



PRÉPASIMPLE

prepasimple.fr

ESSEC

Mathématiques 2

1992

Sujet de mathématiques approfondies

Mathématiques approfondies — Prépa ECG

Annales de concours · Sujet

Dans tout le problème, on désigne par n un nombre entier naturel non nul.

PARTIE 1

On se propose dans cette partie de calculer pour tout entier $p \geq 1$ la somme :

$$S_n(p) = 1^n + 2^n + \dots + p^n.$$

Soient $\mathbb{R}[X]$ l'espace vectoriel des fonctions polynômes à coefficients réels et (P_k) la suite de fonctions polynômes définie par $P_0(X) = 1$ et, si $k \geq 1$, par :

$$P_k(X) = \frac{X(X-1)\dots(X-k+1)}{k!} = \frac{1}{k!} \prod_{j=0}^{k-1} (X-j).$$

On considère l'application Δ de $\mathbb{R}[X]$ dans lui-même associant à la fonction polynôme P la fonction polynôme ΔP définie par :

$$(\Delta P)(X) = P(X+1) - P(X).$$

- a. Prouver que Δ est un endomorphisme de $\mathbb{R}[X]$, puis déterminer son noyau.
- b. Soient j, k des entiers naturels. On rappelle que $\Delta^j = \Delta \circ \Delta \circ \dots \circ \Delta$, avec j facteurs, et que Δ^0 désigne par convention l'application identité de $\mathbb{R}[X]$.

Déterminer ΔP_k , puis $\Delta^j P_k$ en distinguant les cas $j \leq k$ et $j > k$.

- c. Prouver que $(P_0, P_1, \dots, P_n)$ est une base du sous-espace vectoriel de $\mathbb{R}[X]$ constitué des fonctions polynômes de degré inférieur ou égal à n , et établir la formule suivante pour toute fonction polynôme P de degré inférieur ou égal à n :

$$P(X) = \sum_{k=0}^n (\Delta^k P)(0) P_k(X).$$

- d. En déduire que Δ est surjectif, puis, pour toute fonction polynôme P de degré n , établir l'existence d'une unique fonction polynôme Q , de degré $n + 1$, telle que :

$$Q(X + 1) - Q(X) = P(X) \quad \text{et} \quad Q(0) = 0.$$

On note Q_n la fonction polynôme telle que :

$$Q_n(X + 1) - Q_n(X) = X^n \quad \text{et} \quad Q_n(0) = 0.$$

- e. Exprimer $S_n(p)$ à l'aide de Q_n et de p , et expliciter Q_n et $S_n(p)$ pour $1 \leq n \leq 3$.

PARTIE 2

On établit dans cette partie quelques résultats probabilistes.

1. Fonction génératrice d'une variable aléatoire

On considère un entier naturel p et une variable aléatoire X à valeurs dans l'ensemble $\{0, \dots, p\}$. On lui associe la fonction suivante, dite fonction génératrice de X :

$$t \mapsto G_X(t) = \sum_{k=0}^p P(X = k) t^k.$$

Déterminer la fonction génératrice de X lorsque :

1. (a) X suit une loi de Bernoulli de paramètre λ , où $\lambda \in]0, 1[$.

1. (b) X suit une loi binomiale de paramètres n et λ , où $\lambda \in]0, 1[$.

2. Fonction génératrice d'une somme de variables aléatoires indépendantes

On considère désormais des entiers naturels $p_1, p_2, \dots, p_n$ et des variables aléatoires $X_1, X_2, \dots, X_n$ définies sur un même espace probabilisé, mutuellement indépendantes et à valeurs dans $\{0, 1, \dots, p_1\}, \{0, 1, \dots, p_2\}, \dots, \{0, 1, \dots, p_n\}$ respectivement. On note alors :

$$Y_n = X_1 + X_2 + \dots + X_n.$$

2. (a) Pour $n \geq 2$, établir que la fonction génératrice de $Y_2 = X_1 + X_2$ est le produit des fonctions génératrices de X_1 et de X_2 .

2. (b) Exprimer la fonction génératrice de $Y_n = X_1 + X_2 + \dots + X_n$ à l'aide des fonctions génératrices de $X_1, X_2, \dots, X_n$.

2. (c) Retrouver le résultat obtenu en 1. (a) en utilisant le résultat obtenu en 1. (b).

