Présentation MATh.en.JEANS

Sujet 3 :
Manger le plus de pâtisseries

Aïda CADI
Amina LAMRI
Ghiles DEGHEB
Maha BENHAMIDA
Meriem BERKANE
Mounir GOZIM
Tadla MEZOUED
- en 1ère générale spécialité Mathématiques

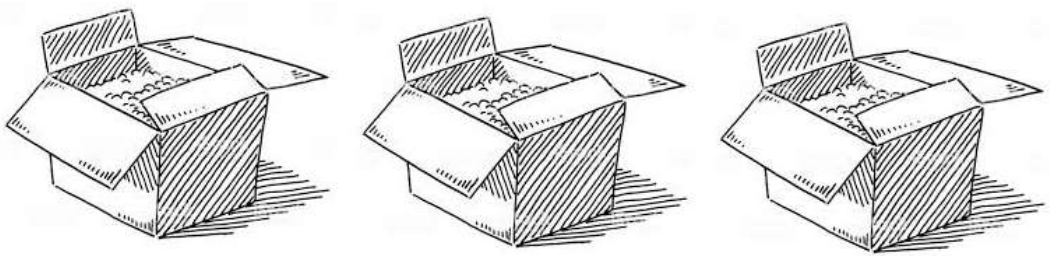
Atelier M.e.J Lycée International Alexandre Dumas (ALGERIE)

Présentation du sujet

SUJET 3 : **Manger le plus de pâtisseries**

Vous disposez de **n boîtes** fermées dont chacune contient un certain nombre de **pâtisseries**.

On suppose que tous ces nombres sont différents et non connus. On vous propose alors de jouer au **jeu** suivant : **vous ouvrez une boîte, puis une seconde, puis une troisième etc.**



A chaque instant, vous avez le droit de choisir entre **deux options** :

- soit vous prenez les pâtisseries de la boîte que vous venez de choisir et le jeu s'arrête,
- soit vous les refusez et vous continuez à ouvrir la boîte suivante.

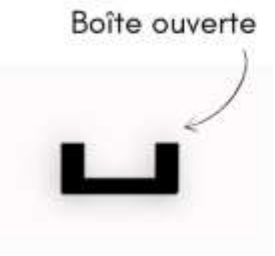
Vous ne prenez que le contenu de la dernière boîte ouverte.

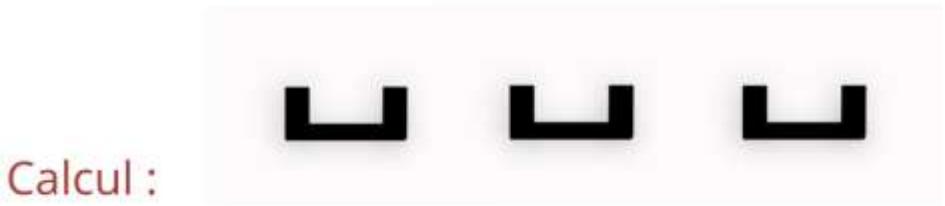
OU

PROBLÉMATIQUE

Quelle est la meilleure façon de jouer, c'est-à-dire celle qui vous permet de trouver la boîte qui contient le plus grand nombre de pâtisseries ?

Méthodologie





Calcul :

$$\frac{\text{nombre total de pâtisseries - nombre de pâtisseries des boîtes ouvertes}}{\text{nombre de boîtes restantes}}$$

Étape préliminaire :

Test d'une 1ère stratégie :

Calculer le nombre moyen de pâtisseries par boîte restante après chaque ouverture :

- Arrêter de jouer si le nombre de pâtisseries de la boîte ouverte est **supérieur** à ce nombre
- Continuer le jeu si le nombre de pâtisseries de la boîte ouverte est **inférieur** à ce nombre

ÉCHEC de l’étape préliminaire

Abandon de la stratégie :

- Donnée non fournie par l’énoncé : nombre total de pâtisseries
- Test sur Python : stratégie inefficace

