- TD1 : Modélisation de réseaux -

- Probabilités discrètes -

- Exercice 1 - Normalisation -

Calculer c dans chacun des cas pour que $(p_k)_{k\in\mathbb{N}}$ soit la loi de probabilité d'une variable aléatoire $X : \Omega \to \mathbb{N}$:

1. $p_k = c.\frac{\lambda^k}{k!}$ (en sachant que $\sum_{k=0}^{\infty} \frac{\lambda^k}{k!} = e^{\lambda}$)
2. $p_0 = 0$ et pour $k \geq 1$, $p_k = c.(1-p)^{k-1}$, pour $0 < p < 1$. Dans un premier temps, on pourra développer $(1-x)(1+x+x^2+\cdots+x^k)$.
3. $p_0 = 0$ et pour $k \geq 1$, $p_k = c.\frac{1}{(k)^a}$, pour $a > 1$.

- Exercice 2 - Espérance -

Calculer l'espérance de la variable aléatoire X dans chacun des cas suivant :

1. A chaque tirage de 2 dés, X est le gain du jeu suivant : si la somme des dés est 2,3,4,5,10,11 ou 12, on gagne 12 $, sinon, on perd 10 $.
2. X suit une loi binomiale.
3. X suit une loi de Poisson.
4. X suit une loi géométrique.

- Exercice 3 -

Montrer que si $p_n = \lambda/n$ alors la loi binomiale de paramètre p_n tend vers une loi de Poisson de paramètre λ quand n tend vers $+\infty$, en sachant que $(1-\frac{\lambda}{n})^n$ tend vers $e^{-\lambda}$ quand n tend vers $+\infty$.

- Exercice 4 -

1. Montrer la linéarité de l'espérance pour des variables aléatoires $\Omega \to \mathbb{N}$.
2. Prouver l'inégalité de Markov pour une variable aléatoire $X : \Omega \to \mathbb{N}$.
3. Montrer que $\texttt{var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$.
4. En admettant l'inégalité de Markov pour des variables aléatoires $\Omega \to \mathbb{R}^+$, prouver l'inégalité de Tchebychev.

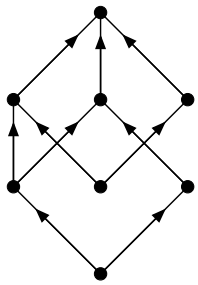
- Structure des graphes orientés -

- Exercice 5 - Graphes orientés acycliques -

Soit D un graphe orienté acyclique (i.e. sans circuit orienté).

1. Montrer que D contient un sommet de degré entrant nul (source) et un de degré sortant nul (puit).

2. Montrer qu'il existe une partition $(X_0, \cdots, X_p)$ de D telle que :
   (a) Pour tout i, X_i est un stable (pas d'arc entre deux éléments de X_i).
   (b) Pour tout arc xy de D avec $x \in X_i$ et $y \in X_j$, on a $i < j$.
   (c) Pour $i > 0$, tout sommet de X_i a un voisin entrant dans X_{i-1}.
3. Calculer cette décomposition sur le cube (orienté) moins deux arcs parallèles :

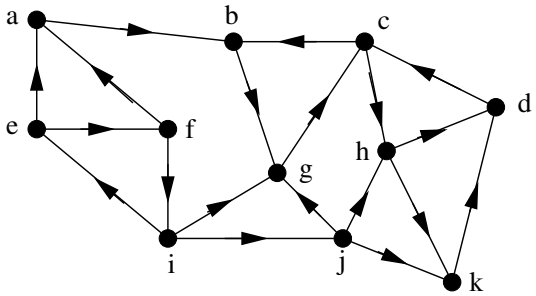


4. Proposer un algorithme linéaire pour calculer cette décomposition.
5. Proposer un algorithme linéaire pour calculer un plus long chemin dans un graphe orienté acyclique.

- Exercice 6 - Composantes fortement connexes -

Soient D un graphe orienté, V son ensemble de sommets et A son ensemble d'arcs. Une *marche* dans D est une suite $P = x_1, x_2, \ldots, x_k$ de sommets de D telle que, pour $i = 1, \ldots, k-1$, $x_i x_{i+1}$ soit un arc de D. Si les sommets x_i sont disjoints, alors P est appelé un *chemin* de D.

1. Soient x et y deux sommets de D, montrer que si il existe une marche de x à y alors, il existe un chemin de x à y.
2. On considère la relation $\mathcal{R}$ définie par $x\mathcal{R}y$ si il existe un chemin de x à y et un chemin de y à x.
   Montrer que $\mathcal{R}$ est une relation d'équivalence. Une classe d'équivalence de $\mathcal{R}$ est appelée une *composante fortement connexe* de D (si D a une seule composante, il est dit *fortement connexe*).
3. Calculer les composantes fortement connexes et le graphe quotient du digraphe ci-dessous.