3. Étude d'un cas particulier

On suppose de plus que, pour tout entier k tel que $1 \leq k \leq n$, la variable aléatoire X_k suit une loi uniforme sur $\{0, 1, \dots, k-1\}$, autrement dit :

$$P(X_k = 0) = P(X_k = 1) = \dots = P(X_k = k-1) = \frac{1}{k}.$$

3. (a) Calculer l'espérance et la variance des variables aléatoires X_k , pour $1 \leq k \leq n$, puis de la somme $Y_n = X_1 + X_2 + \dots + X_n$.

On convient de noter la fonction génératrice G_n de Y_n sous la forme suivante :

$$G_n(t) = \frac{1}{n!} \sum_{k=0}^{n(n-1)/2} c_{n,k} t^k.$$

3. (b) En exprimant G_{n+1} en fonction de G_n , déterminer $c_{n+1,k}$ en fonction des coefficients $c_{n,k}, c_{n,k-1}, \dots, c_{n,k-n}$.

Par convention, on posera $c_{n,k} = 0$ lorsque $k < 0$ ou lorsque $k > \frac{n(n-1)}{2}$. Pour G_{n+1} , on considère la même construction avec $n+1$ variables mutuellement indépendantes.

3. (c) Montrer que les coefficients $c_{n,k}$ sont entiers naturels et calculer leur somme pour $0 \leq k \leq \frac{n(n-1)}{2}$.

PARTIE 3

On considère un tableau $s = (s[1], s[2], \dots, s[n])$ dont les valeurs sont entières et où les n entiers $1, 2, \dots, n$ sont affectés dans un ordre quelconque aux cases $s[1], s[2], \dots, s[n]$.

Dans les programmes Python ci-dessous, le tableau comporte $n+1$ cases : $s[0]$ est inutilisée et les valeurs à trier sont toujours $s[1], \dots, s[n]$. L'entier n est fourni aux fonctions. Les bornes supérieures de range sont exclues.

L'objet de cette partie est l'étude de la complexité de l'algorithme de tri suivant, visant à ranger dans l'ordre croissant les éléments du tableau s :

Python

```
def echanger(s, j):
    temp = s[j]
    s[j] = s[j + 1]
    s[j + 1] = temp

def tri_bulles(s, n):
    for k in range(1, n):
        for j in range(1, n - k + 1):
```

```
if s[j] > s[j + 1]:  
    echanger(s, j)
```

La fonction `echanger(s, j)` permute les valeurs des cases $s[j]$ et $s[j + 1]$. La fonction `tri_bulles` modifie directement le tableau fourni.

1. Exemples de mise en œuvre de l'algorithme

On suppose dans cette question que $n = 3$. Indiquer les valeurs contenues dans $s[1], s[2], s[3]$ après chaque appel de la fonction `echanger` lorsque l'on affecte initialement à $(s[1], s[2], s[3])$ les six valeurs suivantes :

$(1, 2, 3) ; (2, 3, 1) ; (3, 1, 2) ; (1, 3, 2) ; (3, 2, 1) ; (2, 1, 3).$

On précisera à chaque fois le nombre total d'appels de la fonction `echanger`.

On présentera les résultats obtenus dans six tableaux distincts.

2. Action de l'algorithme

2. (a) Déterminer la valeur contenue dans $s[n]$ à l'issue du passage dans la boucle suivante :

Python

```
for j in range(1, n):  
    if s[j] > s[j + 1]:  
        echanger(s, j)
```

2. (b) En déduire la valeur contenue en fin d'algorithme dans $s[k]$ pour $1 \leq k \leq n$.

3. Complexité de l'algorithme

3. (a) Exprimer en fonction de n le nombre de comparaisons d'entiers $>$ effectuées au cours de l'algorithme.
3. (b) Exprimer en fonction de n les nombres maximal et minimal d'appels de la fonction `echanger` au cours de l'algorithme. On explicitera des tableaux s pour lesquels ces maximum et minimum sont effectivement atteints.

Dans la suite, on associe au tableau s et à tout entier k tel que $1 \leq k \leq n$:

- le nombre $U_k(s)$ des entiers i tels que $1 \leq i < k$ et $s[i] > s[k]$, avec $U_1(s) = 0$;
- la somme $V_n(s) = U_1(s) + U_2(s) + \dots + U_n(s)$.