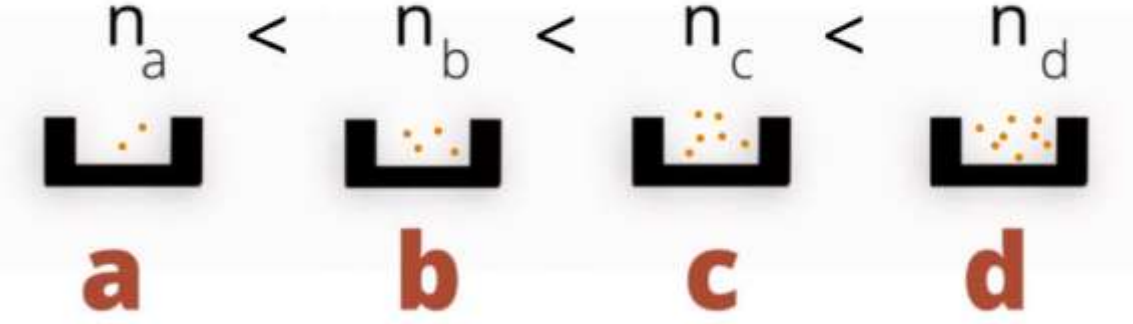
Méthode d’investigation : Déterminer des stratégies et les tester à l’aide d’un exemple

Stratégies : critères d’arrêt (Quand arrêter de jouer ?) basés sur le repérage d'augmentation(s) ou diminution de valeurs, d'une boîte à l'autre.

- sur une 1ère augmentation
- sur une 2e augmentation
- sur une 3e augmentation
- sur un nouveau maximum
- sur un 2e changement de maximum
- à la 2e augmentation consécutive
- sur la 1ère augmentation suivant une baisse
- à partir de la 3e boîte si changement ou 2e changement de maximum

Test des stratégies : Pour **n=4**

Les boîtes sont classées dans l'ordre croissant de leur contenance en pâtisseries :



La boîte d est celle qui contient le plus de pâtisseries

Les boîtes sont par la suite mélangées

a	b	c	d
a	b	d	c
a	c	b	d
a	c	d	b
a	d	b	c
a	d	c	b
b	a	c	d
b	a	d	c
b	c	a	d
b	c	d	a
b	d	a	c
b	d	c	a
c	a	b	d
c	a	d	b
c	b	a	d
c	b	d	a
c	d	a	b
c	d	b	a
d	a	b	c
d	a	c	b
d	b	a	c
d	b	c	a
d	c	a	b
d	c	b	a

Nombre de façons de choisir toutes ces boites l'une après l'autre :

Il y a 24 manières de mélanger ces 4 boîtes.

4! = 4×3×2×1= 24

Pour chaque combinaison, le jeu est gagné si le joueur choisit de s’arrêter sur la boîte **d**.

a	(b)	c	d	non
a	(b)	d	c	non
a	(c)	b	d	non
a	(c)	d	b	non
a	(d)	b	c	OUI
a	(d)	c	b	OUI
b	a	(c)	d	non
b	a	(d)	c	OUI
b	(c)	a	d	non
b	(c)	d	a	non
b	(d)	a	c	OUI
b	(d)	c	a	OUI
c	a	(b)	d	non
c	a	(d)	b	OUI
c	b	a	(d)	OUI
c	b	(d)	a	OUI
c	(d)	a	b	OUI
c	(d)	b	a	OUI
d	a	(b)	c	non
d	a	(c)	b	non
d	b	a	(c)	non
d	b	(c)	a	non
d	c	a	(b)	non
d	c	b	(a)	non

Calcul de la **probabilité** de **gagner** au jeu avec toutes les stratégies déterminées

<u>Exemple 1</u> : Calcul de la probabilité de gagner au jeu en utilisant la stratégie “s’arrêter sur la première augmentation”

- La boîte à ouvrir en dernier pour gagner le jeu est en rouge
- La dernière boîte ouverte en suivant la stratégie est *entourée*

La stratégie permet de gagner le jeu dans **10** issues sur **24**

La probabilité de gagner en suivant cette stratégie est donc :

