Feuille de TD N^o 1 : Généralités

- *2020 : fait en cours : exos 1, 2, 4*

1 Combinatoire

— Quel est le nombre d'arêtes d'un graphe non-orienté complet d'ordre n ?
Quel est le nombre d'arcs d'un graphe orienté complet d'ordre n ?
- *Il y a autant d'arêtes dans le graphe complet non orienté que de façons de choisir deux sommets distincts : c'est $\frac{n(n-1)}{2}$, ou $\binom{n}{2}$.*
Il y a autant d'arcs dans le graphe complet non orienté que de façons de choisir d'abord le point d'origine de l'arc puis son point d'arrivée, donc $n(n-1)$ arcs si on ne met pas de boucles et n^2 sinon.

— $|S_1|=n_1$, $|S_2|=n_2$. Quel est le nombre d'arêtes du graphe non-orienté (S_1,S_2) biparti complet ?
- *Il faut choisir un point de S_1 et un point de S_2 pour chaque arête : il y en a donc $n_1 * n_2$.*

— (maison) Montrer que si un graphe biparti $G(S_1, S_2, A)$ est k-régulier, avec $k > 0$, alors $|S_1| = |S_2|$.
Indication, comptez quelque chose de deux manières différentes et dire que le résultat est le même.
- *Chaque arête compte exactement une extrémité dans l'ensemble S_1 ; or il y a $k|S_1|$ extrémités d'arêtes dans S_1 car G est k-régulier, donc $|A| = k|S_1|$.*
Or par le même raisonnement, $|A| = k|S_2|$. Donc $k|S_1| = k|S_2|$ et comme $k \neq 0$, $|S_1| = |S_2|$.

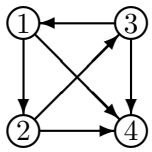
— (maison) G est un graphe non-orienté complet à 6 sommets, dont les arêtes sont coloriées, soit en bleu, soit en rouge. Montrez que G contient un triangle monochrome.
Indication : il y a des sommets x, $y1$ $y2$ $y3$ tq $(x, y1)$, $(x, y2)$ et $(x, y3)$ sont de la même couleur. Pourquoi ? Et donc ?
- *Soit x un sommet de G, alors x a 5 arêtes qui lui sont adjacentes. Comme elles sont toutes colorées en rouge ou en bleu, au moins trois de ces arêtes sont rouges ou au moins trois de ces arêtes sont bleues. Quitte à échanger les couleurs, on les suppose bleues et on les note xy_1, xy_2 et xy_3.*
Considérons alors les trois arêtes y_1y_2, y_2y_3 et y_3y_1. Soit elles sont toutes les trois rouges, et dans ce cas $y_1y_2y_3$ forment un triangle monochrome rouge, soit au moins une des trois est bleue (par exemple y_1y_2) et elle referme un triangle monochrome bleu (dans l'exemple c'est xy_1y_2).
Donc dans tous les cas G contient un triangle monochrome.

— Soit $d_1, d_2, \ldots, d_n$ les degrés des sommets d'un graphe G. Montrez que $\sum_{i=1}^{n} d_i = 2m$.
- *On remarque que le degré d'un sommet correspond au nombre d'extrémités d'arêtes qui lui sont adjacentes, donc $\sum_{i=1}^{n} d_i$ est le nombre d'extrémités d'arêtes dans tout le graphe G. Or chaque arête possède deux extrémités donc ce nombre est $2m$.*

— Montrez que dans un graphe G, il y a toujours un nombre pair de sommets de degré impair.
- *D'après la question précédente, $\sum_{i=1}^{n} d_i = 2m$ donc en particulier $\sum_{i=1}^{n} d_i$ est pair. Or une somme est paire ssi elle a un nombre pair de termes impairs, donc il y a un nombre pair de sommets de degrés impairs.*
(on peut aussi travailler par récurrence en ajoutant les arêtes une par une)

— (maison) Montrez que tout graphe non-orienté avec au moins deux sommets possède au moins deux sommets de même degré.
Indication : combien y a-t-il de sommets ? de degrés différents possibles ? Et donc ? Résoudre la difficulté qui apparait.
- *Dans un graphe non-orientés à n sommets ($n > 1$), le degré d'un sommet est compris entre 0 et $n-1$. Ça nous donne n degrés différents possibles, donc pour qu'il n'y est pas deux sommets de même degré il faudrait que tous ces degrés soient représentés.*
Or si c'était le cas, on aurait un sommet de degré $n-1$, donc voisin de tous les autres sommets. En particulier, ça signifierait qu'il est voisin du sommet de degré 0 : il y a une contradiction et ce n'est pas possible.
Donc si $n > 1$ il y a au moins deux sommets de même degré.

