Tri par tas

Réf : Cormen

Prop : Le tri par tas permet de trier en place un tableau en temps $O(n \log n)$

Démo

- Tout d'abord, on montre comment faire correspondre un tableau à une structure arborescente en considérant que chaque position dans le tableau est un noeud et les fonctions suivantes sont définies : $\text{PARENT}(i) = \lfloor i/2 \rfloor$, $\text{GAUCHE}(i) = 2i$, $\text{DROITE}(i) = 2i+1$.

- On commence par construire la procédure ENTASSER-MAX dont les autres procédures auront besoin : quand on appelle ENTASSER-MAX(i), on suppose que G[i] et D[i] sont des tas max mais A[i] peut être plus petit que ses enfants et il faut donc le faire descendre :

```
ENTASSER-MAX (A, i)
    m ← argmax(A[i], A[GAUCHE(i)], A[DROITE(i)])
    Si m ≠ i ALORS
        ÉCHANGER A[i] ↔ A[m]
        ENTASSER-MAX (A, m)
```

La complexité de ENTASSER-MAX sur un noeud de hauteur h est $O(h)$ car la hauteur diminue de 1 à chaque appel récursif. Comme un tas est presque complet, la hauteur du tas est en $\log_2 n$, d'où une complexité pire cas pour ENTASSER en $O(\log n)$.

- À l'aide de cette procédure, on va maintenant voir comme construire un tas max à partir d'un tableau quelconque, et ce en temps linéaire. D'après la déf d'un tas, les éléments de $A[(\lfloor \frac{n}{2} \rfloor + 1) \dots n]$ sont tous des feuilles, donc il suffit d'appeler ENTASSER sur les autres éléments, à profondeur décroissante, ie indice décroissant :

```
CONSTRUIRE-TAS-MAX (A)
    n ← |A|
    POUR i ← ⌊n/2⌋ À 1 FAIRE
        ENTASSER-MAX (A, i)
```

* La correction de cet algorithme se montre avec l'invariant de boucle suivant : au début de l'itération d'une boucle, les noeuds $i+1, \dots, n$ sont des racines d'un tas max.
* La complexité nécessite une étude en détail des appels à ENTASSER :
étant donné un tas, au plus $\lceil \frac{n}{2^{h+1}} \rceil$ éléments sont à hauteur h. Ainsi, comme chaque appel à ENTASSER est linéaire en la hauteur de l'élément, on a :

$$\sum_{h=0}^{\lfloor \log n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h} \right) \quad \text{or} \quad \sum_{h=0}^{\infty} \frac{h}{2^h} = 2$$

Soit le temps d'exécution de CONSTRUIRE-TAS-MAX en $O(n)$.

- Pour trier avec des tas, on applique une fois CONSTRUIRE-TAS-MAX pour construire la structure de tas, puis comme l'élément $A[1]$ est la racine, l'élément maximal, on l'échange avec $A[n]$ pour le mettre à la bonne position. Si on "oublie" $A[n]$, le tableau $A[1..n-1]$ représente presque un tas max à l'exception de $A[1]$ qui n'est pas forcement au bon endroit. On applique alors ENTASSER-MAX$(A, 1)$ pour le rééquilibrer et on itère.

```
TRI-PAR-TAS (A)
    CONSTRUIRE-TAS-MAX(A)
    POUR i ← |A| À 2 FAIRE
        ECHANGER A[1] ↔ A[i]
        taille[A] ← taille[A] - 1
        ENTASSER-MAX (A, 1)
```

* La correction est donnée par la correction de CONSTRUIRE et ENTASSER
* La complexité en est $O(n)$ pour la construction et $O(\log n)$ pour chaque appel à ENTASSER, soit une complexité en $O(n \log n)$.

□