# Informatique Tronc Commun
# Révisions quotidiennes

Conseil : Vous pouvez cocher la case lorsque la question n'a pas été convenablement traitée et de prévoir de la refaire 3-4 jours plus tard.

## I Semaine 1

### I.1 Jeudi 2 avril ♥

1–❑ Écrire une fonction `Moyenne(L)` qui renvoie la moyenne des éléments de la liste $L$ (liste non vide de nombres).

2–❑ Écrire une fonction `DepasseMoyenne(L)` qui renvoie la liste des éléments de $L$ strictement plus grand que la moyenne.
Attention : la fonction `Moyenne(L)` devra être appelée une et une seule fois.

3–❑ Quelle est la complexité de ces deux fonctions ?

### I.2 Vendredi 3 avril ♥

1–❑ Écrire une fonction `Maximum(L)` qui renvoie le maximum d'une liste non vide de nombres `L`.

2–❑ Écrire une fonction `Indices(L)` qui renvoie la liste des indices des éléments égaux au maximum de `L`.
On n'appellera qu'une seule fois la fonction `Maximum`.

### I.3 Samedi 4 avril ♥

1–❑ Écrire une fonction `Occurrences(chaine,x)` qui renvoie le nombre d'occurrences de `x` dans une chaine de caractères chaine.

2–❑ Écrire la fonction `Minimum(d)` qui renvoie la clé du dictionnaire `d` dont la valeur associée est minimale.

### I.4 Dimanche 5 avril ♥

1–❑ Écrire une fonction `DictionnaireOccurrences(chaine)` qui prend comme paramètre une chaîne de caractères et renvoie un dictionnaire dont les clefs sont les caractères de la chaîne et la valeur associée est le nombre d'occurrences de ce caractère dans la chaine.

2–❑ Soit texte une chaîne de caractères, écrire une fonction `EltMajoritaire(texte)` qui renvoie l'un des caractères qui apparait le plus souvent souvent dans cette `chaîne`.

## II Semaine 2

### II.1 Lundi 6 avril ♥

1–❑ Considérons le graphe définie le dictionnaire d'adjacence :
`Dico={"A" :["B","C","D"], "B" :["A","C","E"], "C" :[], "D" :["B","C"], "E" :["C","D"]}`

a) Faites un dessin du graphe.

b) Donner la liste des points qui seront visités si on fait un parcours en largeur en partant de `A`.

c) Donner la liste des points qui seront visités si on fait un parcours en profondeur partant de `B`.

2–❑ Écrire une fonction `sommets(G)` qui renvoie la liste des sommets du dictionnaire d'adjacence `G` d'un graphe.

### II.2 Mardi 7 avril ♥

1–❑ Écrire la fonction `degre_succ(G)` qui renvoie le dictionnaire dont les clés sont les sommets de `G` et les valeurs le nombre de successeurs du sommet.

2–❑ Écrire la fonction `predecesseurs(G,x)` qui renvoie la liste des prédécesseurs de `x` dans `G`.

3–❑ Écrire la fonction `transpose(G)` qui renvoie la liste des prédécesseurs des sommets de `G`.

## II.3 Mercredi 8 avril ♥

1–❑ Écrire une fonction `ImageBlanche(h,w)` qui définit une image blanche de taille $h \times w$ ($h$ lignes et $w$ colonnes), ne comportant que des 255.

2–❑ Écrire une fonction `MaxCoeffMatrice(M)` qui, à une matrice `M`, renvoie le plus grand coefficient de `M`, sa ligne et sa colonne.

3–❑ On considère la matrice `M` d'adjacence d'un graphe orienté et pondéré. Écrire une fonction `Transposee(M)` qui renvoie la transposée de la matrice `M` (c'est-à-dire la matrice du graphe dont on a inversé le sens des flèches).

## II.4 Jeudi 9 avril

Soit G un graphe non orienté et pondéré, dont les sommets sont numérotés de 0 à $n-1$, ($n$ étant donc l'ordre du graphe) codé par sa matrice d'adjacente notée `M`.

1–❑ ♥ À quoi correspond la valeur de `M[i][j]` dans le graphe ?

2–❑ ♥ Quelle propriété a la matrice `M` du fait que le graphe soit non orienté ?

3–❑ ♥ Quelle commande permet de récupérer l'ordre du graphe ?