2 Représentations



— Soit le graphe G suivant :

1. Donnez sa matrice d'adjacence.
   - $$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
2. Representez ce graphe par sa liste d'adjacence (les sommets d'une liste d'adjacence sont rangés consécutivement dans des tableaux).
   - *Liste d'adjacence :* $[2, 4, 3, 4, 1, 4]$
     Tête : $[2, 4, 6, 6]$

— Comparez les deux représentations en terme d'efficacité (espace mémoire, complexité en temps pour savoir si $i \to j$, pour connaître $deg^+(s)$, $deg^-(s)$, pour énumérer tous les successeurs, tous les prédécesseurs d'un sommet s).

- *Espace mémoire :* $\theta(n^2)$ *pour la matrice,* $\theta(n+m)$ *pour les listes d'adjacence, donc les listes sont généralement meilleures.*
  $i \to j$ *:* $\theta(1)$ *pour la matrice et* $O(n)$ *pour les listes, donc la matrice est meilleure.*
  $deg^+(s)$ *:* $\theta(n)$ *pour la matrice et* $\theta(1)$ *pour les listes (pour la plupart des implémentations), donc les listes sont meilleures.*
  $deg^-(s)$ *:* $\theta(n)$ *pour la matrice (lire la colonne s) et* $\theta(m)$ *pour les listes (il faut parcourir toutes les listes d'adjacence), donc la matrice est généralement meilleure.*
  Successeurs : $\theta(n)$ *pour la matrice et* $\theta(deg^+(s))$ *pour les listes, donc les listes sont généralement meilleures.*
  Prédécesseurs : comme pour $\deg^-(s)$.

— M est la matrice d'adjacence de G et K un entier. Que représente M^K ?
Combien de produits de matrices faut-il effectuer pour la calculer ?
(maison) Si G possède N sommets, combien de produits de matrices faut-il pour calculer sa clôture transitive et réflexive (matrice qui aura un 1 en case (x, y) ssi y est accessible depuis x) ? sa clotûre transitive (1 en case (x, y) ssi il y a un chemin non vide de x à y)
Indication : $(1+x)(1+x^2)(1+x^4)(1+x^8)....$ =?

- *La matrice* M^K *a pour coefficient* (i, j) *le nombre de chemins de longueur* K *entre les sommets* i *et* j.
  Pour la calculer rapidement, on procède en deux étapes :
  1. *on calcule d'abord les matrices* M^2, M^4, M^8, ... , $M^{2^{\lfloor \log_2(K) \rfloor}}$, *en faisant* $M^{2^{(k+1)}} = M^{2^k} \times M^{2^k}$ *à chaque étape* $\to \lfloor \log_2(K) \rfloor$ *multiplications*
  2. *soit* $b_{\lfloor \log_2(K) \rfloor} \dots b_1$ *l'écriture en binaire de* K *(avec* $b_i \in \{0, 1\}$*). On sait que* $b_{\lfloor} \log_2(K) \rfloor = 1$ *(premier chiffre non nul). On a alors* $M^K = M^{\sum_{i=1}^{\lfloor \log_2(K) \rfloor} b_i 2^i}$, *donc* $M^K = \prod_{i=1}^{\lfloor \log_2(K) \rfloor} M'_i$ *avec* $M'_i = M^{2^i}$ *si* $b_i = 1$ *et* Id *sinon (i.e* M^K *est le produit des* M^{2^i} *pour les* 2^i *qui apparaissent dans l'écriture de* K *comme somme de puissance de* 2*).*
     Donc à partir de l'écriture en binaire de K *et des* M^{2^i} *calculés précédemment, on peut retrouver* M^K *en* $b(K)-1$ *produits supplémentaires, avec* $b(K)$ *le nombre de 1 dans l'écriture binaire de* K, *donc* $1 \leqslant b(K) \leqslant \lfloor \log_2(K) \rfloor$.

