Math ECG 2. 2023-2024

Mathématiques Appliquées - F. Gaunard
http://frederic.gaunard.com
ENC Bessières, Paris 17e.

Concours Blanc n°1 - Math 2

Solution

*Ce sujet reprend l'épreuve de **ESSEC II 2011**, à laquelle j'ai ajouté la première partie en Python (et donc supprimé les questions en Turbo-Pascal.*
On trouvera ci-dessous les solutions des questions de la première partie. En revanche, pour toute la suite, des solutions proposées par des collègues existent et on y renvoie.
Par exemple, celle de Pierre Veuillez, disponible ici ou encore celle-ci que j'avais proposée la dernière fois que j'ai posé ce sujet, et dont - hélas - j'ai perdu trace du nom de l'auteur.

Partie I : Simulation d'insertions avec Python

(1) **Évolution des positions successives de C_N au cours du temps**.

(a) La variable `position` renseigne sur la position actuelle de la carte C_N. Après chaque insertion, on ajoutera la position à la liste `L` qui stocke les positions successives occupées par C_N.
Si la carte est en position 1, elle passe dans une une position entre 1 et N de manière équiprobable. Un appel de la loi uniforme $\mathcal{U}(\llbracket 1, N \rrbracket)$ par la commande `rd.randint(1, N+1)` nous donne alors sa nouvelle position aléatoire.
Si la carte est à une autre position, elle ne sera déplacée (d'un cran vers le haut) que si la nouvelle position de la carte sur le dessus du paquet (choisie elle aussi uniformément entre 1 et N) est égale à celle de C_N ou en dessous.
Ceci donne le programme ci-dessous :

```
def simul_position(N, n):
    position = N
    L=[position]
    for j in range(1, n):
        if position == 1 :
            position = rd.randint(1, N+1)
        else :
            k=rd.randint(1, N+1)
            if k>= position :
                position = position -1
        L.append(position)
    return L
```

(b) On lit graphiquement que C_N passe à la position 51 à la 61−ième insertion, à la position 50 à la 98−ième insertion et se retrouve sur le dessus de la pile pour la première fois après la 242−ème insertion. On a donc

$$T_1 = 61, \qquad T_2 = 98, \qquad T = 242 + 1 = 243.$$

(2) **Simulation de T et conjectures**.

(a) On va atteindre la position 1 et ensuite ajouter 1. Pour savoir si la position de C_N baisse d'un cran car la carte est remontée vers le haut, on regarde si la carte de dessus de la pile est placée en dessous de C_N. Comme précédemment, on a le programme ci-dessous :

```
def simul_T(N):
    position = N
    t=0
    while position > 1 :
        if rd.randint(1, N+1) >=position :
            position = position -1
        t=t+1 # une insertion de plus à faire
    return t+1
```

(b) La fonction `mystere` renvoie la moyenne d'un échantillon de taille 1000 de T, soit une *estimation* ou valeur approchée de $E(T)$.

(c) Les commandes permettent l'affichage de l'évolution de l'estimation de $E(T)$ en fonction de N (avec N entre 32 et 100) et cette évolution semble être contrôlée par celles de $N\ln(N)$ en bas et $N\ln(N)+N$ en haut. Ces deux quantités étant équivalentes à $N\ln(N)$ lorsque $N \to +\infty$, on peut conjecturer (et on le démontre dans la Partie 3 par une comparaison série/intégrale on ne peut plus classique) que

$$E(T) \sim N\ln(N), \qquad N \to +\infty.$$

Remarques et commentaires

Dans la suite, on montre que $T = 1 + \sum_{i=1}^{N-1} \Delta_i$ avec le fait que $\Delta_i \hookrightarrow \mathcal{G}\left(\frac{i}{N}\right)$. On aurait alors aussi pu, une fois ces informations connues, simuler T de cette façon

```
def simul_T_alternatif(N):
    return np.sum([rd.geometric(i/N) for i in range(1, N)]) +1
```

On aurait aussi pu écrire une fonction permettant de calculer et de représenter graphiquement la fonction $x \mapsto P(T > x)$. Mais à un moment, il faut que le sujet s'arrête...

```
def depassement(x, N):
   L=[0]*1000
   for k in range(1000):
       if simul_T(N)>x:
           L[k]=1
   return np.mean(L)

N=52
abs=[k for k in range(52, 400)]
ord=[depassement(x, N) for x in abs]
Z=[N*np.exp(-k/N) for k in abs]
plt.plot(abs, ord)
plt.show()
```

Affichage Python