ALGO1 – Tas de Fibonacci

François Schwarzentruber

October 22, 2020

Problème : on paie $O(\log |S|)$ pour mettre à jour la diminution, pour chaque arc dans Dikstra.
Solution : payer $O(1)$ pour chaque arc, en complexité amortie, en rendant moins rigide les tas binomiaux.

1 Définition

Définition 1 *Un* tas de Fibonacci H *est un ensemble d'arbres qui vérifient la condition de tas. En plus, il y a un pointeur vers la racine prioritaire, et certains nœuds sont marqués.*

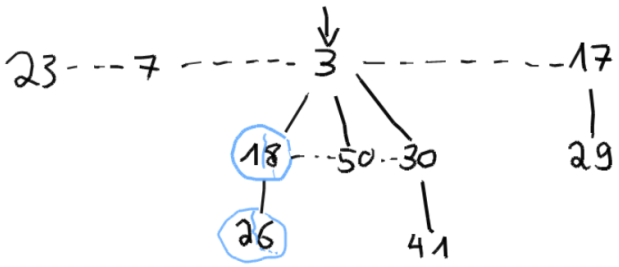
Notations.

n	nombre de nœuds
$d(x)$	degré d'un nœud
$d(H)$	degré maximal d'un nœud
$a(H)$	nombre d'arbres
$m(H)$	nombre de nœuds marqués

Implémentation.

Liste doublement chaînée pour la *liste des racines*
Pointeur vers la racine prioritaire
Listes doublement chaînées pour les nœuds frères
Chaque nœud a un pointeur vers le parent
Dans chaque nœud x, on inscrit son degré $d(x)$ dans $x.deg$
Nombre d'éléments inscrit dans $H.n$

Exemple 2



Définition 3 (Potentiel) *Le potentiel $\varphi(H)$ d'un tas de Fibonacci H est $a(H) + 2m(H)$.*

2 Complexité amortie

On effectue n opérations

$$H_0 \to H_1 \to \dots H_{i-1} \to H_i \to \cdots \to H_n.$$

Définition 4 (complexité amortie) *La complexité amortie de la i-ème opération $H_{i-1} \to H_i$ par rapport à une fonction de potentiel φ est*

$$\hat{c}_i := c_i + \varphi(H_i) - \varphi(H_{i-1})$$

où c_i est la complexité réelle de la i-ème opération.

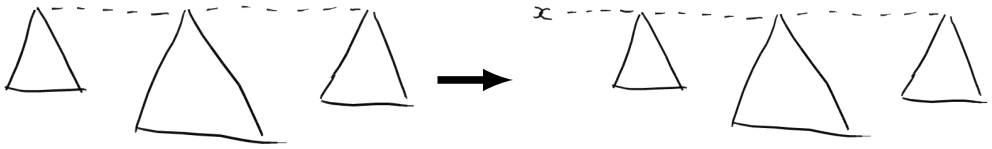
Théorème 5 *Si $\varphi(H_n) \geq \varphi(H_0)$, alors la complexité amortie totale majore la complexité réelle $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$.*

3 Opérations

$$H \to H'$$

3.1 Enfiler x

Ajouter un arbre-racine x et mettre à jour le pointeur vers la racine prioritaire si besoin.

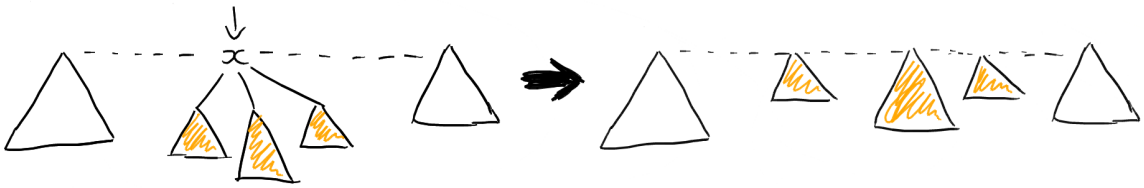


Proposition 6 *Pour l'opération enfiler, on a :*

Complexité réelle : $O(1)$ *Différence de potentiel :* 1 *Complexité amortie :* $O(1)$

