Correction

Examen d'algorithmique

EPITA ING1 2011 RATTRAPAGE S1; A. DURET-LUTZ

Durée : 1 heure 30

Juillet 2009

Consignes

- Cet examen se déroule **sans document** et **sans calculatrice**.
- Répondez sur le sujet dans les cadres prévus à cet effet.
- Il y a 5 pages d'énoncé et 1 page d'annexe dont vous ne devriez pas avoir besoin.
  Rappelez votre nom en haut de chaque feuille au cas où elles se mélangeraient.
- Ne donnez pas trop de détails. Lorsqu'on vous demande des algorithmes, on se moque des points-virgules de fin de commande etc. Écrivez simplement et lisiblement. Des spécifications claires et implémentables sont préférées à du code C ou C++ verbeux.
- Le barème est indicatif et correspond à une note sur 24.

1 Remplacement d'équipement (12 points)

Pour répondre à une commande d'un nouveau type de produit, une usine a besoin d'une machine particulière (un biglotron) au cours des cinq prochaines années. Le biglotron est coûteux à l'achat, mais aussi à l'entretien. De plus, l'usine a pour politique de remplacer les machines *au plus tard* tous les trois ans : la machine est alors revendue puis remplacée par une neuve. L'objectif de ce problème est de calculer les années de remplacement optimales en fonction des tarifs d'achat, de vente, et d'entretien.

Dans un premier temps, nous traiteront un cas particulier, avec des tarifs connus. Ensuite, on généralisera pour construire un algorithme prenant les tarifs en entrée.

Un biglotron coûte $a = 1000$ euros à l'achat (neuf). Son entretien demande $e_1 = 60$ euros la première année, $e_2 = 80$ la seconde, et $e_3 = 120$ la troisième (le biglotron sera remplacé au plus tard après trois ans). Un biglotron se revend $v_1 = 800$ euros s'il n'a qu'un an, $v_2 = 600$ s'il en a deux, et $v_3 = 500$ au bout de trois ans.

Pour simplifier on considère que les décisions d'acheter ou de revendre le biglotron ne se font que le premier jour de chaque année. On notera t le numéro de l'année, en commençant à 0 pour la première année.

Le scénario dans lequel le biglotron serait renouvelé au bout de 3 ans, puis revendu après les 2 dernières années coûterait 1300 euros :

$$\underbrace{(a+e_1)}_{t=0} + \underbrace{(e_2)}_{t=1} + \underbrace{(e_3)}_{t=2} + \underbrace{(-v_3+a+e_1)}_{t=3} + \underbrace{(e_2)}_{t=4} + \underbrace{(-v_2)}_{t=5} = 1060 + 80 + 120 + 560 + 80 - 600 = 1300$$

Le scénario dans lequel le biglotron serait renouvelé tous les 2 ans, puis revendu après la dernière année coûterait 1340 euros :

$$\begin{aligned}\underbrace{(a+e_1)}_{t=0} + \underbrace{(e_2)}_{t=1} + \underbrace{(-v_2+a+e_1)}_{t=2} + \underbrace{(e_2)}_{t=3} + \underbrace{(-v_2+a+e_1)}_{t=4} + \underbrace{(-v_1)}_{t=5} &= 1060 + 80 + 460 + 80 + 460 - 800 \\ &= 1340\end{aligned}$$

À $t = 0$, il faut forcément acheter et entretenir le biglotron : le coût sera toujours $a + e_1$. À $t = 5$ la production est terminée et la machine est forcément revendue, à un coût qui dépend de son âge. Pour $t \in \{1, 2, 3, 4\}$ la direction a deux possibilités : soit conserver le biglotron actuel (à condition toutefois qu'il ait moins de 3 ans), soit le revendre et en acheter un nouveau. Ce choix doit naturellement être fait de façon à *minimiser les coûts*.

1. **(1 point)** Combien existe-t-il de scénarios possibles ?

Réponse :

Pour chaque $t \in \{1, 2, 3, 4\}$, il y a deux choix possibles : remplacer ou conserver. Cela fait donc 2^4 possibilités. De ces possibilités, il faut retirer les trois où trois décisions « conserver » (ou plus) se suivent. Cela fait donc $2^4 - 3 = 13$ scénarios différents.

