DS d'informatique N°1 – Réseaux sociaux

1.

```
reseauA = [5, [[0,1],[0,2],[0,3],[1,2],[2,3]]]
reseauB = [6, [[0,1],[1,2],[2,4],[1,3],[3,4],[2,3]]]
```

2.

```
 def creerReseauVide(n):
   return [n,[]]
```

3.

```
def estUnLienEntre(paire, i, j):
    return paire==[i,j] or paire==[j,i]
```

4.

```
def sontAmis(reseau, i, j):
    for p in reseau[1]:
        if estUnLienEntre(p,i,j):
            return True
    return False
```

Cette fonction contient une boucle de m itérations dans le pire des cas (i et j ne sont pas amis), sa complexité est donc $O(m)$.

5.

```
def declareAmis(reseau, i, j):
  if not sontAmis(reseau, i, j):
    reseau[1].append([i,j])
```

Cette fonction utilise la fonction `sontAmis`, elle est donc aussi de complexité $O(m)$.

6.

```
def listeDesAmisDe(reseau, i):
    amis = []
    for p in reseau[1]:
        if i == p[0]:
            amis.append(p[1])
        elif i == p[1]:
            amis.append(p[0])
    return amis
```

La boucle `for` parcourt les m liens du réseau, la complexité de cette fonction est donc $O(m)$.

7.

```
def listeDesAmisDeOrdre(reseau, i, n):
    dejaAmis = [i]      # Ceux qui sont déjà pris en compte
    amis = [i]          # les Amis d'ordre n
    for k in range(n):
        nouvAmis = []   # les Amis d'ordre n+1
        for a in amis:  # On parcourt les amis d'ordre n
            for b in listeDesAmisDe(reseau, a):
                if b not in dejaAmis:   # On rajoute leurs amis s'ils ne sont pas déjà comptés
                    dejaAmis.append(b)
                    nouvAmis.append(b)
```

```
            amis = nouvAmis
    return amis
```

La complexité de cette fonction est au maximum $O(n^3 m)$ (il ne faut pas oublier la complexité de `in`).

8. `SELECT id2 FROM LIENS WHERE id1=x;`
9. `SELECT nom, prenom FROM INDIVIDUS JOIN LIENS ON id2=id WHERE id1=x;`
10. `SELECT b.id2 FROM LIENS AS a JOIN LIENS AS b ON a.id2=b.id1 WHERE a.id1=x;`