Cet article est rédigé par des élèves. Il peut comporter des oublis et imperfections, autant que possible signalés par nos relecteurs dans les notes d'édition.

Construire des nombres entiers sous contraintes

Année 2023-2024

Romane POUZOULET, Jeanne LOUGE-SOULÉ, Lexina GONZALES, Mélanie CABRIT, Mathilde VIALA, Charline DELAPORTE, Kenzo SENAT, Louis BEDUER, Tymon LECOMTE, Mathis FRANCOUAL, Alexis MAHOUDEAUX,Alexandre AUBER-DEBELLE,Eleny ANCIAUX,Thais BASTOS DA SILVA JULIANN

Établissements : Lycée Raymond Savignac et Collège Georges Pompidou

Enseignants : Mme.VERNHET, M.THOMAS, M.BEDUER et M.LABIT

Chercheuse : Martine KLUGHERTZ, chercheuse à l'IRES

1. Introduction

1.1 Présentation

- Prenons 5 fois le nombre 5.
- Avec les opérations habituelles (+, -, x, ÷) et en utilisant les 5 nombres 5, comment obtenir le nombre 24 ?
- On observe deux catégories de nombres. Si on fait la même chose avec 6 fois le nombre 6, qu'est-ce qu'on remarque ?
- En généralisant à n fois le nombre entier n, comment décrire toutes les possibilités ?
- On ne prend pas en compte les résultats négatifs et les décimaux.

1.2. Problème posé

Quels sont tous les nombres entiers que l'on peut obtenir en utilisant 5 fois le nombre 5 avec les opérations habituelles ?

2. Nos premières recherches

2.1. Comment obtenir 24 avec × ; ÷ ; + ; -?

Avec 5 fois le chiffre 5 :

(5×5×5-5)÷5=24

Avec 6 fois le chiffre 6 :

6x6-6-6+6-6 = 24

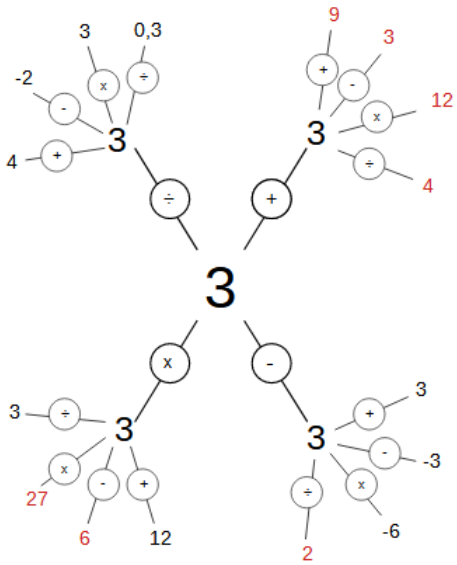
2.2. Simplification du sujet.

Nous avons simplifié notre sujet de différentes manières :

- Pour commencer, nous avons simplifié avec 3 fois le nombre 3.
- Nous avons également simplifié en prenant : 1 opérateur, 2 opérateurs différents, 3 opérateurs différents pour arriver à 4 opérateurs différents. Nous avons ensuite regroupé les résultats dans différents tableaux afin de regrouper et trier les résultats obtenus.

2.3. Simplification du problème avec 3 fois le nombre 3.

Nous avons commencé nos recherches avec 3 fois le nombre 3. Nous avons cherché tous les calculs sans parenthèses possibles qui ont été présentés sur l'arbre suivant :



3. Nos propriétés

Grâce à nos recherches, nous avons trouvé quelques propriétés :

Tout d'abord, il est toujours possible d'obtenir 0 avec n fois le nombre n strictement supérieur à 1. En effet, il suffit d'ajouter n puis de l'annuler et ensuite de multiplier cela par n autant que nécessaire : (n-n)×n×n×n×…=0. Cette propriété ne fonctionne uniquement lorsqu'on utilise les parenthèses.

Ensuite, nous pouvons toujours obtenir 1 avec n pair strictement supérieur à 0. En effet, il suffit de diviser n par n puis d'ajouter n et de le supprimer : n÷n+n-n+n-n…=1

Pour continuer, nous pouvons toujours obtenir 2 avec n impair supérieur ou égal à 3. En effet, il nous faut ajouter n à n puis diviser cette addition par n. On ajoute enfin n puis on l'annule autant de fois que nécessaire : (n+n)÷n+n-n…=2. Cette propriété ne fonctionne qu'avec des parenthèses.

