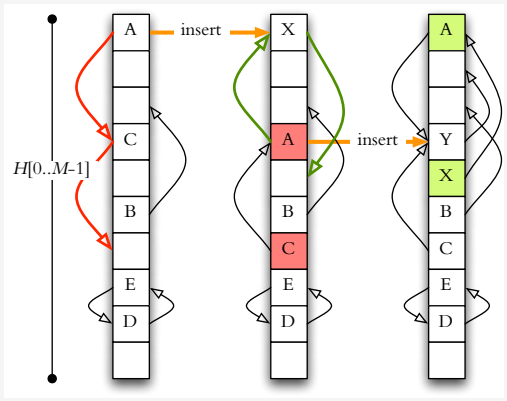
18 Hachage II

18.1 $O(1)$ recherche au pire : hachage coucou [Pagh]

W(en)



Hachage coucou (*cuckoo hashing*) est une méthode d'adressage ouvert où chaque clé est associée avec deux cases possibles, selon deux fonctions de hachage h_1, h_2. Pour insérer une clé x, on examine la case $h_1(x)$ et on y met x. Si la case était occupée, la clé occupante est déplacée à sa case altérnative en utilisant le même principe.

```
INSERT(x)                    // (insertion d'une nouvelle clé x)
C1  pos ← h1(x)
C2  tandis que H[pos] ≠ null
C3      y ← H[pos] ; H[pos] ← x
C4      si pos = h1(y) alors pos ← h2(y)
        sinon pos ← h1(y)
C5      x ← y
C6  H[pos] ← x
```

La boucle de la ligne C2 ne se termine pas si les liens $h_1(x) - h_2(x)$ suivis forment un cycle (v. E-D sans l'illustration). Pour cela, on pose une borne supérieure K sur les itérations permises. Après K opérations, on sort de la boucle et performe une expansion de tableau avec d'autres choix de fonctions h_i (rehachage). Altérnativement, on peut stocker les éléments dont l'insertion a échoué après K itérations dans une liste chaînée à côté. Heureusement, cette liste ne devient pas trop grande !

Pour étudier le comportement de cette technique, on définit le **graphe coucou** $\mathcal{G}(h_1, h_2) = (\mathcal{V}, \mathcal{E})$ dont les sommets correspondent aux cases $\mathcal{V} = \{0, \dots, M-1\}$, et les arêtes aux placements altérnatifs des clés dans le tableau $\mathcal{E} = \Big\{\big(h_1(x), h_2(x)\big) : x \text{ est une clé}\Big\}$. (On permet des arêtes multiples entre deux nœuds.) L'insertion d'une clé correspond à un chemin dans le graphe coucou : le chemin commence par le sommet $h_1(x)$ et comprend les arêtes $\big(h_1(y), h_2(y)\big)$ des déplacements.

Lemme 18.1. *Il existe un chemin de longueur ℓ entre deux sommets $i, j \in \{0, \dots, M-1\}$ avec probabilité moins que $\frac{(2\alpha)^\ell}{M}$, où $\alpha = n/M$ est la facteur de charge.*

Démonstration. On montre que la borne $p_\ell = (2\alpha)^\ell/M$ est valide pour tout ℓ par induction. Un chemin de longueur 1 ($\ell = 1$) est juste une arête entre i et j. Une telle arête existe avec une probabilité bornée par $p_1 = \frac{n}{M(M+1)/2} < 2\alpha/M$. Un chemin de longueur $\ell > 1$ comprend un chemin $i \rightsquigarrow k$ de longueur $(\ell-1)$ et une arête entre k et j. Donc la probabilité d'avoir un chemin de longueur ℓ est bornée par $p_\ell = \sum_{k=0}^{M-1} p_{\ell-1} \cdot p_1 = M p_{\ell-1} p_1$. Or, $(2\alpha)^\ell/M = M\big((2\alpha)^{\ell-1}/M\big)\big(2\alpha/M\big)$. ■

Théorème 18.2. INSERT(x) *prend $O\big(1/(1-2\alpha)\big)$ temps en moyenne, si le graphe ne contient pas de cycles.*

Démonstration. Par la lemme 18.1, il existe un chemin entre deux nœuds i, j dans le graphe coucou avec une probabilité bornée par $p_{i \rightsquigarrow j} = M^{-1} \sum_{\ell=1}^{\infty} (2\alpha)^\ell = \frac{2\alpha}{M(1-2\alpha)}$. En espérance, il existe au plus $r = \sum_j p_{i \rightsquigarrow k} = 2\alpha/(1-2\alpha)$ nœuds j connexes à i, donc la boucle de C2 s'exécute r fois au plus en moyenne.

Avec la ligne C6, il y a $(r+1)$ affectations de genre $H[\mathsf{pos}] \leftarrow x$, et l'algorithme s'exécute en $O(1+r)$ temps. ■

Exercice 18.1. *Montrer comment supprimer une clé dans hachage coucou.*

18.2 Hachage pour compter [Flajolet]

Problème : comment compter le nombre d'éléments différents sur une grande liste ?
Exemples : nombre de pages Web, nombre d'addresses IP différents dans un ensemble de paquets envoyés (diagnostique d'attaques sur l'internet).
Formellement : on a une grande liste $x_1, \ldots, x_n$ et on voudrait savoir le nombre d'éléments différents (**cardinalité**). On sait que $x_i \in U$ ou U est un ensemble fini (p.e., addresses IP).
Solution possible : tableau booléen de taille $|U|$ — prend trop de mémoire

Loglog counting. Idée de base : on utilise une bonne fonction de hachage h. La valeur $h(x_i)$ est un nombre binaire sur t bits. Chaque bit de $h(x_i)$ est 0 ou 1 avec des probabilités 50–50%. Nombre de bits 0 au début : au moins k avec probabilité 0.5^k. Soit $\rho(h(x))$ le nombre de 0s au début de $h(x)$. Si $x_i = x_j$, alors $h(x_i) = h(x_j)$. Supposons qu'on a m éléments différents parmi les n. Si on regarde $\rho(h(x_i))$, on trouvera en moyenne $m/2^k$ éléments différents avec $\rho(h(x_i)) \geq k$. $\Rightarrow$ le maximum $\max_{i=1,\ldots n} \rho(h(x_i))$ devrait être de l'ordre $\lg m$.

1 **Cardinalité**$(x_1, \ldots)$
2 initialiser $\rho_{\max} \leftarrow 0$
3 **pour** $x_1, x_2, \ldots,$ **faire** $\rho_{\max} \leftarrow \max\{\rho_{\max}, \rho(h(x_i))\}$
4 estimer $m \leftarrow \frac{2^{\rho_{\max}}}{\phi}$.

La constante de la ligne 4 $\phi = e^{-\gamma}\sqrt{2} \approx 0.78$ vient de l'analyse précise de la distribution de $\rho_{\max}$. Si $\rho_{\max} \leq r$, on peut le stocker sur $\lg(r+1)$ bits $\Rightarrow$ avec $O(\log\log M)$ bits on peut estimer une cardinalité jusqu'à M.