2. **(2 points)** On note $f(t, x)$ le coût **minimal** de l'utilisation du biglotron entre l'année t et la fin de la production (date 5) si l'on sait que le biglotron est vieux de x ans à la date t. Ainsi on a $f(5, x) = -v_x$ car lorsque $t = 5$ la production est terminée et il ne reste plus qu'à revendre le biglotron. De même, $\forall t \in \{1, 2, 3, 4\}$, $f(t, 3) = -v_3 + a + e_1 + f(t + 1, 1)$ car il faut forcément replacer un biglotron vieux de 3 ans.

   Donnez une définition de $f(t, x)$ pour $t \in \{1, 2, 3, 4\}$ et $x \in \{1, 2\}$, en fonction des valeurs de f pour $t + 1$. Rappelons qu'il y a deux possibilités (soit on conserve le biglotron une année de plus, soit on le remplace) et qu'on veut choisir la moins coûteuse.

Réponse :

$$\forall t \in \{1,2,3,4\},\ \forall x \in \{1,2\},\ f(t,x) = \min(\underbrace{e_{x+1} + f(t+1, x+1)}_{\text{on conserve}}, \underbrace{-v_x + a + e_1 + f(t+1,1)}_{\text{on remplace}})$$

3. **(3 points)** Le coût minimal de l'utilisation du biglotron sur 5 ans est donc de $a + e_1 + f(1, 1)$.

   Completez les valeurs de $f(t, x)$ dans le tableau suivant :

	$x = 1$	$x = 2$	$x = 3$
$t = 5$	-800	-600	-500
$t = 4$	-540	-380	-240
$t = 3$	-300	-120	20
$t = 2$	-40	140	
$t = 1$	220		

4. **(2 point)** Déduisez-en le coût minimal de l'utilisation du biglotron sur 5 ans. À quel scénario ce coût correspond-il ?

Réponse :

$a + e_1 + f(1, 1) = 1000 + 60 + 220 = 1280.$

Un scénario qui donne ce coût est celui où la machine est renouvelée à $t = 1$ et $t = 2$. Un autre est celui celui où la machine est renouvelée à $t = 3$ et $t = 4$.

5. **(2 points)** On généralise maintenant le problème à une machine qu'on achète pour une durée de n années, qu'on s'oblige à renouveler au moins toutes les k années, et pour laquelle on dispose aussi d'un coût d'achat a, ainsi que de coûts de ventes v_i et d'entretien e_i pour les différentes années.

   On calcule le coût minimal avec le même algorithme que ci-dessus. Quelle est la complexité de cet algorithme en fonction de n et k ?

Réponse :

On construit un tableau de taille $n \times k$.
Déterminer chaque cellule du tableau demande au plus un min et quelques addition, c'est-à-dire un nombre constant d'opérations.
Au total on effectue donc $\Theta(nk)$ opérations.

6. **(2 points)** Comment doit-on modifier cet algorithme pour être capable d'indiquer un scénario qui correspond au coût minimal trouvé ?

Réponse :

On construit un second tableau qui indique, pour chaque cellule du premier, si la valeur qui y est indiquée a été obtenue en renouvelant la machine, ou en la conservant.

2 Tas spécial (6 points)

Normalement, un tas de n éléments est représenté par un tableau de n cases. Le père de l'élément situé à l'indice i se trouve à l'indice $Parent(i) = \lfloor i/2 \rfloor$ et peut être accédé en temps constant.

Pour mémoire, voici l'algorithme d'insertion d'une valeur v dans un tas représenté par les n premiers éléments d'un tableau A.

```
HeapInsert(A, n, v)
i ← n + 1
A[i] ← v
while i > 1 and A[Parent(i)] < A[i] do
    A[Parent(i)] ↔ A[i]
    i ← Parent(i)
```

1. **(1 point)** Quelle est la complexité de cet algorithme lorsque le tas possède n éléments.

Réponse :

$O(\log n)$, c'est du cours.

2. **(3 points)** Imaginez maintenant que l'on remplace le tableau A par une liste doublement chaînée. L'accès au père de l'élément i se fait alors en $\Theta(i/2)$ opérations parce qu'il faut remonter la liste de $i/2$ positions.