On obtient 2 sortes de nombres qui sont des multiples de n et les autres qui ne le sont pas. Nous avons remarqué que si l'on veut trouver un résultat non-multiple de n, alors il faut utiliser un ÷.

4. **5 fois le nombre 5**

4.1. Nos résultats

Sans parenthèses pour 5 fois le nombre 5, on a 4 opérations à faire avec 4 choix d'opérations. Il y a 256 calculs possibles en effet 4x4x4x4=256.
Au total, nous avons trouvé 28 résultats différents sans utiliser les parenthèses et en ne prenant pas en compte les résultats négatifs ou non entiers.
Nous avons classé en 2 catégories ces 28 résultats : les multiples de 5 et les non multiples de 5.

<u>Les multiples de 5 sont :</u>

- 5
- 10
- 15
- 20
- 25
- 30
- 40
- 45
- 55
- 100
- 115
- 125
- 135
- 150
- 620
- 630
- 3125

<u>Les non-multiples de 5 sont :</u>

- 3
- 4
- 6
- 7

- 14
- 16
- 21
- 29
- 31
- 124
- 126

4.2. **Formule et matrices**

Après avoir trouvé tous les résultats possibles avec 5 fois le nombre 5 sans parenthèse, nous avons voulu créer une formule qui nous permettrait de retrouver nos résultats.

Pour cela, nous avons commencé par écrire nos résultats sous forme de puissance de 5. Par exemple,
$125=5^3$ $100=5^3-5^2$

Ensuite nous avons créer la formule suivante :

$\sum_{j=1}^{k} a_{jk} * 5^j$ avec $k \epsilon \{1;2;3;4;5\}$

Par exemple pour k=3; $a_{13}*5^1+a_{23}*5^2+a_{33}*5^3$

Nous avons créé un tableau sous forme de matrice qui recense toutes les valeurs de a en fonction de j et k.

En revanche, avec ce fonctionnement, nous n'arrivons pas à trouver certains résultats. Les nombres pour lesquels il fallait calculer une puissance puis l'annuler, comme $5=5^2-5^2+5^1$, n'étaient pas réalisables. Pour résoudre cette problématique, nous nous sommes rendus compte qu'il fallait mettre une somme de nombre sous la forme d'un facteur devant les puissances. Par exemple, $a_{jk}=(3+1-1)$. Pour les résultats non multiples de 5, il faut que l'un des facteurs a soit sous forme de fraction par exemple a=1/5 . En effet, pour trouver un résultat non-multiple de 5 il faut utiliser au moins une fois l'opérateur ÷ dans le calcul.

Voici la matrice des résultats multiples de 5 :

	k					
j		1	2	3	4	5
	1	a_{11} 5 (3+1-1)	a_{12} 1 (2-1) (-2+1) 1 ou -1 3 ou -3	a_{13} (1-1) 0	a_{14} 1 ou -1	a_{15} 0
	2	a_{21} 0	a_{22} (1-1) 2 1	a_{23} 0 1 ou -1	a_{24} 0	a_{25} 0
	3	a_{31} 0	a_{32} 0	a_{33} 1 0	a_{34} 0	a_{35} 0
	4	a_{41} 0	a_{42} 0	a_{43} 0	a_{44} 1	a_{45} 0
	5	a_{51} 0	a_{52} 0	a_{53} 0	a_{54} 0	a_{55} 1

Voici la matrice des résultats non-multiples de 5 :

	k			
j		1	2	3
	1	a_{11} (1/5+3) (-1/5+3) (2/5+1) (-2/5+1) (-1/5+2-1)	a_{12} (1-1/5) (-1+1/5) (1+1/5) 1	a_{13} 1/5 ou -1/5
	2	a_{21} 0	a_{22} 1 $1/5^2$	A_{23} 0
	3	a_{31} 0	a_{32} 0	a_{33} 1

Cette matrice se limite à 3 colonnes et 3 lignes car en recensant toutes les valeurs de a, nous nous sommes rendus compte qu'elles étaient toutes égales à 0 pour k et j étant égal à 4 et 5(1).

5. **Nos programmes**

5.1. Avec Python

Nous avons fait un programme Python en 4 fonctions. Notre programme nous permet d'obtenir une liste où l'on trouve à la fois les opérateurs utilisés pour ce calcul et le résultat. Cependant, la difficulté que nous avons rencontrée est que, dans notre programme, nous ne respectons pas les priorités opératoires, il n'y a pas de parenthèses, ainsi nous ne trouvons pas les mêmes résultats avec notre programme et notre formule.

