Concours National Commun (CNC), Maroc

Session 2019, filière TSI

Informatique

Ce que traite le sujet

- Exercice : somme et puissance
- Partie I : base de données et langage SQL
- Partie II : réseau routier

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/2019/info

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

Épreuve d'Informatique – Session 2019 – 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 deux 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.

≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈

Exercice : (4 points)

Somme et puissance

1 pt **Q1-** Écrire la fonction **somme (L)** qui reçoit en paramètre une liste L de nombres entiers, et qui retourne la somme des éléments de **L**.

Exemple :

La fonction **somme ([7, 0 , -1, 5 , -3])** renvoie la valeur **8 = 7 + 0 + (-1) + 5 + (-3)**

0.5 pt **Q2-** Déterminer la complexité de la fonction **somme (L)**, et justifier votre réponse.

1 pt **Q3-** Écrire la fonction **list_puissances (L, p)** qui reçoit en paramètres une liste **L** de nombres entiers, et un entier **p** strictement positif. La fonction renvoie une nouvelle liste qui contient les éléments de **L** élevés chacun à la puissance **p**.

Exemple :

La fonction **list_puissances ([7, 0 , -1, 5 , -3] , 2)** renvoie la liste **[49 , 0 , 1 , 25 , 9]**

1 pt **Q4-** Écrire la fonction **som_puiss (L, p)** qui reçoit en paramètres une liste **L** de nombres entiers, et un entier **p** strictement positif. La fonction renvoie la somme des éléments de **L** élevés chacun à la puissance **p**.

Exemple :

La fonction **som_puiss ([7, 0 , -1, 5 , -3], 2)** renvoie le nombre **84 = $7^2 + 0^2 + (-1)^2 + 5^2 + (-3)^2$**

0.5 pt **Q5-** Déterminer la complexité de la fonction **som_puiss (L, p),** et justifier votre réponse.

Partie I :

Base de données et Langage SQL

On considère la base de données relationnelle composée de deux tables : la table **Pays** et la table **Villes**.

Pays	
code	*(texte)*
nom_pays	*(texte)*
superficie	*(réel)*
continent	*(texte)*

Villes	
cod_ville	*(texte)*
nom_ville	*(texte)*
cod_pays	*(texte)*

Structure de la table **'Pays'** :

La table '**Pays**' contient des informations sur les pays, elle est composée de **4** champs :

- Le champ **code** contient un code unique pour chaque pays ;
- Le champ **nom_pays** contient le nom de chaque pays ;
- Le champ **superficie** contient la superficie de chaque pays, exprimée en Km² ;
- Le champ **continent** contient le continent de chaque pays.

Exemples d'enregistrements dans la table **'Pays'** :

code	nom_pays	superficie	continent
MAR	Maroc	446 550	Afrique
DEU	Allemagne	357 386	Europe
ITA	Italie	301 338	Europe
ARG	Argentine	2 780 000	Amérique
RUS	Russie	1 710 000	Asie
...	...	...	...

Structure de la table **'Villes'** :

La table '**Villes**' contient des informations sur les villes des pays, elle est composée de **3** champs :

- Le champ **cod_ville** contient un code unique pour chaque ville ;
- Le champ **nom_ville** contient les noms des villes ;
- Le champ **cod_pays** contient le code du pays de chaque ville.

Exemples d'enregistrements dans la table **'Villes'** :

cod_ville	nom_ville	cod_pays
CIA	Rome	ITA
CMN	Casablanca	MAR
SVO	Moscou	RUS
RAK	Marrakech	MAR
TXL	Berlin	DEU
...	...	...

I. 1- Déterminer les clés primaires et les clés étrangères dans les tables **Pays** et **Villes**.

Justifier votre réponse.

I. 2- Écrire, en algèbre relationnelle, la requête qui donne le résultat suivant :

Les noms et les superficies des pays du continent '**Europe**'.

Écrire, en langage SQL, les requêtes qui donnent les résultats suivants :

I. 3- Les noms des villes, sans répétition, dans les quelles le caractère **'m'** se trouve à la **2^{éme}** et à la **dernière** position. Exemple : A**m**sterda**m**.

I. 4- Les codes et les noms des villes du pays '**Espagne**', triés dans l'ordre alphabétique des noms des villes.

I. 5- Les noms des pays, et le nombre de villes dans chaque pays, ayant le nombre de villes compris strictement entre **100** et **1000**, triés dans l'ordre décroissant des nombres de villes.

I. 6- Dans certains pays, on peut trouver une ville ayant le même nom que celui de son pays. Donner les noms de ces pays.