P = 10/24

a	(b)	c	d	non
a	(b)	d	c	non
a	(c)	b	d	non
a	(c)	d	b	non
a	(d)	b	c	OUI
a	(d)	c	b	OUI
b	a	(c)	d	non
b	a	(d)	c	OUI
b	(c)	a	d	non
b	(c)	d	a	non
b	(d)	a	c	OUI
b	(d)	c	a	OUI
c	a	b	(d)	OUI
c	a	(d)	b	OUI
c	b	a	(d)	OUI
c	b	(d)	a	OUI
c	(d)	a	b	OUI
c	(d)	b	a	OUI
d	a	(b)	c	non
d	a	(c)	b	non
d	b	a	(c)	non
d	b	(c)	a	non
d	c	a	(b)	non
d	c	b	(a)	non

<u>Exemple 2</u> : Calcul de la probabilité de gagner au jeu en utilisant la stratégie “s’arrêter sur un changement de maximum”

- La boîte à ouvrir en dernier pour gagner le jeu est en rouge
- La dernière boîte ouverte en suivant la stratégie est *entourée*

La stratégie permet de gagner le jeu dans **11** issues sur **24**

La probabilité de gagner en suivant cette stratégie est donc :

P = 11/24

De la même manière, on calcule la probabilité de gagner en suivant chaque stratégie :

On trouve : La probabilité de gagner en suivant la stratégie :

- arrêt sur une augmentation : **P=10/24**
- arrêt sur une 2e augmentation : **P=7/24**
- arrêt sur une 3e augmentation : **P=1/24**
- arrêt sur un nouveau maximum : **P=11/24**
- arrêt sur un 2e changement de maximum : **P=6/24**
- arrêt à la 2e augmentation consécutive : **P=5/24**
- arrêt sur la 1ère augmentation suivant une baisse : **P=5/24**
- arrêt à partir de la 3e boîte si changement ou 2e changement de maximum : **P=10/24**

Les deux meilleures stratégies découvertes dans nos essais sont :

- “Arrêt sur la première augmentation” (P=10/24)
- “Arrêt sur un nouveau maximum” (P=11/24)

Objectifs fixés pour finaliser notre recherche :

- Tester ces stratégies à l’aide d’un algorithme
- Modéliser mathématiquement la première stratégie

Méthodologie de validation des résultats

Algorithme de dénombrement

Cet algorithme nous permet de déterminer la probabilité théorique de succès en suivant la stratégie 1 avec un nombre de boîtes donné.

```
from itertools import *

nbBoites = int(input("Combien de boites faut-il remplir ?\n"))
nbAugmentations = int(input("Au bout de combien d'augmentations faut il sélectionner la boite ?\n"))
score = 0
lettre = 97

tableau = []
c=0
for loop in range(nbBoites):



    r = chr(lettre)
    tableau.append(r)

    lettre = lettre+1
```

```
for y in permutations(tableau):
    i=0
    r=0
    c=c+1
    augmentation = 0

    print(y)
    while i < len(tableau)-1 and augmentation < nbAugmentations:

        if y[i+1] > y[i]:
            augmentation = augmentation +1
            if   augmentation == nbAugmentations:
                r = y[i+1]

        if r == tableau[-1]:
            score = score + 1

        i=i+1



print(score)
print(c)
print("La probabilité de tomber sur la plus grande boite en choisissant\nune boite après",nbAugmentations,"augmentation(s) est de",score/c)
```

```
Combien de boites faut-il remplir ?
4
Au bout de combien d'augmentations faut il sélectionner la boite ?
1
('a', 'b', 'c', 'd')
('a', 'b', 'd', 'c')
('a', 'c', 'b', 'd')
('a', 'c', 'd', 'b')
('a', 'd', 'b', 'c')
('a', 'd', 'c', 'b')
('b', 'a', 'c', 'd')
('b', 'a', 'd', 'c')
('b', 'c', 'a', 'd')
('b', 'c', 'd', 'a')
('b', 'd', 'a', 'c')
('b', 'd', 'c', 'a')
('c', 'a', 'b', 'd')
('c', 'a', 'd', 'b')
('c', 'b', 'a', 'd')
('c', 'b', 'd', 'a')
('c', 'd', 'a', 'b')
('c', 'd', 'b', 'a')
('d', 'a', 'b', 'c')
('d', 'a', 'c', 'b')
('d', 'b', 'a', 'c')
('d', 'b', 'c', 'a')
('d', 'c', 'a', 'b')
('d', 'c', 'b', 'a')
10
24
La probabilité de tomber sur la plus grande boite en choisissant
une boite après 1 augmentation(s) est de 0.4166666666666667
```