Par exemple : En respectant les prioritées opératoires : 5/5+5/5+5=7

- Dans notre programme : (((5/5)+5)/5)+5=6,2

```
def CONSTRUCTION ():
    """
    fonction qui permet de créer une liste qui contient toutes les listes d'opérateurs
    """
    liste_opérateurs=[]
    for i in range (1,5):
        for j in range (1,5):
            for k in range (1,5):
                for l in range (1,5):
                    liste_opérateurs.append([i,j,k,l])
    return liste_opérateurs
```

```
>>> CONSTRUCTION()
[[1, 1, 1, 1],
 [1, 1, 1, 2],
 [1, 1, 1, 3],
 [1, 1, 1, 4],
 [1, 1, 2, 1],
 [1, 1, 2, 2],
 [1, 1, 2, 3],
 [1, 1, 2, 4],
 [1, 1, 3, 1],
 [1, 1, 3, 2],
 [1, 1, 3, 3],
 [1, 1, 3, 4]
```

Cette première fonction nous permet de créer une liste avec 4 nombres rangés dans un ordre aléatoire, pour cela nous avons utilisé la fonction « for i in range » pour parcourir toutes les possibilités. Nous avons ensuite stocké toutes ses combinaisons dans une liste.

Cette deuxième fonction associe chaque numéro stocké de notre liste, à un opérateur :

- Le "1" représente le "+" ;
- Le "2" représente le "-" ;
- Le "3" représente le "x" ;
- Le "4" représente le "÷".

Chaque numéro de la liste ajoute, soustrait, multiplie ou divise le nombre demandé (dans notre cas 5) au calcul trouvé précédemment.

```python
def operation(operateur,n):
    """
    fonction qui permet d'associer chaque nombre à une opération
    """
    S=n
    for A in operateur :
        if A==1:
            S=S+n
        elif A==2:
            S=S-n
        elif A==3:
            S=S*n
        else:
            S=S/n
    return(S)
```

Cette troisième fonction permet d'ajouter à une nouvelle liste, à la fois les opérateurs utilisés, mais aussi le résultat du calcul.

```python
def ENLEVER(n):
    """
    Cette fonction permet de créer une nouvelle liste dans laquelle on enlève les nombres négatifs, décimaux et les doublons.
    """
    bilan=RESULTATS(n)
    resultat=[]
    result = []
    for i in range (len (bilan)):
        calcul=bilan[i]
        if calcul[1]>0 and calcul[1]%1==0.0 and calcul[1] not in result: #calcul [1] correspond à la somme dans une ligne
            resultat.append(calcul) #ajouter si ils ne sont pas décimaux ni négatifs
            result.append(calcul[1])
    return resultat
```

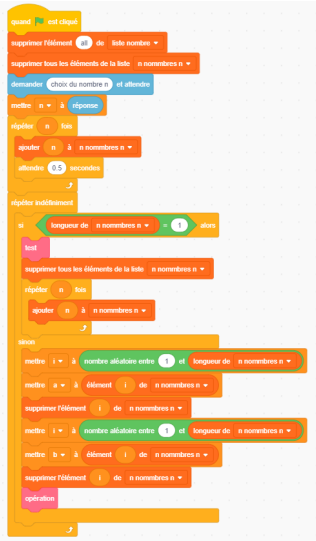
```
>>> ENLEVER(5)
[([1, 1, 1, 1], 25),
 ([1, 1, 1, 2], 15),
 ([1, 1, 1, 3], 100),
 ([1, 1, 1, 4], 4.0),
 ([1, 1, 2, 2], 5),
 ([1, 1, 2, 3], 50),
 ([1, 1, 2, 4], 2.0),
 ([1, 1, 3, 1], 80),
 ([1, 1, 3, 2], 70),
 ([1, 1, 3, 3], 375),
 ([1, 1, 4, 1], 8.0)
```

