Cours algorithmique

VAN DE MERGHEL Robin

2023

Contents

Cours du 28 avril 2023

Correction suppression `(A,x)` :

Remarque

- Dans le cas 2.1, on se ramène aux […]

Exemple

- Sur le schémas ci-dessus, on a une rotation gauche sur B

Exemple 2

Exemple 3

Exemple 4

Complexité de correction suppression

Le nombre d'itérations est borné par le nombre de fois où on a appliqué (2.2 + 2.1) + ($\leq$ 1 fois le Cas 2.3) + ($\leq$ 1 fois le Cas 2.4)

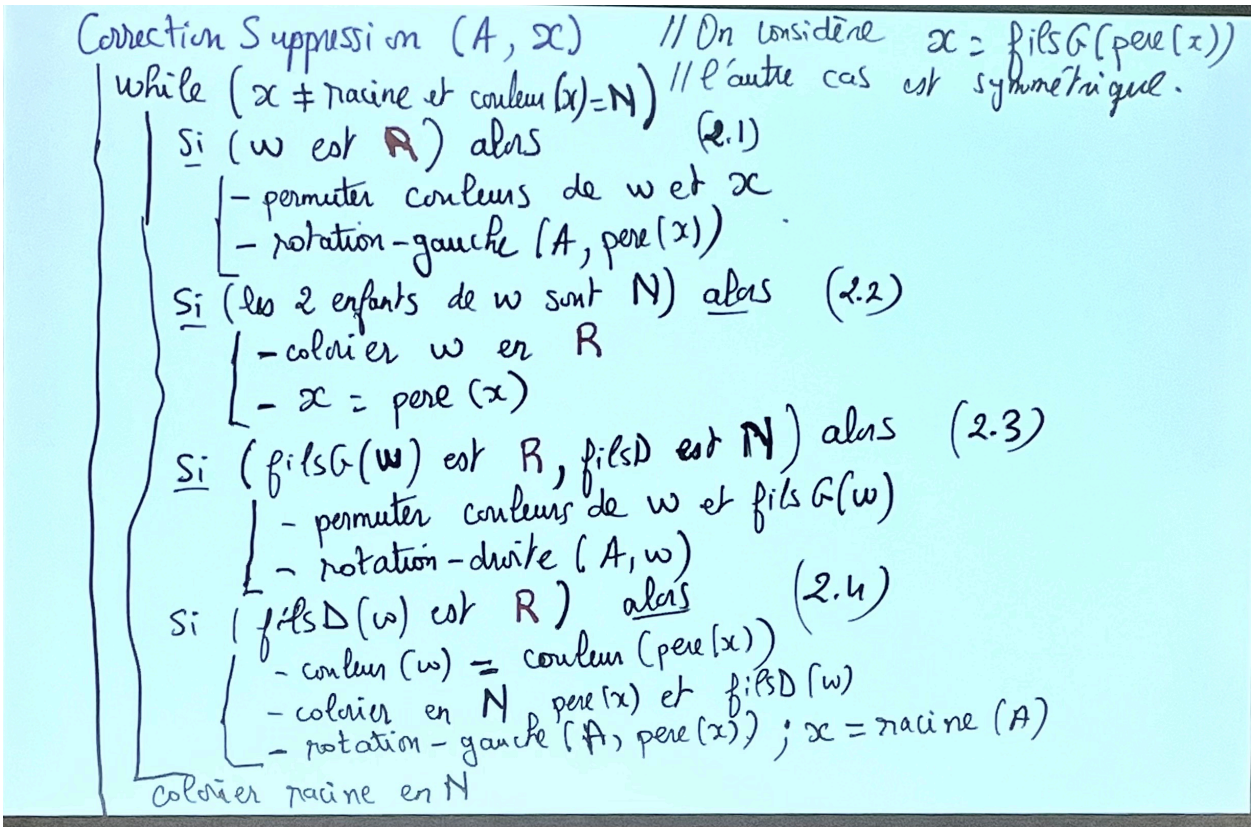

Figure 1: Correction suppression

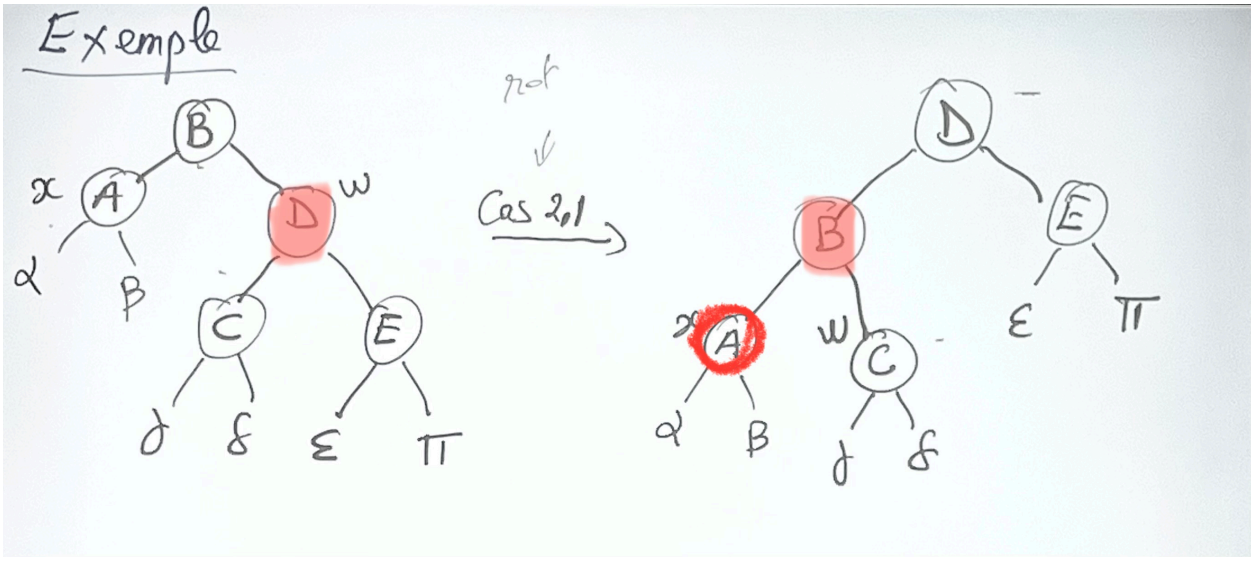

Figure 2: Exemple

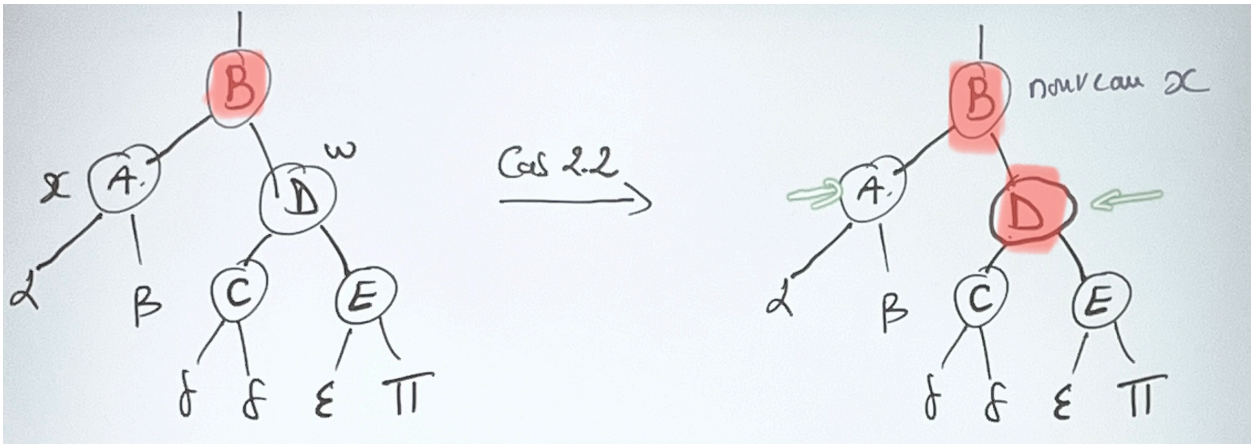

Figure 3: Exemple 2

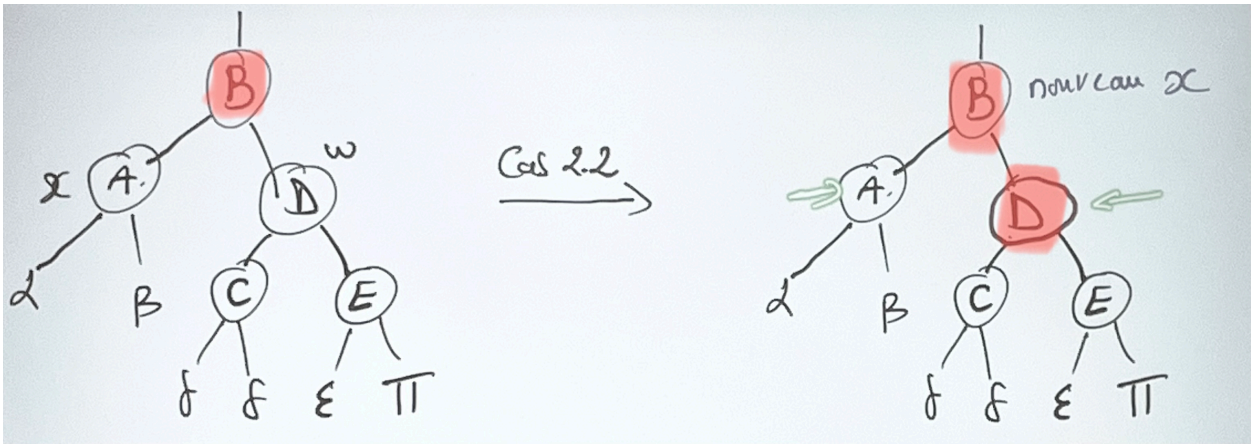

Figure 4: Exemple 3

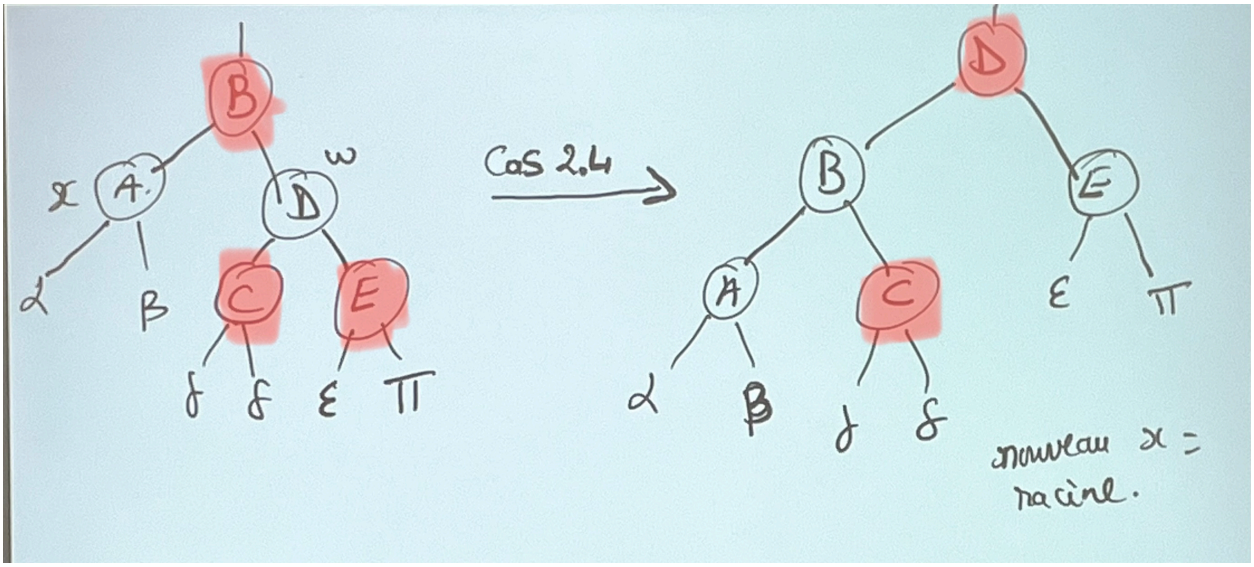

Figure 5: Exemple 4