Test de l'algorithme de dénombrement pour la stratégie 1

La stratégie 1 fonctionne bien et donne la bonne probabilité pour n=4

Suite méthode : Approche théorique de la stratégie 1

Les valeurs (u_i) sont croissantes: $u_1 < u_2 < u_3 < \dots < u_n$

Les boîtes sont mélangées puis numérotées. (les valeurs u_i ne seront alors plus rangées dans l'ordre croissant)

Supposons que u_n soit dans la $k^{ème}$ boîte.

Si on suit la stratégie **1**, combien de permutations nous font gagner ?
Il suffit de compter de combien de manières on peut avoir aucune augmentation de la 1ère à la (k-1)ème boîte. C'est-à-dire k-1 valeurs choisies parmi les (n-1) valeurs (dans toutes les boîtes sauf celles contenant u_n). Sachant que ces (k-1) valeurs sont disposées par ordre décroissant.

Cela revient à trouver le nombre de parties à k-1 éléments dans un ensemble à n-1 éléments :

$$\binom{n-1}{k-1} = \frac{(n-1)!}{(k-1)!\,((n-1)-(k-1))!}$$

Et pour chacune de ces possibilités, il y a encore (n-k)! manières de disposer les valeurs restantes (qui peuvent être échangées entre elles dans la liste).

C’est-à-dire :

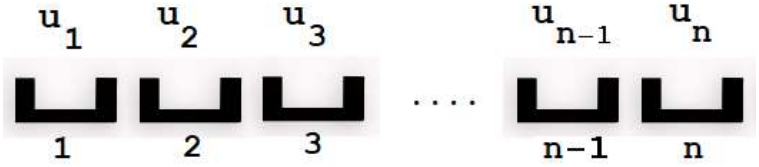
$$\frac{(n-1)(n-2)....(n-k+1)}{(k-1)(k-2)....3x2x1}$$

Et pour chacune des possibilités, il y a (n-k)! manières de disposer les valeurs restantes. On obtient :

$$\binom{n-1}{k-1} \times (n-k)!$$

Formule donnant le nombre de listes gagnantes avec u_n en $k^{ème}$ position.

u_n en position 1	nombre de listes gagnantes	0
u_n en position 2	nombre de listes gagnantes	$(n-1)\times(n-2)!$
u_n en position 3	nombre de listes gagnantes	$\binom{n-1}{2}\times(n-3)!$
u_n en position 4	nombre de listes gagnantes	$\binom{n-1}{3}\times(n-4)!$
. . .		
u_n en position n-1	nombre de listes gagnantes	$\binom{n-1}{n-2}\times(n-(n-1))!$
u_n en position n	nombre de listes gagnantes	$\binom{n-1}{n-1}\times(n-n)!$



$$\sum_{i=2}^{n}\binom{n-1}{i-1}(n-i)!$$

Nombre de listes gagnantes pour la stratégie liée à la première augmentation

Exemple : Pour 5 boîtes (n=5)

Nombre de gains sur la 1ère augmentation:

$$\sum_{i=2}^{5} \binom{4}{i-1}(5-i)! = \binom{4}{1}3! + \binom{4}{2}2! + \binom{4}{3}1! + \binom{4}{4}0!$$

$$= 4\times(3\times2\times1) + \frac{4\times3}{2}\times2\times1 + \frac{4\times3\times2}{3\times2\times1}\times1 + \frac{4\times3\times2\times1}{4\times3\times2\times1}\times1$$