4–❑ ♪ ♪ ♪ Un graphe est complet si tous les sommets sont reliés à l'ensemble des autres sommets.
Écrire une fonction `EstComplet(M)` qui teste si graphe `G` représenté par sa matrice d'adjacence `M` est complet. La fonction renvoie `True` s'il est complet, `False` sinon.

## II.5 Vendredi 10 avril ♥

On considère le graphe `G={0: [1,2,3], 1 : [0,2,4], 2 : [], 3 : [1,2], 4 : [2,3]}`

1–❑ Donner la matrice d'adjacence `M` de `G`.

2–❑ Écrire une fonction `degre_mat(M:list,s:int)->int` qui prend en argument un graphe représenté par sa matrice d'adjacence `M` et un sommet `s`, et renvoie le degré du sommet `s`.

3–❑ Écrire une fonction `voisin_mat(M:list,s:int)->list` qui prend en entrée une matrice d'adjacence et un sommet `s`, et renvoie la liste des voisins du sommet `s`.

4–❑ Écrire une fonction `Isole(G)` qui permet de renvoyer la liste des sommets isolés du graphe `G`.

## II.6 Samedi 11 avril ♥

1–❑ Écrire la fonction `distance(X:list,Y:list)->float` qui renvoie la distance entre les deux listes `X` et `Y`.

2–❑ Écrire la fonction `Distances(X:list,Z:list[list])->list[float]` qui renvoie la liste des distances entre `X` et chaque liste de la liste `Z` en utilisant la fonction précédente.

3–❑ Évaluer la complexité de cette dernière fonction en fonction du nombre d'éléments de `Z` et de la longueur de `X`.

## II.7 Dimanche 12 avril

Soit `L=[['ville',temperature],... ]` qui donne la température (un flottant) un jour donné dans la ville `'ville'`.

1–❑ ♥ Écrire la fonction `moyenne(L:list[list])` qui renvoie la moyenne des températures du jour.

2–❑ Écrire la fonction `ecart_type(L:list[list])` qui renvoie l'écart-type des températures du jour. On veillera à obtenir une fonction de complexité linéaire.

3–❑ Écrire la fonction `froid(L:list[list])->str` qui renvoie la ville où la température a été la plus froide.

# III Semaine 3

## III.1 Lundi 13 avril ♥

On considère une image `img` représentée par un tableau (=liste de listes) de $h$ lignes et de $w$ colonnes. Chaque élément du tableau est une liste de trois entiers $[r, g, b]$ (=pixel). Ces entiers sont codés sur 8 bits.

1–❑ Comment obtient-on le nombre de lignes et le nombre de colonnes de `img` ?

2–❑ Que contient `img[0][0][0]` ? `img[0][0][1]` ? `img[0][0]` ?

3–❑ Écrire une fonction `inv(pixel)` qui renvoie le négatif d'un pixel (représenté par une liste de trois entiers $[r, g, b]$), c'est-à-dire les couleurs inversées (par exemple le niveau de rouge $r$ est remplacé par $255 - r$).

4–❑ Écrire une fonction `negatif(img)` qui renvoie le négatif de l'image.

## III.2 Mardi 14 avril

On considère une image `img` en niveau de gris : c'est un tableau de taille $h \times w$ dont chaque élément est un entier codé sur 8 bits (le pixel est d'autant plus sombre que cet entier est petit)..

1–❑ ♥Écrire la fonction `nb(img,seuil)` qui convertit l'image en noir et blanc selon le critère de seuil : le pixel est blanc (255) au-delà du seuil, il est noir (0) sinon.

2–❑ Écrire la fonction `sombre(img,seuil)` qui renvoie la liste des coordonnées (numéro de ligne, numéro de colonne) des pixels noirs.

## III.3 Mercredi 15 avril

1–❑ ♥Soit un `texte` une chaîne de caractères, écrire une fonction `Presence(lettre,texte)` qui vérifie si `texte` contient le caractère `lettre`.

2–❑ Écrire une fonction `RienAVoir(L1,L2)` qui renvoie `True` si les deux listes `L1` et `L2` n'ont aucun élément en commun et `False` sinon. La complexité est attendue en $\mathcal{O}\Big(\max(\text{len}(L_1), \text{len}(L_2))\Big)$.

