Annales des examens et partiels de l'UE 4M017
années 2018-2016

L'examen comporte quatre exercices. Les trois premiers sont connexes.
La présentation générale et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies. Pensez à justifier tous vos résultats.

◇ Préliminaires ◇

(**1-**) L'ordre lexicographique $\prec$ sur les suites finies d'entiers naturels est défini de manière récursive comme suit : $(x_1, x_2, \ldots, x_n) \prec (y_1, y_2, \ldots, y_m)$ si l'une des trois conditions suivantes est vérifiée

(a) $n = 0$ et $m \neq 0$;

(b) $n \neq 0, m \neq 0$ et $x_1 < y_1$;

(c) $n \neq 0, m \neq 0$, $x_1 = y_1$ et $(x_2, \ldots, x_n) \prec (y_2, \ldots, y_m)$

(**2-**) Une **partition** de l'entier naturel non nul n est une suite finie $(a_1, a_2, \ldots, a_k)$ d'entiers naturels non nuls telle que

(a) $a_1 \geq a_2 \geq \ldots \geq a_k$;

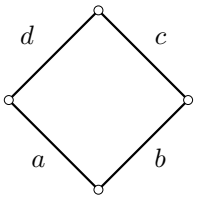
(b) $\sum_i a_i = n$

Par exemple les partitions de 5 sont

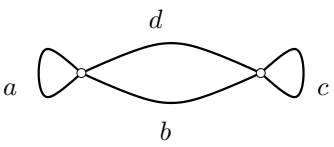
(5)
$(4, 1)$
$(3, 2)$
$(3, 1, 1)$
$(2, 2, 1)$
$(2, 1, 1, 1)$
$(1, 1, 1, 1, 1)$.

Le nombre de chiffres en base 10 du nombre de partitions de l'entier n est de l'ordre de $1.114008620105007861\ldots\sqrt{n}$ tandis que le nombre de chiffres en base 10 du nombre de partitions de l'ensemble $[\![n]\!]$ est de l'ordre de $n \log_{10}(n)$.

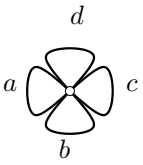
(**3-**) Nous rappelons que si G est un graphe et si F est un sous-ensemble de l'ensemble des arêtes de G alors G/F est le graphe obtenu à partir de G en identifiant les extrémités des arêtes de F. Ainsi



G



$G/\{a, c\}$



$G/\{a, b, c\}$

◇◇

Exercice 1. Donner la liste des partitions de n pour $n = 1, 2, 3$ et 4.

A toute partition π de $[\![n]\!]$, n entier naturel non nul, on associe la partition $\nu(\pi)$ de n dont les termes sont les cardinaux des classes de la partition π.

Expliquer pourquoi l'application ν est bien définie.

Utiliser la liste des partitions de $[\![1]\!], [\![2]\!], [\![3]\!], [\![4]\!]$ donnée dans les notes de cours page 47 pour calculer les cardinaux des $\nu^{-1}(\tau)$ lorsque τ décrit l'ensemble des partitions des entiers $1, 2, 3$ et 4.

Soit

$$\tau = (\underbrace{n_1, n_1, \ldots, n_1}_{a_1 \text{ termes}}, \underbrace{n_2, n_2, \ldots, n_2}_{a_2}, \ldots, \underbrace{n_k, n_k, \ldots, n_k}_{a_k})$$

avec $n_1 > n_2 > \cdots > n_k > 0$ une partition de l'entier n. Exprimer en fonction des n_i et des a_i le cardinal de $\nu^{-1}(\tau)$ [on pourra commencer par le cas $a_1 = a_2 = \ldots = a_k = 1$].

Exercice 2. A toute partition $\tau = (a_i)$ de l'entier n dont au moins un terme est strictement supérieur à 1 on associe la partition $\tau' = (b_i)$ de n définie par

$$(b_i) = (a_1, a_2, \ldots, a_{k-1}) \oplus \underbrace{(a_k - 1, a_k - 1, \ldots, a_k - 1)}_{q+1 \text{ termes}} \oplus \begin{cases} (r) & \text{si } r \neq 0 \\ () & \text{sinon} \end{cases}$$

où $\oplus$ est l'opérateur de concaténation dans l'ensemble des suites finies d'entiers et où

- **(1-)** k est le nombre de termes de la partition τ non égaux à 1;
- **(2-)** q et r sont le quotient et le reste de la division euclidienne de $m + 1$ par $a_k - 1$ où m est le nombre de termes de la partition τ égaux à 1;

Calculer la suite $\tau, \tau', \tau'', \ldots$ pour $\tau = (6)$ [on indiquera les valeurs successives des entiers $k, m + 1, a_k - 1, q$ et r].

Montrer que τ' est bien définie et que τ' est le successeur de τ dans la liste décroissante [pour l'ordre lexicographique] des partitions de l'entier n.

Exercice 3. Soit $m_n = 2^{\binom{n}{2}}$ le nombre de graphes (simples, sans boucles et à renommage près des arêtes) sur l'ensemble $[\![n]\!]$ et soit c_n le nombre de graphes connexes sur l'ensemble $[\![n]\!]$.

Dessiner les graphes sur $[\![n]\!]$ pour $n = 1, 2$ et 3. En déduire les valeurs de c_1, c_2 et c_3.

Montrer que les nombres m_n et c_n sont reliés par les relations

$$m_n = \sum_{\pi} \prod_{B \in \pi} c_{|B|}$$

où π décrit l'ensemble des partitions de $[\![n]\!]$. Expliciter ces relations pour $n = 1, 2, 3$ et 4 sous forme de polynômes en les c_i. En déduire les valeurs de c_1, c_2, c_3 et c_4. Plus généralement déduire de ces relations un algorithme de calcul de c_n. Etudier la complexité de votre algorithme en le nombre cumulé d'opérations élémentaires sur les entiers (addition, mutiplication et division).

Exercice 4. Soit G un graphe connexe muni d'une fonction poids $w : E \to \mathbb{R}$ injective sur l'ensemble E de ses arêtes.

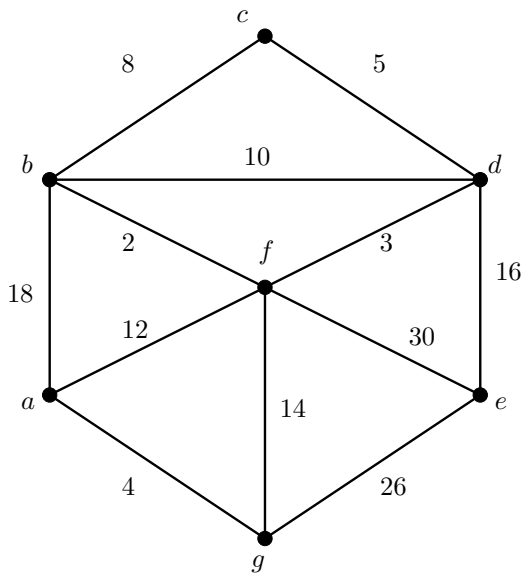
Question 1.– Rappeler succintement pourquoi G admet un unique arbre couvrant minimal et montrer que pour tout sommet v de G l'arête de poids minimal dans l'ensemble des arêtes incidentes à v et qui ne sont pas des boucles appartient à l'ensemble des arêtes de l'arbre couvrant minimal de G.

Pour G graphe connexe muni d'une fonction poids $w : E \to \mathbb{R}$ injective sur l'ensemble E de ses arêtes on note φ_G l'application qui à tout sommet v de G associe l'arête de poids minimal dans l'ensemble des arêtes incidentes à v et qui ne sont pas des boucles.

