Exercice 1 :

1 : Pas quasi parfait
2 : Pas quasi parfait
3 : Pas ordonné
4 : Pas quasi parfait
5 : Pas ordonné
6 : Tas
7 : Tas

Quasi parfait : Tous les niveaux sont remplis et le dernier l'est par la gauche.
Presque parfait : Comme quasi parfait sauf remplissage en bas aléatoire.
Tas : Arbre quasi parfait ordonné (Tous les fils inférieurs ou supérieurs)

Exercice 2 :

1/ Tas d-aire : si la racine est en T[1], si la racine est T[0]

```
def pere(i) :
        return i//d   return (i+1)//d-1
def fils(i, k) :
        return d*i+(k-1)      return d*(i+1)+(k-2)
```

2/ $\sum_{i=0}^{h} d^i = \frac{d^{h+1}-1}{d-1}$

3/ $n \leq \frac{d^{h+1}-1}{d-1}$

$n(d-1) + 1 \leq d^{h+1}$
$\log_d(n(d-1)+1) \leq h+1$
$h = [\log_d(n(d-1)+1)]-1$

4/ Complexité : haut du tas
On met le nouvel élément le plus en bas à gauche puis on l'échange avec son père tant qu'il est plus grand que lui.

5/ Complexité : d h
On supprime la racine est remplace par l'élément le plus bas à droite. Tant qu'un de ses filles est supérieur on l'échange avec son fils max

6/ On gagne en insertion mais pas en extraction avec le tas binaire. Mais comme on a tendance a plus insérer on préfère le tas binaire.

Exercice 3 :

```
def fusion(L):
    M = [range(len(L))]
    sort(M) #trié selon L[i][0]
    A = []
    while L != []:
        A += [L[M[0]][0]]
```

```
    if len(L[M[0]]) == 1 :
        remove(L, M[0])
        M = M[1:]
    else:
        L[M[0]] = L[M[0]][1:]
        M = insert(M[1:], M[0]) # ajout en maintenant l'ordre
     return A
```

On trie les listes et ajoute le premier élément de la bonne liste.