

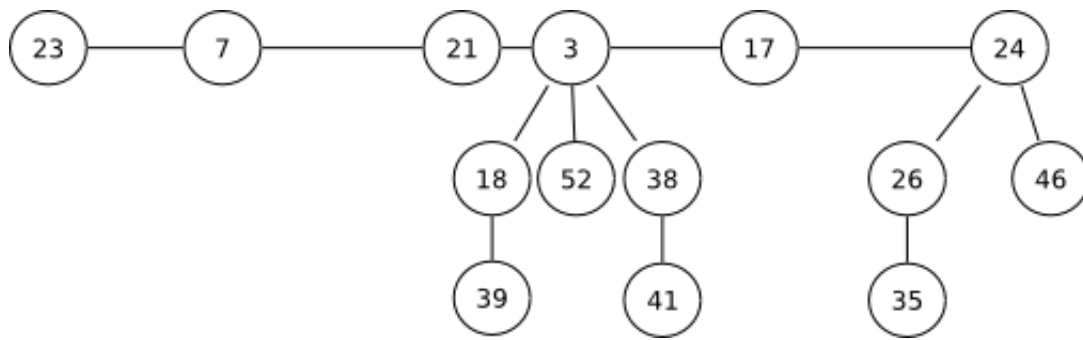
Séance de TD 11

Exercice

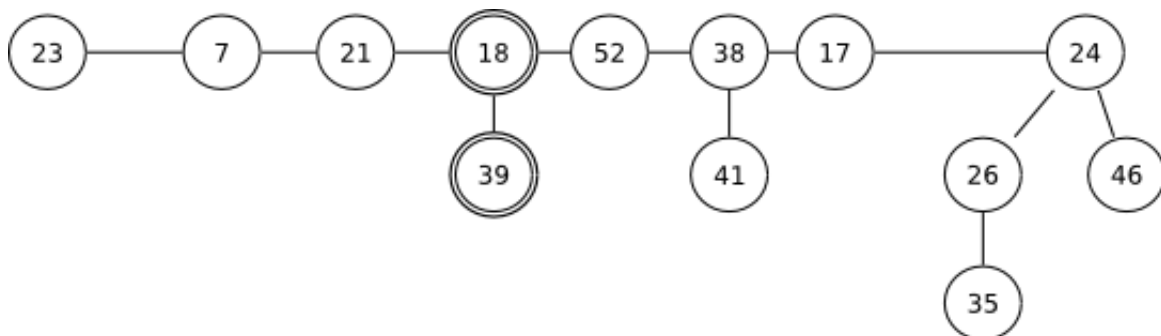
Extraction du noeud minimal dans un tas de Fibonacci

Exemple

Nous allons tenter de résoudre l'exercice sur l'exemple suivant :



- On extrait le min $\min[T]$ de la liste des racines de T
- On ajoute le fils de $\min[T]$ à la liste des racines de T
- On consolide T en reliant les arbres de même degré de T pour obtenir un tas de Fibonacci.

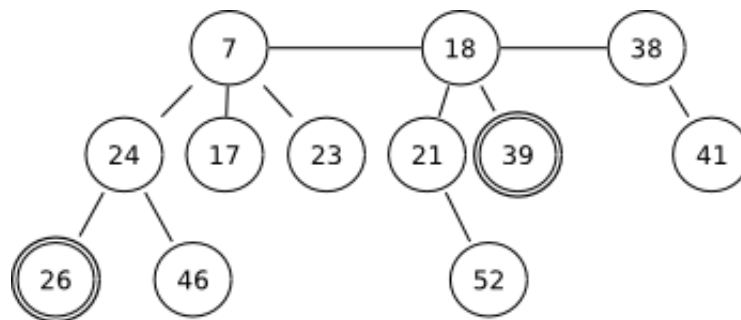


Les noeuds doublement entourés sont les noeuds marqués, i.e. ayant déjà perdu un de leurs fils, et/ou où la clé a été diminuée.