  Ainsi, en tout, on effectue entre $\lfloor \log_2(K) \rfloor$ *et* $2\lfloor \log_2(K) \rfloor$ *multiplications.*

  Pour calculer une clôture réflexive transitive, il suffit de calculer $Id + M + M^2 + \dots + M^k$ *avec* $k \geqslant n$ *(on peut s'arrêter à tout* $k \geqslant n$ *car si il n'y a pas de chemin entre* x *et* y *de longueur* n, *il n'y en a pas du tout) et d'appliquer ensuite à chaque coefficient de la matrice obtenue la fonction qui à 0 associe 0 et à tout autre entier associe 1.*
  Or $Id + M + M^2 + \dots + M^{2^{\lfloor \log_2(K) \rfloor + 1} - 1} = (1+M)(1+M^2)(1+M^4)\dots(1+M^{2^{\lfloor \log_2(K) \rfloor}})$ *et* $2^{\lfloor \log_2(K) \rfloor + 1} - 1 \geqslant n$, *donc par la méthode décrite précédemment, il suffit d'effectuer* $\lfloor \log_2(K) \rfloor$ *produits pour calculer les* M^{2^i}, *puis* $\lfloor \log_2(K) \rfloor$ *produits pour calculer la somme qu'on cherche. On arrive donc à* $2\lfloor \log_2(K) \rfloor$ *produits de matrice.*
  Le même nombre de produits suffit à obtenir la clôture transitive, qu'on retrouve en calculant $M + M^2 + \dots + M^{2^{\lfloor \log_2(K) \rfloor + 1} - 1} = (1+M)(1+M^2)\dots(1+M^{2^{\lfloor \log_2(K) \rfloor}}) - Id$.

— Soit un graphe G, codé par les listes d'adjacence utilisant deux tableaux `Tete` et `Succ` (le tableau `Tete` à n elements donne pour tout sommet l'indice dans `Succ` où finissent ses successeurs).

```
pour y := 1 a n
    pour k := Tete[y-1]+1 a Tete[y]       /* avec gendarme Tete[0]=0 */
        si Succ[k] = x alors
            write(x, 'a pour predecesseur ', y);
        fin_si;
    fin_pour;
fin_pour;
```

1. Qu'affiche l'algorithme ci-dessus ?
   - *L'algorithme affiche tous les prédécesseurs de x.*
2. Quelle est sa complexité ?
   - *L'algorithme parcourt une fois chaque élément de `Succ` et deux fois ceux de `Tete` : on obtient une complexité en $\theta(n+m)$.*
3. Modifiez l'algorithme pour construire les listes de prédécesseurs, en faisant varier x de 1 a n. Quelle est la complexité du nouvel algorithme ?

```
predec <- liste(n, [])
pour x := 1 à n
 pour y := 1 a n
     pour k := Tete[y-1]+1 a Tete[y]       /* avec gendarme Tete[0]=0 */
         si Succ[k] = x alors
             ajouter y à predec[x];
         fin_si;
     fin_pour;
 fin_pour;
fin pour;
renvoyer predec;
```

   - *Ce nouvel algorithme est de complexité $\theta(n(n+m))$.*
4. Donnez un algorithme qui calcule en temps $\theta(m+n)$ les degrés ENTRANTS de tous les sommets et les range dans un tableau DegInf[1..n].

```
DegInf <- liste(n,0)
pour k = 1 à Tete[n]
   DegInf[Succ[k]] = DegInf[Succ[k]] + 1;
fin pour;
renvoyer DegInf;
```

