Les exercices marqués par (*) sont à traiter en travail personnel.

Exercice 1

Montrer que dans tout graphe simple non orienté ayant au moins deux sommets, il existe toujours deux sommets de même degré.

Propriétés combinatoires des arbres

Exercice 2

1. Dessiner tous les arbres (non-isomorphes) à n sommets pour $n = 1, 2, 3, 4, 5, 6$.
2. Quel est le degré maximum d'un arbre à n sommets ? Justifier. Quelle est la structure des arbres qui, parmi tous les arbres à n sommets, ont le degré maximum le plus grand possible ? Pour une valeur de n donnée, un tel arbre est-il unique ?
3. Montrer que tout arbre à $n \geq 2$ sommets possède au moins deux sommets de degré 1. Quelle est la structure des arbres possédant exactement deux sommets de degré 1 ? Pour une valeur de $n \geq 2$ donnée, un tel arbre est-il unique ?
4. Soit G un arbre de degré maximum Δ. Montrer que G possède au moins Δ sommets de degré 1. Quelle est la structure des arbres qui, parmi tous les arbres de degré maximum Δ, possèdent un nombre minimum possible de sommets de degré 1 ? Pour une valeur de Δ donnée, un tel arbre est-il unique ?

Notation $O()$, notion de complexité

Exercice 3

Soit $f(n) = \frac{1}{3}n\left(n + \frac{1}{2}\right)(n+1)$. Montrer que $f(n) = \mathcal{O}(n^3)$

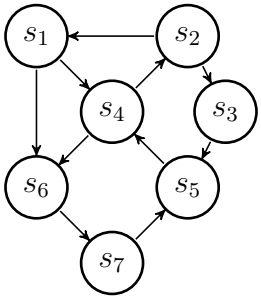
Les relations suivantes sont elles vraies ?

i. $f(n) = \mathcal{O}(n^{10})$
ii. $f(n) = \frac{1}{3}n^3 + \mathcal{O}(n^2)$
iii. $\mathcal{O}(n^2) = \mathcal{O}(n^3)$
iv. $\mathcal{O}(n^3) = \mathcal{O}(n^2)$
v. $\frac{1}{3}n^3 + \mathcal{O}(n^2) = \mathcal{O}(n^3)$

Représentation de graphes orientés

Exercice 4

Soit G_1 le graphe orienté dessiné à droite. Donner les listes de successeurs, la matrice d'adjacence et la matrice d'incidence de G_1.



Exercice 5

(*) Montrer que dans un graphe orienté, la somme des degrés entrants est égale à la somme des degrés sortants.

Calcul à partir de la matrice d'adjacence, de la matrice d'incidence, des listes d'adjacences

Exercice 6

Soit G un graphe (non-orienté) représenté par la matrice d'adjacence A.
Pour chaque algorithme demandé, donner sa complexité.

Proposer un algorithme qui décide si G contient ou pas un sommet isolé (un sommet de degré 0).

Proposer un algorithme qui calcule le degré maximum de G.

(*) Proposer un algorithme qui calcule le degré minimum de G.

Exercice 7

Soit G un graphe (non-orienté) représenté par la matrice d'incidence B.
Pour chaque algorithme demandé, donner sa complexité.

Proposer un algorithme qui calcule le degré maximum de G.

Proposer un algorithme qui calcule le degré minimum de G.

(*) Proposer un algorithme qui décide si G contient ou pas un sommet isolé.

Exercice 8

Soit G un graphe (non-orienté) représenté par les listes d'adjacence $Adj[\,]$.
Pour chaque algorithme demandé, donner sa complexité.

Proposer un algorithme qui calcule le degré minimum de G.

Proposer un algorithme qui décide si G contient ou pas un sommet isolé.

(*) Proposer un algorithme qui calcule le degré maximum de G.

Exercice 9

(*) On se donne un graphe orienté à n sommets et m arcs. On suppose que le degré (nombre d'arcs entrant ou sortant) de chaque sommet est au plus d.

Donner l'algorithme et une majoration (la plus basse possible en fonction de n, m, d) pour la complexité de chacune des opérations suivantes :

1. faire afficher tous les successeurs d'un sommet s donné
2. faire afficher tous les prédécesseurs d'un sommet s donné
3. déterminer si un graphe est sans boucle (une boucle est un arc $s \to s$)

suivant qu'on utilise, pour représenter le graphe, une matrice d'adjacence `A[i,j]`, une matrice d'incidence `B[u,e]`, ou des listes de successeurs `Adj[u]`.

Solutions des exercices optionnels

Correction de l'exercice 5

Considérons la représentation par matrice d'adjacence (ne contenant que des valeurs 0 ou 1) Sur chaque ligne i le nombre de valeurs égales à 1 correspond au degré sortant du sommet numéro i. Sur chaque colonne j le nombre de valeurs égales à 1 correspond au degré entrant du sommet numéro j. Si on calcule la somme des valeurs non nulles sur les lignes (somme des degrés sortants) elle est forcement égale à la somme des valeurs non nulles sur les colonnes (somme des degrés entrants).