3–❑ ♥Soit `D` un dictionnaire dont les clefs et les valeurs sont toutes des chaînes de caractères. Écrire une fonction `PlusLong(D)` qui renvoie la clef dont la valeur associée est de longueur maximale.

## III.4 Jeudi 16 avril ♥

1–❑ Écrire la fonction `Distances(X:list,Z:list[list])->list[float]` qui renvoie la liste des distances entre `X` et chaque liste de la liste `Z`. On n'utilisera pas de fonctions auxiliaires.

2–❑ Écrire une fonction `EstTriee(L)` qui renvoie `True` si la liste de nombre `L` est triée par ordre croissant et `False` sinon.

3–❑ Quelle est la complexité de cette fonction ?

## III.5 Vendredi 17 avril ♪ ♪ ♪

On considère une liste `L` de flottants.

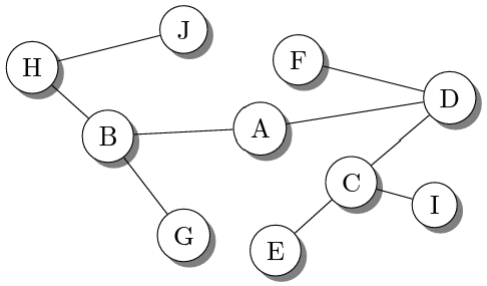
1–❑ Écrire une fonction `PlusProches(L)` qui prend en entrée une liste de flottants et renvoie un couple (tuple) d'indices distincts de valeurs les plus proches dans cette liste.

2–❑ Écrire une fonction `SecondMax(L)` qui prend en entrée une liste de flottants et renvoie la valeur du second maximum de cette liste (la seconde valeur maximale après le maximum).

## III.6 Samedi 18 avril ♪ ♪ ♪

1–❑ Qu'est-ce qu'une pile ? qu'est-ce qu'une file ? S'aider d'une représentation de la vie quotidienne !

2–❑ Effectuer le parcours en largeur et en profondeur du graphe suivant à partir de A :




3–❑ Compléter la fonction suivante qui effectue de façon itérative le parcours en largeur d'un graphe représenté par un dictionnaire l'adjacence.

```
def parcours_largeur(G,s0):
    parcours=[] # liste décrivant le parcours
    vu={} # dictionnaire  des sommets visités
    file=deque([s0]) # file qui garde les sommets en attente
    while ... : # tant que la file n'est pas vide
        s=... # on enlève le sommet au début de la file (file.poplef())
        if ... : # si le sommet s n'a pas déjà été visité
            .... # on l'ajoute à la liste du parcours
            .... # on l'ajoute au dictionnaire vu
            ... : # il faut visiter les voisins v du sommet s
                if ... : # le voisin v de s n'a pas déjà été visité
                    ... # on le place dans la file des sommets en attente d'être visité (en fin de file)
    return visite
```

4–❑ Compléter la fonction suivante qui effectue de façon itérative le parcours en profondeur d'un graphe représenté par un dictionnaire l'adjacence.

```
def parcours_profondeur(G,s0):
    parcours=[] # liste décrivant le parcours
    vu={} # dictionnaire  des sommets visités
    pile=[] # pile qui garde les sommets en attente
    ... # on ajoute au sommet de la pile le sommet de départ
    while ... : # tant que la pile n'est pas vide
        s=... # on enlève et enlève le sommet au sommet de la pile (pile.pop())
        if ... : # si le sommet s  n'a pas déjà été visité
            .... # on l'ajoute à la liste des sommets visités
            .... # on l'ajoute au dictionnaire des sommets vus
            ... : # il faut visiter les voisins v du sommet s
                if ... : # si le voisin v de s n'a pas déjà été visité
                    ... # on le place dans la pile des sommets en attente d'être visité
    return parcours
```

Informatique Tronc Commun

# Révisions quotidiennes (SQL)

Conseil : Vous pouvez cocher la case lorsque la question n'a pas été convenablement traitée et de prévoir de la refaire 3-4 jours plus tard.

Toutes les questions porteront sur la base de données ci-dessous des matériaux magnétiques.