   Quelle est la complexité de l'algorithme d'insertion lorsque A est une liste doublement chaînée ? **Justifiez votre réponse.**

Réponse :

Au pire l'insertion va nous forcer à partir de $A[n]$ et à remonter jusqu'à la racine. Depuis $A[n]$ on trouve $A[Parent(n)]$ en $\Theta(n/2)$ opérations. À partir de là on trouve ensuite $A[Parent(Parent(n))]$ en $\Theta(n/4)$ opérations. Au total on aura fait $\frac{n}{2^1} + \frac{n}{2^2} + \cdots + \frac{n}{2^{\lfloor \log n \rfloor}} = \Theta(n)$ opérations.
La complexité de l'algorithme est donc $O(n)$.

3. **(2 points)** On considère le tas représenté par le tableau suivant :

10	7	9	4	3	6	2	1

Donnez l'état du tas après les *suppressions* successives de ses trois plus grandes valeurs.

6	4	1	2	3

3 Tris (6 points)

1. **(1.5 points)** Donnez les noms de trois algorithmes de tri *stables* et indiquez leur complexité.

Réponse :

Tri par insertion $O(n^2)$.
Tri fusion $\Theta(n \log n)$.
Tri par dénombrement $\Theta(n)$.

2. **(1.5 points)** Donnez les noms de trois algorithmes de tri *instables* et indiquez leur complexité.

Réponse :

Tri rapide $O(n^2)$.
Tri par sélection $\Theta(n^2)$.
Tri par tas $O(n \log n)$.
Tri introspectif $O(n \log n)$.

3. **(3 points)** Indiquez une façon de rendre stable n'importe quel algorithme de tri. Quelle complexité (en temps et espace) cette modification ajoute-t-elle à un algorithme de tri lorsqu'il faut trier n valeurs ?

Réponse :

Il suffit de modifier les clefs des valeurs à trier pour y inclure leur indice d'origine. La fonction de comparaison doit alors être adaptée pour comparer ces indices lorsque les parties originales des clefs sont identiques.
Ce changement demande $\Theta(n)$ mémoire supplémentaire (on ne peut donc plus prétendre que le tri soit *en place*, car l'occupation mémoire dépend de n), en revanche il ne modifie pas la complexité du temps d'exécution.

FIN

Notations asymptotiques

$$\mathrm{O}(g(n)) = \{f(n) \mid \exists c \in \mathbb{R}^{+\star}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq f(n) \leq cg(n)\}$$
$$\Omega(g(n)) = \{f(n) \mid \exists c \in \mathbb{R}^{+\star}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq cg(n) \leq f(n)\}$$
$$\Theta(g(n)) = \{f(n) \mid \exists c_1 \in \mathbb{R}^{+\star}, \exists c_2 \in \mathbb{R}^{+\star}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \iff g(n) \in \mathrm{O}(f(n)) \text{ et } f(n) \notin \mathrm{O}(g(n)) \qquad f(n) \in \mathrm{O}(g(n)) \iff g(n) \in \Omega(f(n))$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \iff f(n) \in \mathrm{O}(g(n)) \text{ et } g(n) \notin \mathrm{O}(f(n)) \qquad f(n) \in \Theta(g(n)) \iff \begin{cases} f(n) \in \Omega(g(n)) \\ g(n) \in \Omega(f(n)) \end{cases}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \in \mathbb{R}^{+\star} \iff f(n) \in \Theta(g(n))$$

Ordres de grandeurs

constante	$\Theta(1)$	
logarithmique	$\Theta(\log n)$	
polylogarith.	$\Theta((\log n)^c)$	$c > 1$
	$\Theta(\sqrt{n})$	
linéaire	$\Theta(n)$	
	$\Theta(n \log n)$	
quadratique	$\Theta(n^2)$	
	$\Theta(n^c)$	$c > 2$
exponentielle	$\Theta(c^n)$	$c > 1$
factorielle	$\Theta(n!)$	
	$\Theta(n^n)$	

Identités utiles

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$
$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1} \qquad \text{si } x \neq 1$$
$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad \text{si } |x| < 1$$
$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2} \qquad \text{si } |x| < 1$$
$$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log n)$$
$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

Définitions diverses

La complexité d'un problème est celle de l'algorithme le plus efficace pour le résoudre.