4. Montrer que le graphe quotient $D_{\mathcal{R}}$ est sans circuit (les sommets de $D_{\mathcal{R}}$ sont les composantes fortement connexes de D, deux sommets sont reliés par un arc si un tel arc existe entre les composantes correspondantes).
5. Comment le nombre de composantes fortement connexes d'un graphe peut-il évoluer lorsqu'on ajoute ou retire un arc ?

- Exercice 7 - Calcul des composantes fortement connexes [Tarjan, 1972] -
Le but de l'exercice est de donner un algorithme rapide de calcul des composantes fortement connexes.

1. Ecrire l'algorithme itératif de parcours en profondeur pour un graphe orienté. Préciser les dates de début d et de fin f de traitement.
2. Rappeler les propriétés d'une arborescence issue d'un parcours en profondeur. Effectuer un tel parcours sur le graphe de l'exercice précédent en partant du sommet b.
3. Proposer un algorithme en temps linéaire pour tester si un graphe orienté est fortement connexe ou non.
4. Montrer que si deux sommets sont dans une même composante fortement connexe alors aucun chemin orienté les liant ne sort de cette composante.
5. Montrer que si deux sommets x et y sont dans une même arborescence lors d'un parcours en profondeur avec $f(x) \leq f(y)$ et si il existe un chemin dans le graphe orienté de x à y, alors il existe un chemin de x à $ppac(x, y)$, l'ancêtre commun de x et y dans leur arborescence commune ayant la plus petite date de fin.
6. Lors d'un parcours en profondeur, montrer que chaque composante fortement connexe forme une sous-arborescence du parcours.
7. Pour un sommet u de D, on note $\alpha(u)$, *l'aïeul de* u le sommet v atteignable depuis u par un chemin et de $f(v)$ maximale.
   (a) Montrer que $\alpha(u)$ est un ancêtre de u dans l'arborescence en profondeur.
   (b) Montrer l'équivalence : "$\alpha(u) = \alpha(v) \Leftrightarrow u$ et v sont dans la même composante fortement connexe".
8. On considère l'algorithme suivant :

Algorithme : CFC
Données : Un graphe orienté $D = (V, A)$.
Résultat : Les composantes fortement connexes de D.
début
Effectuer un parcours en profondeur de D suivant les arcs sortants à partir d'une racine quelconque;
Pour chaque sommet v, on note $f(v)$ la date de fin de traitement de v dans le parcours précédent;
Effectuer un parcours en profondeur de D suivant les arcs entrants, à partir du sommet de valeur f maximale. Au cours du parcours, en cas de choix, on choisit le sommet de valeur f la plus grande;
retourner *Les composantes obtenues lors du dernier parcours*;
fin

   (a) Faire tourner l'algorithme CFC sur le graphe de l'exercice précédent.
   (b) Prouver que la première composante renvoyée par l'algorithme CFC est une composante fortement connexe de D. En déduire qu'il en est de même pour les autres composantes retournées par CFC.
   (c) Préciser le temps d'exécution de l'algorithme.
   (d) Comment obtenir le graphe quotient ?

- Modèles de graphes aléatoires -

	Internet	Actors	Co-auth	Protein	Web
n	228263	392340	16401	2113	325729
m	320149	15038083	29552	2203	1090108
α	2.3	2.2	2.4	2.4	2.3
C	0.06	0.785	0.638	0.153	0.466
C_{rd}	0.00001	0.0002	0.0002	0.001	0.00002
C_{dd}	0.001	0.0057	0.001	0.007	0.017
C_{ab}	0.0002	0.0015	0.003	0	0.0005
C_{ws}	0.06	0.74	0.523	0.06	0.461
d	9.2	3.6	7.18	6.74	7
d_{rd}	11.96	2.97	7.57	10.4	5.47
d_{dd}	5.6	2.95	5.77	5.73	4.48
d_{ab}	7.2	2.93	5.5	8.15	5.1
d_{ws}	15.6	2559	2269	509	11.23

FIGURE 1 – Relevés statistiques sur certains réseaux (d'après *Optimisation et modélisation du graphe du Web*, M.Lyckova et al.)

- Exercice 8 -

Soit G un graphe aléatoire, obtenu selon le modèle d'Erdős-Renyi avec paramètre p. Calculer :

1. la loi de probabilité de la distribution de degrés.
2. le degré moyen espéré.
3. le cœfficient de clustering.
4. la distance moyenne entre deux sommets.

- Exercice 9 -

Soit G un graphe aléatoire, obtenu selon le modèle de Watts et Strogratz sur un graphe de Cayley $C(n, k)$ et avec paramètre p. Calculer :

1. le cœfficient de clustering lorsque $p = 0$.
2. en déduire une borne inférieure sur ce cœfficient dans le cas p quelconque.