Correction de l'exercice 6

Soit G non orienté ayant n sommets numérotés de 1 à n représenté par matrice d'adjacence $A[i, j]$.

1. Existence d'un sommet isolé :
```
pour tout i de 1 à n faire:
    existeSuccesseur <- FAUX
    pour tout j de 1 à n faire:
        si A[i,j] = 1 alors:
           existeSuccesseur <- VRAI
           sortir // de la boucle la plus interne
    si non existeSuccesseur alors:
       retourner VRAI
retourner FAUX
```
2. Calcul du degré maximum :
```
degreMax <- 0
pour tout i de 1 à n faire:
    degre <- 0
    pour tout j de 1 à n faire:
        degre <- degre + A[i,j]
    si degre > degreMax alors:
        degreMax <- degre
retourner degreMax
```
3. Calcul du degré minimum :
```
degreMin <- 0
pour tout j de 1 à n faire:
        degreMin <- degreMin + A[1,j]
pour tout i de 2 à n faire:
    degre <- 0
    pour tout j de 1 à n faire:
        degre <- degre + A[i,j]
    si degre < degreMin alors:
        degreMin <- degre
retourner degreMin
```

Évaluation de complexité.

1. Matrice d'adjacence : dans le cas le pire (il n'y a pas de sommet isolé) il faut effectuer un parcours complet de la matrice, donc $\mathcal{O}(n^2)$; dans le meilleur des cas (le premier sommet est isolé) il suffit de parcourir la première ligne, donc $\Omega(n)$.
2. Il faut parcourir entièrement la matrice, donc $\mathcal{O}(n^2)$.
3. Il faut parcourir entièrement la matrice, donc $\mathcal{O}(n^2)$.

Correction de l'exercice 7

Soit G non orienté ayant n sommets numérotés de 1 à n et m arêtes numérotées de 1 à m représenté par matrice d'incidence $B[i,k]$.

1. Calcul du degré maximum :

```
degreMax <- 0
pour tout sommet i de 1 à n faire:
    degre <- 0
    pour toute arête k de 1 à m faire:
           degre <- degre + B[i, k]
    si degre > degreMax alors:
        degreMax <- degre
retourner degreMax
```

2. Existence d'un sommet isolé : on cherche l'existence d'une ligne ne contenant aucun 1 :

```
pour tout sommet i de i à n faire:
    existeSuccesseur <- FAUX
    pour toute arête k de 1 à m faire:
        si B[i,k] = 1 alors:
           existeSuccesseur <- VRAI
    si non existeSuccesseur alors:
        retourner VRAI
retourner FAUX
```

3. Calcul du degré minimum :

```
degreMin <- 0
pour tout k de 1 à m faire:
        degreMin <- degreMin + B[1,k]
pour tout i de 2 à n faire:
    degre <- 0
    pour tout k de 1 à m faire:
        degre <- degre + B[i,k]
    si degre < degreMin alors:
        degreMin <- degre
retourner degreMin
```

Évaluation de complexité.

1. Il faut parcourir entièrement la matrice d'incidence, donc $\mathcal{O}(n * m)$.
2. Dans le cas le pire il faut effectuer un parcours complet de la matrice, donc $\mathcal{O}(n * m)$; dans le meilleur des cas il suffit de parcourir la première ligne, donc $\Omega(m)$.
3. Il faut parcourir entièrement la matrice d'incidence, donc $\mathcal{O}(n * m)$.

Correction de l'exercice 8

Soit G non orienté ayant n sommets et représenté par listes d'adjacence $Adj[]$

1. Calcul du degré minimum :

```
degMin <- 0
soit u un sommet de V(G)
pour tout sommet v de Adj[u] faire:
    si v = u alors :
        degMin <- degMin + 2
    sinon
        degMin <- degMin + 1
```

```
pour tout sommet s de V(G) \ {u} faire:
    degre <- 0
    pour tout sommet v de Adj[s] faire:
        si v = s alors:
            degre <- degre + 2
        sinon
            degre <- degre + 1
    si degre < degreMin alors:
        degreMin <- degre
retourner degreMin
```

2. Existence d'un sommet isolé : il faut chercher s'il existe un sommet dont la liste d'adjacence est vide.

```
pour tout sommet s de i à n faire:
    si Adj[s] est vide alors:
       retourner VRAI
retourner FAUX
```

3. Calcul du degré maximum :

```
degreMax <- 0
pour tout sommet s de 1 à n faire:
    degre <- 0
    pour tout sommet v de Adj[s] faire:
        si v = s alors:
            degre <- degre + 2
        sinon
            degre <- degre + 1
    si degre > degreMax alors:
        degreMax <- degre
retourner degreMax
```

Évaluation de complexité.

1. Pour tout sommet s, on parcourt sa liste d'adjacence. L'ensemble de ces parcours est en $\mathcal{O}(m)$. De plus il faut regarder chaque liste (même vide), donc le nombre d'accès aux listes est en $\mathcal{O}(n)$. On a donc en tout $\mathcal{O}(n+m)$.
2. On obtient $\mathcal{O}(n)$ s'il n'y a pas de sommet isolé et pour chaque liste on teste si elle est vide ou pas (test en temps constant), $\Omega(1)$ si le premier sommet est isolé et on effectue uniquement un test.
3. On parcourt entièrement les listes d'adjacence, contenant en tout autant d'éléments que le double du nombre d'arêtes du graphe (sauf pour les boucles qui n'apparaissent qu'une fois), pour faire cela on parcourt aussi la liste de tous les sommets du graphe et pour chacun on effectue un test entre les deux compteurs, d'où $\mathcal{O}(m+n)$.

