L2 Informatique

Algorithmique 3 – Feuille de TD n° 1

1 Listes triées (révisions)

On considère des listes d'objets de type `Element`, sur lesquels il existe un ordre total. Une liste est dite triée si tous les éléments sont rangés dans l'ordre croissant. Une liste triée peut contenir des doublons.

1. Écrire un sous-programme qui permet d'ajouter un élément e à une liste triée.
2. Écrire un sous-programme qui permet de supprimer les occurrences d'un élément e d'une liste triée.
3. Écrire un sous-programme qui corrige une liste non nécessairement triée, c'est-à-dire qui supprime tous les éléments strictement inférieurs à un élément précédent dans la liste.
4. Écrire un sous-programme qui fusionne deux listes triées en une liste triée. Les deux listes initiales n'auront pas à être conservées après cette opération.

2 Tableaux de listes – Compression d'images

Une image en 256 couleurs est codée par une matrice de pixels, chaque pixel prenant une valeur entre 0 et 255. On note `NL` le nombre de lignes et `NC` le nombre de colonnes de l'image.

On veut passer de cette représentation matricielle à une représentation par un tableau de listes chaînées. La i-ème variable du tableau représente la i-ème ligne de l'image et contient un pointeur vers la liste des pixels de la ligne. Le stockage d'une image étant onéreux en place mémoire, chaque liste chaînée correspond à une représentation compressée de la ligne obtenue en remplacçant toute suite de n pixels de même couleur (et donc de même valeur v) par un couple (v, n).

Par exemple, la ligne `(5 5 5 5 5 5 5 5 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7)` sera remplacée par `((5, 9) (2, 13) (7, 17))`.

1. Donner les types nécessaires à cette représentation.
2. Écrire un sous-programme permettant de compresser une ligne d'une image, c'est-à-dire permettant de passer d'un tableau de `NC` valeurs à une liste chaînée.
3. Écrire un sous-programme permettant de compresser une image complète.
4. Écrire les sous-programmes permettant de décompresser une ligne et une image.

Questions supplémentaires (travail personnel)

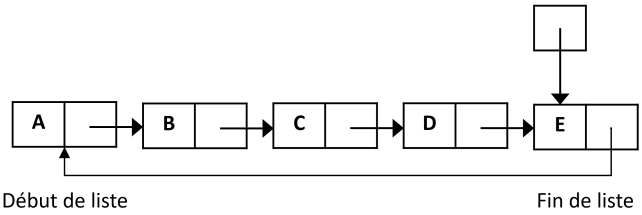
Écrire un programme complet permettant de générer aléatoirement une image (réfléchir à un moyen d'obtenir globalement des zones de pixels identiques, afin que la compression prenne sens), de la compresser et la décompresser, puis afficher le ratio de compression (rapport entre la taille des données représentant l'image non compressée et compressée). Compléter en vérifiant la correspondance de l'image obtenue avec l'image initiale.

3 Piles d'entiers

1. Écrire un sous-programme permettant d'afficher les éléments d'une pile d'entiers.
2. Écrire, avec les opérations sur le type Pile, un sous-programme permettant de copier, dans une nouvelle pile d'entiers Q, tous les nombres pairs d'une pile donnée P. Ces nombres pairs devront être dans Q dans l'ordre où ils figurent dans P et le contenu de la pile P devra être inchangé après exécution de l'algorithme.

4 Listes circulaires, Files

On considère des listes chaînées circulaires : dans la dernière cellule de la liste le pointeur à NULL est remplacé par un pointeur vers la première cellule de la liste. De plus la liste est représentée par un pointeur vers son dernier élement (qui permet facilement de retrouver la tête de la liste).



1. Donner les déclarations de type pour cette structure de liste circulaire. Comment est représentée la liste vide ? Comment est représentée une liste contenant un seul élément ?
2. Écrire un sous-programme qui affiche une liste.
3. Écrire un sous-programme qui calcule la longueur d'une liste. (a) une version itérative (b) une version récursive
4. Expliquer pourquoi cette structure de liste circulaire avec un pointeur sur le dernier élément est bien adaptée pour représenter une file.
5. Donner pour cette structure l'implémentation des primitives sur les files :
   (a) `creer`, qui initialise une file (vide) ;
   (b) `estVide`, qui vérifie si une file est vide ;
   (c) `premier`, qui retourne le premier élément d'une file non vide (sans le supprimer) ;
   (d) `ajouter`, qui ajoute un élément à une file ;
   (e) `retirer`, qui retire un élément d'une file non vide ;
   (f) `vider`, qui retire tous les éléments d'une file.

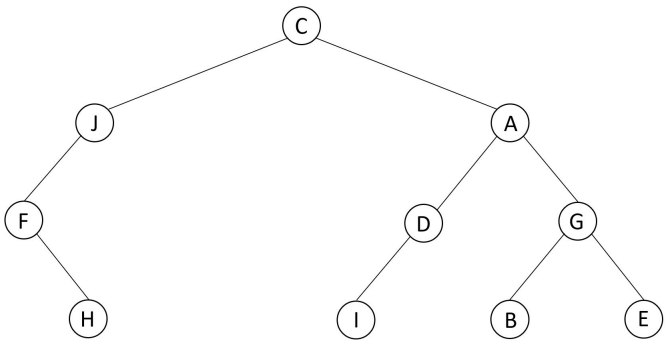
5 Files (autres implémentations)

Une file peut également être implémentée via une liste chaînée et un double pointeur tête-queue, ou bien au moyen d'un tableau circulaire (via une structure constituée d'un tableau d'entiers et de deux indices représentant la tête et la queue de la file).

Rappeler l'implémentation de chacune des deux structures, c'est-à-dire la définition du type `File` et l'écriture des primitives (en y incluant une opération d'affichage) : `creer` ; `estVide` ; `estPleine` (le cas échéant) ; `premier` ; `ajouter` ; `retirer` ; `afficher`.

6 Propriétés des arbres binaires

1. Dessiner tous les arbres binaires non étiquetés de taille 3.
2. Dessiner un arbre filiforme de taille 5. Combien y a-t-il d'arbres filiformes non étiquetés de taille n ?
3. Dessiner un arbre binaire localement complet de taille 9, n'étant ni un peigne ni un arbre parfait.
4. Combien y a-t-il d'arbres complets non étiquetés dont la taille est comprise entre 1 et 100 ? Et d'arbres parfaits ?
5. Quelle est la hauteur minimale et maximale d'un arbre binaire de taille 10 ?
6. Quelle est la hauteur d'un arbre binaire parfait de taille 10 ?
7. Quelle est la taille minimale et maximale d'un arbre binaire de hauteur 10 ?
8. Combien de feuilles possède un arbre binaire complet de hauteur 10 ?
9. Combien de feuilles possède au minimum et au maximum un arbre binaire localement complet de hauteur 10 ?
10. Indiquer l'ordre de traitement des nœuds selon les différents parcours (largeur, profondeur préfixe, infixe et postfixe) de l'arbre suivant :



7 Parcours préfixe itératif

Au moyen d'une **pile**, écrire un algorithme non récursif qui affiche tous les éléments d'un arbre binaire selon un parcours en profondeur d'abord avec un ordre préfixe.