Concours National Commun (CNC), Maroc

Session 2021, filière TSI

Informatique

Durée : 2 heures

Ce que traite le sujet

- Exercice : développement limité de l'exponentielle
- Partie I : calcul numérique
- Partie II : bases de données et SQL
- Partie III : problème Python

Sujet reproduit sans retouche à partir de la page suivante.

Résumé des parties, chapitres de cours liés et autres années :
prepahelp.com/annales/cnc/tsi/2021/info

Prepa Help ne revendique aucun droit sur le sujet, propriété de l'organisateur du concours.

ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵍⵎⵖⵔⵉⴱ
ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵣⵔⴼⵜ ⴰⵏⴰⵎⵓⵔ ⵏ ⵜⵙⴷⴰⵡⵉⵜ ⴷ ⵜⵙⵙⵏⵜⵉⵔⵉⵜ ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵣⵔⴼⵜ ⵏ ⵜⵙⵙⵏⵜⵉⵔⵉⵜ ⴷ ⵜⵙⵙⵏⵜⵉⵔⵉⵜ

المملكة المغربية
وزارة التربية الوطنية والتكوين المهني
والتعليم العالي والبحث العلمي

Royaume du Maroc

Ministère de l'Éducation Nationale, de la Formation Professionnelle, de l'Enseignement Supérieur et de la Recherche Scientifique

Département de l'Enseignement Supérieur et de la Recherche Scientifique

وزارة التجهيز و النقل و اللوجستيك و الماء
Ministère de l'Equipement, du Transport, de la Logistique et de l'Eau

المدرسة الحسنية للأشغال العمومية
Ecole Hassania des Travaux Publics

CNC 2021

Concours National Commun

d'Admission dans les Établissements de Formation d'Ingénieurs et Établissements Assimilés

Épreuve d'**Informatique**

Filière : **TSI**

Durée **2 heures**

Cette épreuve comporte **10 pages au format A4**, en plus de la page de garde

Épreuve d'Informatique – Session 2021 – Filière TSI

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- ✓ Cette épreuve est composée d'un exercice et de trois parties tous indépendants ;
- ✓ Toutes les instructions et les fonctions demandées seront écrites en Python ;
- ✓ Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- ✓ Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈

Important : Le candidat doit impérativement traiter l'exercice ci-dessous, et inclure les réponses dans les premières pages du cahier de réponse.

Exercice : (4 points)

Développement limité de 'exponentielle'

Le développement limité de $\mathbf{e^x}$, à l'ordre **n**, est :

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

1 pt **Q1-** Écrire la fonction **factoriel (k)** qui reçoit en paramètre un entier positif **k**, et qui retourne la valeur de factorielle **k : k! = 1 * 2 * 3 * ... * (k-1) * k.**

NB : La fonction **factoriel (0)** retourne **1**

0.75 pt **Q2-** Déterminer la complexité de la fonction **factoriel (k)**, et justifier votre réponse.

0.75 pt **Q3-** Écrire la fonction **calcul (x, k)** qui reçoit en paramètres un réel **x** et un entier positif **k**, et qui retourne la valeur de l'expression suivante :

$$\frac{x^k}{k!}$$

1.5 pt **Q4-** Écrire la fonction **exponentielle (x, n)** qui reçoit en paramètres un réel **x** et un entier positif **n**, et qui retourne la valeur du développement limité de $\mathbf{e^x}$, à l'ordre **n**.

Partie I : Calcul numérique

Modélisation d'une épidémie

Le modèle 'SIR'

La crise sanitaire mondiale du Coronavirus Covid-19 a démontré le rôle des modélisations mathématiques dans la prise de décisions politiques et sanitaires.

Les modèles à compartiments font partie des premiers modèles mathématiques, à avoir été utilisés en épidémiologie. L'idée est de diviser une population en plusieurs compartiments (groupes d'individus), et définir les règles d'échanges entre ces compartiments.

Le modèle SIR

Dans ce modèle, les individus sont répartis en trois compartiments :

- **S**, dans lequel se trouvent les individus sains et susceptibles d'être infectés ;
- **I**, pour les individus infectés et infectieux ;
- **R**, pour les individus retirés du modèle, non susceptibles d'être infectés, car guéris et immunisés ou décédés.