I. 7- Supprimer toutes les villes du pays '**Belgique**'.

≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈

Partie II :

Réseau routier

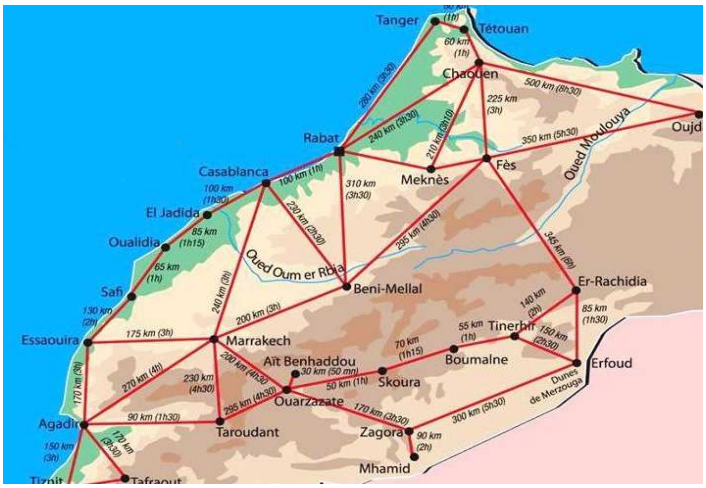


Figure 1 : Extrait du réseau routier du Maroc

Un réseau routier peut être représenté par un dessin qui se compose de points et de traits continus reliant deux à deux certains de ces points : les **points** sont les **villes**, et les **lignes** sont les **routes**. On considère que toutes les routes sont à double sens. Chaque route peut être affectée par une valeur qui peut représenter le temps ou la distance entre deux villes, …

Étant donné un tel réseau, on pourra s'intéresser, par exemple, à la résolution des problèmes suivants :

- Quel est le plus court chemin entre deux villes ?
- Entre deux villes, quel est le nombre de chemins passant par un nombre de routes donné ?
- Est-il possible de passer par toutes les villes du réseau, sans passer deux fois par une même ville ?, si oui, quel est le plus court chemin ?
- Entre deux villes, quel est le chemin ayant le moindre coût ?
- …

Modélisation d'un réseau routier

On considère un réseau routier composé de **n** villes (avec **n ≥ 2**). Les villes du réseau routier sont numérotées par des entiers allant de **0** à **n-1**.

Exemple :

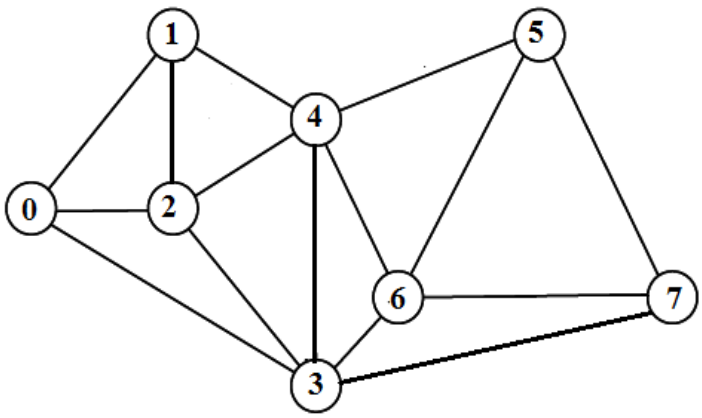


Figure 2 : Réseau routier composé de **15** routes, et de **8** villes numérotées de **0** à **7**.

Pour plus de clarté, tous les exemples de cette partie seront appliqués sur le réseau routier de la **figure 2**.

Représentation du réseau routier par une matrice

Pour représenter un réseau routier de **n** villes, on utilise une matrice symétrique **R** d'ordre **n** (**n** lignes et **n** colonnes), telle que :

Pour toutes les villes **i** et **j**, telles que **0 ≤ i < n** et **0 ≤ j < n**, on a :

- `R[i,j]= R[j,i]=1`, s'il existe une route qui relie entre la ville **i** et la ville **j** ;
- `R[i,j]= R[j,i]=0`, sinon.

Exemple :

Le réseau routier de la **figure 2** est représenté par la matrice symétrique **R** suivante :

i/j	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
1	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**
2	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
3	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**
4	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**
5	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**
6	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
7	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**

***NB :** Dans la matrice symétrique **R**, les lignes **i** et les colonnes **j** représentent les villes du réseau routier.*

II. 1- Représentation de la matrice du réseau routier par une liste de listes

Pour représenter la matrice symétrique **R** d'ordre **n**, on utilise une liste composée de **n** listes toutes de même longueur **n**.

