Mathématiques PT. 2025 - 2026

Mathématiques & Informatique - F. Gaunard
http://frederic.gaunard.com
Lycée Voltaire, Paris 11e.

Préparation à l'oral

Algorithmique

Ces exercices sont **à préparer** de sorte à être présentés **clairement** lors des séances en classe entière, selon le planning en ligne. On en profitera pour faire, en classe, les rappels de cours (avec les énoncés complets et précis) correspondants.
On se doute bien qu'il n'est pas possible d'être exhaustif en quatre ou cinq exercices...

Exercice 1

Référence : R-INFO-1

Origine : Math II 2025

Thèmes : Manipulation de listes

1. Écrire une fonction `voeux` qui prend en argument une liste `voyageurs` (telle que `voyageurs[i]` donne le train j dans lequel il aimerait voyager), et un nombre de trains `rt`.
   La fonction doit renvoyer une liste `train` telle que `train[i]` donne la liste des voyageurs qui veulent voyager dans ce train, ordonnés de façon croissante.

2. Écrire une fonction `possible1` d'argument une liste `voyageurs` et une liste `capacite` telle qu'elle renvoie un booléen indiquant s'il est possible ou non de remplir les trains.

Désormais, si le `train[i]` ne peut pas accueillir tous les passagers, les passagers en surplus se dirigent vers le train suivant.

3. Écrire une fonction `possible2` d'argument une liste `voyageurs` et une liste `capacite` telle que `possible2` renvoie un booléen selon la méthode décrite ci-dessus.

4. Écrire une fonction `affecte` d'argument les listes `voyageurs` et `capacite` qui renvoie une liste `affectation` telle que `affectation[i]` donne la composition du train i.

Exercice 2

Référence : R-INFO-2

Origine : Math II 2018

Thèmes : Méthode d'Euler, représentations graphiques

On cherche à étudier numériquement les solutions de l'équation différentielle avec conditions initiales suivante :

$$(\mathcal{P}) \qquad y'(t) = t^2 - y(t)^3 \quad \text{avec} \quad y(-1.5) = a.$$

1. Pour $a = 2$ et un pas de discrétisation $h = \frac{1}{4}$, puis $h = \frac{1}{8}$, représenter les solutions approchées du problème $(\mathcal{P})$ obtenues par la méthode d'Euler sur l'intervalle $[-1.5, 2.5]$.

2. En utilisant la fonction `odeint` du module `scipy.integrate` de `Python`, résoudre numériquement le problème $(\mathcal{P})$ pour $a = 2$. On prendra garde à bien définir soigneusement les arguments de cette fonction en lisant attentivement l'aide en ligne.

3. Sur la figure existante, rajouter la courbe de la solution numérique obtenue à la question précédente.

4. Rajouter ensuite les courbes des solutions numériques pour $a = 1.3$ et $a = 0.1$. Qu'observe-t-on ?

5. Pour résoudre $y'(t) = f(t, y(t))$ avec $y(t_0) = a$, on utilise maintenant la suite (y_n) définie par $y_0 = a$ et

$$y_{k+1} = y_k + \frac{h}{2}\left(f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k))\right).$$

Représenter, pour $a = 2$ et les mêmes pas de discrétisation qu'à la question 1, les solutions approchées obtenues par cette méthode. Expliquer pourquoi le résultat semble meilleur.

Exercice 3

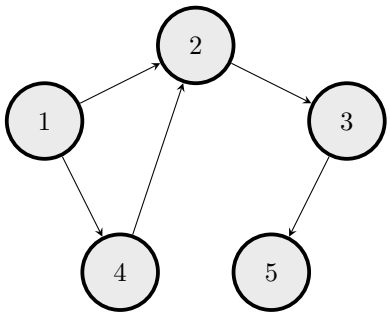
Référence : R-INFO-3

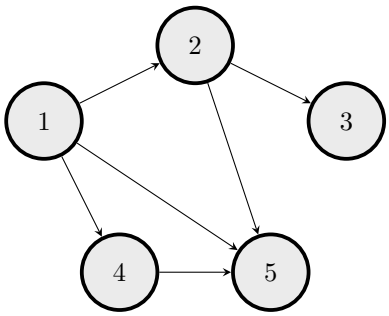
Origine : Math II 2025

Thèmes : Graphes

Un graphe est dit *correctement orienté* si chaque arc de i vers j on a i inférieur à j.

