Fiche d'exercices *II*: Listes, Arbres et Graphes

Exercice 1 (Liste chaînée) *Construire un programme Pascal qui lit n entiers en lecture et construit une liste chaînée de ces entiers. Procédure récursive puis itérative. Comparer vos deux programmes.*

Exercice 2 (Liste doublement chaînée) *Implanter les opérations* Ajouter, Retirer *et* Premier *en utilisant les listes doublement chaînées.*

Exercice 3 (Opérations sur les polynômes) *Définir un type mônome puis polynôme. Ordonner le polynôme par mônome croissant dans une liste chaînée. Implanter les algorithmes suivants:*

- *Addition:* $P_1 + P_2$.
- *Multiplication:* $P_1 \times P_2$.
- *Dérivation:* ∂P.
- *Intégration:* $\int P$.

Exercice 4 (Incrément) *Coder un nombre en base b par une liste chaînée et donnez un algorithme pour incrémenter.*

Exercice 5 (Test d'égalité d'arbres) *Donner un algorithme pour tester si deux arbres* $\mathcal{T}_1$ *et* $\mathcal{T}_2$ *sont égaux ou pas (égalité physique et logique).*

Exercice 6 (Hauteur d'un arbre) *Écrire une fonction récursive puis itératique qui donne la hauteur d'un arbre* $\mathcal{T}$.

Exercice 7 (Compter les nœuds) *Donner un algorithme qui compte le nombre de nœuds internes, de feuilles, d'un arbre binaire* $\mathcal{T}$.

Exercice 8 (Nombre de nœuds) *Prouver que dans un arbre binaire le nombre maximal de nœuds à hauteur k est* 2^k. *Donner le nombre maximal de nœuds d'un arbre binaire de hauteur h.*

Exercice 9 (Inégalité) *Soit* $\mathcal{T}$ *un arbre binaire,* f *le nombre de feuilles et* n_i *le nombre de nœuds internes. Prouver l'inégalité* $f \leq n_i + 1$. *Quand a-t-on égalité?*

Exercice 10 (Cheminement interne/externe) *Soit $\mathcal{T}$ un arbre binaire. On définit la longueur de cheminement externe par*

$$\mathbf{LCE}(\mathcal{T}) = \sum_{x \text{ feuille de } \mathcal{T}} h(x),$$

et la longueur de cheminement interne par

$$\mathbf{LCI}(\mathcal{T}) = \sum_{x \text{ nœud interne de } \mathcal{T}} h(x).$$

Montrer que si $\mathcal{T}$ est strictement binaire alors $\mathbf{LCE}(\mathcal{T}) = \mathbf{LCI}(\mathcal{T}) + 2n_i$ où n_i est le nombre de nœuds internes.

Exercice 11 *Démontrer que l'ordre des feuilles lors d'un parcours prefixé, infixé et postfixé est le même. Quels sont alors les avantages de ces différents parcours.*

Exercice 12 (Arbre binaire parfait) *Un* arbre binaire parfait *est un arbre binaire dont les feuilles se situent sur deux niveaux: le dernier et l'avant dernier.*

1. *Montrer que si les nœuds d'un arbre binaire parfait sont numérotés de gauche à droite et en descendant (ordre hiérarchique total) alors un nœud i à son fils gauche $2i$ et son fils droit $2i+1$.*

2. *Comment stocker un arbre binaire parfait dans un tableau de taille n?*

Exercice 13 (Arbre partiellement ordonné) *On appelle* arbre partiellement ordonné *un arbre tel que la valeur associée à tout nœud est inférieure ou égale à celle de ses fils. Un* **tas** *est un arbre partiellement ordonné. Donner un algorithme pour l'adjonction d'une feuille et la suppression de la racine d'un tas qui conserve la propriété de tas. On pourra utiliser comme structure de données pour le tas un tableau. Quelle est la complexité des opérations.*

Exercice 14 (Mots de Dick) *Associons à x l'action empiler et à $\overline{x}$ l'action dépiler. On part d'une pile vide et on lit le mot de gauche à droite en effectuant les opérations associées aux lettres. Montrer que l'on peut ainsi reconnaitre si un mot m est un mot de Dick. En déduire que tout historique de la pile terminant par une pile vide est un mot de Dick. Notons d_i le nombre de mots de Dick de longueur $2i$. Prouver l'égalité $d_{i+1} = \sum_{k=0}^{i} c_k c_{i-k}$. En déduire que d_n est le n-^{e} nombre de Catalan. Donner une analogie entre le nombre de mots de Dick de longueur $2n$ et le nombre d'arbres binaires à n nœuds.*

Exercice 15 (Stockage sur fichier) *On désire stocker un arbre sur un fichier et reconstruire l'arbre à partir du fichier. Chaque enregistrement dispose de trois champs: la valeur courante du nœud (ou étiquette) et deux champs booléens indiquant la présence du fils gauche ou droit. On effectue un parcours prefixé de l'arbre. Donner les algorithmes correspondants.*

Exercice 16 (Parcours en largeur/profondeur) *Donner un graphe et ses parcours en largeur et profondeur.*

Exercice 17 (Algorithme itératif de parcours) *Donner un algorithme itératif pour parcourir un graphe en profondeur en utilisant une pile de sommets.*

Exercice 18 (Graphe planaire/Triangulation) *Un graphe planaire est un graphe qui admet une représentation plane, c'est-à-dire que les sommets sont représentés par des points et les arêtes par des arcs ne s'intersectant pas (sauf possiblement aux extrêmités). La célébre formule d'Euler est valable pour les graphes planaires connexes; soit s le nombre de sommets, a le nombre d'arêtes et r le nombre de régions, on a:*

$$s - a + r = 2$$

- *Exemplifiez la formule d'Euler et montrer que $r \leq \frac{2}{3}a$, $a \leq 3s - 6$ et $r \leq 2s - 4$.*
- *Si tout sommet est de degré au moins 3 alors on a $s \leq \frac{2}{3}a$, $a \leq 3r - 6$ et $s \leq 2r - 4$.*
- *Si le graphe planaire connexe est une triangulation dont le contour est un triangle, alors $a = 3s - 6$.*

