TP08 Petits jeux électoraux

Pensez à utiliser GithubDesktop pour faire un « Pull » et récupérer le dossier du TP qui aura été déposé sur Gitlab par le Général Kléber.

Partie I

Algorithme d'Euclide

L'objectif est de renvoyer le PGCD de deux entiers naturels a et b. Rappelons que le PGCD de a et b noté $a \wedge b$ est le plus grand diviseur commun aux entiers a et b. Cet algorithme est basé sur lemme suivant dû à Euclide :

Lemme (d'Euclide)**.** *Soient $a \in \mathbb{N}$ et $b \in \mathbb{N}^*$. La division euclidienne de a par b s'écrit*

$$\begin{cases} a = b\,q + r \\ r \in [\![0, b-1]\!]. \end{cases}$$

Les diviseurs communs aux entiers a et b sont les mêmes que les diviseurs communs aux entiers b et r. Autrement dit, on a $a \wedge b = b \wedge r$.

Euclide procède alors de la manière suivante :

- On pose $r_0 = a$.
- On pose $r_1 = b$.
- Pour tout $n \geq 1$, si $r_n \neq 0$ on note r_{n+1} le reste de la division euclidienne de r_{n-1} par r_n.

Cette suite ainsi définie est à valeurs dans $\mathbb{N}$. Comme $r_{k+1} < r_k$ tant que $r_k \neq 0$ (étant donné que l'on montre (par l'absurde) qu'il n'existe pas de suite strictement décroissante à valeurs dans $\mathbb{N}$) il existe N tel que $r_N \neq 0$ et $r_{N+1} = 0$. D'après le lemme d'Euclide, on a

$$a \wedge b = r_0 \wedge r_1 = r_1 \wedge r_2 = \cdots = r_N \wedge r_{N+1}.$$

Comme $r_{N+1} = 0$ on a donc $a \wedge b = r_N \wedge r_{N+1} = r_N$. Rappelons à toute fins utiles qu'en particulier $0 \wedge 0 = 0$. Programmer une procédure `pgcd(a,b)` s'appliquant à deux entiers naturels et renvoyant l'entier naturel `a` $\wedge$ `b`.

Stop Gitlab

Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie II

Crible d'Ératostène

L'algorithme du crible d'Ératostène procède par élimination. Dans un premier temps on construit la liste `L` des entiers compris entre 2 et n puis on effectue les opérations suivantes :

1. On considère le premier élément de la liste, à savoir 2 et on supprime de la liste tous les multiples de 2 hormis 2 évidemment.
2. On considère le successeur de 2 dans `L`, c'est-à-dire 3. On constate qu'il est premier puisqu'il n'a pas de diviseur strict autre que 1. On supprime alors de `L` tous les multiples de 3 hormis 3.

3. On considère le successeur de 3 dans `L`, c'est-à-dire 5. C'est un nombre premier, puisqu'il n'a pas été supprimé lors des deux premières étapes. On supprime de `L` tous les multiples de 5 hormis 5.

On continue le procédé. On s'arrête lorsque l'on atteint un élément p de `L` tel que $p^2 > n$. Les nombres restant dans la liste `L` sont tous premiers. Implémenter une procédure `eratosthene(n)` qui renvoie la liste des nombres premiers inférieurs ou égaux à n en respectant l'algorithme ci-dessus. On utilisera une liste annexe pour stocker l'état (premier ou non) d'un nombre donné sous forme d'un booléen et éviter ainsi de barrer les multiples d'un nombre qui n'est déjà plus premier. Pour visualiser concrètement ce qu'il se passe, on pourra consulter la page suivante :

`http://commons.wikimedia.org/wiki/File:New_Animation_Sieve_of_Eratosthenes.gif`

Stop Gitlab

Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie III

Wall-Street

Ce problème s'inspire d'une épreuve d'informatique de l'École Polytechnique. Vous ne devez pas utiliser les fonctions `min` et `max` appliquées à des listes.

On cherche à calculer le gain maximum possible (à la bourse) sur une action pendant une période de n jours. Vous ne faites qu'une opération d'achat et de vente d'une seule action pendant la période étudiée. On suppose que les cours quotidiens de cette action sont enregistrés dans un tableau `cot` d'entiers naturels contenant n éléments ($0 \leq i \leq n-1$).

On définit l'*amplitude* de la variation du cours comme le maximum de la valeur absolue de la variation de ce cours pendant la période observée, c'est-à-dire la quantité suivante :

$$amplitude = \max_{0 \leq i \leq j \leq n-1} |\texttt{cot}_j - \texttt{cot}_i| = \max_{0 \leq i \leq n-1} \texttt{cot}_i - \min_{0 \leq i \leq n-1} \texttt{cot}_i.$$

1. Écrire une fonction `amplitude(cot)` qui renvoie comme résultat l'amplitude de la variation du cours représenté par la liste `cot`.