   - *Cet algorithme parcourt la liste `Succ` et, à chaque fois qu'elle rencontre l'élément `x` (qui indique une arête arrivant en `x`), incrémente le compteur `DegInf[x]`.*
5. (maison) Donnez un algorithme qui construit les listes de prédécesseurs en temps $\theta(m+n)$.
   - *On utilise le même principe que précédemment, sauf qu'on garde en mémoire le point de départ des arêtes étudiées pour les mettre dans la liste des prédecesseurs du point d'arrivée :*

```
predec <- tableau(n,file vide);
nb_predec <- tableau(n,0);
pour y := 1 à n faire
   pour k := Tete[y-1]+1 à Tete[y] faire
       ajouter y à predec[Succ[k]];
       nb_predec[Succ[k]]++;
   fin pour;
fin pour;
cpt <- 0;
Pred <- tableau(m,null);
pour y = 1 à n faire
    pour i = 1 à nb_predec[y] faire
        x <- pop(predec[y]);
```

```
            Pred[cpt+i] <- x;
        fin pour;
        cpt <- cpt + nb_predec[y];
    fin pour;
    renvoyer Pred;
```

3 Algorithmes

On supposera que l'on peut faire la liste de tous les sommets en temps $\theta(n)$, et qu'on peut faire la liste des voisins d'un sommet s en un temps $\theta(\text{degré}(s))$.

— Le rayon en u d'un graphe est $max\{$ distance de u à $v \mid v \in S\}$.
Donner un algorithme **linéaire** pour calculer le rayon en u d'un arbre.

— (maison) Le diamètre d'un graphe est $max\{$ distance de u à $v \mid u, v \in S\}$.
Donner un algorithme **linéaire** pour calculer le diamètre d'un arbre.
Indication : remonter deux informations en parallèle lors d'un parcours

- *Pour le rayon en u, l'idée est de calculer la profondeur de l'arbre A enraciné en u.*
Pour ça, on donne un algorithme auquel on donne un sommet de départ x et un sommet v tel que si v est un voisin de x, v n'est pas considéré à l'étape suivante (en fait, on fait un parcours en profondeur qui ne dit pas son nom), et qui calcule récursivement la profondeur de l'arbre qui est la composante connexe de x dans $A \setminus v$, enraciné en x. On applique ensuite cet algorithme à (u, u) (u n'est pas un voisin de lui-même donc on n'évite aucun voisin de u).

```
profondeur(x,v):
    prof <- 0
    pour tout voisin y de x faire
        si y != v alors
            prof <- max(prof, profondeur(y,x)+1)
                /* la profondeur du sous-arbre en v est le max des */
                /* profondeurs des sous-arbres en ses fils + 1 */
        fin si
    fin pour
    renvoyer prof

rayon(u):
    renvoyer profondeur(u,u)
```

- *On sait que le diamètre d'un arbre A, dont u est un sommet et dont les composantes connexes de $A \setminus u$ sont $A_1, \ldots, A_k$, est $max(diametre(A_i), profondeur(A_j) + profondeur(A_l))$, où i est tel que $diam(A_i)$ est maximal et j et l sont deux indices différents tels que $profondeur(A_j)$ est maximale et $profondeur(A_l) = \max_{s \neq j}(profondeur(A_s))$ (par contre on peut avoir $i = j$ ou $i = l$). Ces valeurs correspondent au cas où le plus long chemin possible est entièrement contenu dans un des sous-arbre (A_i) et celui où il passe par le sommet u (et donc commence dans A_j et finit dans A_l).*
Pour calculer efficacement le diamètre de l'arbre par récursivité, on a donc également besoin de faire remonter l'information de la profondeur de chaque sous-arbre.
On utilise donc la même technique que précédemment : l'algorithme ***diam_aux(x,v)*** *renvoie la profondeur et le diamètre du sous-arbre de $A \setminus xv$ contenant x et enraciné en x.*

```
diam_aux(x,v):
    prof <- 0;          /* profondeur du sous-arbre en x */
    diam <- 0;          /* diamètre max d'un fils */
    pmax2 <- 0;         /* profondeur de la feuille la plus profonde qui est dans */
                        /* un autre sous-arbre que la feuille la plus profonde */
    pour tout voisin y de x faire
        si y != v faire
            (p,d) <-  diam_aux(y,x);
            si p+1 > pmax2 alors
                si p+1 > prof alors
```

```
                    pmax2 <- prof;
                    prof <- p+1;
                sinon
                    pmax2 <- p;
                fin si;
            fin si;
            diam <- max(d,diam);
        fin si;
    fin pour;
    diam <- max(diam,prof+pmax2);
    renvoyer (prof,diam);

diametre(A):
    x <- un sommet quelconque de A;
    (p,d) <- diam_aux(x,x);
    renvoyer d;
```

4 Récurrences

On appelle arbre binaire un arbre enraciné ordonné dans lequel chaque noeud a soit 2 fils, soit aucun. On considère la proposition Q suivante :

Q : Dans un arbre binaire de profondeur p, toutes les feuilles sont à la profondeur p.