Soit G un graphe connexe muni d'une fonction poids $w : E \to \mathbb{R}$ injective sur l'ensemble E de ses arêtes. On définit une suite $E_0, E_1, \ldots$ de parties de E comme suit

(**1-**) $E_0 = \varphi_{G_0}(V_0)$ où $G_0 = G$ et V_0 est l'ensemble des sommets de G_0; et
(**2-**) $E_{i+1} = \varphi_{G_{i+1}}(V_{i+1})$ où $G_{i+1} = G_i/E_i$ et V_{i+1} est l'ensemble des sommets de G_{i+1}.

Question 2.– Calculer la suite des G_i, φ_{G_i} et E_i dans le cas où G est le graphe suivant :

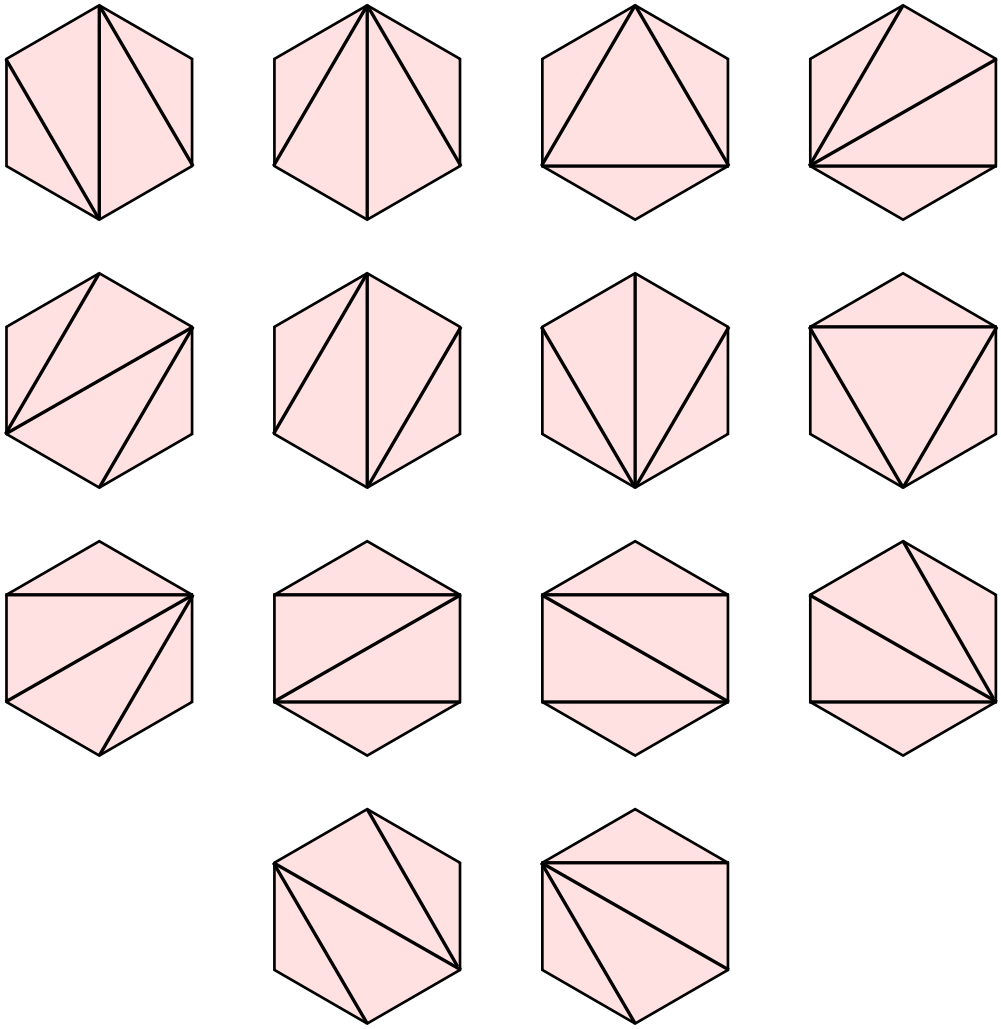


Question 3.– Montrer que l'union des E_i est l'ensemble des arêtes de l'arbre couvrant minimal de G. En déduire un algorithme de calcul de l'arbre couvrant minimal de G et discuter sa complexité. Comparer la complexité de votre algorithme avec celle des algorithmes de Kruskal et de Prim. Comment modifier votre algorithme pour obtenir des complexités comparables? meilleures?

L'examen comporte deux exercices.
La présentation générale et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies. Pensez à justifier tous vos résultats.

◇ Terminologie ◇

Une triangulation d'un polygone convexe à n sommets est une décomposition du polygone en $n-2$ triangles par ajout de $n-3$ diagonales. La figure ci-dessous décline les quatorze triangulations d'un hexagone.



Les sous-arbres **pendants à gauche et à droite** d'un nœud interne ν d'un arbre binaire A sont les sous-arbres gauche et droit du sous-arbre de A de racine ν.

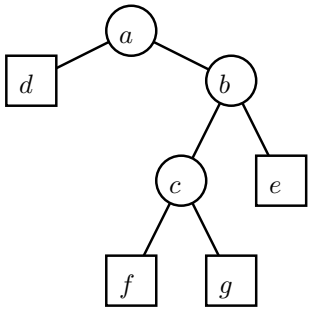
◇◇

Exercice 1. Soit $x_1x_2\dots x_n$ un polygone convexe à n sommets muni d'une fonction poids w sur l'ensemble des triangles $x_ix_jx_k$ engendrés par les triplets x_i, x_j, x_k de sommets du polygone. Le **poids d'une triangulation** est la somme des poids de ses triangles et une **triangulation minimale** est une triangulation de poids minimal dans l'ensemble des triangulations du polygone.

Question 1.– Soit w_{ij}, $1 \le i < j \le n$, $j - i \ge 2$, le poids d'une triangulation minimale du polygone $x_ix_{i+1}\dots x_j$. Montrer que $w_{ij} = \min_{i<k<j}\{w_{ik} + w_{kj} + w(x_ix_jx_k)\}$ où, par convention, $w_{i,i+1} = 0$. En déduire un algorithme de calcul du poids d'une triangulation minimale d'un polygone à n sommets de complexité cubique. Comment faut-il modifier votre algorithme pour obtenir, toujours en temps cubique, une triangulation minimale?

Question 2.– Montrer que le nombre de triangulations d'un polygone convexe est un nombre de Catalan. Lequel ?

Exercice 2. Le **vecteur de Loday** $\mathrm{Ly}(A)$ d'un arbre binaire A de taille n est le n-vecteur à coordonnées entières $(c(\nu_1), c(\nu_2), \dots, c(\nu_n))$ où $\nu_1, \nu_2, \dots, \nu_n$ est la suite croissante pour l'ordre infixe des nœuds internes de A et où l'entier $c(\nu_i)$, appelé la **coordonnée de Loday** de ν_i, est le produit des nombres de nœuds externes des sous-arbres pendants à gauche et à droite de ν_i. Par exemple le vecteur de Loday de l'arbre



est le vecteur $(3, 1, 2)$. Notez que le vecteur de Loday d'un arbre binaire ne dépend que de la classe d'isomorphisme de cet arbre binaire.