$$= 24+12+4+1$$

$$= 41$$

Sur un total de 5×4×3×2×1=120 permutations

Probabilité : 41/120

Vérifié avec l'algorithme

```
Combien de boites faut-il remplir ?
5
Au bout de combien d'augmentations faut il sélectionner la boite ?
1
('a', 'b', 'c', 'd', 'e')
('a', 'b', 'c', 'e', 'd')
('a', 'b', 'd', 'c', 'e')
('a', 'b', 'd', 'e', 'c')
('a', 'b', 'e', 'c', 'd')
('a', 'b', 'e', 'd', 'c')
('a', 'c', 'b', 'd', 'e')
('a', 'c', 'b', 'e', 'd')
('a', 'c', 'd', 'b', 'e')
('a', 'c', 'd', 'e', 'b')
('a', 'c', 'e', 'b', 'd')
('a', 'c', 'e', 'd', 'b')
('a', 'd', 'b', 'c', 'e')
('a', 'd', 'b', 'e', 'c')
('a', 'd', 'c', 'b', 'e')
('a', 'd', 'c', 'e', 'b')
('a', 'd', 'e', 'b', 'c')
('a', 'd', 'e', 'c', 'b')
('a', 'e', 'b', 'c', 'd')
('a', 'e', 'b', 'd', 'c')
('a', 'e', 'c', 'b', 'd')
('a', 'e', 'c', 'd', 'b')
('a', 'e', 'd', 'b', 'c')
('a', 'e', 'd', 'c', 'b')
('b', 'a', 'c', 'd', 'e')
('b', 'a', 'c', 'e', 'd')
('b', 'a', 'd', 'c', 'e')
('b', 'a', 'd', 'e', 'c')
('b', 'a', 'e', 'c', 'd')
('b', 'a', 'e', 'd', 'c')
('b', 'c', 'a', 'd', 'e')
('b', 'c', 'a', 'e', 'd')
('b', 'c', 'd', 'a', 'e')
('b', 'c', 'd', 'e', 'a')
('b', 'c', 'e', 'a', 'd')
('b', 'c', 'e', 'd', 'a')
('b', 'd', 'a', 'c', 'e')
('b', 'd', 'a', 'e', 'c')
('b', 'd', 'c', 'a', 'e')
('b', 'd', 'c', 'e', 'a')
('b', 'd', 'e', 'a', 'c')
('b', 'd', 'e', 'c', 'a')
('b', 'e', 'a', 'c', 'd')
('b', 'e', 'a', 'd', 'c')
('b', 'e', 'c', 'a', 'd')
('b', 'e', 'c', 'd', 'a')
('b', 'e', 'd', 'a', 'c')
('b', 'e', 'd', 'c', 'a')
('c', 'a', 'b', 'd', 'e')
('c', 'a', 'b', 'e', 'd')
('c', 'a', 'd', 'b', 'e')
('c', 'a', 'd', 'e', 'b')
('c', 'a', 'e', 'b', 'd')
('c', 'a', 'e', 'd', 'b')
('c', 'b', 'a', 'd', 'e')
('c', 'b', 'a', 'e', 'd')
('c', 'b', 'd', 'a', 'e')
('c', 'b', 'd', 'e', 'a')
('c', 'b', 'e', 'a', 'd')
('c', 'b', 'e', 'd', 'a')
('c', 'd', 'a', 'b', 'e')
('c', 'd', 'a', 'e', 'b')
('c', 'd', 'b', 'a', 'e')
('c', 'd', 'b', 'e', 'a')
('c', 'd', 'e', 'a', 'b')
('c', 'd', 'e', 'b', 'a')
('c', 'e', 'a', 'b', 'd')
('c', 'e', 'a', 'd', 'b')
('c', 'e', 'b', 'a', 'd')
('c', 'e', 'b', 'd', 'a')
('c', 'e', 'd', 'a', 'b')
('c', 'e', 'd', 'b', 'a')
('d', 'a', 'b', 'c', 'e')
('d', 'a', 'b', 'e', 'c')
('d', 'a', 'c', 'b', 'e')
('d', 'a', 'c', 'e', 'b')
('d', 'a', 'e', 'b', 'c')
('d', 'a', 'e', 'c', 'b')
('d', 'b', 'a', 'c', 'e')
('d', 'b', 'a', 'e', 'c')
('d', 'b', 'c', 'a', 'e')
('d', 'b', 'c', 'e', 'a')
('d', 'b', 'e', 'a', 'c')
('d', 'b', 'e', 'c', 'a')
('d', 'c', 'a', 'b', 'e')
('d', 'c', 'a', 'e', 'b')
('d', 'c', 'b', 'a', 'e')
('d', 'c', 'b', 'e', 'a')
('b', 'd', 'e', 'c', 'a')
('b', 'e', 'a', 'c', 'd')
('b', 'e', 'a', 'd', 'c')
('b', 'e', 'c', 'a', 'd')
('b', 'e', 'c', 'd', 'a')
('b', 'e', 'd', 'a', 'c')
('b', 'e', 'd', 'c', 'a')
('c', 'a', 'b', 'd', 'e')
('c', 'a', 'b', 'e', 'd')
('c', 'a', 'd', 'b', 'e')
('c', 'a', 'd', 'e', 'b')
('c', 'a', 'e', 'b', 'd')
('c', 'a', 'e', 'd', 'b')
('c', 'b', 'a', 'd', 'e')
('c', 'b', 'a', 'e', 'd')
('c', 'b', 'd', 'a', 'e')
('c', 'b', 'd', 'e', 'a')
('c', 'b', 'e', 'a', 'd')
('c', 'b', 'e', 'd', 'a')