W propose de montrer Q par récurrence sur la profondeur de l'arbre : C'est vrai si l'arbre est de profondeur 0 (c'est l'arbre réduit à une feuille). Supposons Q vraie pour les arbres de profondeur p. On construit alors un arbre A de profondeur $p+1$ en prenant la racine, en mettant à sa gauche un sous-arbre AG de profondeur p et à sa droite un arbre AD de profondeur p. A est bien de profondeur $p+1$. Par hypothèse de récurrence, toutes les feuilles de AG et de AD sont à profondeur p dans AG et dans AD. Elles sont donc à profondeur $p+1$ dans A. Comme il n'y a pas dans A d'autres feuilles que celles de AG et de AD, le résultat est prouvé.

1. La proposition Q étant clairement fausse, quelle est l'erreur de W ?
2. Que se passe-t-il pour les arbres binaires complets ?

- *On doit prouver que TOUS les arbres de taille p vérifient la propriété. À partir d'arbres de taille p, on en construit des de taille $p+1$ mais on ne se soucie pas de savoir si on les a tous construits. Ici, ce n'est pas le cas.*
  Une bonne preuve, ce n'est pas "prenons un objet de taille p vérifiant HR , ajoutons quelque chose, c'est un objet de taille p+1", mais prenons un objet quelconque de taille $p+1$ et enlevons qqch pour se ramener a dimension p qui verifie HR.
  Ici, soit A de profondeur $p+1$, soient B et C ses sous-arbres, alors un au moins est de profondeur p, mais pas forcement les deux, d'où l'erreur de la démonstration, précisément dans les cas que la construction ne construit pas.
  Le résultat ne sera garanti vrai que pour ceux qui ont été construits d'après le rang d'avant à partir d'objets qui marchent, donc à partir d'objets qui ont été construits a partir d'objets qui ont été construits .... qui ont été construits a partir du rang de base.
  Les bugs se répercutent : SI je construit A à partir de B et C mais que j'ai eu un bug pour B, la récurrence n'a pas validé B et donc je n'ai pas forcement le résultat pour A.
  Les seuls qui sont se construisent correctement depuis le début sont les arbres binaires complets. Ce sont les seuls pour lesquels la propriété est vraie.
  Et on peut prouver qu'ils marchent en disant :
  C'est bon pour un arbre complet de profondeur 0.
  Si A est complet de profondeur $p+1$, alors soient B et C ses deux sous-arbres. On a alors que B et C sont complets et de profondeur p, etc.
  Ici, il n'y a pas de problème, puisque tous les arbres complets de profondeur $p+1$ sont de la forme B – racine – C avec B et C des arbres complets de profondeur p.

5 Arbres

— Montrez qu'un graphe admet un arbre couvrant si et seulement si il est connexe.

- *Sens facile : si un graphe G admet un arbre couvrant, alors soient x et y deux sommets, il existe un chemin entre x et y dans l'arbre (qui est connexe) donc dans G. Donc G est connexe. Réciproque : soit G connexe. Alors tant que G admet au moins un cycle, on peut retirer une arête du cycle sans modifier la connexité de G (puisque tout chemin qui passait par l'arête peut faire un détour par le reste du cycle). Or G possède un nombre fini d'arêtes, donc on ne peut pas continuer à l'infini, donc quand on est obligé de s'arrêter parce qu'il n'y a plus de cycles, c'est qu'on a trouvé un arbre couvrant de G (c'est-à-dire un sous-graphe de G connexe et sans cycle).*

— Montrez que tout arbre d'ordre $n \geq 2$ a au moins 2 sommets pendants. 6 preuves sont possibles.

