TNSI Devoir Maison – Piles et files

à rendre pour le lundi 13 novembre 2023

On attend un travail <u>personnel</u> sur l'ensemble du DM et des programmes commentés et lisibles.

Calculatrice HP48SX
On note la présence d'une pile dont les 4 derniers niveaux sont visibles ici

Exercice 1 : Notation polonaise inverse

La notation polonaise inverse (notation postfixe) est une manière de noter les calculs sans utiliser de parenthèses. Cette notation a été utilisée par certaines calculatrices, notamment de Hewlett-Packard.
Exemple : calcul de `7 8 * 2 +`

On lit les deux premiers nombres et l'opérateur, on calcule 7×8 = 56
On garde le 56 en tête, on lit le nombre et l'opérateur suivant : 56 + 2 = 58 qui est le résultat du calcul

Pour cet exercice, on n'utilisera que des nombres positifs et pas de division (pour éviter la division par 0). L'évaluation d'une expression est simple et utilise une pile :

⇒ Initialement la pile est vide
⇒ Si on trouve un opérateur, on dépile deux fois pour trouver les deux opérandes (attention à l'ordre pour la soustraction, non symétrique). On effectue l'opération, et on empile le résultat
⇒ Sinon c'est un nombre et on l'empile
⇒ Le résultat de l'opération se lit en sommet de pile.

1) Sur papier, donner le résultat de `1 2 3 4 + x 5 x - 7 +`. Ecrire ce calcul sous forme infixe (c'est-à-dire de la manière usuelle avec les parenthèses).
2) Ecrire une fonction Python `calculRPN(chaine)` qui, étant donnée une chaîne de caractères `chaine` exprimant un calcul sous forme postfixe (polonaise inverse), donne le résultat du calcul. On supposera que l'expression est bien formée, et ne contient pas de division.

`calculRPN("7 8 * 2 +")` doit donner `58.0`.
`calculRPN("11 8.5 - 3 * 7 +")` doit donner `14.5`.

<u>Rappel</u> : la méthode `split` de la classe `str` permet d'obtenir une liste comportant les différents "mots" d'une chaîne de caractères.
Utilisée sans arguments, le séparateur est l'espace : `'un deux trois'.split()` renvoie `['un', 'deux', 'trois']`.

Exercice 2 : Pas de bol

Vous avez sans doute déjà remarqué qu'il arrivait qu'une personne arrivée aux caisses après vous et ayant choisi une autre file que la votre passe avant vous en caisse. Est-ce le signe que vous n'avez vraiment pas de chance ou est-ce que la probabilité pour que cet évènement arrive est assez importante ?
Dans cet exercice, nous nous proposons de déterminer à l'aide d'une simulation la probabilité qu'un tel évènement arrive (qu'un client arrivé après vous et se plaçant dans une autre file passe en caisse avant vous).
Vous avez déjà suffisamment d'explication pour faire l'exercice (c'est à vous de prendre des initiatives et fixer certains paramètres éventuellement), mais si vous ne voyez pas comment faire, vous pouvez faire les questions guidées qui suivent.

Les files d'attentes seront représentées par des files (au sens informatique du terme) et stockées dans un tableau `files` de `N` files où `N` est un entier que nous pourrons ajuster par la suite. Toutes les files ont la même longueur, sauf la première sur laquelle on se trouve qui est plus courte d'un élément au moment où on arrive (raison pour laquelle on a choisi cette file).
On va représenter les clients remplissant les files par des nombres. Tous les clients arrivés avant nous ont le nombre « `0` », notre personnage a le nombre « `1` » et ceux qui arrivent après lui les nombres `2`, puis `3`, `4`, …
Le temps d'attente (en nombre de tours de simulation) sera tiré aléatoirement entre `attente_min` et `attente_max` pour chaque caisse à chaque fois qu'un nouveau client se présente et stocké dans un tableau `attentes`.

1) Ecrire une fonction `initialisation(N : int, longueur :int) -> list(File)` qui :
   - crée un tableau `files` de `N` files d'attentes ;
   - remplit les files d'attentes de `longueur` éléments valant tous `0` ;
   - place le nombre 1 à la fin de la première file, puis le nombre 2 à la fin de la deuxième file, le nombre 3 à la fin de la troisième file …
   - supprime le premier élément de la première file d'attente (c'est celle qu'on a choisit car elle était plus courte) ;
   - renvoie le tableau `Files`.

Numéro de file	Contenu de la file
0	`0,0,0,1`
1	`0,0,0,0,2`
2	`0,0,0,0,3`
3	`0,0,0,0,4`
4	`0,0,0,0,5`
5	`0,0,0,0,6`

Exemple avec N = 6 et longueur = 4

2) Ecrire une fonction `simulation(files:list(File), attente_min:int, attente_max:int) -> int` qui :
   - initialise une variable entière `nb_clients_plus_rapides` à 0 ;
   - initialise une variable entière `N` à la longueur du tableau `files` ;
   - crée un tableau `attentes` de `N` entiers, chacun tiré aléatoirement entre `attente_min` et `attente_max` ;
   - fait une boucle de simulation tant que le client numéro 1 n'est pas sorti de sa file :
     - on passe en revue chaque valeur du tableau attente :
       - la valeur est décrémentée de 1 ;
       - si la valeur vaut 0 :
         - la file correspondante est défilée et on note le numéro du client qui vient de sortir :
           - si le client est le 1, on sort de la boucle principale (après avoir fini l'analyse de toutes les files) ;
           - si le numéro est 2 ou plus, on incrémente la variable `nb_clients_plus_rapides` et on enfile un client de même numéro dans la même file ;
           - si le nombre est nul, on ne fait rien.
         - on lui remet une valeur entre `attente_min` et `attente_max` ;
   - renvoie `nb_clients_plus_rapides.`

3) Ecrire maintenant le programme principal qui définit les variables utiles (`N`, `longueur`, `attente_min`, `attente_max`) puis effectue un nombre réglable mais important (1000 par exemple) de simulations et en déduit (et affiche) la probabilité qu'une personne arrivée après nous passe en caisse avant ainsi que le nombre moyen de personnes étant passées avant nous (alors qu'elles étaient arrivées après).
4) Faire différentes simulations en changeant un peu les paramètres et conclure : est-il normal qu'assez souvent des personnes arrivées plus tard passent avant nous en caisse ?