Psi 999 – 2021/2022
`http://www.psi945.fr`

ψ

TP Python

Algorithmes de tri... et un exercice de probabilités

Vendredi 15 et samedi 16 octobre 2021

Buts du TP

— Faire un premier exercice de probabilités pythonisées.
— Coder effectivement les algorithmes de tri quadratique.
— Tester expérimentalement la complexité.

Dans tout ce TP, les tableaux sont implicitement constitués d'entiers ou de flottants.

Exercice 1. *Créer (au bon endroit) un dossier associé à ce TP. Y placer une copie du fichier récupéré dans le dossier partagé de travail de la classe :* `squelette_tris.py`.
Lancer Spyder/Pyzo/Idle, ouvrir le fichier précédent, le sauvegarder sous un autre nom dans le bon dossier. Observer et comprendre le contenu.
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).

Le fichier généreusement offert contient un peu de maths (probas) et de quoi vous aider pour faire les tests des différentes fonctions. En basculant progressivement le contenu vers la zone non commentée, vous pourrez avancer de façon efficace.

Je rappelle qu'il est IMPORTANT de copier/coller les résultats de la fenêtre d'exécution VERS la fenêtre d'édition. Vous n'avez pas compris la phrase précédente ? Alors il est urgent de commencer à apprendre à travailler : je vous suggère de demander des précisions MAINTENANT, et pas vers le 15 juin. À bon entendeur...

1 Pénaux (pas plus de 40 minutes)

Exercice 2. *Le petit Olivier et le petit Franz s'affrontent lors d'une compétition de penaltys. 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.*

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

On change les règles ! On choisit celle 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 3. *Changer la fonction de simulation, et estimer la probabilité pour qu'Olivier (resp. Franz) gagne la séance de tirs au but.*

Une dernière règle ?

Exercice 4. *Enfin, on prend pour règle : celui qui marque le plus de penaltys sur 5 tirs a gagné (et s'ils ont réussi le même nombre de tirs, il y a match nul).*
Estimer expérimentalement les probabilités de victoires des deux joueurs, et la probabilité d'un match nul.

FIGURE 1 – Platoche, avec 50 kg de moins !

2 Tris « sélection » (pas plus de 40 minutes)

On commence par le « tri-bulles ». Il s'agit d'un tri « en place » : le programme reçoit un tableau sur lequel il agit, mais ne retourne rien.

Entrées : T
pour i *de* $|T|-1$ *à* 1 **faire**
  pour j *de* 0 *à* $i-1$ **faire**
    si $T[j] > T[j+1]$ **alors**
      $T[j] \leftrightarrow T[j+1]$

Exercice 5. Tri-bulles
Programmer effectivement le tri-bulles ; le tester et le chronométrer. Noter les résultats puis commenter éventuellement la partie du script qui lance les tests ou contient les résultats chronométrés.

Avant de coder le tri par sélection classique, quelques fonctions préliminaires (seule la dernière sera effectivement utilisée dans la suite). Il s'agit essentiellement de s'échauffer le cerveau, et de se poser les bonnes questions :

— Pour tel type de problème, de quelles variables vais-je avoir besoin ?
— Que vont-elles représenter ?
— Comment vais-je les initialiser ?
— Comment vont-elles évoluer en cours de route ?

Exercice 6. *Écrire des fonctions calculant et renvoyant respectivement :*
— *la valeur maximale d'un tableau non vide (cette fonction ne sera* pas *utilisée dans la suite) ;*
— *la (une) position du maximum ;*
— *la (une) position du maximum parmi les k premières cases d'un tableau (la fonction reçoit le tableau et k).*
Tester les programmes (sur des valeurs mettant potentiellement votre programme en difficulté, sinon ça ne sert à rien !)

Voici enfin l'algorithme du tri par sélection, qui est à nouveau un tri en place :

Entrées : T
pour j *de* $|T|$ *à* 2 **faire**
  Trouver la position p du maximum des j premiers éléments du tableau
  $T[p] \leftrightarrow T[j-1]$

Exercice 7. *Coder effectivement le tri par sélection. Le tester et le chronométrer.*

3 Tris « insertion »

Dans les tris par insertion, on place chaque nouvel arrivant dans le tableau à sa position, dans une zone déjà triée : par exemple dans l'exemple suivant, on doit insérer 6 dans un sous-tableau déjà trié :

0	5	**9**	**10**	**13**	**6**	*	*	$\longrightarrow$	0	5	**6**	**9**	**10**	**13**	*	*

Pour réaliser l'opération précédente (décalage de $T[5]$ vers la position 2) on peut écrire

```
for j in range(5, 2, -1):
    T[j-1], T[j] = T[j], T[j-1]
```

Mais on peut aussi brutaliser, à l'aide du slicing offert par Python :

```
T[2: 6] = [ T[5] ] + T[2: 5]
```

Pour insérer $T[i]$ dans $T[:i]$, il faut commencer par *trouver la position* de ce nouvel élément $T[i]$; par exemple comme ceci :

Entrées : T, i
$pos \leftarrow i$ # la position d'affectation ; au départ : à droite !
tant que $pos > 0$ *et* $T[pos-1] > T[i]$ **faire**
  $pos \leftarrow pos - 1$
Résultat : pos

Exercice 8. *Écrire une fonction prenant en entrée un tableau T et un indice $i \in [\![1, |T|-1]\!]$ et retournant la position où on doit insérer $T[i]$ dans $T[:i]$ supposé déjà trié.*

On peut maintenant écrire le tri par insertion :

Entrées : T
pour i *de* 1 *à* $|T|-1$ **faire**
  $pos \leftarrow position(T, i)$
  décaler $T[i]$ en position pos

Exercice 9. *Programmer effectivement le tri par insertion. Le tester et le chronométrer.*

On rappelle la variante suivante, qui consiste à trouver la position d'insertion par une recherche dichotomique de la position :

Entrées : T, i
si $T[i] < T[0]$ **alors**
  Résultat : 0
si $T[i] > T[i-1]$ **alors**
  Résultat : i
$g, d \leftarrow 0, i-1$ # les bords gauche et droit de l'intervalle de recherche
tant que $g < d-1$ *# On s'assure : $d \geqslant g+2$, et $T[g] \leqslant T[i] \leqslant T[d]$* **faire**
  $m \leftarrow (g+d)//2$ # l'indice du milieu ; $g < m < d$
  si $T[m] \leqslant T[i]$ **alors**
    $g \leftarrow m$ # continuer à droite
  sinon
    $d \leftarrow m$ # continuer à gauche
Ici, $d = g+1$, et $t[g] \leqslant t[i] \leqslant t[d]$
Résultat : d

Exercice 10. *Programmer cette fonction d'insertion ; tester et chronométrer le nouveau programme de tri par insertion ainsi obtenu.*