Le modèle **SIR** décrit l'évolution dans le temps du nombre d'individus dans chaque compartiment, et part du schéma simplifié suivant :



- **β** : représente le taux de transmission de la maladie d'un individu infecté à un individu sain ;
- **γ** : représente le taux de guérison.

Un individu est d'abord sain (**S**) ; il est infecté (**I**) avec la probabilité (**β**) ; enfin, il guérit (**R**), avec la probabilité (**γ**).

Il paraît plus naturel de travailler avec le nombre des individus dans chaque catégorie, mais certains calculs seront plus simples si on utilise plutôt la proportion des individus dans chaque catégorie, ce qui nous permet de connaître tout aussi bien la progression de l'épidémie.

On note donc $S(t)$, $I(t)$ et $R(t)$ les proportions des individus dans chaque catégorie à l'instant t.

L'évolution des trois catégories de population peut alors être décrite par le système d'équations différentielles suivant :

$$(E)\quad \begin{cases} S'(t) = -\beta.S(t).I(t) & (1) \\ I'(t) = \beta.S(t).I(t) - \gamma.I(t) & (2) \\ R'(t) = \gamma.I(t) & (3) \end{cases}$$

Les dérivées permettent de connaître la variation (c'est à dire si c'est croissant ou décroissant) des fonctions $S(t)$, $I(t)$ et $R(t)$ en fonction du temps t, afin d'en décrire l'évolution au cours du temps.

Le produit $S(t).I(t)$ représente le nombre de contacts entre des individus sains et des individus infectés. β étant le taux de transmission, il y a dès lors $\beta.S(t).I(t)$ individus nouvellement infectés. Ceux-ci se soustraient des individus sains (équation 1), et s'ajoutent aux individus infectés (équation 2).

De même, parmi les individus infectés, certains vont guérir. γ étant le taux de guérison, il a $\gamma.I(t)$ individus nouvellement guéris qui se soustraient des individus infectés (équation 2) et s'ajoutent aux individus retirés (équation 3).

On suppose que les modules **numpy** et **matplotlib.pyplot** sont importés :

```
import numpy as np
import matplotlib.pyplot as pl
```

La méthode Euler :

Dans cette partie, nous allons nous intéresser à la méthode **Euler** permettant de résoudre numériquement les équations différentielles du premier ordre, avec une condition initiale, sous la forme :

$$\begin{cases} y' = f(t, y(t)) \quad ; \quad \forall\, t \in [t_0, t_0 + T] \\ y(t_0) = y_0 \end{cases}$$

On note **I** = $[t_0, t_0 + T]$ l'intervalle de résolution. Pour un nombre de nœuds **N** donné, soit $t_n = t_0 + n*h$, avec $n = 0, 1, 2, \ldots, N$, une suite de nœuds de **I** induisant une discrétisation de **I** en sous-intervalles $I_n = [t_n, t_{n+1}]$. La longueur h de ces sous-intervalles est appelée **pas** de discrétisation. Le pas de discrétisation h est donné par $h = \frac{T}{N}$.

Soit y_n l'approximation au nœud t_n de la solution exacte $y(t_n)$. Le schéma de la méthode **Euler** est donné par :

- y_0 est donné
- $y_{n+1} = y_n + h * f(t_n, y_n)$

Q.1- Écrire la fonction **def euler (f, t_0, T, y_0, N):** qui reçoit en paramètres la fonction $f : (t , y) \rightarrow f(t , y)$, t_0 et T tels que $[t_0 , t_0 + T]$ est l'intervalle de résolution, la condition initiale y_0 et N est le nombre de nœuds. La fonction retourne $tt = [t_0 , t_1 , \dots , t_N]$ et $Y = [y_0 , y_1 , \dots , y_N]$ en utilisant le schéma d'Euler.

Q.2- Le but de cette question est d'utiliser la fonction précédente **euler**, pour résoudre numériquement le système d'équations différentielles **(E)**.

Q.2.a- En posant $Y(t) = [S(t) , I(t) , R(t)]$, montrer que le système d'équations différentielles **(E)** peut s'écrire sous la forme : $Y'(t)=F(t, Y(t))$ avec

$$F : I * IR^3 \rightarrow IR^3$$
$$(t , X = [X[0] , X[1] , X[2]]) \rightarrow [z_0 , z_1 , z_2]$$

z_0 , z_1 et z_2 sont à préciser en fonction de **X[0], X[1]** et **X[2]**.