- 1. *Faire une récurrence en enlevant une arête*
  2. *Faire une récurrence en enlevant un sommet*
  3. *Utiliser $m = n - 1$*
  4. *Partir d'un sommet, avancer sans revenir en arrière*
  5. *prendre x puis y a distance maximale de x*
  6. *prendre x et y réalisant $max_{u,v} d(u, v)$*

  (1) : Par récurrence, soit $n \geqslant 2$, on suppose que c'est vrai pour tout arbre de taille $2 \leqslant k \leqslant n$ (le cas des arbres de taille 2 est trivial). Alors soit A un arbre de taille $n+1$, $s_1 s_2$ une arête de A. On retire $s_1 s_2$ de A, on obtient alors deux arbres plus petits A_1 et A_2 (avec A_i contenant s_i).
  Si A_1 et A_2 sont de taille au moins 2, alors d'après la propriété de récurrence ils ont chacun au moins eux sommets pendant, dont au moins un qui n'est ni s_1 ni s_2, qu'on notera respectivement t_1 (dans A_1) et t_2 dans A_2. Alors quand on relie A_1 et A_2 par $s_1 s_2$ pour former A, on n'ajoute aucune arête adjacente à t_1 ou à t_2, donc ils restent tous deux des sommets pendants et A possède au moins deux sommets pendants.
  Supposons que A_1 soit de taille 1, alors A_2 est de taille au moins 2 car A est de taille au moins 3. Dans ce cas, par hypothèse de récurrence, A_2 possède au moins deux sommets pendants, dont au moins un qui n'est pas s_2, qu'on notera t. Alors t et s_1 sont des sommets pendants de A.

  (2) : Même idée : on retire un sommet, et chaque nouvelle composante connexe contient au moins deux sommets pendants, dont un qui est toujours pendant dans l'arbre original. Si il y a au moins deux composantes connexes c'est bon, sinon c'est que le sommet retiré était lui-même un sommet pendant ; dans tous les cas on en a au moins deux.

  (3) : On rappelle que $\sum d_i = 2m$, donc ici $\sum d_i = 2n-2$. De plus, comme le graphe est connexe et $n > 1$, aucun sommet n'est de degré nul. Donc il existe au moins deux sommets de degré 1, sinon on aurait $\sum d_i \geqslant 2n - 1$.

  (4) : On parcourt l'arbre en passant par les arêtes, sans jamais passer deux fois sur le même sommet et en partant d'un sommet quelconque. Puisque le nombre de sommets est fini, on va forcément être obligé de s'arrêter à un moment.
  Si on ne peut plus avancer quand on arrive en un sommet qu'on nomme s, c'est que tous les voisins ont déjà été parcourus. Supposons que s ait au moins deux voisins distincts v_1 et v_2, alors il existe un chemin C ne passant pas par s entre les deux, puisqu'ils ont été tous deux parcourus avant d'arriver en s. Or le chemin C auquel on ajoute $v_1 s$ et $s v_2$ est un cycle, ce qui n'est pas possible dans un arbre. Donc ss est bien un sommet pendant de l'arbre.
  Pour trouver un deuxième sommet pendant, on recommence le même type de parcours en partant de s : on arrive à quitter s puisque l'arbre est connexe et de taille au moins 2, et on s'arrêtera sur un autre sommet pendant.

  (5) : Soient x et y deux sommets de l'arbre A (de taille au moins 2) tel que x est choisi quelconque et y est à distance maximale de x.
  Soit u le premier sommet sur le chemin P de y à x (le chemin P est unique puisqu'on est dans un arbre). Alors u est un voisin de y. Si y possède un autre voisin $v \neq u$, alors un chemin entre v et x est P auquel on ajoute l'arête vy. Or ce chemin est unique, donc la distance entre x et v est $|P| + 1 = d(x, y) + 1$ donc dans ce cas y n'est pas à distance maximale de x. Donc y a pour unique voisin u et est un sommet pendant.

Pour trouver un deuxième sommet pendant, on considère z un sommet de l'arbre à distance maximale de y, et on prouve qu'il est également pendant et différent de y.

(6) : Même chose que précédemment sauf qu'on prouve en parallèle que x et y sont pendants.