

Proposition de corrigé

Concours : X-ENS

Année : 2016

Filière : MP - PC

Épreuve : Informatique

Ceci est une proposition de corrigé des concours de CPGE, réalisée bénévolement par des enseignants de Sciences Industrielles de l'Ingénieur et d'Informatique, membres de l'[UPSTI](http://www.upsti.fr) (Union des Professeurs de Sciences et Techniques Industrielles), et publiée sur le site de l'association :

<https://www.upsti.fr/espace-etudiants/annales-de-concours>

A l'attention des étudiants

Ce document vous apportera des éléments de corrections pour le sujet traité, mais n'est ni un corrigé officiel du concours, ni un corrigé détaillé ou exhaustif de l'épreuve en question.

L'UPSTI ne répondra pas directement aux questions que peuvent soulever ces corrigés : nous vous invitons à vous rapprocher de vos enseignants si vous souhaitez des compléments d'information, et à vous adresser à eux pour nous faire remonter vos éventuelles remarques.

Licence et Copyright

Toute représentation ou reproduction (même partielle) de ce document faite sans l'accord de l'UPSTI est **interdite**. Seuls le téléchargement et la copie privée à usage personnel sont autorisés (protection au titre des [droits d'auteur](#)).

En cas de doute, n'hésitez pas à nous contacter à : corrigesconcours@upsti.fr.

Informez-vous !

Retrouvez plus d'information sur les [Sciences de l'Ingénieur](#), l'[orientation](#), les [Grandes Ecoles](#) ainsi que sur les [Olympiades de Sciences de l'Ingénieur](#) et sur les [Sciences de l'Ingénieur au Féminin](#) sur notre site : www.upsti.fr

L'équipe UPSTI

ALGORITHMIQUE & PROGRAMMATION

CORRIGÉ INFOX-ENS 2016

MARDI 15 MARS 2016

Q1.



```
reseau_A = [5, [ [0,1],[0,2],[0,3],[1,2],[2,3] ] ]
reseau_B = [5, [ [0,1],[1,2],[1,3],[2,3],[2,4],[3,4] ] ]
```

Q2.



```
def creerReseauVide(n) :
    """ Création d'un réseau vide
    n : int nombre d'invidus du reseau """
    return [n,[]]
```

Q3.



```
def estUnLienEntre(paire, i, j) :
    """
    paire : list avec len(paire) = 2
    i : int
    j : int
    Fonction qui teste si la paire représente le lien entre i et j
    """
    if [i, j] == paire or [j, i] == paire : ## On teste les deux possibilités d'écriture
        return True
    else :
        return False
```

Q4.



```
def sontAmis(reseau,i,j) :
    """
    reseau : list , repesante un reseau social
    i : int,
    j : int,
    Renvoie True s'il existe un lien d'amitié entre i et j
    """

    if [i,j] in reseau[1] or [j,i] in reseau[1] :
        return True
    else :
        return False
```

Ce code est de complexité m , car dans le pire des cas il faut tester tout les liens d'amitié

Remarque : On peut aussi utiliser la fonction **estUnLienEntre**

Q5.



```
def declareAmis(reseau,i,j) :
    """
    reseau : list , repesante un reseau social
    i : int
    j : int
    Ajoute (si necessaire) au reseau le lien entre i et j
    """

    if not sontAmis(reseau,i,j) : ## On teste si le lien d'amitié existe déjà
        reseau[1].append([i,j]) ## On ajoute à reseau[1] (liste des liens) le lien
```

Ce code est de complexité $O(m)$, car dans le pire des cas il faut tester tout les liens d'amitié pour voir si le lien existe (même complexité que la fonction **sontAmis**)

Q6.



```
def listeDesAmisDe(reseau,i) :
    """
    reseau : list repesantant un reseau social
    i : int
    Renvoie la liste des amis de i
    """

    ami = [] ## Initialisation de la liste des amis de i
    for lien in reseau[1] : ## Pour chacun des liens du reseau
```



```

if lien [0] == i : ## Le lien peut s'écrire [i,j]
    ami.append(lien [1])
elif lien [1] == i : ## ou [j,i]
    ami.append(lien [0])
return ami

```

Ce code est de complexité $O(m)$, car il faut tester tout les liens d'amitié pour voir si le lien existe.