('c', 'd', 'a', 'b', 'e')
('c', 'd', 'a', 'e', 'b')
('c', 'd', 'b', 'a', 'e')
('c', 'd', 'b', 'e', 'a')
('c', 'd', 'e', 'a', 'b')
('c', 'd', 'e', 'b', 'a')
('c', 'e', 'a', 'b', 'd')
('c', 'e', 'a', 'd', 'b')
('c', 'e', 'b', 'a', 'd')
('c', 'e', 'b', 'd', 'a')
('c', 'e', 'd', 'a', 'b')
('c', 'e', 'd', 'b', 'a')
('d', 'a', 'b', 'c', 'e')
('d', 'a', 'b', 'e', 'c')
('d', 'a', 'c', 'b', 'e')
('d', 'a', 'c', 'e', 'b')
('d', 'a', 'e', 'b', 'c')
('d', 'a', 'e', 'c', 'b')
('d', 'b', 'a', 'c', 'e')
('d', 'b', 'a', 'e', 'c')
('d', 'b', 'c', 'a', 'e')
('d', 'b', 'c', 'e', 'a')
('d', 'b', 'e', 'a', 'c')
('d', 'b', 'e', 'c', 'a')
('d', 'c', 'a', 'b', 'e')
('d', 'c', 'a', 'e', 'b')
('d', 'c', 'b', 'a', 'e')
('d', 'c', 'b', 'e', 'a')
('d', 'c', 'b', 'a', 'e')
('d', 'c', 'b', 'e', 'a')
('d', 'c', 'e', 'b', 'a')
('d', 'c', 'e', 'a', 'b')
('d', 'c', 'e', 'b', 'a')
('d', 'c', 'e', 'a', 'b')
('d', 'c', 'b', 'e', 'a')
('d', 'c', 'b', 'a', 'e')
('d', 'c', 'a', 'e', 'b')
('d', 'c', 'a', 'b', 'e')
('d', 'b', 'e', 'c', 'a')
('d', 'b', 'e', 'a', 'c')
('d', 'b', 'c', 'e', 'a')
('d', 'b', 'c', 'a', 'e')
('d', 'b', 'a', 'e', 'c')
('d', 'b', 'a', 'c', 'e')
('d', 'a', 'e', 'c', 'b')
('d', 'a', 'e', 'b', 'c')
('d', 'a', 'c', 'e', 'b')
('d', 'a', 'c', 'b', 'e')
('d', 'a', 'b', 'e', 'c')
('d', 'a', 'b', 'c', 'e')
('c', 'e', 'd', 'b', 'a')
('c', 'e', 'd', 'a', 'b')
('c', 'e', 'b', 'd', 'a')
('c', 'e', 'b', 'a', 'd')
('c', 'e', 'a', 'd', 'b')
('c', 'e', 'a', 'b', 'd')
('c', 'd', 'e', 'b', 'a')
('c', 'd', 'e', 'a', 'b')
('c', 'd', 'b', 'e', 'a')
('c', 'd', 'b', 'a', 'e')
('c', 'd', 'a', 'e', 'b')
('c', 'd', 'a', 'b', 'e')
('c', 'b', 'e', 'd', 'a')
('c', 'b', 'e', 'a', 'd')
('c', 'b', 'd', 'e', 'a')
('c', 'b', 'd', 'a', 'e')
('c', 'b', 'a', 'e', 'd')
('c', 'b', 'a', 'd', 'e')
('b', 'e', 'd', 'c', 'a')
('b', 'e', 'd', 'a', 'c')
('b', 'e', 'c', 'd', 'a')
('b', 'e', 'c', 'a', 'd')
('b', 'e', 'a', 'd', 'c')
('b', 'e', 'a', 'c', 'd')
('b', 'd', 'e', 'a', 'c')
('b', 'd', 'e', 'c', 'a')
('b', 'd', 'c', 'e', 'a')
('b', 'd', 'c', 'a', 'e')
('b', 'd', 'a', 'e', 'c')
('b', 'd', 'a', 'c', 'e')
('b', 'c', 'e', 'd', 'a')
('b', 'c', 'e', 'a', 'd')
('b', 'c', 'd', 'e', 'a')
('b', 'c', 'd', 'a', 'e')
('b', 'c', 'a', 'e', 'd')
('b', 'c', 'a', 'd', 'e')
('a', 'e', 'd', 'c', 'b')
('a', 'e', 'd', 'b', 'c')
('a', 'e', 'c', 'd', 'b')
('a', 'e', 'c', 'b', 'd')
('a', 'e', 'b', 'd', 'c')
('a', 'e', 'b', 'c', 'd')
('a', 'd', 'e', 'c', 'b')
('a', 'd', 'e', 'b', 'c')
('a', 'd', 'c', 'e', 'b')
('a', 'd', 'c', 'b', 'e')
('a', 'd', 'b', 'e', 'c')
('a', 'd', 'b', 'c', 'e')
('a', 'c', 'e', 'd', 'b')
('a', 'c', 'e', 'b', 'd')
('a', 'c', 'd', 'e', 'b')
('a', 'c', 'd', 'b', 'e')
('a', 'c', 'b', 'e', 'd')
('a', 'c', 'b', 'd', 'e')
('a', 'b', 'e', 'd', 'c')
('a', 'b', 'e', 'c', 'd')
('a', 'b', 'd', 'e', 'c')
('a', 'b', 'd', 'c', 'e')
('a', 'b', 'c', 'e', 'd')
('a', 'b', 'c', 'd', 'e')
41
120
La probabilité de tomber sur la plus grande boite en choisissant
une boite après 1 augmentation(s) est de 0.3416666666666667
```