Question 1.– Donner une définition récursive du vecteur de Loday d'un arbre binaire. Soit A_n la suite d'arbres binaires définie par $A_0 = \square$ et la relation de récurrence $A_{n+1} = (\circ, A_n, \square)$. Calculer la coordonnée de rang k du vecteur de Loday de A_n en fonction de k. Même question avec la suite d'arbres binaires définie par $A_0 = \square$ et $A_{n+1} = (\circ, A_n, A_n)$. [Indication : commencer par déterminer la taille de A_n puis le rang de la racine de A_n dans la suite croissante pour l'ordre infixe des nœuds internes de A_n et enfin la coordonnée de Loday de la racine de A_n.]

Question 2.– Quel est l'effet d'une rotation sur le vecteur de Loday d'un arbre binaire? En déduire que les vecteurs de Loday des arbres binaires de taille n se situent dans un même hyperplan de $\mathbb{R}^n$.

Question 3.– Soit $(a_1, a_2, \dots, a_n)$ le vecteur de Loday d'un arbre binaire A de taille $n \ge 1$. Soit G^m le chemin de recherche du nœud externe de A minimal pour l'ordre infixe, soient A_i et B_i les sous-arbres pendants à gauche et à droite du nœud interne ν_i de A de chemin de recherche G^{m-i} $(i = 1, \dots, m)$, soient λ_i et μ_i les nombres de nœuds externes de A_i et B_i et, finalement, soit α_i le rang de ν_i dans la suite croissante pour l'ordre infixe des nœuds internes de A. Les assertions suivantes sont-elles exactes? Si oui justifier; sinon corriger.

(**1-**) A_1 est un arbre vide;
(**2-**) $\alpha_1 = 1$;
(**3-**) $a_1 = \lambda_1 \times \mu_1$;
(**4-**) $\alpha_i = \lambda_i$;
(**5-**) $a_{\alpha_i} = \lambda_i \times \mu_i$;
(**6-**) $\lambda_{i+1} = \lambda_i + \mu_i$ avec $\lambda_{m+1} = n + 1$;
(**7-**) $(a_1, a_2, \ldots, a_n) = (a_{\alpha_1}) \oplus C_1 \oplus (a_{\alpha_2}) \oplus C_2 \ldots \oplus (a_{\alpha_m}) \oplus C_m$ où C_i est le vecteur de Loday de B_i et où $\oplus$ est l'opération de concaténation dans l'ensemble des suites finies d'entiers.

Question 4.– En déduire que le vecteur de Loday d'un arbre binaire détermine cet arbre binaire à isomorphisme près ainsi qu'un algorithme prenant en entrée un n-vecteur entier a et retournant 'oui' ou 'non' selon que a est ou n'est pas le vecteur de Loday d'un arbre binaire (on cherchera à obtenir la meilleure complexité possible). Quelle est la complexité de votre algorithme? Décrire le déroulement de cet algorithme sur le vecteur $(1, 2, 12, 1, 2, 3, 21, 1, 2)$? Comment faut-il modifier votre algorithme pour que celui-ci retourne de plus un arbre binaire de vecteur de Loday a si a est le vecteur de Loday d'un arbre binaire? Quelle est la complexité de votre algorithme modifié?

Le partiel comporte 5 exercices.
La présentation générale et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies. Pensez à justifier tous vos résultats.

◇ Préliminaires ◇

(**1-**) Un **alphabet** est un ensemble dont les éléments sont appelés **lettres**.

(**2-**) Le monoide des **mots** sur l'alphabet A est l'ensemble A^* des suites finies $u_1u_2\dots u_n$, $n \in \mathbb{N}$, d'éléments de A, muni de l'opération de concaténation

$$u_1u_2\dots u_n \oplus v_1v_2\dots v_m = w_1w_2\dots w_{n+m}$$

où $w_i = u_i$ si $i \in [\![n]\!]$; v_{n-i} sinon. La concaténation est associative et unifère, d'élément neutre la suite vide, notée ϵ.

(**3-**) Un mot $v \in A^*$ est **facteur gauche** d'un mot $u \in A^*$ si il exite un mot $w \in A^*$ tel que $u = v \oplus w$. Ainsi le mot $u_1u_2\dots u_n$ admet exactement $n+1$ facteurs gauches : $\epsilon, u_1, u_1u_2, u_1u_2u_3, \dots, u_1u_2\dots u_n$.

◇◇

Exercice 1. Montrer que pour tout entier $n \geq 2$

$$\lfloor \lg n \rfloor - \lfloor \lg n - 1 \rfloor = \begin{cases} 1 & \text{si } n \text{ est une puissance de deux;} \\ 0 & \text{sinon.} \end{cases}$$

Tabuler la fonction $\lfloor \lg n \rfloor$ sur l'intervalle $[\![16]\!]$.

Exercice 2. Quel est le nombre d'extensions linéaires de la somme disjointe de k ordres linéaires sur des ensembles de tailles $n_1, n_2, \dots, n_k$?

Exercice 3. Montrer que l'ensemble $C(A)$ des chemins de recherche d'un arbre binaire A vérifie les quatre propriétés suivantes

(**1-**) $\epsilon \in C(A)$;

(**2-**) tout facteur gauche d'un élément de $C(A)$ est un élément de $C(A)$;

(**3-**) pour tout mot X sur l'alphabet $\{G, D\}$ le mot XG appartient à $C(A)$ si et seulement si le mot XD appartient à $C(A)$;

(**4-**) $C(A)$ est fini.

Réciproquement, montrer que tout ensemble C de mots sur l'alphabet $\{G, D\}$ vérifiant les quatre propriétés ci-dessus est l'ensemble des chemins de recherche d'un arbre binaire.

Exercice 4. Une $\star$-arborescence d'ordre n est une arborescence d'ordre n dont les sous-arborescences d'ordre k, $k \in [\![n]\!]$, sont de hauteur $\lfloor \lg k \rfloor$. Une suite d'ajouts et d'additions pondérées est valide si, appliquée à une partition intialement vide, elle produit une $\star$-arborescence.

Les suites

(**1-**) makeset(1)
(**2-**) makeset(1), makeset(2), link(1, 2)
(**3-**) makeset(1), makeset(2), link(1, 2), makeset(3), link(3, 2)
(**4-**) makeset(1), makeset(2), link(1, 2), makeset(3), makeset(4), link(3, 4), link(4, 2)

sont-elles valides ? Montrer de manière constructive que pour tout entier naturel $n \geq 1$ il existe une suite σ valide qui, appliquée à une partition initialement vide, produit une $*$-arborescence $p : E \to E$ d'ordre n. Quelle est la longueur de la suite σ de votre solution à la question précédente? Quels sont les degrés entrants de p (les cardinaux des $p^{-1}(\nu)$, $\nu \in E$)?

Exercice 5. Diverses variantes à la compression des chemins ont été proposées dans la littérature pour implanter de manière efficace l'opération d'identification de la classe d'un élément donné d'une partition. En voici une :

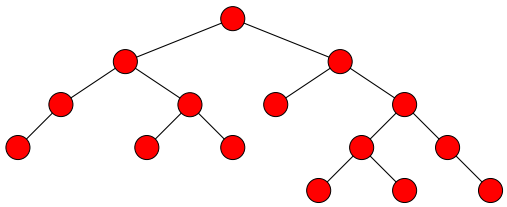
```
element function find (element x);
1.    y := x /* variable auxilliaire initialisée à x */
2.    while P(P(y)) ≠ P(y) do
3.       P(y) := P(P(y));
4.       y := P(y);
5.    end
6.    return P(y)
end find;
```

Quelle modification apporte-elle à l'arborescence de la classe de x?

L'examen comporte 4 exercices. Les exercices 1, 2 et 3 sont connexes. On pourra admettre une question (un exercice) pour traiter les suivantes (les suivants).
La présentation générale et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies. Pensez à justifier tous vos résultats.