Q7.



```

## 0 1 2 3 4 5 6 7 8 9
parent_A = [5,1,1,3,4,5,1,5,5,7]
parent_B = [3,9,0,3,9,4,4,7,1,9]

```

Les représentants sont 5, 4, 1, et 3 pour la représentation filiale A et 9, 7 et 3 pour la représentation filiale B.

Q8.



```

def creerPartitionsEnSingletons (n) :
    """ n : int
    Renvoie le tableau parent où chaque element est son propre représentant """

    parent=[]
    for elt in range(n) :
        parent.append(elt)
    return parent

## autre possibilité en une ligne : return [i for i in range(n)]

```

Q9.



```

def representant (parent, i) :
    """
    parent : list
    i : int
    Renvoie le représentant de i """
    ancetre = parent[i]
    while ancetre!=parent[ancetre] : ## Le représentant est son propre parent
        ancetre=parent[ancetre]
    return ancetre

```


Complexité : $O(n)$, au pire des cas on a une filiation en ligne exemple : $\text{parent}=[1,2,3,\dots,n-1,n-1]$

Q10.

```
## Q10 :
def fusion (parent, i, j) :
    """
    parent : list
    i : int
    j : int
    Fusionne les partition contenant i et j"""
    p = representant (parent, i)
    q = representant (parent, j)
    parent [p] = q
```

Q11.

```
fusion (parent, 0, 1)
fusion (parent, 0, 2)
...
fusion (parent, 0, n-1)
```

Q12.

```
def representant (parent, i) :
    """
    parent : list
    i : int
    Renvoie le représentant de i"""
    ancetre = parent[i]
    while ancetre != parent[ancetre] : ## Le representant est son propre parent
        ancetre = parent[ancetre]
    parent [i] = ancetre ## Modification de la question 12
    return ancetre
```

En terme de complexité, cette modification ne comporte qu'une opération supplémentaire par rapport à la question 10. La complexité reste la même, on peut dire que c'est « gratuit ».

Q13.

```
def listeDesGroupes (parent) :
    """
    parent : list
    Renvoie les groupes de parents"""
```



```

groupe = []
lrepresentant = []
## Droit de modifier parent ?
for i in range(len(parent)) :
    ancetre = representant(parent,i)
    if ancetre in lrepresentant :
        if i!=ancetre :
            groupe[lrepresentant.index(ancetre)].append(i)
    else :
        lrepresentant.append(ancetre)
        if ancetre!=i :
            groupe.append([ancetre,i])
        else :
            groupe.append([ancetre])
return groupe

```

Q14.



```

from random import randint

def coupeMiniumRandomisee(reseau) :
    # Etape 1 :
    P = creerPartitionsEnSingletons (reseau[0])
    # Etape 2 : Aucune action nécessaire
    # Etape 3 :
    lien = reseau[1]
    m = len(lien) # Nombre de lien non marqués
    while len(listeDesGroupes(P))>2 and m!=0 : ## Nombre de groupes au moins 3 et des liens non marqués
        choix = randint(0,m-1) ## Choix du lien au hasard
        [i,j]=lien[choix] ## Valeur de i et de j
        ri , rj = representant(P,i), representant(P,j)
        if ri!=rj :
            fusion(P,i,j)
        m=m-1
        lien = lien[0:choix] + lien[choix+1:]+[lien[choix]]
    ## Etape 4 : Fusion des groupes (pas de critere ?)
    while len(listeDesGroupes(P))>2 :
        groupe = listeDesGroupes(P)
        fusion(P,groupe[0][0],groupe[1][0])
    ## Etape 5 : Return
    return P

```

Q15.



```
def tailleCoupe ( reseau , parent ) :
    nb = 0 ## Initialisation du compteur
    for lien in reseau[1] : # Parcours de tous les liens
        if representant ( parent , lien [0])!=representant ( parent , lien [1]) : ## Lien entre les deux groupes si representant diff
            nb = nb+1
    return nb
```

Q16.

SQL

```
SELECT id2 FROM liens WHERE id1 = x
```

Q17.

SQL

```
SELECT nom, prenom
FROM individus JOIN liens
ON individus.id=liens.id1
WHERE liens.id2=x
```

Q18.

SQL

```
SELECT distinct id1 from liens JOIN
(SELECT id1 from liens where id2 = x) as ami
ON liens.id2=ami.id1 where liens.id1<>x
```