Q.3- On suppose que **beta** et **gamma** sont deux variables globales, initialisées par les valeurs suivantes :

beta = 0.5

gamma = 0.1

En utilisant la fonction précédente **euler**, écrire un script python qui permet de tracer la représentation graphique ci-dessous (Figure 1), des fonctions **S(t)**, **I(t)** et **R(t)**. Le nombre de points générés dans chaque courbe est $N=10^3$.

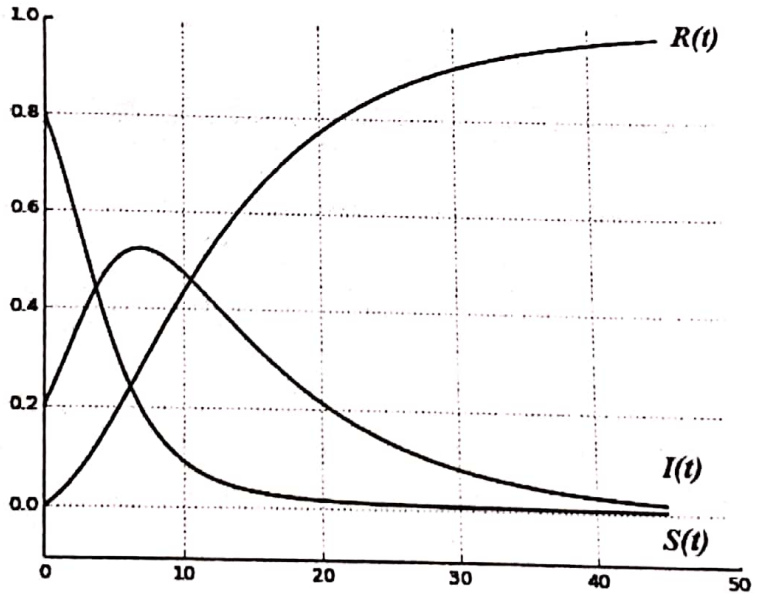


Figure 1

Partie II : Bases de données et langage SQL

Dans le but de suivre l'évolution de la pandémie du Covid-19 dans chaque pays, on propose d'utiliser une base de données composée de 3 tables : **Pays**, **Villes** et **Quotidien**.

a- Structure de la table 'Pays' :

La table **'Pays'** est composée de trois champs : Le champ '**codPays**' de type texte. Ce champ représente la clé primaire de la table 'Pays'. Le champ '**nom**' de type texte, contient le nom de chaque pays.

Exemples :

codPays	nom
DEU	Allemagne
FRA	France
ESP	Espagne
ARN	Suède
ITA	Italie
...	...

b- Structure de la table 'Villes' :

La table **'Villes'** est composée de 4 champs : Le champ '**codVille**' de type texte. Il représente la clé primaire de la table 'Ville'. Le champ '**nom**' de type texte, il contient le nom de chaque ville. Le champ '**population**' de type entier, il contient le nombre de personnes dans chaque ville. Le champ '**codP**' de type texte, il contient le code du pays de chaque ville.

Exemples :

codVille	nom	population	codP
CIA	Rome	2 844 395	ITA
NAP	Napoli	954 244	ITA
PAR	Paris	2 148 271	FRA
TXL	Berlin	3 748 148	DEU
LYS	Lyon	1 669 730	FRA
	...	...	...

c- Structure de la table 'Quotidien' :

La table 'Quotidien' est composée de 5 champs : Le champ '**jour**' de type date, il contient la date de chaque jour. Le champ '**infectés**' de type entier, il contient le nombre de nouveaux cas de personnes infectées par le virus, chaque jour dans chaque ville. Le champ '**rétablis**' de type entier, il contient le nombre de nouveaux cas de personnes guéries du virus, chaque jour dans chaque ville. Le champ '**décédés**' de type entier, il contient le nombre de nouveaux cas de personnes décédées à cause du virus, chaque jour dans chaque ville. Le champ '**codV**' de type texte, il contient les codes des villes.

Exemples :

jour	infectés	rétablis	décédés	codeV
2020-07-15	4653	3458	172	CIA
2020-07-15	6074	4892	493	TXL
2020-07-15	1762	2059	651	LYS
2020-07-16	3721	5973	85	CIA
...	...			...