Pour consolider on définit un tableau A ; $A = [0, 1 \dots D(n)]$, où $D(n)$ est le degré maximal du noeud minimal du tas de Fibonacci. Dans notre cas, $D(n) = 3$, c'est le nombre de fils de $\min[T]$.

Initialement, $A[i] = u_0 \forall i \in [0; D(n)]$

Cela nous donne finalement :



Algorithmes

ExtraireTasFib(T)

 z <- min[T]

 Si z != Nil Alors

 Pour tout enfant x de z Faire

 ajouter x à la liste des racines de T

 P(x) <- Nil

 FinPour

 Supprimer z de la liste des racines de T

 Si z = Droite(z) Alors

 min[T] <- Nil

 Sinon

 min[T] <- Droite(z)

 Consolider(T)

 Finsi

 n[T] <- n[T] - 1

 retourner z

 Finsi

Fin

Consolider(T)

 Pour i de 0 à D(n) Faire

 A[i] <- Nil

 Finpour

 Pour chaque noeud w de la liste des racines de T Faire

 x <- w

 d <- degré(x)

 Tant que A(d) != Nil Faire

 Si clé(x) > clé(y) Alors

 permuter(x,y)

 Finsi

```

    RelierTasFils(T,y,x)
    A(d) <- Nil
    Fintantque
    A(d) <- x
    d <- d + 1
  Finpour
  min[T] <- Nil
  Pour i de 0 à D(n) Faire
    Si A[i] != Nil Alors
      Ajouter A[i] à la liste des racines de T
      Si min[T] != Nil ou clé(A[i]) < clé(point) Alors
        min[T] <- A[i]
      Finsi
    Finsi
  Finpour
Fin

RelierTasFibo(T,y,x)
  Supprimer y de la liste des racines de T
  Faire de y un enfant de x
  degré(x) <- degré(x) + 1
  marqué(y) <- Faux
Fin

```

Analyse du coût des algorithmes

L'algorithme ExtraireMin traite $D(n)$ noeuds. Sa complexité est donc en $O(D(n))$.

L'algorithme Consolider traite $D(n) + R(T) - 1$ élément, avec $R(T)$, le nombre de noeuds dans la liste des racines de T.

Le coût réel est de la forme : $O(D(n) + R(T))$.

On s'intéresse maintenant au coût amorti ; définissons la fonction potentiel $\phi(T)$ comme suit :

$$\phi(T) = R(T) + 2 \cdot m(T)$$

- $R(T)$ est le nombre de noeud contenus dans la liste des racines de T.
- $m(T)$ est le nombre de noeuds marqués de T.

Avant l'extraction de min :

$$\phi_{avant}(T) = R(T) + 2 \cdot m(T)$$

Après l'extraction de min :

$$\phi_{aprs}(T) = D(n) + 1 + 2 \cdot m(T)$$

Coût amorti :

$$\begin{aligned} O(D(n) + R(T)) + O(\phi_{aprs}(T) - \phi_{avant}(T)) \\ = O(D(n)) \\ = O(\log(n)) \end{aligned}$$

Corollaire : le degré maximal $D(n)$ d'un noeud d'un tas de Fibonacci à n noeuds est en $O(\log(n))$.

Démonstration : Soit x un noeud d'un tas de Fibonacci, $k = \text{degr}(x)$, admettons que :

$$\text{taille}(x) \geq \Phi^k, \text{ avec } \Phi = \frac{1 + \sqrt{5}}{2}$$

, Φ étant le nombre d'Or.

On a $n \geq \text{taille}(x) \geq \Phi^k$

En utilisant le logarithme de base Φ , on obtient :

$$n \geq \Phi^k \Rightarrow \log_{\Phi}(n) \geq k$$

Comme $k \in \mathbb{N}$, alors :

$$k \leq \lfloor \log_{\Phi}(n) \rfloor \tag{1}$$

$D(n)$ a le degré maximal de x , alors :

$$k \leq D(n) \tag{2}$$

$$(1) \text{ et } (2) \Rightarrow D(n) = \log(n)$$