

# Tas, files et plus courts chemins

OPTION INFORMATIQUE - TP n° 3.6 - Olivier Reynet

## À la fin de ce chapitre, je sais :

- ☞ définir un tas-min et un tas-max
- ☞ trier un tableau pas tas
- ☞ utiliser un tas pour créer une file de priorité
- ☞ appliquer les tas à l'algorithme de Dijkstra
- ☞ programmer impérativement en OCaml

## A Implémentation d'un tas max et tri par tas

- A1. Créer une variable `heap_test` de type `Array` contenant les entiers de 1 à 10 dans l'ordre décroissant. On générera automatiquement cette variable (pas in extenso). Est-ce un tas? Si oui, de quel type?
- A2. Écrire une fonction de signature `swap : 'a array -> int -> int -> unit` qui échange la place de deux éléments dans un type `Array`. Tester cette fonction sur le tableau `heap_test`.  
Comme en OCaml les `Array` sont indexés à partir de 0, on choisit de placer la racine du tas en 0. Puis, si le père est  $k$  alors les fils seront en  $2k + 1$  et  $2k + 2$ . Si un fils est en  $k$ , alors le père est en  $\lfloor \frac{k-1}{2} \rfloor$ .  
Il faut noter qu'un tas peut ne pas être rempli. Dans ce cas, on prendra soin de relever l'indice de la première case non remplie.
- A3. Écrire une fonction de signature `up : 'a array -> int -> unit` qui, si cela est possible, fait monter un élément d'après son indice dans le tas tout en préservant la structure du tas.
- A4. Écrire une fonction de signature `down : 'a array -> int -> int -> unit` qui, si cela est possible, fait descendre un élément dans un tas max tout en préservant la structure du tas. Le premier entier de la signature est l'indice de la première case non remplie du tas, le second l'indice de l'élément à faire descendre.

On cherche à créer un tas de deux manière différente :

- soit en considérant que le premier élément du tableau est un tas que l'on fait grossir avec les éléments de droite du tableau,
  - soit en considérant que les feuilles sont des tas à un élément que l'on fait grossir au fur et à mesure avec les éléments de gauche du tableau. On remarque dans ce cas que quelle que soit la taille  $n$  du tas, l'élément d'indice  $n/2$  est toujours la première feuille. On va donc insérer les  $n/2$  premiers éléments dans leurs feuilles et faire apparaître la structure de tas.
- A5. Effectuer à la main la transformation du tableau `[|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]` en tas-max par les deux méthodes.

- A6. Écrire une fonction de signature `heap_make_up : 'a array -> unit` qui transforme un tableau en un tas en faisant monter les éléments. Quelle la complexité de cette fonction?
- A7. Écrire une fonction de signature `heap_make_down : 'a array -> unit` qui transforme un tableau en un tas en faisant descendre les éléments. Quelle est la complexité de cette fonction <sup>1</sup>?
- A8. Écrire une fonction de signature `heap_sort : 'a array -> unit` qui effectue le tri par tas d'un tableau. Quelle la complexité de cette fonction?
- A9. Tester le tri par tas sur un tableau de 10000 valeurs entières choisies aléatoirement entre 0 et 1000000.

---

1. On suppose que le nombre de nœuds ayant la hauteur  $h$  est majoré par  $\lceil \frac{n}{2^{h+1}} \rceil$