Tris récursifs ; et deux exercices de probabilités

Vendredi 26 et samedi 27 novembre 2021

Buts du TP

— Reprendre les tris récursifs.
— Faire un petit peu de « probabilités expérimentales ».

Exercice 1. *Créer (au bon endroit) un dossier associé à ce TP. Lancer Spyder/Pyzo/Idle, sauvegarder au bon endroit le fichier* `tdtrisbisouuntruccommeça.py` *; écrire une commande absurde, de type* `print( 6*7 )` *dans l'éditeur, sauvegarder et exécuter. En profiter pour importer ce qu'il faut :* Si vous êtes sous Spyder, changez (via F6) les options d'exécution, pour avoir à chaque exécution une nouvelle console et garder la main dessus. (il y a donc deux cases à vérifier/cocher). *Récupérer le fichier-cadeau sur le réseau.*

Exercice 2. *Vérifier que le premier exo a effectivement bien été fait : tout manquement donnera lieu de ma part à une agitation néfaste pour tout le monde...*

1 Tri « fusion »

On rappelle le principe du tri *fusion* ou *dichotomique* : si un tableau a une taille $\geqslant 2$, on le casse en deux, on trie récursivement les deux morceaux T_1 et T_2, puis on les fusionne via l'algorithme suivant, où on adjoint (via la méthode `append`) à un tableau initialement vide successivement les éléments de T_1 et T_2 :

Entrées : T_1, T_2
$Res \leftarrow []$ # le tableau qui sera rendu
$i_1, i_2 \leftarrow 0, 0$ # les indices qu'on regarde dans les tableaux T_1 et T_2
tant que $i_1 < |T_1|$ *ou* $i_2 < |T_2|$ **faire**
  si $i_2 = |T_2|$ *ou* ($i_1 < |T_1|$ *et* $T_1[i_1] < T_2[i_2]$) **alors**
    Adjoindre $T_1[i_1]$ à Res
    $i_1 \leftarrow i_1 + 1$
  sinon
    Adjoindre $T_2[i_2]$ à Res
    $i_2 \leftarrow i_2 + 1$
Résultat : Res

Exercice 3. *Écrire une fonction réalisant la fusion de deux tableaux supposés triés.*

C'est presque fini !

Exercice 4. *Écrire une fonction réalisant le tri-fusion d'un tableau. Cette fonction doit renvoyer un nouveau tableau (et non pas trier en place comme dans les deux premières parties).*

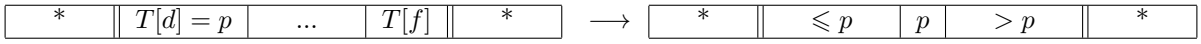
Exercice 5. *Tester et chronométrer le tri fusion.*

```
>>> foo = tableau_aleatoire(10)
>>> foo, tri_fusion(foo)
[74, 11, 20, 49, 49, 40, 70, 59, 18, 50], [11, 18, 20, 40, 49, 49, 50, 59, 70, 74]
>>> test_tri(tri_fusion, 10**3)
0.0021212100982666016
```

```
>>> test_tri(tri_fusion, 10**4)
0.022387584050496418
>>> test_tri(tri_fusion, 10**5)
0.26934083302815753
```

2 Tri rapide

Le principe du tri rapide consiste, pour trier une zone $[\![d, f]\!]$ d'un tableau, si $f > d$, à effectuer une première phase de préparation de la zone, à l'issue de laquelle les éléments de cette zone ont été permutés pour avoir le schéma :



En plus de la préparation de cette zone, l'algorithme doit retourner la nouvelle position du pivot (ce qui sera crucial pour lancer les deux appels récursifs ultérieurs pour trier la zone).

Un algorithme permettant de préparer la zone $[\![d, f]\!]$ consiste à manipuler deux indices i_1 et i_2 délimitant les fins respectives de zones d'éléments respectivement *majorés par* et *strictement minorés par* le pivot $p = T[d]$:

L'invariant préservé est :

$$(d < i \leqslant i_1 \implies T[i] \leqslant p) \quad \text{et} \quad (i_1 < i \leqslant i_2 \implies T[i] > p)$$

Entrées : T, d, f
$i_1, i_2 \leftarrow d, d$ # Si, vraiment !
tant que $i_2 < f$ **faire**
  si $T[i_2 + 1] \leqslant p$ **alors**
    $T[i_1 + 1] \leftrightarrow T[i_2 + 1]$
    $i_1 \leftarrow i_1 + 1$
  $i_2 \leftarrow i_2 + 1$
$T[d] \leftrightarrow T[i_1]$
Résultat : i_1

On note que cet algorithme est à « effets de bord » (il modifie le tableau qui a été donné) et retourne également un résultat (l'indice du pivot à la fin).