◇ Terminologie et rappels ◇

(**1-**) L'**épine gauche** d'un arbre binaire est le chemin de recherche du nœud externe minimal pour l'ordre infixe. L'**épine droite** d'un arbre binaire est le chemin de recherche du nœud externe maximal pour l'ordre infixe. Par exemple les épines gauche et droite de l'arbre binaire suivant (nous n'avons pas indiqué les nœuds externes) :



sont $GGGG$ et $DDDDD$.

(**2-**) Les **ancêtres** d'un nœud ν d'un arbre binaire sont les nœuds de chemins de recherche les $X_1X_2\dots X_j$, $j \in \{0,1,2,\dots,k\}$, où $X_1X_2\dots X_k$, $k \geq 0$, est le chemin de recherche du nœud ν. Notez qu'un nœud est ancêtre de lui-même.

(**3-**) Je rappelle que $\ln n < H_n < 1 + \ln n$ pour $n > 1$.

(**4-**) Je rappelle que si une variable aléatoire Z est la somme de variables de Bernoulli indépendantes alors $\text{Prob}[Z \geq cE[Z]] < e^{-c\ln(c/e)E[Z]}$ pour tout $c > 1$ (borne dite de Chernoff).

◇◇

Exercice 1. Montrer que la suppression d'une clé dans un arbre binaire de recherche requiert d'appliquer à l'arbre un nombre de rotations égal à la somme de la longueur de l'épine droite du sous-arbre gauche du nœud support de la clé et de la longueur de l'épine gauche du sous-arbre droit du nœud support de la clé. (Par exemple, pour l'arbre ci-dessus, la longueur de l'épine droite du sous-arbre gauche de la racine est 3, la longueur de l'épine gauche du sous-arbre droit de la racine est 2 et le nombre de rotations requises pour la suppression de la clé affectée à la racine est 5.) En déduire que pour une arbre binaire de recherche donné la moyenne sur l'ensemble des clés du nombre de rotations requises pour supprimer une clé est majorée par 2.

Exercice 2. Soit A un arbre binaire de recherche sur un ensemble de $n \geq 1$ clés :

$$k_1 < k_2 < \dots < k_n$$

où $<$ est l'ordre total sur les clés. Pour ν et ν' nœud internes de A on note

(**1-**) $D(\nu)$ le nombre d'ancêtres de ν;

(**2-**) $S(\nu)$ la taille du sous-arbre de racine ν;

(**3-**) $P(\nu,\nu')$ la longueur de l'unique chemin dans l'arbre joignant ν à ν';
(**4-**) $SL(\nu)$ la longueur de l'épine droite du sous-arbre gauche de ν et $SR(\nu)$ la longueur de l'épine gauche du sous-arbre droit de ν.

On suppose maintenant que A est obtenu en insérant les clés k_i dans un arbre initialement vide selon l'ordre $k_{\sigma(1)}, k_{\sigma(2)}, \ldots, k_{\sigma(n)}$ où σ est une permutation des entiers de 1 à n, aléatoire pour la loi uniforme sur l'ensemble des permutations des entiers de 1 à n. L'unique entier p_i tel que $k_i = k_{\sigma(p_i)}$ est appelé la priorité de k_i, ainsi $p_i = \sigma^{-1}(i)$. Le nœud support de k_i est noté ν_i.

Question 1.– Dessiner A dans le cas $n = 15$ et $\sigma = [7,9,3,2,8,5,13,6,4,1,14,11,10,15,12]$.

On introduit les variables aléatoires suivantes :

(**1-**) $A_{i,j} = 1$ si ν_i est un ancêtre de ν_j; 0 sinon.
(**2-**) $C_{i,l,m} = 1$ si ν_i est un ancêtre commun à ν_l et ν_m; 0 sinon.

Notez que pour $1 \le l, m \le n$ on a $D(\nu_l) = \sum_{1\le i\le n} A_{i,l}$ et $S(\nu_l) = \sum_{1\le j\le n} A_{l,j}$

Question 2.– Montrer pour $1 \le l, m \le n$, et $l < m$, les égalités suivantes

(**1-**) $P(\nu_l,\nu_m) = 1 + \sum_{1\le i<m}(A_{i,l} - C_{i,l,m}) + \sum_{l<i\le n}(A_{i,m} - C_{i,l,m})$;
(**2-**) $SL(\nu_l) = \sum_{1\le i<l}(A_{i,l-1} - C_{i,l-1,l})$;
(**3-**) $SR(\nu_l) = \sum_{l<i\le n}(A_{i,l+1} - C_{i,l,l+1})$.

Question 3.– Soit $1 \le i, j \le n$. Montrer que ν_i est un ancêtre de ν_j si et seulement si p_i est minimal dans l'ensemble des p_k pour k compris au sens large entre i et j.

Question 4.– Soit $1 \le l, m, i \le n$ avec $l < m$. Montrer que ν_i est un ancêtre commun à x_l et x_m si et seulement si

$$\begin{array}{ll} p_i = \min\{p_k \mid 1 \le k \le m\} & \text{si } 1 \le i \le l, \\ p_i = \min\{p_k \mid l \le k \le m\} & \text{si } l \le i \le m, \\ p_i = \min\{p_k \mid l \le k \le i\} & \text{si } m \le i \le n. \end{array}$$

Question 5.– Montrer que $E[A_{i,j}] = 1/(|i-j|+1)$ et que

$$E[C_{i,l,m}] = \begin{cases} 1/(m-i+1) & \text{si } 1 \le i \le l, \\ 1/(m-l+1) & \text{si } l \le i \le m, \\ 1/(i-l+1) & \text{si } m \le i \le n. \end{cases}$$

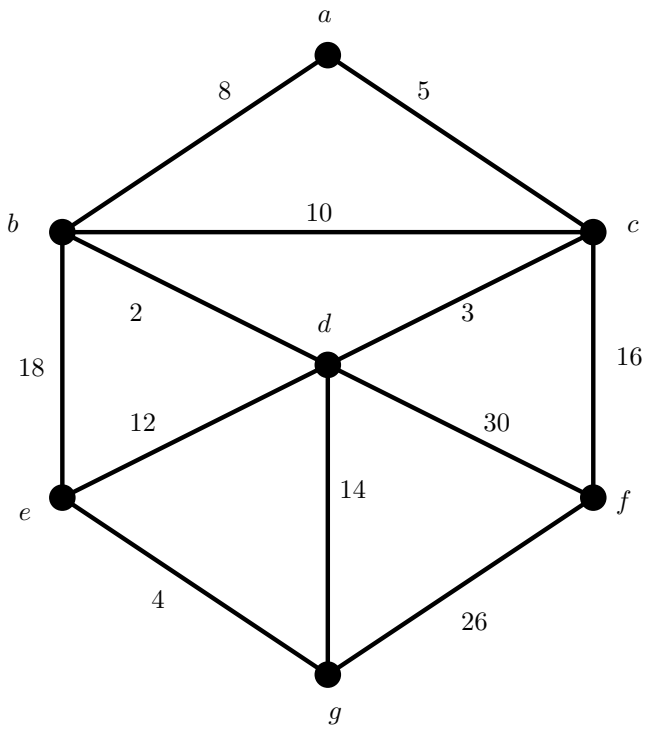
Question 6.– Déduire des questions précédentes que

(**1-**) $E[D(\nu_l)] = H_l + H_{n+1-l} - 1$;
(**2-**) $E[S(\nu_l)] = H_l + H_{n+1-l} - 1$;
(**3-**) $E[P(\nu_l,\nu_m)] = 4H_{m-l+1} - (H_m - H_l) - (H_{n+1-l} - H_{n+1-m}) - 3$;
(**4-**) $E[SL(\nu_l)] = 1 - 1/l$;
(**5-**) $E[SR(\nu_l)] = 1 - 1/(n+1-l)$.

