S4 exam-B7 Algo-Comps 2026-03-03

Graded

Student

Total Points

13 / 20 pts

Question 1

L'union fait la force - istreeUF

3.5 / 3.5 pts

Propriétés testées

✔ **+ 2 pts** 2 propriétés - fonctionne

+ 1.5 pts 1 petite erreur
ou opération en plus
ou 3 propriétés testées

+ 1 pt erreur + / trop d'opérations
ou 1 seule propriété (acyclique ou connexe) ok

+ 0.5 pts plusieurs erreurs
ou juste test longueur `L` ok

---

optimisation

✔ **+ 1.5 pts** arrêt au plus tôt (dès que cycle trouvé)

+ 1 pt `build` complet - arrêt dès premier représentant trouvé

+ 0.5 pts `build` complet - arrêt dès que 2ème représentant trouvé

+ 0 pts Pas d'opti (ex : parcours complet `L` puis `P`)
Ou trop d'erreurs / mal géré

0 à l'exo

+ 0 pts tests propriétés faux = beaucoup trop d'erreurs

+ 0 pts exo vide

Question 2

Big component - largest_SCC 7 / 7 pts

structure de base

- **– 1 pt** erreur +
- **– 1 pt** erreur +
- **– 0.5 pts** petite erreur
- **– 0.5 pts** petite erreur

---

calcul tailles composantes

conservation du maximum

- **– 1 pt** erreur +
- **– 1 pt** erreur +
- **– 0.5 pts** petite erreur
- **– 0.5 pts** petite erreur

---

complexité / optimisation

- **– 1 pt** Plus d'un parcours
- **– 1 pt** Opérations inutiles
  Ex : parcours vecteur de composantes
- **– 0.5 pts** Pas d'arrêt si $n//2$ dépassé

– 0.5 pts Opérations inutiles
ex : parcours liste des composantes

+ 1 pt Bonus optimisation

---

0 à l'exo

– 7 pts exo vide

– 7 pts pas le bon algo
ne fonctionne pas du tout
beaucoup trop d'erreurs

✔ **– 0 pts** Tout est parfait !

1 Bien!

Question 3

What is this? **1** / 5 pts

3.1 Application: mystery(Gmyst) — 1 / 2.5 pts

(a)

	0	1	2	3	4	5	6	7	8	9	10	11
D	0	4	1	2	2	3	1	3	2	3	4	4

+ 1 pt correct
`0 4 1 2 2 3 1 3 2 3 4 4`
ou indication **n'affiche rien**

+ 0.5 pts 1 erreur

✔ **+ 0 pts** 2 erreurs / faux

(b)

	0	1	2	3	4	5	6	7	8	9	10	11
C	True		True		True					True		

✔ **+ 1 pt** correct
`True` pour 0 2 4 9
ou indication **n'affiche rien**

+ 0.5 pts 1 erreur

+ 0 pts 2 erreurs / faux

(c) `mystery(Gmyst)`

+ 0.5 pts `False`

✔ **+ 0 pts** autre...

+ 0 pts tout vide

3.2 **mystery(G) - spécifications** **0** / 2.5 pts

(a) `D[s]` :

+ 1 pt profondeur de `s` dans l'arbre du DFS

+ 0.5 pts profondeur de `s` sans précision

+ 0.5 pts (b) `C[s]` : `s` est point d'articulation

+ 1 pt (c) `mystery(G) == True` `G` a au plus 2 points d'articulation

✔ **+ 0 pts** Tout faux / vide

Question 4

Warshall - connect_me **1.5** / 4.5 pts

boucle externe

k = nb composantes

n = ordre

+ 2 pts k itérations

+ 1.5 pts arrêt dès que n sommets vus

+ 1 pt erreurs +
ou opérations en plus

✔ **+ 0.5 pts** n itérations dans tous les cas
ou plusieurs erreurs / trop
d'opérations en plus

---

parcours ligne matrice / boucle interne

+ 1 pt ok / optimisé

✔ **+ 0.5 pts** parcours ligne entière
ou petite erreur

+ 0 pts trop d'erreurs / d'opérations

---

construction liste d'arêtes

+ 1.5 pts correct et optimisé

+ 1 pt 1 erreur
ou opérations en plus
ex : parcours *listes* de composante $+O(k)$

✔ **+ 0.5 pts** erreurs +
ou trop d'opérations
ex : parcours *vecteur* des composantes ($+O(n)$)