1. Les deux graphes ci-dessous sont ils bien orientés ? Donner leurs matrices d'adjacence.





2. Écrire une fonction `GO` ; prenant en argument un graphe G représenté par sa matrice d'adjacence `M`, qui détermine si un graphe est bien orienté. *La fonction renvoie donc un booléen.*

3. Peut-on toujours correctement orienter un graphe non orienté ? Écrire un code qui prend un graphe non orienté et l'oriente correctement.

4. Écrire une fonction qui détermine si il y a un cycle de longueur 3. *La fonction prendra en argument la matrice d'adjacence d'un graphe non orienté.*
Même question avec un cycle de longueur maximale.

Exercice 4

Référence : R-INFO-4

Origine : Math II 2018

Thèmes : Déplacements aléatoires

Soit Ω le sous-ensemble $\{(1,0),(-1,0),(0,1),(0,-1)\}$ de l'espace vectoriel $\mathbb{R}^2$ et (V_n) une suite de variables aléatoires indépendantes suivant la même loi uniforme sur Ω_V.
On définit une suite de variables aléatoires (X_n) en posant :

$$X_0 = (0,0), \quad \text{et} \quad \forall n \in \mathbb{N}, \quad X_{n+1} = X_n + V_n.$$

On appelle *trajectoire de longueur n* la ligne brisée aléatoire reliant les points de coordonnées $X_0, X_1, \ldots, X_n$.

1. Écrire une fonction `tirage` sans argument et renvoyant une des listes suivantes avec équiprobabilité :

`[1,0]`, `[-1,0]`, `[0,1]` et `[0,-1]`.

On pourra utiliser les fonctions du module `numpy.random` de `Python`.

2. En déduire une fonction `trajectoire` d'argument n renvoyant le tirage d'une trajectoire de longueur n, sous forme d'une liste de couples `[x,y]`.

3. Tracer quelques trajectoires de longueurs 10, 100, 1000, puis 10000.

4. Écrire une fonction `premierRetour` d'argument m renvoyant le plus petit entier non nul $n \leq m$ tel que $X_n = (0,0)$ s'il existe, et -1 sinon.

5. On note T le premier retour en $(0,0)$. Au moyen d'un programme, conjecturer si la variable aléatoire T est bien définie.

Exercice 5

Référence : R-INFO-5

Origine : Math II 2021

Thèmes : Tri

1. Écrire une fonction `parcours` d'argument une liste $L = [a_0, a_1, \ldots, a_{n-1}]$ qui modifie cette liste en permutant a_k et a_{k+1} si $a_{k+1} < a_k$, pour k variant de 0 à $n-2$, et qui renvoie le nombre de permutations effectuées.
Par exemple, si `L = [5, 4, 1, 6]`, alors `parcours(L)` renvoie le nombre 2 et la liste `L` devient `[4, 1, 5, 6]`.

2. Que peut-on dire du dernier élément d'une liste à laquelle on a déjà appliqué une fois la fonction `parcours` ?

3. On veut utiliser cette fonction pour trier par ordre croissant une liste selon le principe suivant : pour une liste L donnée en entrée on répète l'application de la fonction parcours jusqu'à ce que `L` devienne invariante par cette fonction.
Écrire une fonction `tri` qui ordonne une liste donnée en entrée selon ce principe et qui renvoie le nombre d'itérations effectuées.

4. Expliquer pourquoi la fonction `tri` donne bien le résultat voulu en un temps fini.
Évaluer son coût de calcul dans le meilleur des cas et dans le pire des cas.

5. Écrire une fonction `tri1` comme une amélioration de la fonction `tri` en évitant les comparaisons inutiles.