Question 7.– Montrer que la probabilité que $D(x_l)$ soit supérieur ou égal à $1 + 2c\ln n$, $c > 1$, est majorée par $2(n/e)^{-c\ln(c/e)}$. (Indication : commencer par montrer que les $A_{i,l}$ ne sont pas nécessairement indépendantes - puis traiter les cas $l = 1$ et $l = n$ avant de traiter le cas général - penser à utiliser le résultat de l'exercice 1.14.)

Exercice 3. Analyser le nombre de rotations requises pour insérer une nouvelle clé dans un arbre binaire de recherche randomisé.

Exercice 4. Dérouler l'algorithme de Prim sur le graphe suivant en choisissant pour nœud initial le nœud a. Quel est sur cet exemple le nombre d'appels à l'opération **decrement**?



L'examen comporte 5 exercices.
Une très grande attention sera accordée à la qualité de la rédaction.

◇ Préliminaires ◇

(**1-**) Une **bipartition** est une partition de cardinalité 2.

(**2-**) Un graphe G est **biparti** si il existe une bipartition de l'ensemble des sommets de G telle que toute arête de G a une extrémité dans chacune des deux classes de la bipartition.

(**3-**) Dans un graphe un **parcours** de longueur $k \geq 0$ est une suite alternée

$$v_0 e_1 v_1 e_2 v_2 \dots e_k v_k$$

de sommets et d'arêtes, de longueur $2k+1$, où, pour $i \geq 1$, v_{i-1} et v_i sont les extrémités de l'arête e_i. Le sommet v_0 est le **sommet initial** du parcours et v_k est son **sommet terminal.** Le parcours est dit fermé si $v_0 = v_k$.
L'**inverse** du parcours $v_0 e_1 v_1 e_2 v_2 \dots e_k v_k$ est le parcours $v'_0 e'_1 v'_1 e'_2 v'_2 \dots e'_k v'_k$ défini par $v'_i = v_{k-i}$ et $e'_i = e_{k-i+1}$.
Le **concaténé** du parcours $v_0 e_1 v_1 e_2 v_2 \dots e_k v_k$ et du parcours $v'_0 e'_1 v'_1 e'_2 v'_2 \dots e'_{k'} v'_{k'}$, où v'_0 est supposé être égal à v_k, est le parcours $v''_0 e''_1 v''_1 e''_2 v''_2 \dots e''_{k''} v''_{k''}$ de longueur $k'' = k + k'$ défini par

$$v''_i = \begin{cases} v_i & \text{si } 0 \leq i \leq k; \\ v'_{i-k} & \text{sinon,} \end{cases} \qquad \text{et} \qquad e''_i = \begin{cases} e_i & \text{si } 1 \leq i \leq k; \\ e'_{i-k} & \text{sinon.} \end{cases}$$

Un parcours fermé $v_0 e_1 v_1 e_2 v_2 \dots e_k v_k$ est dit **simple** si les v_i, $0 \leq i \leq k-1$, sont distincts deux à deux.

(**4-**) La suite de Fibonnaci $F_0, F_1, F_2, \dots$ est la suite d'entiers naturels définie par les relations $F_0 = 1$, $F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$ pour tout $n \in \mathbb{N}$.

(**5-**) Soit G un graphe simple et sans boucles. Un **couplage** de G est un sous-ensemble d'arêtes de G deux à deux non-adjacentes. [Deux arêtes sont adjacentes si elles ont une extrémité commune.]

(**6-**) La **représentation binaire** de l'entier naturel m est l'unique suite $(\epsilon_i)_{i \in \mathbb{N}}$, $\epsilon_i \in \{0,1\}$, telle que

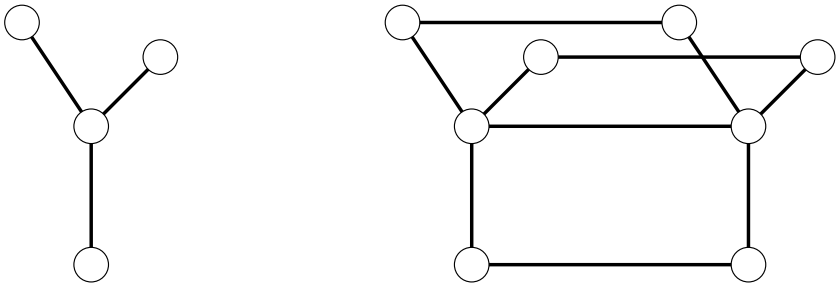
$$m = \sum_{i=0}^{\infty} \epsilon_i 2^i.$$

Ainsi ϵ_i est le reste de la division euclidienne de m_i par 2 où $m_0 = m$ et où m_{i+1} est le quotient de la division euclidienne de m_i par 2.

(**7-**) Un **bit** est un élément de $\{0,1\}$ et une **chaîne** de n bits est une suite de n bits. Notez que le nombre de chaînes de n bits est 2^n.

◇◇

Exercice 1. Soit $G = (V, E)$ un graphe simple sans boucles. Soit $G^\star$ le graphe obtenu à ajoutant à la somme disjointe de G et d'une copie G' de G les arêtes vv' où v décrit V et où v' est la copie de v. Par exemple



G $G^\star$

Montrer que si G est biparti alors $G^\star$ est également biparti.

Exercice 2. "La somme de deux entiers naturels est paire si et seulement si les deux entiers ont la même parité": vrai ou faux? (On ne demande pas de preuve.)

Exercice 3. Quel rapport y a-t-il entre les cycles d'un graphe et ses parcours fermés simples?

Montrer que les cycles d'un graphe G sont de longueurs paires si et seulement si les parcours fermés de G sont de longueurs paires. (Indication : on pourra pour le parcours fermé $v_0e_1v_1e_2 \dots e_kv_k$, $k \geq 1$, considérer une paire ij d'indices qui minimise $j - i$ dans l'ensemble des paires ij, $i < j$, telles que $v_i = v_j$ ... sans oublier de démontrer qu'une telle paire existe.)

Montrer qu'un graphe G d'ordre ≥ 2 est biparti si et seulement si ses cycles sont de longueurs paires. (Indication pour la condition suffisante : on pourra démontrer que deux parcours ayant des extrémités communes ont des longueurs de même parité puis démontrer que la relation binaire "être les extrémités d'un parcours de longueur paire", définie sur l'ensemble des sommets, est une relation d'équivalence dont le nombre de classes d'équivalence est de 2 par composante connexe de G d'ordre ≥ 2.)

Soit G un graphe biparti d'ordre n ayant k composantes connexes. Quel est le nombre de bipartitions de l'ensemble des sommets de G pour lesquelles G est biparti?

Exercice 4. Montrer que le nombre de couplages d'un graphe linéaire d'ordre n est le nombre de Fibonnaci de rang n. Quel est le nombre de couplages d'un graphe cyclique d'ordre $n \geq 3$?

Quel est le nombre d'arêtes d'un couplage maximal d'un graphe cyclique d'ordre $n \geq 3$?