Exemple :

La matrice symétrique **R,** du réseau routier de la **figure 2**, est représentée par la liste **R**, composée de **8** listes, de taille **8** chacune :

```
R = [ [0,1,1,1,0,0,0,0],
      [1,0,1,0,1,0,0,0],
      [1,1,0,1,1,0,0,0],
      [1,0,1,0,1,0,1,1],
      [0,1,1,1,0,1,1,0],
      [0,0,0,0,1,0,1,1],
      [0,0,0,1,1,1,0,1],
      [0,0,0,1,0,1,1,0]
    ]
```

`R[i][j]` est l'élément de **R**, à la **i**ème ligne et la **j**ème colonne.
Exemples : **R[0][0]** est l'élément : **0**, et **R[0][2]** est l'élément : **1**

`R[i]` est la ligne d'indice **i** dans **R**.
Exemple : **R[0]** est la liste : **[0 , 1 , 1 , 1 , 0 , 0 , 0 , 0]**, qui représente la ligne d'indice **0** dans **R**.

Q 1- À partir de la matrice **R**, donner les résultats des expressions suivantes :

R[4][2] , R[1] , len(R[2]) , len(R)

II. 2- Villes voisines

***i** et **j** sont deux villes dans un réseau routier représenté par une matrice symétrique **R**.*

*Les villes **i** et **j** sont **voisines**, s'il existe une route entre la vaille **i** et la ville **j**.*

Q 2. a- Écrire la fonction **`voisines(i,j,R)`**, qui reçoit en paramètres une ville **i** d'un réseau routier représenté par la matrice symétrique **R**. La fonction renvoie **True** si les villes i et j sont voisines, sinon, la fonction renvoie **False**.

Exemples :

- La fonction **voisines (3, 0, R)** renvoie **True** ; *(les villes 3 et 0 sont voisines)*
- La fonction **voisines (3, 5, R)** renvoie **False**. *(les villes 3 et 5 ne sont pas voisines)*

Q 2. b- Écrire la fonction **`list_voisines(i,R)`**, qui reçoit en paramètres une ville **i** d'un réseau routier représenté par la matrice symétrique **R**. La fonction renvoie la liste de toutes les villes voisines à la ville **i**.

Exemple :

La fonction **list_voisines (3, R)** renvoie la liste : **[0 , 2 , 4 , 6 , 7]**

II. 3- Degré d'une ville

*Dans un réseau routier, le **degré** d'une ville **i** est le nombre de villes voisines à la ville **i**.*

Q 3- Écrire la fonction **`degre(i,R)`**, qui reçoit en paramètres une ville **i** d'un réseau routier représenté par la matrice symétrique **R**. La fonction renvoie le degré de la ville **i**.

Exemples :

- La fonction **degre (3, R)** renvoie le nombre : **5** *(La ville **3** possède **5** villes voisines)*
- La fonction **degre (0, R)** renvoie le nombre : **3** *(La ville **0** possède **3** villes voisines)*
- La fonction **degre (2, R)** renvoie le nombre : **4** *(La ville **2** possède **4** villes voisines)*

II. 4- Liste des degrés des villes

Q 4- Écrire la fonction **`list_degres(R)`**, qui reçoit en paramètres la matrice symétrique **R** qui représente un réseau routier. La fonction renvoie une liste **D** contenant des tuples. Chaque tuple est composé de deux éléments : une **ville** du réseau routier, et le **degré** de cette ville.

Exemple :

La fonction **list_degres (R)** renvoie la liste **D** suivante :

D = [(0, 3**), (1,** 3**), (2,** 4**), (3,** 5**), (4,** 5**), (5,** 3**), (6,** 4**), (7,** 3**)]**

II. 5- Tri des villes

Q 5- Écrire la fonction **`tri_degres(D)`** , qui reçoit en paramètre la liste **D** des degrés des villes. La fonction trie les tuples de la liste **D** dans l'ordre décroissant des degrés des villes.

Exemple :

D = [(0 , 3**) , (1 ,** 3**) , (2 ,** 4**) , (3 ,** 5**) , (4 ,** 5**) , (5 ,** 3**) , (6 ,** 4**) , (7 ,** 3**)]**

Après l'appel de la fonction **tri_degres (D)**, on obtient la liste suivante :

[(3 , 5**) , (4 ,** 5**) , (2 ,** 4**) , (6 ,** 4**) , (0 ,** 3**) , (5 ,** 3**) , (1 ,** 3**) , (7 ,** 3**)]**

II. 6- Chemin entre deux villes

Dans un réseau routier, un **chemin** est un tuple **T** qui contient des villes du réseau, et qui satisfait les deux conditions suivantes :

- **c1** : Le tuple **T** contient au moins deux villes du réseau routier ;
- **c2** : Deux villes consécutives dans **T** sont voisines.