Exercice 19 (Propriétés des graphes) *Soit $\mathcal{G}$ un graphe non orienté à n sommets $S_1, ..., S_n$. Montrer que $\sum_{i=1}^{n} Degré(S_i) \bmod 2 = 0$ et que le nombre de sommets à degré impair est pair.*

Exercice 20 (Formule d'Euler) *Donner une preuve de la formule d'Euler pour les graphes planaires connexes.*

Exercice 21 (Automate à états fini) *Un* automate à états fini *sur un alphabet Σ est un graphe où de chaque sommet partent n arcs valués par des éléments de Σ. On part d'un sommet distingué appelé* état initial *et on désire réaliser des* transitions *jusqu'à arriver à un* état final *(ensemble d'états finaux). Un mot $m \in \Sigma^*$ est reconnu par l'automate s'il peut être lu par la concaténation des lettres des arcs de transition menant de l'état initial à un état final. Soit $\Sigma = \{'0', ..., '9'\} \cup \{'.'\}$.*

- *Donner un automate reconnaissant les nombres décimaux.*
- *Écrire un algorithme qui prend en entrée une chaîne de caractères et qui indique si cette chaîne est un nombre pair en utilisant un automate.*

Exercice 22 (Fermeture transitive d'un graphe) *On appelle* fermeture transitive *de $\mathcal{G}$ le graphe $\mathcal{G}'$ dont les sommets sont ceux de $\mathcal{G}$ et tel que deux sommets S_i et S_j de $\mathcal{G}'$ sont reliés ssi il existe un chemin de S_i à S_j dans $\mathcal{G}$. Montrer que l'on peut simplement modifier l'algorithme des plus courts chemins de Floyd pour obtenir la fermeture transitive d'un graphe (algorithme de Warshall).*

Exercice 23 (Fermeture transitive via les matrices) *On utilise la matrice d'adjacence M pour coder un graphe $\mathcal{G}$. Montrer que pour tout $k \geq 1$ la matrice $M^k = M^{k-1} \times M = (a_{ij}^{(k)})_{i,j}$ (on pose $M^0 = Id$) peut s'interpreter comme suit: $a_{ij}^{(k)}$ est le nombre de chemins reliant le sommet S_i au sommet S_j composé de k arcs. Réaliser les opérations de multiplication, addition en booléen... En déduire un algorithme pour tester si un graphe est acyclique ou pas.*

Exercice 24 (Voyageur de commerce) *Un problème soulevé par Dijkstra. Soient $V_1, ..., V_n$ un ensemble de n villes. Le voyageur de commerce veut visiter toutes les villes en n'y passant qu'une fois et une seule et revenir à la ville de départ. Soit $d(V_i, V_j)$ la distance entre les villes V_i et V_j. Le voyageur de commerce désire minimiser la distance parcourue. Donner un algorithme et étudier la complexité de celui-ci. Qu'en pensez-vous ($\rightarrow$heuristique)?*

Exercice 25 (Maris Jaloux) *3 couples se trouvent sur une rive et désirent passer sur l'autre rive à l'aide d'une barque ne pouvant contenir que deux personnes. Les maris, extrêmement jaloux, ne veulent pas laisser leur femme avec d'autres hommes sans leur présence. Donner une solution à ce problème en le modélisant à l'aide d'un graphe (la solution est alors un chemin dans ce graphe). Combien y a-t-il de solutions? Donner un algorithme dans le cas général de n couples.*

Exercice 26 (Optimisation de stock) *Chaque mois, une entreprise doit s'approvisionner en matière première de manière à continuer d'assurer sa production. On suppose que les matières premières sont comptées à l'unité et que chaque unité produite par l'entreprise nécessite une unité de matière première. L'entreprise dispose d'une capacité maximale de stockage N. Chaque mois,* **la** *matière première change de prix. Soit p_i le prix de cette matière première au début du mois m_i pour $i \in \{1, ..., n\}$ et soit q_i la quantité de matière à produire pour le mois m_i. Le niveau du stock au début de m_0 est x_0 et on souhaite avoir un niveau x_n en stock à la fin du mois m_n. Déterminer les quantités $a_1, ..., a_n$ de matière première à acheter au début de chaque mois de manière à minimiser le coût total d'achat sur la période. On montrera que le problème revient à calculer un plus court chemin dans un graphe que l'on définira.*

Exercice 27 (Tri topologique) *En utilisant une structure de données par niveaux des graphes acycliques, donner un algorithme donnant une extension linéaire de l'ordre induit par le graphe. Quelle est sa complexité?*

Exercice 28 (Bis repetita!) *Soit $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ un graphe acyclique orienté (DAG) connexe. On considère la procédure suivante:*

procédure *TriTopologique(s*:sommet*);*
var s':sommet*;*
début
$marque(s) \leftarrow vrai$*;*
tant que $\exists s' \in \mathcal{S}$ *tel que* $(s, s') \in \mathcal{A}$
et non marqué(s')**faire**
TriTopologique(s')
AjouterTête$(s, \mathcal{L})$*;*
fin*;*

Initialement $\mathcal{L} = \emptyset$ *et tous les sommets ne sont pas marqués. Montrer que l'algorithme précédent effectue un tri topologique et donner sa complexité. Tenir compte de la non connexité de* $\mathcal{G}$*.*