Exercice 5. [**Code de Gray.**] Un **code de Gray** est une liste des 2^n chaînes de n bits dans laquelle deux chaînes consécutives diffèrent exactement d'un bit. Par exemple la liste

$$000, 001, 011, 010, 110, 111, 101, 100$$

est un code de Gray des 8 chaînes de 3 bits. Pour $n = 1, 2, \ldots$, nous définissons une liste $\mathcal{L}_n$ de chaînes de n bits comme suit

- (**1-**) $\mathcal{L}_1 = 0, 1$.
- (**2-**) $\mathcal{L}_{n+1}$ est la concaténation de la liste, notée $0 \oplus \mathcal{L}_n$, obtenue en préfixant d'un bit égal à 0 les chaînes de la liste $\mathcal{L}_n$, et de la liste, notée $1 \oplus \overline{\mathcal{L}}_n$, obtenue en préfixant d'un bit égal à 1 les chaînes de l' inverse $\overline{\mathcal{L}}_n$ de la liste $\mathcal{L}_n$.

Par exemple

$\mathcal{L}_1 = 0, 1$ $\qquad\qquad$ $\overline{\mathcal{L}}_1 = 1, 0$

$\mathcal{L}_2 = 00, 01, 11, 10$ $\qquad\qquad$ $\overline{\mathcal{L}}_2 = 10, 11, 01, 00$

$\mathcal{L}_3 = \underbrace{000, 001, 011, 010}_{0\oplus\mathcal{L}_2}, \underbrace{110, 111, 101, 100}_{1\oplus\overline{\mathcal{L}}_2}$ $\qquad\qquad$ $\overline{\mathcal{L}}_3 = 100, 101, 111, 110, 010, 011, 001, 000$

Question 1.– Montrer que la liste $\mathcal{L}_n$ est un code de Gray.

Question 2.– Soit $\epsilon_0\epsilon_1\epsilon_2\epsilon_3\ldots$ la représentation binaire de l'entier naturel m, et soit $\ldots e_3e_2e_1e_0$ la chaîne de rang m de $\mathcal{L}_n$. [La chaîne de rang m de $\mathcal{L}_n$ est son $m+1$-ième terme. Ainsi la chaîne de rang 0 est $\underbrace{00\ldots0}_{n\text{ “0”}}$, la chaîne de rang 1 est $\underbrace{00\ldots0}_{n-1\text{ “0”}}1$, la chaîne de rang 2 est $\underbrace{00\ldots0}_{n-2\text{ “0”}}11$, etc.] Montrer que

$$e_i = \epsilon_i + \epsilon_{i+1} \bmod 2, \qquad (i = 0, 1, 2, \ldots).$$

Question 3.– Quelle est la chaîne de rang 398 de la liste $\mathcal{L}_n$? Quel est le rang de la chaîne 00100111000 de la liste $\mathcal{L}_{11}$?

La **distance de Hamming** $h(s,t)$ entre deux chaînes $s = s_1s_2\ldots s_n$ et $t = t_1t_2\ldots t_n$ de n bits est le nombre de positions où les bits de s et t diffèrent, c'est-à-dire le cardinal de l'ensemble des i telles que $s_i \neq t_i$. Le rang dans la liste $\mathcal{L}_n$ de la chaîne de n bits s est noté $\rho(s)$. On pose

$$f(m) = \min_{h(s,t)\geq m} \{|\rho(s) - \rho(t)|\} \qquad (m \geq 1).$$

Question 4.– Montrer que $f(m) \geq \lceil 2^m/3 \rceil$. Est-ce la meilleure borne possible?

Le partiel comporte deux exercices.
La présentation générale et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies. Pensez à justifier tous vos résultats.

Exercice 1. Je rappelle que le cœfficient binomial $\binom{n}{p}$ est le nombre de parties à p éléments d'un ensemble à n éléments ($n, p \in \mathbb{N}$).

Question 1.– Démontrer que pour tout $n \geq 0$ et tout $p \geq 0$

$$\binom{n}{p} + \binom{n}{p+1} = \binom{n+1}{p+1}.$$

(On donnera une preuve combinatoire ou une preuve calculatoire.) 2

Question 2.– Quelle est la sortie de la procédure suivante? 2

```
procedure TRIANGLE_DE_PASCAL(var A : array[0..n, 0..n] of integer);
entrée  : un tableau bidimensionnel d'entiers
sortie  :
  var i, j : integer;
  begin
1.  for i := 0 to n do A[i, 0] := 1;
2.  for i := 0 to n do
3.    for j := 1 to n do A[i, j] := 0;
4.  for i := 1 to n do
5.    for j := 1 to i do A[i, j] := A[i − 1, j] + A[i − 1, j − 1];
  end; {TRIANGLE_DE_PASCAL}
```

Quelle est sa complexité exacte en nombre d'additions? en nombre d'affectations? 2

Question 3.– Soit B_n le nombre de partitions de $[\![n]\!]$, $n \in \mathbb{N}$. Quelles sont les valeurs de B_0, B_1, B_2, B_3 et B_4? . . . . . 1 Démontrer la formule de récurrence suivante pour B_n 4

$$B_{n+1} = \sum_{p+q=n} \binom{n}{p} B_q.$$

En déduire les valeurs de B_5 et B_6. 2 Ecrire (ou décrire) une procédure qui prend en entrée un tableau d'entiers de taille $n+1$ et retourne le tableau des B_i, $i \in \{0, 1, \ldots, n\}$. . . . 4 Quelle est la complexité de votre procédure en nombre d'additions et de multiplications? 2 Cette complexité est-elle polynomiale en n? Si non, donner une procédure de complexité polynomiale en n.

Exercice 2. Je rappelle que la suppression d'une clé dans un arbre binaire de recherche s'effectue comme suit : "Dans le cas particulier où le nœud support, appelons-le ν, de la clé à supprimer a pour fils gauche et fils droit, appelons-les ν' et ν'', des nœuds externes la suppression consiste simplement à remplacer le sous-arbre de racine ν par un nœud externe. Le cas général se réduit au cas particulier au moyen d'une séquence de rotations en l'arête $\nu\nu'$ si ν'' est un nœud externe, en l'arête $\nu\nu''$ si ν' est un nœud externe, en l'arête $\nu\nu'$ ou l'arête $\nu\nu''$ (à notre guise) sinon." Ainsi il y a en général plusieurs façons d'effectuer la suppression d'une clé dans un arbre binaire de recherche.

Question 1.– Décliner toutes les façons d'effectuer la suppression de la racine de l'arbre binaire de recherche 3

$$\Psi_2(A, \Psi_2(B, \Psi_2(D, \square, \square), \square), \Psi_2(C, \square, \square)).$$

Question 2.– Montrer que si le sous-arbre gauche/droit du nœud support ν de la clé à supprimer est un arbre vide alors la suppression revient à substituer à l'arbre de racine ν son sous-arbre droit/gauche? (Indication: procéder par induction.) 4

Question 3.– Montrer que la complexité de la suppression d'une clé est un grand-O de la hauteur de l'arbre. 4

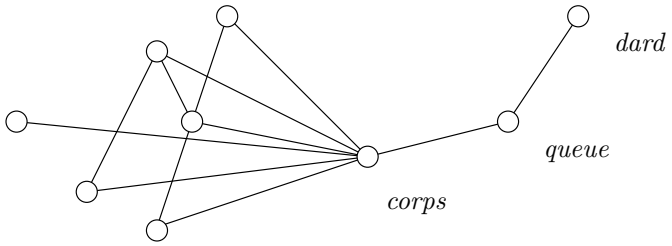
L'examen comporte quatre exercices.
La présentation, la qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies.

Exercice 1. Soit G un graphe d'ordre n à k arêtes dont k' boucles. Quel est le nombre d'orientations de G en fonction de n, k et k'?