Exemples :

- Le tuple **(2**, 0, 1, 4**, 3)** est un chemin entre la ville **2** et la ville **3**, composé de **4** routes ;
- Le tuple **(7,** 6, 4, 5, 6**,** 3**, 4)** est un chemin entre la ville **7** et la ville **4**, composé de **6** routes ;
- Le tuple **(** 3, 1, 4, 6 **)** n'est pas un chemin.

Q 6- Écrire la fonction **`chemin_valide(T,R)`** , qui reçoit en paramètre un tuple **T** contenant des villes du réseau routier représenté par la matrice symétrique **R.** La fonction renvoie **True** si le tuple **T** satisfait les deux conditions **c1** et **c2** citées ci-dessus, sinon, la fonction renvoie **False**.

II. 7- Chemin simple entre deux villes

*Dans un réseau routier, un **chemin simple** est un chemin qui passe une seule fois par la même ville.*

Exemples :

- Le chemin **(2**, 0, 1, 4**, 3)** est un chemin simple ;
- Le chemin **(6,** 7, **3**, 4, 2, **3**, **6,** 5 **)** n'est pas un chemin simple.

Q 7- Écrire la fonction **`chemin_simple(T,R)`** , qui reçoit en paramètres un chemin **T** dans un réseau routier représenté par la matrice symétrique **R.** La fonction renvoie **True** si le chemin **T** est un simple, sinon, la fonction renvoie **False**.

II. 8- Plus court chemin entre deux villes

On suppose que la fonction **`liste_chemins(i,j,R)`**, reçoit en paramètres deux villes **i** et **j** d'un réseau routier représenté par la matrice symétrique **R.** Cette fonction renvoie une liste qui contient tous les chemins simples entre la ville **i** et la ville **j.**

Exemple :

La fonction **liste_chemins (1, 5, R)** renvoie la liste de tous les chemins simples entre les villes **1** et **5** :

[(**1**,0,2,3,4,**5**),(**1**,0,2,3,4,6,**5**),(**1**,0,2,3,4,6,7,**5**),(**1**,0,3,4,6,7,**5**),
(**1**,0,3,6,4,**5**),(**1**,0,3,6,**5**),(**1**,0,3,6,7,**5**),(**1**,2,0,3,6,**5**),(**1**,2,0,3,6,7,**5**),
(**1**,2,0,3,7,**5**),(**1**,2,0,3,7,6,4,**5**),(**1**,2,3,7,6,**5**),(**1**,4,2,0,3,7,**5**),(**1**,4,2,0,
3,7,6,**5**),(**1**,4,2,3,6,**5**),(**1**,4,2,3,6,7,**5**),(**1**,4,2,3,7,**5**),(**1**,4,2,3,7,6,**5**),(
1,4,3,6,**5**),(**1**,4,3,6,7,**5**),(**1**,4,3,7,**5**),(**1**,4,3,7,6,**5**),(**1**,4,**5**),(**1**,4,6,3,7,
5),(**1**,4,6,**5**), … ,(**1**,4,6,7,**5**) **]**

*Le **plus court chemin** entre une ville **i** et une ville **j** est un chemin simple entre la ville **i** et la ville **j**, et qui traverse le moins de villes.*

***NB** : On peut trouver plusieurs plus courts chemins entre deux villes.*

Q 8- Écrire la fonction **`pc_chemins(i,j,R)`**, qui reçoit en paramètres deux villes **i** et **j** d'un réseau routier représenté par la matrice symétrique **R.** Cette fonction renvoie la liste de tous les plus courts chemins entre la ville **i** et la ville **j**.

Exemples :

- La fonction **pc_chemins (2, 6, R)** renvoie la liste : [(2 , 3 , 6) , (2 , 4 , 6)]
- La fonction **pc_chemins (7, 1, R)** renvoie la liste :
  [(7 , 3 , 0 , 1) , (7 , 3 , 2 , 1) , (7 , 3 , 4 , 1) , (7 , 5 , 4 , 1) , (7 , 6 , 4 , 1)]
- La fonction **pc_chemins (1, 4, R)** renvoie la liste : [(1 , 4)]

II- 9. Calcul du nombre de chemins entre deux villes

Dans cette partie, on considère que le module **numpy** est importé :

```
from numpy import *
```

On suppose que le réseau routier est représenté par la matrice symétrique **R**, et que la matrice **R** est créée par la méthode **array()** du module **numpy**.