On utilise une base de données contenant les propriétés de quelques matériaux (`materials`) dotés de propriétés magnétiques. Des fournisseurs (`suppliers`) proposent à la vente des matériaux au kg. On recense les offres de prix (`price`) de ces fournisseurs pour chaque matériau. On suppose qu'un fournisseur ne propose qu'une seule offre de prix par matériau.

| material |
|---|
| id_material |
| name |
| curie_temp |
| density |

| price |
|---|
| id_price |
| id_material |
| id_supplier |
| kg_price |

| supplier |
|---|
| id_supplier |
| name |
| country |

## I Semaine 1 ♥

### I.1 Jeudi 2 avril

1–❑ Identifier les clés étrangères et les clés primaires des différentes tables, et tracer des flèches entre elles pour « voir » les relations.

2–❑ Écrire une requête SQL qui permet d'obtenir le nom de tous les matériaux dont la température de Curie est strictement inférieure à 500 kelvins.

### I.2 Vendredi 3 avril

3–❑ Écrire une requête SQL qui permet d'obtenir le nom de tous les matériaux dont la température de Curie est strictement inférieure à 500 kelvins ou la densité strictement inférieure à 8.

### I.3 Samedi 4 avril

4–❑ Écrire une requête SQL qui permet d'obtenir le prix au kg minimum, moyen et maximum calculés sur la totalité des matériaux disponibles et chez tous les fournisseurs.

### I.4 Dimanche 5 avril

5–❑ Écrire une requête SQL qui permet d'obtenir le nom de tous les matériaux disponibles à la vente. On n'affichera une liste sans doublons dans l'ordre alphabétique inverse des noms de matériaux.

## II Semaine 2 ♥

### II.1 Lundi 6 avril

6–❑ Écrire une requête SQL qui permet d'obtenir le nom des fournisseurs du matériau d'identifiant 8713 ainsi que le prix auquel ils fournissent une tonne de ce matériau.

### II.2 Mardi 7 avril

7–❑ Écrire une requête SQL qui permet d'obtenir le nom du fournisseur, **l'identifiant du matériau** qu'il fournit et le prix au kg de ce matériau. On affichera les résultats ordonnés d'après le nom du fournisseur puis le prix au kg.

### II.3 Mercredi 8 avril

8–❑ Écrire une requête SQL qui permet d'obtenir le nom du fournisseur, **le nom du matériau** qu'il fournit et le prix au kg de ce matériau. On affichera les résultats ordonnés d'après le nom du fournisseur et le prix au kg.

### II.4 Jeudi 9 avril

9–❑ Écrire une requête SQL qui permet d'obtenir le nom de tous les matériaux dont le fournisseur est localisé en France.

### II.5 Vendredi 10 avril

10–❑ Écrire une requête SQL qui permet d'obtenir le nom du matériau et son prix moyen, du plus cher au moins cher.

### II.6 Samedi 11 avril

11–❑ Écrire une requête SQL qui permet d'obtenir le nom des fournisseurs de nickel ainsi que le prix auquel ils fournissent une tonne de nickel.

### II.7 Dimanche 12 avril ♪ ♪

12–❑ Écrire une requête SQL qui permet d'obtenir le nom du matériau et son prix moyen, du moins cher au plus cher, si le prix moyen est inférieur à 100 €.

## III Semaine 3

### III.1 Lundi 13 avril ♪ ♪

13–❑ Écrire une requête SQL qui permet d'obtenir le nom du matériau, le nombre de fournisseurs et le prix moyen, s'il y a plus de trois fournisseurs de ce matériau et si le prix moyen est inférieur à 100 €, du plus cher au moins cher en moyenne.

### III.2 Mardi 14 avril ♥

14–❑ Écrire une requête SQL qui permet d'obtenir le nom du matériau le plus cher.

### III.3 Mercredi 15 avril ♥

15–❑ Écrire une requête SQL qui permet d'obtenir le nom des cinquièmes et sixièmes matériaux les moins chers.

### III.4 Jeudi 16 avril ♪ ♪ ♪

16–❑ Écrire une requête SQL qui permet d'obtenir la liste des noms des matériaux dont le prix est supérieur au prix moyen.

### III.5 Vendredi 17 avril ♪ ♪ ♪

17–❑ Écrire une requête SQL qui permet d'obtenir le pourcentage de matériaux différents dont le prix moyen tout magasin confondu est supérieur ou égal au prix moyen de l'ensemble des tarifs.

### III.6 Samedi 18 avril ♪ ♪

18–❑ Écrire une requête SQL qui permet d'obtenir le matériau le plus cher et le matériau le moins cher.