Faire**
  - **Pour** i **variantDe** 1 **à** n **Faire**
    - $L_1(i) \leftarrow D - 10 \times \texttt{int}\left(\frac{D}{10}\right)$
    - $D \leftarrow \texttt{int}\left(\frac{D}{10}\right)$
  - **FinPour**
  - **Pour** k **variantDe** 1 **à** 10 **Faire**
    - $L_2(k) \leftarrow 0$
  - **FinPour**
  - **Pour** i **variantDe** 1 **à** n **Faire**
    - $L_2(L_1(i) + 1) \leftarrow L_2(L_1(i) + 1) + 1$
  - **FinPour**
  - **Si** $\texttt{Max}(L_2(i)) = n$ **Alors**
    - Ecrire « les chiffres sont tous identiques »
  - **Sinon**
    - $P \leftarrow 0$
    - $G \leftarrow 0$
    - $J \leftarrow 0$
    - **Pour** k **variantDe** 1 **à** 10 **Faire**
      - **Si** $L_2(k) \neq 0$ **Alors**
        - $Z \leftarrow L_2(k)$
        - **TantQue** $Z \neq 0$ **Faire**
          - $G \leftarrow G + (K - 1) \times 10^J$
          - $P \leftarrow P + (K - 1) \times 10^{(n-1)-J}$
          - $J \leftarrow J + 1$
          - $Z \leftarrow Z - 1$
        - **FinTantQue**
      - **FinSi**
    - **FinPour**
    - $C \leftarrow C + 1$
    - $D \leftarrow G - P$
    - $L_3(C) \leftarrow D$
  - **FinSi**
- **FinTantQue**
- Afficher « Consulter la liste L_3 »

Fin