+ 0 pts trop d'erreurs ou beaucoup trop d'opérations

---

0 à l'exo

+ 0 pts exo vide

+ 0 pts beaucoup trop d'erreurs / méthode ne fonctionne pas du tout

2 Beaucoup Beaucoup trop complexe

Question 5

Malus / bonus

0 / 0 pts

✔ **+ 0 pts** OK

Malus

– 0.5 pts propreté copie

– 1 pt propreté copie

ALGO-COMPS
exam B7 - 3/3/2026
Réponses

Réponses 1 **(L'union fait la force – `istreeUF(n, L)` – *3.5 points*)**

```
def istreeUF(n,L):
    if len(L) != n-1:
        return False
    P = [-1]*n
    for (x,y) in L:
        if not union(x,y,P):
            return False
    return True
```

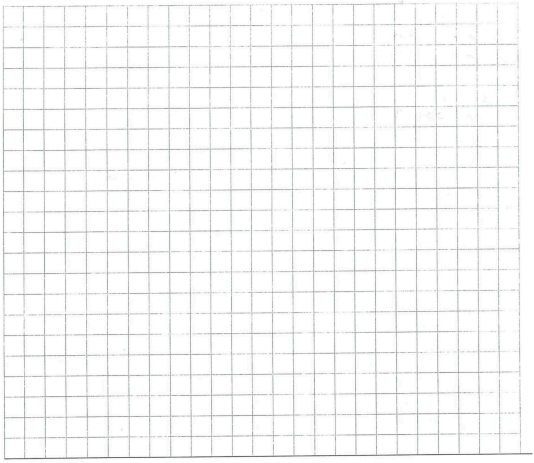
Réponses 2 **(Big component – `largest_SCC(G)` – *7 points*)**

```
def _loopst_scc (G, x, pref, max_composant, cpt, vertexStack):
    cpt += 1
    pref[x] = cpt
    retour_x = pref[x]
    vertexStack.push(x)
    for y in G.adjLists[x]:
        if pref[y] == 0 :
            (retour_y, cpt, max_composant) = _loopst_scc(G,y,pref,max_composant,cpt,vertexStack)
            if max_composant > G.order/2 :
                return (0, 0, max_composant)
            retour_x = min(retour_x, retour_y)
        else :
            retour_x = min(retour_x, pref[y])
    if retour_x == pref[x]:
        y = -1
        nvn = 0
        while y != x :
            y = vertexStack.pop()
            nvn += 1
            pref[y] = G.order * 2
        if nvn > max_composant :
            return (retour_x, cpt, nvn)
    return (retour_x, cpt, max_composant)

def loopst_scc (G):
    pref = [0] * G.order
    max_composant = 0
    cpt = 0
    vertexStack = stack.Stack()
    for s in range(G.order) :
        if pref[s] == 0 :
            (_, cpt, max_composant) = loopst_scc(G, s, pref, max_composant, cpt, vertexStack)
            if max_composant > G.order/2 : 1
                return max_composant
    return max_composant
```

NOM : BENMAHIEDDINE Gaël IUD 30 416.

Éventuelles fonctions supplémentaires pour l'exercice 2. À n'utiliser que si absolument nécessaire.
Ne pas "couper" une fonction sur 2 pages !



***Réponses 3* (What is this ? – *5 points*)**

1. `mystery(Gmyst)`
   (a) *Vecteur* `D` *affiché à la ligne 36 :*

	0	1	2	3	4	5	6	7	8	9	10	11
D	1	11	2	3	5	4	8	6	9	10	7	12

   (b) *Vecteur* `C` *affiché à la ligne 36 (n'indiquer que les valeurs* `True`*) :*

	0	1	2	3	4	5	6	7	8	9	10	11
C	True		True		True					True		

   (c) *`mystery(Gmyst)` retourne :* True

2. `mystery(G)` avec G un graphe connexe quelconque
   (a) `D[s]` = prefix (s)
   (b) `C[s]` = True
   (c) `mystery(G) == True` si
   le nombre d'isthmes est inférieur ou égal à 2.

Réponses 4 **(Warshall, Connect Me – connect_me(M) – *4.5 points*)**

```
def makeMatrixUF(n,L):
    P = [-1]*n
    for (x,y) in L:
        union(x,y,P)
    x = 0
    while P[x] < 0:
        x = p[x]
    toAdd = []
    for y in range(x+1,n):
        if p[y] < 0:
            toAdd.append(x,y)
    return toAdd
def connect_me(M):
    L = []
    for x in range(len(M)):
        for y in range(len(M[0])):
            if M[x][y] == 1 and x != y:
                L.append((x,y))
    return makeMatrixUF(len(M), L)
```

Attention, faux !