Correction de l'exercice 9

1. Tous les successeurs du sommet s :
   (a) Matrice d'adjacence A[i,j] :

```
pour j allant de 1 à n faire:
    si A[s,j]=1, afficher(j)
```

Complexité : $\mathcal{O}(n)$ d'après l'algorithme précédent.

On peut améliorer dans la pratique l'algorithme précédent avec une version tenant compte du degré sortant.

```
j <- 1
compteur <- 0
tant que j <= n et compteur < d
   faire si A[s,j] = 1
         alors afficher(j)
               compteur = compteur + 1
         j <- j + 1
```

Malheureusement, la complexité (dans le pire des cas) de ce second algorithme n'est pas améliorée, car même pour d petit, il peut arriver qu'il soit nécessaire de parcourir toute les colonnes de la ligne de s.

(b) Matrice d'incidence B[i,k] :

```
pour e de 1 à m faire:
    si B[s,e]=-1 alors
       pour i allant de 1 à n faire:
           si B[i,e]=1 alors afficher(i)
```

Complexité : $\mathcal{O}(m + n\deg(s)) = \mathcal{O}(m + nd)$. En effet, le premier "si" est exécuté m fois. S'il est faux, son coût est $\mathcal{O}(1)$. S'il est vrai, son coût est $\mathcal{O}(n)$. Le test est vrai $\deg(s) \leq d$ fois, d'où la complexité annoncée.

On peut aussi tenir compte de d pour accélérer la boucle "pour e de 1 à m", mais la complexité ne change pas dans le pire des cas.

```
k <- 1
compteur <- 0
tant que k <= m et compteur < d
   faire si B[s,k] = -1
         alors compteur = compteur + 1
               pour chaque i de 1 à n
                  faire si B[i,k] = 1
                        alors afficher(i)
                              sortir   // sort de la boucle pour i
         k <- k + 1
```

(c) Listes de successeurs :

```
pour tout sommet v de Adj[s] faire:
        afficher(v)
```

Complexité : $\mathcal{O}(\deg(s)) = \mathcal{O}(d) = \mathcal{O}(n)$.

2. Tous les prédécesseurs du sommet s :

(a) Matrice d'adjacence A[i,j] :

Pareil que pour 1(a) : remplacer "A[s,j]" par "A[j,s]". Même complexité.

(b) Matrice d'incidence B[i,k] :

Pareil que pour 1(b) : échanger le "-1" avec le "+1". Même complexité.

(c) Listes de successeurs :

```
pour tout sommet u de V(G) faire:
    pour tout sommet v de Adj[u] faire:
        si v=s, afficher(u)
```

Complexité : $\mathcal{O}(n+m)$. Le coût pour le deuxième "pour" (le plus imbriqué) est $\mathcal{O}(\deg(u))$, proportionnel au degré de u. Attention, cela n'est pas exactement $\deg(u)$. En particulier lorsque $\deg(u)$ est petit, style 0, le coût est au moins une certaine constante (voir les remarques de Exercice 1). La complexité est : $\sum_u \mathcal{O}(\deg(u)) = \mathcal{O}(n+m)$.

On peut améliorer la temps en utilisant d et en sortant dès que possible. Cependant cela n'améliore pas la complexité dans le pire des cas.

```
Pour tout sommet u de V(G) faire :
        compteur <- 0
        pour tout sommet v de Adj(u) faire :
                si v = s alors
                        afficher(u)
                        compteur = compteur + 1
                        sortir    // sort de la boucle Pour tout sommet v
        si compteur = d alors:
                sortir // sort de la boucle Pour tout sommet u
```

3. Déterminer si un graphe est sans boucle :

   (a) Matrice d'adjacence A[i,j] :

```
pour chaque i de 1 à n
    faire si A[i,i] = 1
          alors retourner FAUX
retourner VRAI
```

   Complexité : $\mathcal{O}(n)$.

   (b) Matrice d'incidence B[i,k] :
   dans la matrice d'incidence, une boucle est caractérisée par une colonne à 0. Par contre, on ne peut pas savoir sur quel sommet est positionnée cette boucle.

```
Pour j de 1 à m faire:
      boucle <- vrai
      pour i de 1 à n faire:
          si B[i,j] != 0 alors
              boucle <- faux
              sortir // sortie de la boucle i
      si boucle alors:
          retourner FAUX
retourner VRAI
```

   (c) Listes de successeurs :

```
pour tout sommet v de V(G) faire:
    pour tout sommet u de Adj[v] faire:
        si u = v alors
            retourner FAUX
retourner VRAI
```

   Complexité : $\sum_u \mathcal{O}(\deg(i)) = \mathcal{O}(n+m)$.