Suite : méthodologie de validation des résultats

```
import random
nbBoites = int(input("Quel est le nombre de boites ?\n"))
nbAugmentation = int(input("Combien d'augmentations ?\n"))
nbEssais = int(input("Combien d'essais voulez vous réaliser ?\n"))
score = 0
for loop in range(nbEssais):
    compteur = 0
    liste = [0]*nbBoites
    aug = 0
    i=0
    maxi = -1
    while compteur < nbBoites :
        r= random.randint(0,500)
        if r not in liste:
            liste[compteur]= r
            compteur = compteur + 1
    print(liste)

    while aug < nbAugmentation and i < len(liste):
        if liste[i] > liste[i-1]:
            aug = aug+1
        if liste[i] > liste[i-1] and aug == nbAugmentation:
            maxi = liste[i]

        i=i+1
        if maxi == max(liste):
            score=score+1
print(score)
```

Algorithme de simulation de la stratégie 1 :

Il permet de simuler notre stratégie en effectuant plusieurs essais. Le pourcentage de réussite trouvé permet de confronter le résultat avec la probabilité théorique.

Résultat d'un test du programme de simulation :

10000 essais du jeu d'ouverture simulé avec 4 boites (avec interruption des ouvertures de boîtes à la 1ère augmentation).

La stratégie est d'interrompre le processus d'ouverture des boîtes dès que la valeur du contenu a augmenté par rapport au contenu de la boîte précédemment ouverte.

Score de 4203.

La fréquence des succès est de : 4203/10000 = 0,4203

La probabilité fournie par le modèle vaut : 10/24 qui vaut environ 0,417 (qui est donc centre d'un intervalle de fluctuation d'échantillonnage des fréquences de demi-amplitude 0,01).