Exercice 2. Montrer que le nombre de comparaisons de tout algorithme d'insertion par comparaisons d'un élément dans une liste triée de n éléments est, dans le pire des cas, supérieur ou égal à la partie entière supérieure du logarithme à base deux de $n+1$:

$$\lceil \lg(n+1) \rceil.$$

Exercice 3. Un **scorpion** est un graphe simple dont trois sommets, notés *dard*, *queue* et *corps*, sont distingués et vérifient les conditions suivantes : *queue* est l'unique sommet adjacent à *dard*; *dard* et *corps* sont les uniques sommets adjacents à *queue*; et *corps* est adjacent à tous les sommets excepté *dard* (deux sommets sont adjacents si ils sont les extrémités d'une même arête). Voici le dessin d'un scorpion d'ordre 9 :



Question 1.– Quel est le nombre de scorpions sur un ensemble de cardinalité n, à renomage près des arêtes? Etablir la liste des classes d'isomorphismes de scorpions d'ordres 3, 4, 5 et 6.

Un sommet d'un graphe simple d'ordre n est **singulier** si il est de degré 1, 2 ou $n-2$.

Question 2.– Soit u un sommet singulier dans un scorpion. Montrer qu'il est égal à *dard,* *queue* ou *corps* ou qu'il est de degré ≤ 2 et est adjacent à *corps*.

Soit G un graphe simple d'ordre $n \geq 5$ et soit u un sommet de G. On note B l'ensemble des sommets de G adjacents à u et S le complémentaire de B dans l'ensemble des sommets de G privé de u.

Question 3.– Montrer que la valeur de vérité de l'assertion "G est un scorpion avec u dans le rôle de *corps*" est décidable en effectuant au plus $3n$ tests d'adjacence entre les sommets de G. Plus généralement, montrer que la valeur de vérité de l'assertion "u est singulier et G est un scorpion" est décidable en effectuant au plus cn tests d'adjacence entre les sommets de G où c est une constante que l'on explicitera au mieux.

Question 4.– On suppose que le sommet u n'est pas singulier. Montrer que si G est un scorpion alors la procédure suivante

```
1.  prendre un élément x dans B et un élément y dans S;
2.  repeat
3.    if x et y sont adjacents
4.      then S := S ∖ {y};
5.           if S est non vide
6.             then prendre un nouvel élément y dans S;
7.             else return error ;
8.      else B := B ∖ {x};
9.           if B est non vide
10.            then prendre un nouvel élément x dans B;
11.            else return y ;
12. until 0 = 1;
```

termine et retourne *dard*.

Question 5.– Montrer que la valeur de vérité de l'assertion "G est un scorpion" est décidable en effectuant au plus cn tests d'adjacence entre les sommets de G où c est une constante que l'on explicitera au mieux.

Exercice 4. L'exercice porte sur l'analyse de complexité d'une variante à l'implémentation de la structure de données union-find donnée en cours. Les procédures makeset et link sont remplacées par les procédures suivantes :

```
procedure makeset (element x);
1.      P(x) := x; R(x) := 0;
end makeset;

element function link (element x, y);
1.      if R(x) > R(y) then échanger le rôle de x et de y;
2.      if R(x) = R(y) then R(y) := R(y) + 1;
3.      P(x) := y;
4.      return y
end link;
```

où, comme on le voit, le choix de l'élément distingué de l'addition ensembliste de deux classes n'est plus guidé par la comparaison des cardinalités des classes additionnées mais par la comparaison des hauteurs des arborescences, représentatives des classes additionnées, obtenues sans la compression des chemins. La procédure find, quant à elle, n'est pas modifiée.

L'entier $\mathcal{R}(x)$ est appelé le rang de l'élément x.

Question 1.– Dessiner les arborescences associées à la suite d'opérations

makeset(1)	makeset(2)	makeset(3)	makeset(4)	makeset(5)	makeset(6)	makeset(7)
makeset(8)	link(1, 2)	link(4, 3)	link(2, 3)	link(5, 6)	link(8, 7)	link(6, 7)
link(3, 7)	find(1)					

en étiquetant chaque nœud de l'élément et du rang de l'élément qu'il représente.

Question 2.– Montrer que pour tout élément x, $\mathcal{R}(x) \leq \mathcal{R}(\mathcal{P}(x))$ avec égalité si et seulement si $x = \mathcal{P}(x)$.

Question 3.– Montrer que cardinal de toute classe est au moins $2^{\mathcal{R}(x)}$ où x est l'élément distingué de la classe. En déduire que le rang de tout élément est au plus logarithmique en le nombre total d'éléments.

Question 4.– Montrer que pour tout entier $k \geq 0$ le nombre d'éléments de rang k est au plus $n/2^k$ où n est le nombre total d'éléments.

Question 5.– La borne démontrée en cours sur le coût d'une suite mixte d'ajouts, d'additions pondérées et d'identifications avec compression des chemins, appliquée à une partition pointée initialement vide est-elle toujours valable dans ce contexte modifié? Justifier votre réponse.

L'examen comporte 4 exercices.

Exercice 1. Expliquer le déroulement de la procédure OPERATEUR suivante :

```
    procedure OPERATEUR(var A : array[1..n] of integer);
entrée    : un tableau de n entiers compris entre 0 et n − 1
sortie    :
    var i, j, k : integer;
        B : array[0..n − 1] of integer;
        D : array[1..n] of integer;
    begin
1.  for i := 0 to n − 1 do B[i] := 0;
2.  for i := 1 to n do B[A[i]] := B[A[i]] + 1;
3.  k := 0;
5.  for i := 0 to n − 1 do
    begin
6.    for j := 1 to B[i] do D[k + j] := i;
7.    k := k + B[i];
    end;
8.  for i := 1 to n do A[i] := D[i];
    end; {OPERATEUR}
```

Quelle est sa complexité asymptotique en nombre d'affectations (:=)?

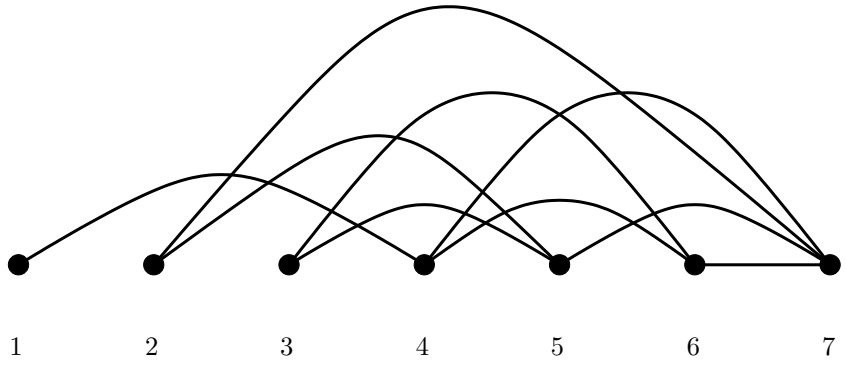
Exercice 2. Soit $X = \{a, b\}$ un alphabet à deux lettres dont l'une est distinguée, disons a. La **valence** d'un mot de X^* est la somme des valences des lettres qui le composent, la valence de la lettre b étant égale à 1 et la valence de la lettre a étant égale à -1. Soit W l'ensemble des mots sur X de valence 1 dont tout facteur droit non vide est de valence ≥ 1. Montrer que W est l'ensemble des mots de Łukasiewicz sur X.

Exercice 3. Soit G un graphe connexe muni d'une valuation $w : E \to \mathbb{R}$ injective sur l'ensemble E de ses arêtes. Montrer que G admet un unique arbre couvrant de valuation minimale.