3.2 Défiler l'élément prioritaire x

1. Supprimer x, le remplacer par ses enfants

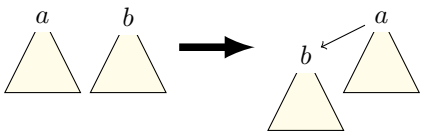


2. *Consolider* la structure : on colle les arbres de même degré, jusqu'à n'avoir que des arbres de degrés différents

,

procédure consolider()
  allouer un tableau $T[0, \dots, d(H)] = [\bullet, \dots, \bullet]$ // on verra que $d(H) = O(\log n)$
  pour toute racine x dans la liste des racines **faire**
    $d = d(x)$
    tant que $T[d] \neq \bullet$ **faire**
      Coller $T[d]$ à x ou x à $T[d]$ selon les priorités
      $T[d] := \bullet$
      $d := d + 1$
    $T[d] := x$
  Reconstruire la liste des racines à partir de T
  Calculer l'élément prioritaire

Coller a à b si a plus prioritaire que b



et démarquer b

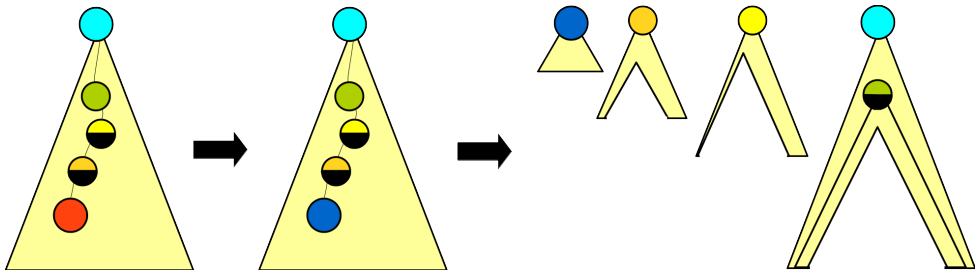
Proposition 7 *Pour l'opération défiler l'élément prioritaire, on a :*

Complexité réelle : $O(a(H) + d(H))$ *Différence de potentiel :* $O(d(H) - a(H))$ *Complexité amortie :* $O(d(H))$

DÉMONSTRATION. A la fin, le nombre d'arbres est $\leq d(H) + 1$
$\varphi(H') - \varphi(H) = a(H') - a(H) \leq d(H) + 1 - a(H)$ ■

Remarque 8 *Jusqu'ici, tous les arbres sont des arbres binomiaux. Du coup,* $d(H) \leq \log_2 n$.

3.3 Mise à jour : Rendre plus prioritaire x



modifier la priorité de x
si la propriété de tas n'est pas respectée,
  décrocher x et le mettre x dans la liste des racines
  démarquer x
  pour tous les sommets y en remontant depuis où était x, jusqu'à trouver un sommet non marqué
    décrocher y et le mettre dans la liste des racines
    démarquer y
  marquer le dernier parent non marqué (sauf si c'est une racine)
mettre à jour le pointeur vers l'élément prioritaire

Proposition 9 *Pour l'opération de mise à jour, on a :*

Complexité réelle : $O(c)$ *Différence de potentiel :* $O(1) - c$ *Complexité amortie :* $O(1)$

où c est le nombre de nœuds que l'on a mis dans la liste

Démonstration.
$\varphi(H') - \varphi(H) := a(H') - a(H) + 2(m(H') + m(H))$
On a $a(H') = a(H) + c$
$m(H') \leq m(H) - c + 2$
D'où : $\varphi(H') - \varphi(H) \leq c - 2c + 4 = 4 - c$ ■

4 Rappel sur la suite de Fibonacci

Définition 10 *La suite de Fibonacci* $(F_k)_{k\in\mathbb{N}}$ *est définie par* $F_0 = 0$, $F_1 = 1$, *et pour tout* $k \geq 2$, $F_k = F_{k-1} + F_{k-2}$.