Comme à chaque fois que l'on applique 2.1, on se réduit aux cas 2.2 à 2.4 et ç chaque application du cas 2.2 on remonte dans l'arbre, le nombre d'itérations est bornée par $2 \times h$. Comme chaque itération c'est en etemps $O(1)$ et hauteur : $O(logn)$, la complexité est en $O(logn)$.

Application des arbres

- On utilise les arbres pour représenter tout objet avec une hiérarchie (arbre génélogique, arbre de décision, …)
- Pour représenter des données sous forme hiérarchique pour une recherche plus rapide dans les données. Ex : texte, dictionnaire…

TDA Dictionnaire

On y stocke des éléments et on est intéressé par les opérations suivantes : insérer, supprimer, et rechercher.

TDA Gestion des partitions, ensemble disjoint

Si E est un ensemble, alors $\{E_1, E_2, ..., E_k\}$ est une partition de E si :

- $\bigcup_{1 \leq i \leq j}^{k} E_i = E$
- $\forall l \neq k$, $E_l \cap E_k = \emptyset$

Le TDA ensemble disjoint nous permet de manipuler les partitions d'un ensemble. On a 3 opérations :

- `creeEnsDisjoint` : crée un ensemble disjoint à partir d'un ensemble donné où chaque partie est un singleton.

Exemple

`creerEnsenbmeDisjoint({1,2,3,4,5})` doit créer la partition $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$

- `union(x,y)`:
  - On recherche les parties de x et de y et on les fusionne.

Exemple

`union({1,2,3,4,5},{1,2,3,4,5})` doit créer la partition $\{\{1, 2, 3, 4, 5\}\}$

On peut uiliser des listes pour implémenter le TDA ensemble disjoint (TD), mais la complexité en temps des opérations pas terrible.