Exemple :

```
R = array([ [0,1,1,1,0,0,0,0],
            [1,0,1,0,1,0,0,0],
            [1,1,0,1,1,0,0,0],
            [1,0,1,0,1,0,1,1],
            [0,1,1,1,0,1,1,0],
            [0,0,0,0,1,0,1,1],
            [0,0,0,1,1,1,0,1],
            [0,0,0,1,0,1,1,0] ])
```

Dans cette partie, on propose de résoudre le problème suivant :

Soit **p** un entier strictement positif.

Quel est le nombre de chemins (simples ou non) entre une ville **i** et une ville **j**, passant exactement par **p** routes ?

Exemple :

Dans le réseau routier de la **figure 2**, pour aller de la ville **1** à la ville **3**, en passant exactement par **5** routes, on peut trouver plusieurs chemins, dont voici quelques uns :

- **(1,**4,5,7,6,**3)**
- **(1,**4,6,5,4,**3)**
- **(1,**2,3,4,2,**3)**
- **(1,**0,2,0,2,**3)**
- **…**

Pour résoudre ce problème, on doit procéder ainsi :

- Calculer la matrice $\mathbf{M = R^p}$ *(matrice symétrique **R** élevée à la puissance **p**)*
- Chaque élément $\mathbf{M[i][j]}$ représente le nombre de chemins, qui partent de la ville **i** à la ville **j**, en passant par **p** routes ;
- La matrice $\mathbf{M}$ est symétrique aussi. Le nombre de chemins, entre la ville **i** et la ville **j**, est le même que celui entre la ville **j** et la ville **i**.

Exemples :

$$\begin{bmatrix} 3 & 1 & 2 & 1 & 3 & 0 & 1 & 1 \\ 1 & 3 & 2 & 3 & 1 & 1 & 1 & \boxed{0} \\ 2 & 2 & 4 & 2 & 2 & 1 & 2 & 1 \\ 1 & 3 & 2 & 5 & 2 & 3 & 2 & 1 \\ 3 & 1 & 2 & 2 & 5 & 1 & 2 & 3 \\ 0 & 1 & 1 & 3 & 1 & 3 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 4 & 2 \\ 1 & \boxed{0} & 1 & 1 & 3 & 1 & 2 & 3 \end{bmatrix}$$

$\mathbf{M = R^2}$

$$\begin{bmatrix} 4 & 8 & 8 & 10 & 5 & 5 & 5 & 2 \\ 8 & 4 & 8 & 5 & 10 & 2 & 5 & \boxed{5} \\ 8 & 8 & 8 & 11 & 11 & 5 & 6 & 5 \\ 10 & 5 & 11 & 8 & 15 & 5 & 11 & 10 \\ 5 & 10 & 11 & 15 & 8 & 10 & 11 & 5 \\ 5 & 2 & 5 & 5 & 10 & 4 & 8 & 8 \\ 5 & 5 & 6 & 11 & 11 & 8 & 8 & 8 \\ 2 & \boxed{5} & 5 & 10 & 5 & 8 & 8 & 4 \end{bmatrix}$$

$\mathbf{M = R^3}$

Par exemple, entre la ville **1** et la ville **7** :

- Il y a **0** chemins qui passent exactement par **deux** routes (*dans la matrice* $\boldsymbol{M = R^2}$) ;
- Il y a **5** chemins qui passent exactement par **trois** routes (*dans la matrice* $\boldsymbol{M = R^3}$).

Rappel : *Le module **numpy** contient la méthode **dot( )** qui calcule le produit matriciel.*

Lorsqu'il s'agit d'une matrice carrée, le calcul de la puissance devient crucial. L'**exponentiation rapide** est un algorithme qui permet de minimiser le nombre de multiplications effectuées dans le calcul de la puissance d'une matrice carrée.

Pour calculer x^n, le principe de l'exponentiation rapide est le suivant :

- Si **n = 1** alors $x^n = x$
- Si **n est pair** alors $x^n = (x^2)^{n/2}$
- Si **n est impair** alors $x^n = x \,.\, (x^2)^{(n-1)/2}$

Q 9. a- Déterminer la complexité de l'algorithme de l'exponentiation rapide. Justifier votre réponse.

Q 9. b- Écrire la fonction **`puissance(R,n)`**, reçoit en paramètres la matrice symétrique **R**, qui représente un réseau routier, et un entier **n** strictement positif**.** La fonction calcule et renvoie la matrice $\mathbf{R^n}$ (**R** puissance **n**), en utilisant le principe de l'exponentiation rapide.

≈ ≈ ≈ ≈ ≈ ≈ **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
- Complexité des algorithmes itératifs et récursifs
- Récursivité
- Tri par fusion et tri rapide

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