On dit que $V_n(s)$ est le nombre des inversions du tableau s , autrement dit le nombre des couples d'entiers (i, k) tels que :

$$1 \leq i < k \leq n \quad \text{et} \quad s[i] > s[k].$$

4. Nombre d'appels de la fonction `echanger`

4. (a) Lorsque $s[j] > s[j+1]$, indiquer l'effet de l'échange des valeurs de $s[j]$ et $s[j+1]$ sur le nombre des inversions du tableau $s = (s[1], s[2], \dots, s[n])$. En déduire le nombre d'appels de la fonction `echanger` lors du tri de s .
4. (b) On suppose dans cette question que $n = 3$. Préciser $(U_1(s), U_2(s), U_3(s))$ et $V_3(s)$ lorsque l'on affecte initialement à $(s[1], s[2], s[3])$ les six valeurs :

$$(1, 2, 3) ; (2, 3, 1) ; (3, 1, 2) ; (1, 3, 2) ; (3, 2, 1) ; (2, 1, 3).$$

4. (c) On suppose que n est donné et que s est un tableau d'entiers comportant $n+1$ cases, avec $s[0]$ inutilisée. Rédiger une fonction Python `inversions(s, n)` qui calcule les valeurs $U_k(s)$ pour $1 \leq k \leq n$ et $V_n(s)$ pour un tableau s donné.

La fonction renverra un tableau U de $n + 1$ cases, dont la case $U[0]$ est inutilisée et dont $U[k]$ contient $U_k(s)$, ainsi que l'entier V égal à $V_n(s)$. Pour créer un tableau entier rempli de zéros, on pourra utiliser `np.zeros(n + 1, dtype=int)` après `import numpy as np`; le paramètre `dtype=int` indique que les cases contiennent des entiers.

On suppose dans la suite que les différentes façons d'affecter les n entiers $1, 2, \dots, n$ aux n cases $s[1], s[2], \dots, s[n]$ sont équiprobables, et l'on note :

- $\mathcal{S}_n$ l'ensemble des différents tableaux s ainsi obtenus ;
- Ω_n l'ensemble des n -uplets $(u_1, u_2, \dots, u_n)$ d'entiers tels que $0 \leq u_k < k$ pour tout entier k tel que $1 \leq k \leq n$.

On considère $U_1, U_2, \dots, U_n$ comme des variables aléatoires sur l'ensemble $\mathcal{S}_n$.

5. Bijectivité de l'application $s \mapsto (U_1(s), U_2(s), \dots, U_n(s))$

5. (a) Vérifier que l'application associant à tout tableau s de $\mathcal{S}_n$ le n -uplet $(U_1(s), U_2(s), \dots, U_n(s))$ prend ses valeurs dans Ω_n .

5. (b) Soit s un tableau de $\mathcal{S}_n$ tel que $U_1(s) = u_1, U_2(s) = u_2, \dots, U_n(s) = u_n$.

Déterminer $s[n]$ en fonction de u_n .

Expliquer de même comment l'on peut obtenir les valeurs de $s[n-1], \dots, s[1]$ en fonction de $u_{n-1}, \dots, u_1$.

5. (c) Montrer que l'application $s \mapsto U(s) = (U_1(s), U_2(s), \dots, U_n(s))$ est une bijection de $\mathcal{S}_n$ sur Ω_n .

En déduire le nombre de tableaux s de $\mathcal{S}_n$ tels que $U_k(s) = u_k$ pour tout entier k tel que $1 \leq k \leq n$ et tout entier u_k tel que $0 \leq u_k < k$.

6. Lois des variables aléatoires $U_1, U_2, \dots, U_n$

6. (a) Déterminer la probabilité $P(U_1 = u_1, U_2 = u_2, \dots, U_n = u_n)$ pour tout n -uplet $(u_1, u_2, \dots, u_n)$ de Ω_n .
6. (b) Déterminer la probabilité $P(U_k = u_k)$ pour tout entier k tel que $1 \leq k \leq n$ et tout entier u_k tel que $0 \leq u_k < k$.

En déduire la loi de U_k et l'indépendance mutuelle des n variables aléatoires $U_1, \dots, U_n$.

6. (c) En déduire en fonction de n l'espérance et la variance du nombre d'appels de la fonction échanger au cours de l'algorithme, et donner des équivalents simples de ces expressions lorsque n tend vers l'infini.