Stop Gitlab

Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Le *gain maximum* est le gain maximum possible sur la période observée, c'est-à-dire la quantité suivante :

$$gain = \max_{0 \leq i \leq j \leq n-1} (\texttt{cot}_j - \texttt{cot}_i).$$

2. Donner un exemple où l'amplitude est différente du gain maximum. Que représente l'amplitude en terme de gain ou de perte ?

3. En suivant textuellement la définition du gain, écrire une fonction `gain(cot)` qui renvoie, en temps quadratique par rapport à n, le gain maximal possible sur le cours représenté par la liste `cot`.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

4. Construire une procédure `gain_date(cot)` en modifiant la procédure précédente afin de renvoyer la liste des couples formés des deux dates `da` et `dv` d'achat et de vente de l'action permettant d'obtenir le gain maximum sur la liste `cot` avec `dv-da` minimum.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Pour tout $i \in [\![0, n-1]\!]$, définissons le gain courant maximum comme le gain maximum possible obtenu en vendant son action au temps i.

5. En calculant progressivement le gain courant maximum, écrire une procédure que l'on nommera `gain1(cot)` qui renvoie, en temps linéaire par rapport à n, le gain maximum possible sur le cours représenté par la liste `cot`.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

6. Construire une procédure `gain1_date(cot)` en modifiant la procédure précédente afin de renvoyer la liste des couples formés des deux dates `da` et `dv` d'achat et de vente de l'action permettant d'obtenir le gain maximum sur la liste `cot` avec `dv-da` minimum.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie IV

Dépouillement d'élections

Le problème suivant — inspiré d'un sujet d'écrit de l'École polytechnique — consiste à faire le décompte du résultat des élections dans le pays des bisounours. La règle électorale y est très laxiste, puisque l'on peut être élu sans s'être présenté. Plus prosaïquement, chaque votant (portant un numéro d'életeur $i \in [\![0, n-1]\!]$) vote pour `vote[i]`, où `vote[i]` est un entier positif caractéristique de la personne pour qui on vote (les habitants du pays des bisounours sont identifiés par un entier réprésentant leur numéro de sécurité sociale qui peut être très grand et est notamment différent de leur numéro d'électeur). On dispose de la liste `vote` qui contient les votes des n habitants du pays des bisounours.

1. Dans le pays des bisounours, le seul candidat ayant obtenu strictement plus de 50% des voix exprimées est élu au premier tour des élections.
   (a) Écrire une fonction `winner(vote,k)` qui renvoie `True` si le candidat `k` possède la majorité absolu et `False` sinon.
   (b) Écrire la fonction `gagnant_tour1(vote)` qui renvoie le gagnant du premier tour s'il existe. S'il n'existe pas, on renvera `None`. On pourra utiliser la fonction de la question précédente. On se contentera d'une fonction faisant le calcul en temps quadratique par rapport à n.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

2. Supposons le tableau `vote` trié dans l'ordre croissant, vérifiant
$$\texttt{vote}[0] \leq \texttt{vote}[1] \leq \cdots \leq \texttt{vote}[n-1].$$
Écrire une fonction `gagnant_tour1_bis(vote)` qui renvoie en temps linéaire par rapport à n le gagnant du premier tour s'il existe et `None` dans le cas contraire.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

3. Au second tour des élections, le candidat ayant obtenu le plus de voix est élu. S'il y a égalité de voix, tous les candidats ayant obtenu le plus de voix sont élus (ex æquo).
   (a) Écrire la fonction `gagnant_tour2(vote)` qui renvoie dans un tableau les gagnants du second tour. On prendra garde à ne faire apparaître qu'une seule fois le nom des gagnants dans la liste finale (par exemple en utilisant un ensemble python (`set`)). On se contentera d'une fonction faisant le calcul en temps quadratique par rapport à n.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

   (b) Supposons le tableau `vote` trié en ordre croissant, vérifiant
$$\texttt{vote}[0] \leq \texttt{vote}[1] \leq \cdots \leq \texttt{vote}[n-1].$$
Écrire une fonction `gagnant_tour2_bis(vote)` qui renvoie en temps linéaire par rapport à n le gagnant du second tour.

Stop Gitlab
Allez sur GithubDesktop pour faire un commit. Choisissez (avec pertinence) le résumé. Pensez, si possible, à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.