Exercice 6. *Écrire une fonction réalisant ce travail de préparation. La tester.*

L'élément principal du tri rapide est écrit ; terminons le travail ! La fonction `tri_rapide` se contentera d'appeler la fonction au cœur de l'algorithme : `tri_zone`, qui fait un appel à la fonction de préparation de zone avant de lancer deux appels récursifs.

Exercice 7. *Écrire une fonction réalisant en place le tri rapide d'un tableau ; tester et chronométrer.*

3 Du foot

Le petit Olivier et le petit Franz s'affrontent lors d'une compétition de penalty. À chaque essai, Olivier marque avec probabilité 5/6, et Franz avec probabilité 4/5. C'est Olivier qui tire en premier. Ensuite, les tirs sont alternés, et le premier qui marque a gagné la compétition.

Exercice 8. *Écrire une fonction simulant la séance de tirs au but.*
Le résultat renvoyé est 1 si Olivier gagne ; 0 si c'est Franz.
À l'aide de cette fonction, estimer la probabilité pour qu'Olivier remporte la séance.

On choisit maintenant la règle de la « mort subite » : « À chaque tour, les deux joueurs tirent. Si l'un marque et pas l'autre alors il a gagné ; sinon on continue. ».

Exercice 9. *Changer la fonction de simulation, et estimer la probabilité pour qu'Olivier (resp. Franz) gagne la séance de tirs au but.*

FIGURE 1 – Platoche, avec 50 kg de moins !

4 Une ronde

Voici un sujet d'oral des Mines PSI (2016) :
On s'intéresse à une ronde de p enfants ($p \geqslant 3$) avec une balle qui circule entre eux.
À l'instant $n = 0$, Alice a la balle. À chaque instant, celui qui a la balle passe à son voisin de droite avec une probabilité b, à son voisin de gauche avec une probabilité c et garde la balle avec une probabilité a.

Soit A_n l'événement : « Alice a la balle à l'instant n. »

Déterminer $\lim_{n\to+\infty} \mathbb{P}(A_n)$.

On peut visualiser la « chaîne de Markov » sous-jacente :

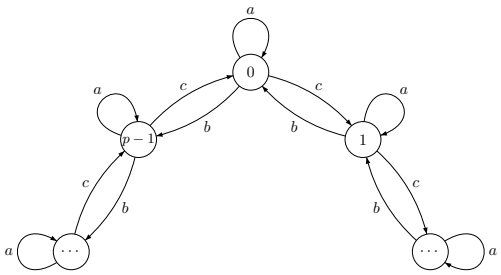
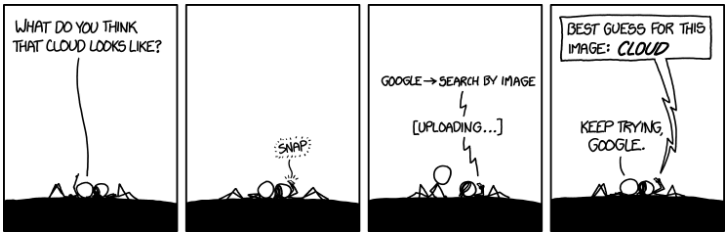


FIGURE 2 – Une chaîne de Markov

(il y a p états, et $a + b + c = 1$). On pourra fixer « en dur » : $(a, b, c) = (1/10, 7/10, 3/10)$ et $p = 6$.

Exercice 10. *Simuler cette chaîne de Markov : la fonction prendra en entrée $p, n \in \mathbb{N}^*$ et retournera l'état après n étapes (l'état initial est 0).*
Évaluer expérimentalement la probabilité pour que le résultat soit égal à 0 (« Alice a la balle après n étapes ») pour $n = 10$ puis pour $n = 100$ (on pourra évaluer la fréquence de réalisation de cet événement sur 10^3, 10^4 puis 10^5 expériences).

Pour finir, vous pouvez exécuter... puis comprendre le script donné à la fin du fichier-cadeau.



XKCD 1444 – *Cloud computing has a ways to go.*