DM6 : Tableaux dynamiques et recherche de collisions

A rendre pour le 22 février 2022 durant le TD. Les deux exercices sont complètement indépendants.

Exercice 1 *Tableaux dynamiques*

Le sujet est celui de l'exercice 4 du TP11. Vous pouvez bien sûr implémenter les fonctions demandées sur machine, afin de les tester, avant de recopier le code produit dans votre copie. Je rappelle au besoin que vous trouverez un compilateur C utilisable en ligne à cette adresse :

`https://www.onlinegdb.com/online_c_compiler`

Exercice 2 *Pollard-rho pour la recherche de collisions*

On considère une fonction de hachage $H : F \to E$ avec F et E deux ensembles tels que E est fini et inclus strictement dans F. On cherche un algorithme permettant d'exhiber une collision pour H (c'est-à-dire, on cherche à calculer deux éléments $a \neq b$ de F tels que $H(a) = H(b)$).

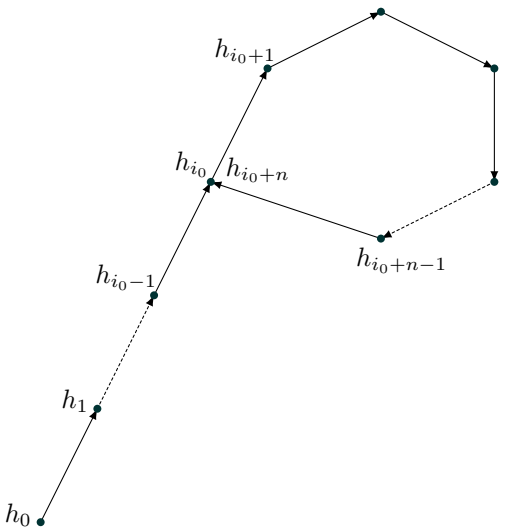
1. Montrer que l'existence d'une collision pour H est garantie.

Notons $\mathcal{H}$ l'image de la fonction H. On définit de manière récursive la suite $(h_i)_{i\in\mathbb{N}} \in \mathcal{H}^{\mathbb{N}}$ suivante : h_0 est un élément quelconque de $\mathcal{H}$ et pour tout $i \geq 0$, $h_{i+1} = H(h_i)$. On définit également la suite $(h'_i)_{i\in\mathbb{N}} = (h_{2i})_{i\in\mathbb{N}}$.

2. a) Montrer que pour tout $i \in \mathbb{N}$ on a $h_i = H^i(h_0)$ où H^i dénote H composée i fois avec elle-même (on rappelle que l'élément neutre pour la composition de fonctions est la fonction identité).

   b) Montrer que la suite $(h_i)_{i\in\mathbb{N}}$ est ultimement périodique ; c'est-à-dire qu'il existe un rang $r \geq 0$ et une période $p > 0$ tels que pour tout $i \geq r$, $h_{i+p} = h_i$.

On note $i_0 \in \mathbb{N}$ le plus petit rang à partir duquel $(h_i)_{i\in\mathbb{N}}$ est périodique et $n > 0$ la plus petite période de $(h_i)_{i\geq i_0}$. Si chaque flèche représente l'application de la fonction H, on se retrouve donc avec la modélisation suivante du comportement de la suite $(h_i)_{i\in\mathbb{N}}$:



Cette forme ressemble à la lettre grecque ρ. L'algorithme de recherche de collisions ici étudié, conceptualisé par le mathématicien John Pollard, s'appelle ainsi classiquement "l'algorithme rho de Pollard" (Pollard-rho algorithm en anglais). On supposera que l'indice i_0 est strictement positif dans la suite.

3. Exhiber deux éléments de la suite $(h_i)_{i\in\mathbb{N}}$ (fonctions de i_0 et n) provoquant une collision pour H.

Une méthode naïve pour trouver une collision est de calculer récursivement tous les éléments de la suite $(h_i)_{i\in\mathbb{N}}$, de stocker l'indice i dans la case h_i d'un tableau de taille $\mathrm{Card}(E)$, et à chaque nouveau calcul de vérifier si la case indicée par la nouvelle valeur obtenue est vide ou non (si elle n'est pas vide, on a trouvé une collision). On obtient alors une complexité spatiale et temporelle pire cas en $O(\mathrm{Card}\, E)$ (le paradoxe des anniversaires assure en fait que la complexité temporelle attendue est en $O(\sqrt{\mathrm{Card}\, E})$). On construit ci-dessous un algorithme qui améliore la complexité spatiale de l'algorithme naïf.

4. a) Montrer que, s'il existe deux indices $i, i' \geq i_0$ tels que $h_i = h_{i'}$, alors $i - i'$ est un multiple de n.
   b) Montrer qu'il existe un indice $j > 0$ tel que $h_j = h'_j$. Montrer qu'un tel indice est nécessairement plus grand que i_0 et est un multiple de n. Montrer que le plus petit de ces indices, noté j_0, vérifie de surcroît $j_0 < i_0 + n$.

5. Etant donnés H et h_0, expliquer (en langage naturel ou en pseudo-code) comment calculer j_0 en utilisant un espace mémoire constant. Montrer que le nombre d'opérations nécessaires à ce calcul est en $O(i_0 + n)$ si on suppose que l'application de la fonction H se fait en temps constant.

6. Une fois calculé l'indice j_0, expliquer comment retrouver la valeur de i_0. On pourra considérer les collisions entre les suites $(h_i)_{i\in\mathbb{N}}$ et $(h_{j_0+i})_{i\in\mathbb{N}}$.

7. Décrire, en utilisant les questions précédentes, un algorithme permettant de trouver une collision pour H. Etudier les complexités temporelle et spatiale de cet algorithme dans le cas où $E = [\![1, N]\!]$.

8. *(Question facultative)* Que se passe-t-il si $i_0 = 0$? Que faire dans ce cas ?

Remarques :

- *L'algorithme permettant de calculer i_0 aux questions 5 et 6 est l'algorithme de détection de cycles de Floyd, aussi appelé algorithme du lièvre et de la tortue : la suite $(h_i)_{i\in\mathbb{N}}$ avance de 1 en 1, c'est la tortue, tandis que la suite $(h'_i)_{i\in\mathbb{N}}$ avance de 2 en 2, c'est le lièvre. La question 4 montre que le lièvre et la tortue se rejoignent nécessairement pour un certain indice, j_0, à partir duquel on peut calculer i_0.*

- *L'algorithme rho de Pollard est en fait un algorithme probabiliste de factorisation d'entiers (qui peut se décliner en algorithme de calcul de logarithmes discrets). Il est efficace pour trouver les petits facteurs d'un nombre, et est d'ailleurs toujours utilisé dans ce contexte. L'idée principale de cet algorithme peut être adaptée à d'autres situations, comme le montre cet exercice.*