Exercice 4. Soit $\mathcal{G}_n$ l'ensemble des graphes simples sur l'ensemble des entiers naturels compris entre 1 et n, $n \geq 1$. On appelle **score** de $G \in \mathcal{G}_n$ la suite d'entiers $(d_1, d_2, \ldots, d_n)$ où d_i est le degré du sommet i dans le graphe G.

L'objet de l'exercice est de décrire un algorithme de complexité polynomiale pour tester si une suite finie d'entiers naturels est le score d'un graphe.

Question 1.– Quel est le score du graphe complet K_n? du graphe biparti complet de bipartition $\{1, 2, \ldots, k\} \cup \{k+1, k+2, \ldots, n\}$? du graphe suivant



Soit $D = (d_1, d_2, \ldots, d_n)$ une suite de $n > 1$ entiers naturels majorée par $n-1$, et soit D' la suite $(d'_1, d'_2, \ldots d'_{n-1})$ définie par

$$d'_i = \begin{cases} d_i - 1 & \text{pour } n - d_n \leq i \leq n-1 \\ d_i & \text{pour } 1 \leq i < n - d_n. \end{cases}$$

Question 2.– Montrer que si D' est le score d'un graphe alors D est aussi le score d'un graphe.

Question 3.– Soit $\mathcal{G}_n(D)$ l'ensemble des $G \in \mathcal{G}_n$ de score D. On suppose que la suite des d_i est croissante. Montrer que si $\mathcal{G}_n(D)$ est non vide il existe un graphe $H \in \mathcal{G}_n(D)$ dont les sommets adjacents à n sont les sommets i tels que $n - d_n \leq i \leq n-1$. (Indication : pour tout graphe $G \in \mathcal{G}_n(D)$ introduire l'entier j_G comme étant le plus grand des entiers j compris entre 1 et $n-1$ non adjacent à n; montrer, sous une hypothèse adéquate à expliciter, l'existence d'un entier $i < j_G$ et d'un entier $k \neq i$ tels que

(**1-**) i est adjacent à n;
(**2-**) k est adjacent à j_G et non adjacent à i;

puis considérer le graphe obtenu à partir de G en supprimant les arêtes kj_G et in et en ajoutant les arêtes ik et $j_G n$.)

Question 4.– Montrer sous l'hypothèse "la suite des d_i est croissante" que si D est le score d'un graphe alors D' est aussi le score d'un graphe.

Question 5.– Existe-t-il un graphe de score $(1, 1, 1, 2, 2, 3, 4, 5, 5)$? Si oui exhiber un tel graphe.

Question 6.– Déduire des questions précédentes un algorithme de complexité polynomiale pour tester si une suite finie d'entiers naturels (pas nécessairement croissante) est un score de graphe. Quelle est la complexité de votre algorithme? Modifier votre algorithme pour exhiber un graphe de score la suite donnée, dans le cas où un tel graphe existe.

Le partiel comporte cinq exercices.
La présentation générale et la précision des raisonnements seront prises en compte de manière essentielle dans l'appréciation des copies. Pensez à justifier tous vos résultats.

Exercice 1. Rappeler la définition de la partie entière supérieure et de la partie entière inférieure d'un réel.

Montrer que $-\lfloor x \rfloor = \lceil -x \rceil$ pour tout réel x.

La fonction partie entière inférieure est notée f.

Montrer que $f(f(x)/n) = f(x/n)$ pour tout réel x et tout entier naturel non nul n (on pourra poser $a = f(x)$ et introduire le quotient q et le reste r de la division euclidienne de a par n).

Exercice 2. [**Conjugués d'un mot.**] Soit w un mot de longueur $n \geq 1$ sur l'alphabet X.

Question 1.– Quel est le nombre de facteurs gauches de w? le nombre de facteurs gauches propres de w?

Soit φ l'application de l'ensemble des facteurs gauches propres de w dans l'ensemble des mots sur X qui associe au facteur gauche propre u de w le mot vu où v est le facteur droit de w tel que $w = uv$.

Question 2.– Tabuler φ dans le cas où $w = abaaba$ avec $a, b \in X$.

Question 3.– Montrer que si φ n'est pas injective alors il existe des mots non vides a et b tels que $w = ab = ba$.

Exercice 3. Quel est le nombre de paires non ordonnées d'un ensemble de n éléments? (Votre réponse doit donner 3 dans le cas $n = 2$ car les paires non ordonnées de l'ensemble $\{a, b\}$ sont aa, bb et $ab (= ba)$.)

Exercice 4. [] Etant donné une première procédure qui retourne un objet aléatoire pour la loi uniforme d'un ensemble donné de n objets, et une deuxième procédure qui retourne "pile" ou "face" avec probabilités respectives de succès p et $1-p$, nous définissons une troisième procédure qui retourne une paire aléatoire non ordonnée d'objets en procédant comme suit: tirer un premier objet aléatoire ω en appelant la première procédure puis tirer à pile ou face en appelant la deuxième procédure : si pile sort retourner la paire non ordonnée $\omega\omega$; sinon, i.e., si face sort, tirer un deuxième objet aléatoire ω' en appelant la première procédure (attention ω' peut très bien être égal à ω) et retourner la paire non ordonnée $\omega\omega' (= \omega'\omega)$.

Question 1.– Soient ω et ω' deux objets distincts. Quelle est la probabilité pour la troisième procédure de retourner la paire non ordonnée $\omega\omega$? la paire non ordonnée $\omega\omega'$?

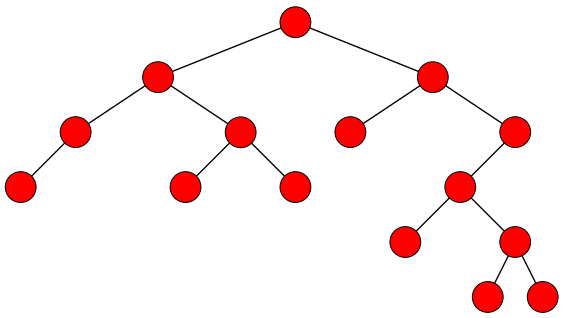
Question 2.– Pour quelle(s) valeur(s) du paramètre p la probabilité sur l'ensemble des paires non ordonnées d'objets définie par la troisième procédure est-elle uniforme?

Exercice 5. [] Soit T un arbre binaire de taille n. On note $(\nu_1, \nu_2, \ldots, \nu_n)$ la suite croissante des nœuds internes de T pour l'ordre infixe et on introduit le vecteur d'entiers

$$w(T) = (w_1, w_2, \ldots, w_n)$$

où w_k est le nombre de nœuds externes du sous-arbre gauche du sous-arbre de T de racine ν_k, le *poids de rang k de l'arbre* en abrégé. On notera que $w(T)$ ne dépend que de la classe d'isomorphisme de T.

Question 1.– Quel est le vecteur ainsi associé à un arbre vide? à l'arbre



Etant donné une paire (A, B) d'arbres binaires on note $A \oplus B$ l'arbre binaire obtenu en substituant au nœud externe de A maximal pour l'ordre infixe, l'arbre $(\circ, B, \square)$. En particulier $\square \oplus B = (\circ, B, \square)$.

Question 2.– Quelle relation lie $w(A), w(B)$, $w(A \oplus B)$ et la taille de B?

Question 3.– Le vecteur $w(T)$ détermine-t-il T à isomorphisme près?

Question 4.– Donner un algorithme qui prend en entrée un tableau de n entiers et donne en sortie le message Erreur s'il n'existe pas d'arbre binaire dont le poids de rang k, pour k compris entre 1 et n, est l'entier de rang k du tableau; un tel arbre sinon. Discuter la complexité de votre algorithme. Votre algorithme est-il optimal? Donner un algorithme optimal.
