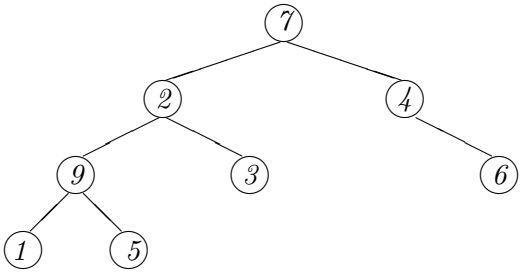
Travaux Dirigés d'informatique n°5

Algorithmique

Tas.

Exercice 1 Chemins de la racine aux feuilles.

On considère un arbre binaire de hauteur inférieure à une constante `MAX`*. Ecrire une fonction* `void Chemin(Arbre T)` *qui permet l'affichage de tous les chemins menant de la racine aux feuilles dans l'ordre d'un parcours en profondeur gauche. Ainsi, pour l'arbre*



on affichera

```
7 2 9 1
7 2 9 5
7 2 3
7 4 6
```

Pour ce faire, on appellera une fonction annexe `void AffCheminAux(Arbre T, int Buffer[], int indice)` *où* `Buffer` *est un tableau de taille* `MAX` *déclaré dans la fonction* `Chemin` *et* `indice` *indique la première case vide de* `Buffer`*.*

Exercice 2 Tas et représentation par un tableau.

1. *Dessiner les arbres binaires parfaits dont le parcours en largeur correspond à ces suites d'entiers:*
   (a) *1, 5, 7, 6, 5, 8, 9, 12, 7, 5, 8, 9*
   (b) *4, 5, 8, 6, 6, 9, 8, 7, 5, 11*
2. *parmi les deux arbres représentés ci-dessus, quels sont ceux qui sont des arbres tournois?*
3. *soit i l'indice d'un tableau qui représente un arbre parfait. Quel est l'indice de son noeud parent?*
4. *soit i l'indice d'un tableau qui représente un arbre parfait. Quels sont les indices de ses fils gauche et fils droit?*
5. *tant donné un tas vide, dessiner le tas et donner le tableau qui le représente après lui avoir ajouté dans l'ordre les valeurs suivantes: 5, 13, 4, 14, 1, 20.*

6. *après lui avoir enlevé sa valeur minimale, dessiner le tas et le tableau qui le représente.*

Pour représenter un tas, on utilise la structure suivante:

```
typedef struct
{
    int taille;
    int arbre[MAX];
} Tas;
```

Exercice 3 Vérification.
Écrire une fonction `int estTas(int tab[], int taille)` *qui vérifie qu'un tableau* `tab` *de taille* `taille` *représente un tas.*

Exercice 4 Fils minimal.
Écrire une fonction `int Fils(Tas T, int indice)` *qui renvoie* -1 *si le noeud représenté par la case* `indice` *n'a pas de fils, sinon renvoie l'indice du noeud fils qui contient la plus petite étiquette.*

Exercice 5 Change.
Écrire une fonction `void Change(Tas *T, int indice, int valeur)` *qui remplace la valeur de la case* `indice` *par* `valeur` *et effectue différents échanges de case afin de respecter la structure de tas. On vérifiera que* `indice` *est inférieur à la taille du tas* `T`*.*

Exercice 6 Ajout.
Écrire une fonction `int Ajout(Tas *T, int valeur)` *qui ajoute une valeur au tas, tout en respectant la structure de tas, on renverra* 1 *si l'ajout est correct et* 0 *sinon.*

Exercice 7 Extrait l'élément minimal.
Écrire une fonction `int ExtraireMin(Tas *T)` *qui extrait la valeur minimum d'un tas et la retourne, tout en respectant la structure d'un tas.*

Exercice 8 Complément.
Reécrire les fonctions précédentes avec la structure suivante

```
typedef struct
{
    int taille;
    int *arbre;
    int Max;
} Tas;
```