Théorème

Avec la représentation par la liste chaînée en supposant qu'à chaque appel à l'union, le représentant de la liste la plus longue est le nouveau représentant, une séquence de m opérations nécessite un temps $O(m + n \log n)$.

Tables de hachage

- Avec les tableaux, on a un accès direct aux valeurs. Mais, grossir un tableau est coûteux
  - Toutes les valeurs sont contiguës en mémoire et on n'est même pas sûr d'avoir une telle possibilité en mémoire. Même s'il existe, il faut recopier les valeurs de l'ancien tableau
- Avec les listes, pas d'accès direct mais modification facile

Les tables de hachage c'est une solution de compromis entre les deux.

Principe

- prendre une tableau ni trop petit ni trop grand. Comment choisir la taille du tableau ?
- Pour chaque valeur, choisir une case du tableau pour y stocker. Comment choisir la case ?

Comme la taille du tableau est plus petite que le nombre de données, il y aura forcément deux valeurs renvoyées à la même case. On appelle ça une collision.

Réponse à la question 3 Pour résoudre les collisions, comme on ne veut pas perdre de données, on stocke toutes les valeurs en collision dans une list, et l'index du tableau où toutes ces valeurs étaient renvoyées pointera sur cette liste.

Comme on veut réduire les temps d'accè il faut que ces listes ne soient pas trop grand : une bonne réponse aux Q1 et Q2.

Réponse à la question 1 et 2 La fonction qui a toute valeur associe un index dans le tableau est appelé fonction de hachage.

Supposons que le tableau est de taille m : $h : U \to \{0, 1, ..., m-1\}$ est la fonction de hachahge. On dira que h est simplement uniforme si pour tout $x \in U$, x a la même probabilité d'être ahché dans chacune des cases 0 à $m-1$.

Avec cette hypothèse d'équi-probabilité, on peut avoir des borne sur les tailles des lites de collisions.

Notons n_i la taille de la litste T_i.

Propriété

- $\sum n_i = |U|$
- Comme il y a équi-probabilité, l'espérance de la taille d'une liste est $\frac{|U|}{m}$

Théorème Si h est une fonction de hachage simplement uniforme et que $h(x)$ se calcule en temps $O(1) \forall x$, alors une recherche infructueuse nécessite en moyenne $O(1+\alpha)$ et une recherche réussie aussi en temps moyen $O(1+\alpha)$ où $\alpha = \frac{|U|}{m}$.

Fonctions de hachage

Bonne fonction c'est une fonction simplement uniforme, mais difficile car :

- On ne sait pas comment les éléments sont distribués
- La distribution n'est pas forcéemnt uniforme : aucune raison que la probabilité de $h(x) = i$ est celle que $h(x) = j$ soit les mêmes

Il faut trouver des heuristiques pour s'en rapprocher.

Extraction On prend la représentation binaire et ensuite p bits dans cette représentation binaire. Si le choix des p bits est uniforme, on a une fonction de hachage simplement uniforme.

Compression On divise en blocs de p bits et on fait une somme de ces blocs. Ensuite on applique un opérateur binaires (ou exclusif, ou, …) pour avoir un nombre de p bits. (ex : SHA)

Division $h(x) = x \ mod \ m$

Comment choisir m :

- $m \neq 2^k$ sinon on a choisi les k bits de poids faible
- $m \neq 2^k - 1$ car une permutation ne changerait rien sur le résultat (si x est interprété en base 2^k)

Donald knuth Il a donné trois propriétés à satisfaire :

- m premier
- $msansdiviseurpremier$
- m ne divise pas $r^k \pm a$ pour des petites valeurs de a et k

(un ou l'autre)

Multiplication $h(x) = \lfloor m \times (x \times A \ mod \ 1) \rfloor$ avec $0 < A < 1$, $xmod1$ calcule la partie décimale

Multiplier par un petit nombre, récupérer la partie décimale et multiplier par m. On peut prendre $m = 2^k$.

$A = \frac{S}{2^w}$, w la taille d'un mot de l'oordinateur et S un entier de w bits.

Ensemble universel de fonction de hachage

Il ne faut pas que la fonction de hachage dépende des données.

H est une famille de fonctions universelle si $\forall x, y | \{h : h(x = h(y)\}| \leq \frac{|H|}{m}$ le nombre d'indices possibles.

Il existe des ensembles universels. Tout ensemble universel est un "vivier" de bonnes fonctions de hachage.

Exemple $p > m$ premer très grand, $a, b \in \mathbb{Z}_p$ $(a \neq 0)$, $h_{a,b}(x) = ((ax + b) \ mod \ p) \ mod \ m$

$$H_{p,m} = \{h_{a,b} : a, b \in \mathbb{Z}_p, a \neq 0\}$$

est un ensemble universel.