Q.1- Le jour du **'17/09/2020'**, la ville de code **'TXL'** a enregistré **2513** nouveaux cas de personnes infectées, **2847** nouveaux cas de personnes rétablies, et **65** nouveaux cas de personnes décédées.

Écrire, en langage **SQL**, la requête qui ajoute cet enregistrement dans la table 'Quotidien'.

Q.2- Écrire, en langage **SQL**, la requête qui donne pour résultat, le compte des villes qui ont enregistré des nombres de nouveaux cas de personnes infectées, composés d'au moins **5** chiffres, et ce le jour du **'27/08/2020'**.

Q.3- Écrire, en langage **SQL**, la requête qui donne pour résultat, le nombre de nouveaux cas de personnes infectées, le nombre nouveaux cas de personnes rétablies et le nombre de nouveaux cas de personnes décédées, chaque jour du mois de **Septembre 2020**, dans toutes les villes du pays **'Italie'**.

Q.4- Écrire, en algèbre relationnelle, la requête de la question **Q.3**

Q.5- *Le **taux d'incidence TI** pour une ville est égal au nombre de nouveaux cas de personnes infectées dans cette ville, pendant une période donnée, divisé par la population de cette ville. Le **TI** s'exprime en nombre de personnes pour **100000** personnes.*

Écrire, en langage **SQL**, la requête qui donne pour résultat, les codes des pays, les noms des villes et le **TI** de chaque ville, pendant l'année **2020**, tels que **TI** est compris entre **500** et **1000**, triés dans l'ordre croissant du **TI**.

Q.6- *Le **taux de guérison TG** pour un pays est égal au nombre total de personnes rétablies dans ce pays, divisé par le nombre total de personnes infectées dans ce même pays. Le **TG** s'exprime en pourcentage.*

Écrire, en langage **SQL**, la requête qui donne pour résultat, les noms des pays et le **TG** pour chaque pays, tels que **TG > 70%**, triés dans l'ordre décroissant du **TG**.

Q.7- *Le **taux de mortalité TM** pour une ville est égal au nombre total de personnes décédées dans cette ville, divisé par le nombre total de personnes infectées dans cette même ville. Le **TM** s'exprime en pourcentage.*

Écrire, en langage **SQL**, la requête qui donne pour résultat, les noms des pays, le plus petit **TM**, le plus grand **TM** et la moyenne des **TM** des villes dans chaque pays, triés dans l'ordre croissant du nombre de villes dans chaque pays.

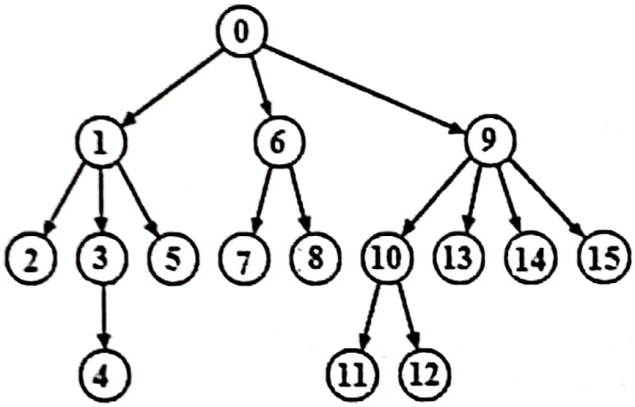
Partie III : Problème Python

Arbre généalogique

Le but de ce problème est de modéliser une structure élémentaire représentant un arbre généalogique « descendant » où on représente sur quelques générations la descendance issue d'un unique ancêtre commun, sans se préoccuper du sexe des différents membres de l'arbre. Ainsi, chaque membre de l'arbre est relié vers le haut à une unique personne qu'on appellera simplement son père, et vers le bas à tous ses enfants. À l'exception de l'ancêtre commun tout en haut de l'arbre qui n'a pas de père.

Si l'arbre généalogique est composé de N membres, alors chaque membre de l'arbre est représenté par un nombre entier positif unique compris entre 0 et N-1. Le plus grand ancêtre est représenté par le nombre 0.