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	$\ldots$
0	1	1	2	3	5	8	13	$\ldots$

Lemme 11 $F_{k+2} = 1 + \sum_{i=0}^{k} F_i$.

Démonstration. Par récurrence. ■

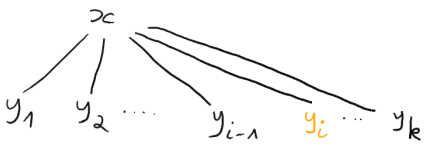
Proposition 12 $F_{k+2} \geq \varphi^k$ *où* $\varphi = (1 + \sqrt{5})/2$.

5 Degré logarithmique

Théorème 13 $d(H) = O(\log n)$.

Lemme 14 *Soit* x *un nœud de degré* d. *Soit* $y_1, \ldots, y_d$ *ses enfants dans l'ordre où ils ont été reliés à* x. *On a :* $d(y_1) \geq 0$, $d(y_i) \geq i - 2$ *pour tout* $i \in \{2, 3, \ldots, d\}$.

Démonstration. On se place à un instant de la structure et on a :



On considère $y_1, \ldots, y_k$ ses enfants courants dans l'ordre (de leur dernier ajout) où ils ont été relié à x. Au moment où y_i est relié à x, on a $d(y_i) = d(x)$ (en effet, ça se fait pendant la consolidation). Or $y_1, \ldots, y_{i-1}$ étaient déjà enfants de x à ce moment là. Donc, à ce moment là, $d(x) \geq i-1$ et donc $d(y_i) \geq i-1$.

Maintenant, peut être que y_i a perdu des enfants depuis. Mais en fait, il ne peut pas en avoir perdu plus d'un ! Supposons par l'absurde que y_i perd plus de deux enfants. A la première perte, y_i est marqué. A la deuxième perte, y_i est dézingué et devient une racine, n'est donc plus enfant de x, ou alors l'est ajouté plus tard... Contradiction avec le fait qu'on a considéré le dernier ajout.

Bref, $d(y_i) \geq i-2$. ■

Lemme 15 *Il y a au moins F_{d+2} éléments dans un sous-arbre dont la racine est de degré d.*

Démonstration. Soit s_d le nombre d'éléments minimum dans un sous-arbre d'un nœud de degré d d'un tas de Fibonacci. Montrons le par récurrence sur d. C'est vrai pour $d = 0$ et $d = 1$ car $s_0 \geq 1$ et $s_1 \geq 2$.

Supposons la propriété vraie pour des valeurs $\leq d-1$.

Considérons un sous-arbre de degré d. On note x sa racine,et $y_1, \ldots, y_d$ ses enfants comme dans le lemme 14. D'ailleurs, par ce lemme 14, cet arbre est composé du nœud y_1 puis des arbres de degré $i-2$ pour $i = 2..d$. Ainsi :

$$s_d \geq 1 + 1 + \sum_{i=2}^{d} s_{i-2} \geq 1 + 1 + \sum_{i=2}^{d} F_i \geq F_{d+2}.$$

en utilisant aussi le lemme 11. ■

Démonstration. (du théorème) Considérons un arbre de la structure et appelons m le nombre d'éléments dans cet arbre et d son degré. On a $\varphi^d \leq F_{d+2} \leq m \leq n$. En passant au log, on a $d = O(\log n)$. ■

6 Notes bibliographiques

Les tas de Fibonacci sont présentés ont été inventés par Fredman et Tarjan [FT87]. Les algorithmes détaillés sont donnés dans le chapitre sur les tas de Fibonacci dans [CLRS09]. Les tas relâchés offrent des complexités similaires mais en complexité pire cas [DGST88].

References

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009.

[DGST88] James R. Driscoll, Harold N. Gabow, Ruth Shrairman, and Robert Endre Tarjan. Relaxed heaps: An alternative to fibonacci heaps with applications to parallel computation. *Commun. ACM*, 31(11):1343–1354, 1988.

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.