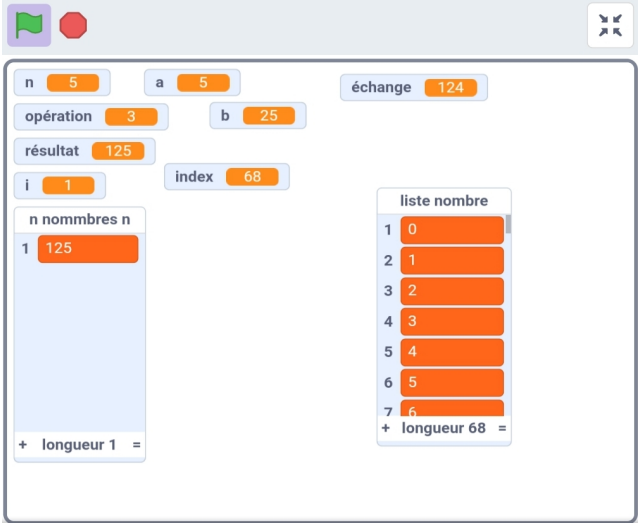
Cette dernière fonction permet de trier les résultats. En effet, nous ne prenons pas en compte les résultats négatifs, décimaux et doubles. Pour enlever les décimaux nous avons utilisé une condition : si le reste de la division euclidienne par 1 est égal à 0 alors le résultat peut potentiellement être ajouté à la liste. Pour qu'il soit ajouté à la liste il faudra aussi qu'il respecte la condition d'être supérieur à 0 et de ne pas avoir déjà existé dans la liste.

Ainsi grâce à notre programme nous avons trouvé tous les résultats possibles avec 5 fois le nombre 5 sans parenthèses, que nous avons pu stocker dans une liste afin de les voir ainsi que la façon dont ils ont été construits.

5.2. Avec Scratch

Nous avons fait un programme sur Scratch. Notre programme nous permet d'obtenir une liste où l'on trouve tous les résultats avec 5 fois le chiffre 5. Cependant, la difficulté que nous avons rencontrée est que le programme ne pouvait pas faire tous les calculs.
Mais grâce au tableau ci-dessous (5.3) nous avons pu améliorer notre programme pour qu'il puisse faire tous les calculs et donc trouver tous les résultats.





Notre programme scratch insère 5 fois le chiffre 5 ou n fois le nombre n dans la liste "n nombre n" et en sort 2 nombres de manière aléatoire puis fait un calcul +, - , ÷ ou × puis les réinsère dans la liste. Il refait cette procédure encore jusqu'à ce qu'il ne reste plus qu'un nombre dans la liste. Et enfin le programme teste le nombre et s'il est entier, positif et qu'il n'est pas déjà intégré dans la liste "liste nombre" alors le programme l'insère dans la liste de nombres.
Lien de notre programme : https://scratch.mit.edu/projects/103022898

5.3. Tableau de rangement des nombres

Ici on a voulu travailler avec un tableau. On a rangé les nombres que l'on a trouvé avec 5 fois le nombre 5 et les opérations habituelles (+, -, ÷ et ×). Nous avons réalisé différents calculs comme par exemple : 5/5+5*5*5=126

Nous avons ensuite classé tous ces résultats dans le tableau ci-dessus. Ils sont classés dans l'ordre, par dizaines. Au total, nous avons trouvé 54 résultats différents .

	A	B	C	D	E	F	G	H	I	J
1	0	1	2	3	4	5	6	7	8	9
2	10	11	12		14	15	16			19
3	20	21		23	24	25	26	27		29
4	30	31				35				
5	40					45				49
6	50	51				55				
7	60									
8	70					75				
9	80									
10						95				
11	100					105				
12						115				
13	120				124	125	126			
14	130					135				
15						145				
16	150					155				
17						175				
18	200									
19						225				
20						245				
21	250					255				
22						275				
23	300									
24						375				
25	500									
26						525				
27	600									
28	620									
29	630									
30	650									
31	750									
32	1250									
33						3125				

6. Conclusion

Premièrement, nous avons réussi à trouver comment obtenir 24, nous avons aussi réussi à faire un programme scratch qui nous a permis de trouver tous les résultats que l'on pouvait obtenir avec n'importe quel nombre et avec les opérations +, -, ×, ÷. Mais ce programme est lent quand les nombres sont trop grands même au delà de 7 le programme est assez lent à trouver tous les résultats.
D'autre part, la formule que nous avons trouvée nous permet de retrouver l'ensemble des résultats sans parenthèses. Calculs que nous avons rangés sous forme de 2 matrices. Cette formule pourrait être généralisée pour n fois le nombre n et nous permettrait de trouver tous ces résultats.

Note d'édition

(1) Dans la troisième colonne de cette matrice il faut lire en deuxième ligne a_{23} à la place de A_{23}