Exemple : Arbre généalogique pour N=16



Représentation de l'arbre généalogique en Python :

En python, pour représenter un arbre généalogique composé de N *membres*, on utilise une liste G de longueur N, telle que pour tout entier $i \in [\![0, N]\!]$:

- Si i est la racine de l'arbre, alors la valeur de l'élément G[i] est None
- Sinon, la valeur de l'élément G[i] est le père de l'élément i dans l'arbre.

Exemple :

L'arbre généalogique, dans l'exemple ci-dessus, sera représenté par la liste G de longueur 16 :

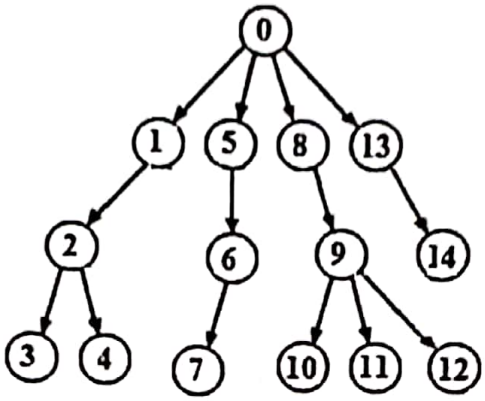
G =	[None,	0,	1,	1,	3,	1,	0,	6,	6,	0,	9,	10,	10,	9,	9,	9]
indices	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*

Les indices de la liste G représentent les membres de l'arbre généalogique :

- ✓ Le membre 0 est la racine de l'arbre, il n'a pas de père : G[0] = None.
- ✓ 0 est le père du membre 1 : G[1] = 0
- ✓ 6 est le père du membre 7 : G[7] = 6
- ✓ 10 est le père du membre 12 : G[12] = 10
- ✓ ...

NB : Pour plus de clarté, dans les exemples des fonctions de cette partie, on utilisera l'arbre généalogique précédent.

Q.1- Donner la liste qui représente l'arbre généalogique suivant :



Q.2- Père d'un membre de l'arbre

Écrire la fonction ***pere (G, m)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **m** un entier positif qui représente un membre de l'arbre **G**. La fonction retourne le père du membre **m** dans l'arbre **G**.

Exemples :

- La fonction **pere (G, 0)** retourne le nombre : **None**
- La fonction **pere (G, 4)** retourne le nombre : **3**

Q.3- Liste des fils d'un membre

Écrire la fonction ***liste_fils (G, p)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **p** un entier positif qui représente un membre de l'arbre **G**. La fonction retourne la liste des fils du membre **p** dans l'arbre **G**.

Exemples :

- La fonction **liste_fils (G, 8)** retourne la liste : **[ ]**
- La fonction **liste_fils (G, 9)** retourne la liste : **[10, 13, 14, 15]**

Q.4- Génération d'un membre

Dans un arbre généalogique, la **génération** de l'ancêtre commun est **0**, la génération de ses enfants est **1**, la génération des enfants de ses enfants est **2**, …

Écrire la fonction ***generation (G, m)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **m** un entier positif qui représente un membre de l'arbre **G**. La fonction retourne la génération de **m** dans l'arbre **G**.

Exemples :

- La fonction **generation (G, 0)** retourne le nombre **0**
- La fonction **generation (G, 4)** retourne le nombre **3**

Q.5- Hauteur de l'arbre généalogique

La **hauteur** de l'arbre généalogique est la plus grande génération des membres de cet arbre.

Écrire la fonction ***hauteur (G)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique. La fonction retourne la hauteur de l'arbre **G**.

Exemple :

La fonction **hauteur (G)** retourne le nombre **4**

Q.6- Descendant d'un membre

Soient **i** et **j** deux membres d'un arbre généalogique **G**. On dit que **i** est un **descendant** de **j** si **j** est le père de **i**, ou si **j** est le père du père de **i**, ou si **j** est le père du père du père de **i**, etc ...

Exemples :

- ✓ Le membre **12** est un descendant du membre **9**
- ✓ Le membre **5** est un descendant du membre **1**
- ✓ Le membre **2** n'est pas un descendant du membre **6**

Écrire la fonction ***descendant (G, i, j)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **i** et **j** deux entiers positifs qui représentent deux membres de l'arbre **G**. La fonction retourne **True** si **i** est un descendant de **j**, sinon, la fonction retourne **False**.