Donc l'expérience confirme le résultat théorique probabiliste.

```
[486, 492, 397, 464]
[2, 8, 251, 353]
[367, 225, 201, 457]
[53, 216, 354, 263]
[376, 448, 382, 251]
[222, 371, 167, 332]
[30, 413, 302, 318]
[471, 118, 151, 218]
[151, 211, 269, 190]
[295, 420, 158, 477]
[68, 182, 416, 378]
[333, 28, 46, 474]
[355, 180, 239, 456]
[331, 365, 288, 47]
[468, 53, 442, 399]
[396, 159, 421, 137]
[277, 371, 227, 52]
[300, 449, 137, 365]
[441, 194, 434, 142]
[238, 494, 23, 375]
[193, 13, 489, 140]
[120, 238, 251, 245]
[124, 323, 308, 81]
[155, 9, 104, 471]
[492, 307, 145, 416]
[197, 324, 313, 380]
[254, 326, 109, 217]
[10, 286, 26, 124]
[224, 484, 274, 299]
[384, 266, 335, 437]
[209, 148, 223, 219]
[22, 372, 151, 306]
[392, 176, 206, 142]
[220, 405, 218, 323]
[267, 463, 352, 159]
[451, 167, 104, 28]
[259, 257, 391, 318]
[125, 151, 112, 6]
4203
>>>
```

Perspectives

Nous avons réalisé un algorithme de dénombrement pour la stratégie 2 qui semble être meilleure lorsque n=4

```
from itertools import *

nbBoites = int(input("Combien de boites faut-il remplir ?\n"))
nbChangements = int(input("Au bout de combien de changements de maximum faut il sélectionner la boite ?\n"))
score = 0
lettre = 97

tableau = []
c=0
for loop in range(nbBoites):


    r = chr(lettre)
    tableau.append(r)

    lettre = lettre+1
```

```
for y in permutations(tableau):
    i=0
    r=0
    c=c+1
    changements = 0
    maximum = y[0]

    print(y)
    while i < len(tableau)-1 and changements < nbChangements:

        if y[i+1] > maximum:
            changements = changements + 1
            maximum = y[i+1]
            if changements == nbChangements and y[i+1]== tableau[-1]:
                score = score + 1



        i=i+1


print(score)
print(c)
print("La probabilité de tomber sur la plus grande boite en choisissant une boite après",nbChangements,"changement(s) est de",score/c)
```

Perspectives

Nous avons également commencé à observer les résultats donnés par l'algorithme de dénombrement pour n=5, n=6 et n=7.

Stratégie 1	Stratégie 2
Pour n = 5 alors p = 41/120 ≃ 0.34	Pour n = 5 alors p = 50/120 ≃ 0.42
Pour n = 6 alors p = 206/720 ≃ 0.29	Pour n = 6 alors p = 274/720 ≃ 0.38
Pour n=7 alors p = 1237/5040 ≃ 0.25	Pour n = 7 alors p = 1764/5040 = 0.35

Nous conjecturons alors que la stratégie 2 est plus efficace que la stratégie 1 pour tout n.

Nous aurions pu traiter cette stratégie mathématiquement avec le combinatoire et les coefficients binomiaux.

Conclusion

- Problématique :
  - Quelle est la meilleure façon de jouer, c'est-à-dire celle qui permet de s'arrêter sur le plus grand nombre de pâtisseries ?

- Réponse(s) à la problématique :
  - Méthode probabiliste
    - Stratégie de l'arrêt sur une première augmentation
    - Stratégie de l'arrêt sur un nouveau maximum
  - Méthode algorithmique
    - Algorithme de dénombrement (stratégie 1)
    - Algorithme de simulation (stratégie 1)
  - Généralisation
    - Approche théorique de la stratégie 1

Conclusion

Découverte d’aspects mathématiques que nous ne connaissions pas auparavant :

- Les factorielles
- Combinatoire et dénombrement

Exploitation d’un langage de programmation :

- Python

Exploitation d’aspects mathématiques vus en classe de seconde :

- Échantillonnage et fluctuations
- Probabilités

MERCI POUR VOTRE ÉCOUTE