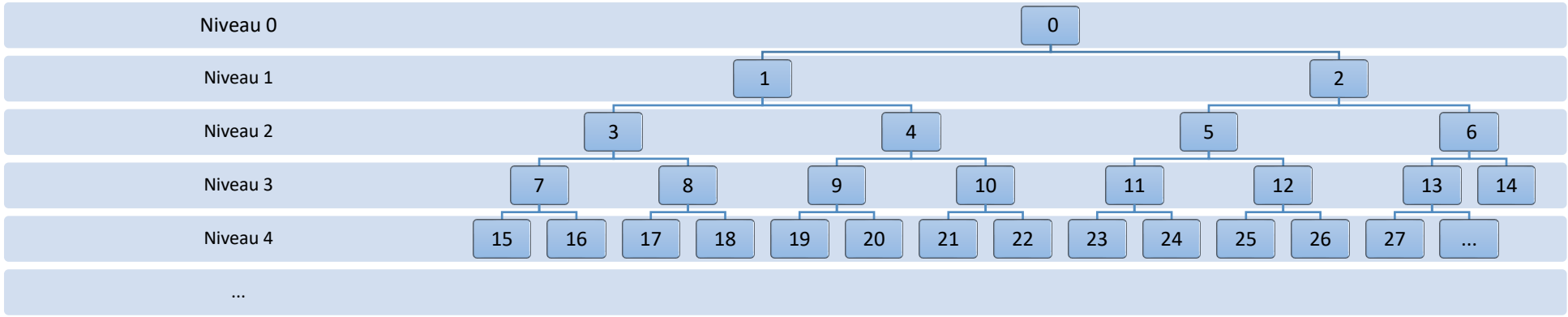
Tas Binaire



30	31	56	60	27	12	73	23	73	30	36
71	13	57	85	52	31	29	61	42	12	53
80	82	16	94	50	49	49	24	67	44	47

File de priorité

Une file de priorités doit permettre de connaître rapidement quelle est la tâche la plus prioritaire parmi un ensemble de tâches. Montrez qu'un simple tableau trié peut répondre rapidement à cette attente.
Quelle est la complexité de l'ajout d'une nouvelle tâche, associée à un niveau de priorité ?
Quelle est la complexité de la suppression de la tâche la plus prioritaire du tableau ?

Tas Binaire

Un tas binaire est une structure de données informatique utilisée pour gérer de manière efficace les files de priorité. Il est également à la base de l'un des algorithmes de tri les plus efficaces de la littérature : le *heapsort*.

Il s'agit d'un arbre binaire. Chaque nœud (sauf la racine) possède un nœud père, et chaque nœud possède au plus deux fils (les feuilles n'ont pas de fils). Il est rempli avec des clefs, niveau par niveau, et de gauche à droite dans le dernier niveau. Seul le dernier niveau peut être incomplet.

La racine de l'arbre (dessinée en haut) porte l'indice 0.

1. Combien de nœuds porte le niveau *k* ?
2. Combien de clefs peuvent être mémorisées dans un arbre ayant *k* niveaux complets ?
3. Quelle est la hauteur *h* de l'arbre (c'est à dire la longueur d'un plus long chemin allant d'une feuille à la racine) correspondant à un tas de *n* clefs ?
4. Quel sont les indices des deux nœuds fils du nœud *i* ?
5. Quel est l'indice du nœud père du nœud *j* ?

Pour gérer efficacement les files de priorité, on va supposer que les clefs correspondent aux priorités des tâches. En particulier, la tâche la plus prioritaire doit pouvoir être déterminée instantanément. Dans un tas-min, la priorité la plus haute correspond à la plus petite valeur de clef. Dans un tas-max, la plus grande priorité correspond à la plus grande valeur de clef. On raisonnera par la suite sur un tas-max, qui doit vérifier la propriété suivante : la clef d'un nœud doit toujours être de valeur inférieure ou égale à celle de son père.

6. Montrez que la valeur maximale de clef se trouve à la racine de l'arbre.
7. Quelle est la complexité de l'opération permettant d'afficher la valeur de la priorité la plus haute ?
8. Etant donné un tas-max, proposez un algorithme permettant d'ajouter une clef au tas, en gardant la propriété du tas-max. Quelle est la complexité de cet algorithme, en fonction de la hauteur de l'arbre *h* ? en fonction du nombre de clefs dans le tas *n* ?
9. Trouvez un algorithme permettant de supprimer du tas la clef ayant la priorité la plus haute, tout en conservant la propriété de tas-max. Quelle est la complexité de cet algorithme, en fonction de la hauteur de l'arbre *h* ? en fonction du nombre de clefs dans le tas *n* ?
10. Proposez un algorithme de tri basé sur un tas binaire. Quelle est sa complexité en temps ? en espace ? pouvez-vous trouver une solution pour trier le tableau « sur place » (sans utilisation d'un tableau annexe) ?