Exemples :

- La fonction **descendant (G, 14, 0)** retourne **True**
- La fonction **descendant (G, 5, 4)** retourne **False**

Q.7- Plus proche ancêtre de deux membres

Soient **i** et **j** deux membres d'un arbre généalogique **G**. Le **plus proche ancêtre** de **i** et **j** est le membre de l'arbre qui est ancêtre commun à **i** et **j**, et ayant la plus grande génération des ancêtres communs à **i** et **j**. Le plus proche ancêtre existe toujours, et il est même unique, puisque tout membre de l'arbre est ancêtre de lui-même.

Exemples :

- ✓ Le membre **1** est le plus proche ancêtre des deux membres **2** et **4**
- ✓ Le membre **0** est le plus proche ancêtre des deux membres **7** et **12**

Écrire la fonction ***plus_proche_ancetre (G, i, j)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **i** et **j** deux entiers positifs qui représentent deux membres de l'arbre **G**. La fonction retourne le plus proche ancêtre de **i** et **j** dans l'arbre **G**.

Exemples :

- La fonction **plus_proche_ancetre (G, 5, 11)** retourne : **0**
- La fonction **plus_proche_ancetre (G, 12, 15)** retourne : **9**
- La fonction **plus_proche_ancetre (G, 6, 7)** retourne : **6**

Q.8- Distance entre deux membres

On considère deux membres **i** et **j** de l'arbre généalogique **G**. La **distance** entre **i** et **j** est la longueur du plus court chemin menant de **i** à **j** dans l'arbre. Ce chemin est celui qui part de **i**, et qui remonte jusqu'au plus proche ancêtre de **i** et **j**, puis qui redescend jusqu'à **j**.

Exemples :

- ✓ La distance entre les membres **4** et **11** est : **6**
- ✓ La distance entre les membres **7** et **8** est : **2**
- ✓ La distance entre les membres **12** et **15** est : **3**

Écrire la fonction ***distance (G, i, j)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **i** et **j** deux entiers positifs qui représentent deux membres de l'arbre **G**. La fonction retourne la distance entre **i** et **j** dans l'arbre **G**.

Exemples :

- La fonction **distance (G, 8, 4)** retourne : 5
- La fonction **distance (G, 5, 2)** retourne : **2**

Q.9- Plus court chemin entre deux membres

On considère deux membres **i** et **j** de l'arbre généalogique **G**. Le **plus court chemin** de **i** à **j** est le chemin partant de **i** et remontant jusqu'au plus proche ancêtre commun de **i** et **j**, puis redescendant jusqu'à **j**.

Exemples :

- ✓ Le plus court chemin du membre **4** au membre **11** est : '4-3-1-0-9-10-11'
- ✓ Le plus court chemin du membre **8** au membre **7** est : '8-6-7'
- ✓ Le plus court chemin du membre **12** au membre **15** est : '12-10-9-15'

Écrire la fonction ***plus_court_chemin (G, i, j)*** qui reçoit en paramètres la liste **G** qui représente un arbre généalogique, et **i** et **j** deux entiers positifs qui représentent deux membres de l'arbre **G**. La fonction retourne la chaîne de caractères qui contient le chemin le plus court du membre **i** au membre **j** dans l'arbre **G**.

Exemples :

- La fonction **plus_court_chemin (G, 5, 9)** retourne : **'5-1-0-9'**
- La fonction **plus_court_chemin (G, 8, 4)** retourne : **'8-6-0-1-3-4'**

≈≈≈≈≈≈ **FIN DE L'ÉPREUVE** ≈≈≈≈≈≈

Travailler les chapitres de ce sujet

Prepa Help suit le programme de TSI chapitre par chapitre : cours détaillé, méthodes, pièges, exercices et problèmes corrigés pas à pas, tirés des concours. L'essai gratuit donne accès aux chapitres offerts de ta filière, sans carte bancaire.

Chapitres liés à ce sujet

- Bases de Python : expressions, types, instructions, fonctions
- Récursivité
- Complexité des algorithmes itératifs et récursifs

Ouvrir l'essai gratuit

Scanne le code ou clique sur le bouton.

prepahelp.com/#essai

Une question sur le concours ou sur ta filière : **écrire sur WhatsApp**

Les autres années et épreuves : prepahelp.com/annales/cnc/tsi