Un tri stable préserve l'ordre relatif de deux éléments égaux (au sens de la relation de comparaison utilisée pour le tri).

Un tri en place utilise une mémoire temporaire de taille constante (indépendante de n).

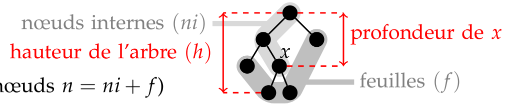
Théorème général

Soit à résoudre $T(n) = aT(n/b) + f(n)$ avec $a \geq 1$, $b > 1$

- Si $f(n) = \mathrm{O}(n^{\log_b a - \varepsilon})$ pour un $\varepsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$.
- Si $f(n) = \Theta(n^{\log_b a})$, alors $T(n) = \Theta(n^{\log_b a} \log n)$.
- Si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ pour un $\varepsilon > 0$, et si $af(n/b) \leq cf(n)$ pour un $c < 1$ et tous les n suffisamment grands, alors $T(n) = \Theta(f(n))$.

(Note : il est possible de n'être dans aucun de ces trois cas.)

Arbres



(nœuds $n = ni + f$)

Pour tout arbre binaire :

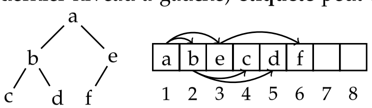
$n \leq 2^{h+1} - 1$ $\qquad h \geq \lceil \log_2(n+1) - 1 \rceil = \lfloor \log_2 n \rfloor$ si $n > 0$

$f \leq 2^h$ $\qquad h \geq \lceil \log_2 f \rceil$ si $f > 0$

$f = ni + 1$ (si l'arbre est *complet* = les nœud internes ont tous 2 fils)

Dans un arbre binaire équilibré une feuille est soit à la profondeur $\lfloor \log_2(n+1) - 1 \rfloor$ soit à la profondeur $\lceil \log_2(n+1) - 1 \rceil = \lfloor \log_2 n \rfloor$. Pour ces arbres $h = \lfloor \log_2 n \rfloor$.

Un arbre parfait (= complet, équilibré, avec toutes les feuilles du dernier niveau à gauche) **étiqueté** peut être représenté par un tableau.



Les indices sont reliés par :

$\text{Père}(y) = \lfloor y/2 \rfloor$

$\text{FilsG}(y) = y \times 2$

$\text{FilsD}(y) = y \times 2 + 1$

Rappels de probabilités

Espérance d'une variable aléatoire X **:** C'est sa valeur attendue, ou moyenne. $\mathrm{E}[X] = \sum_x \Pr\{X = x\}$

Variance : $\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}^2[X]$

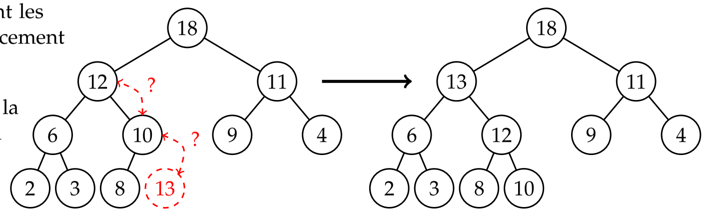
Loi binomiale : On lance n ballons dans r paniers. Les chutes dans les paniers sont équiprobables ($p = 1/r$). On note X_i le nombre de ballons dans le panier i. On a $\Pr\{X_i = k\} = C_n^k p^k (1-p)^{n-k}$. On peut montrer $E[X_i] = np$ et $\mathrm{Var}[X_i] = np(1-p)$.

Tas

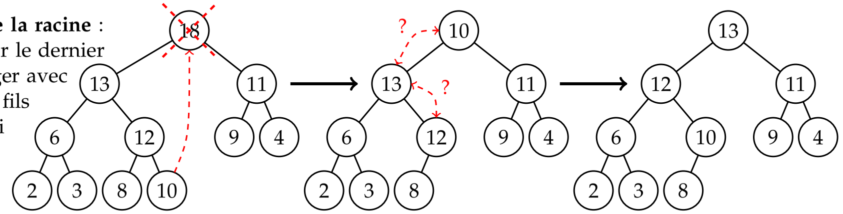
Un tas est un arbre parfait partiellement ordonné : l'étiquette d'un nœud est supérieure à celles de ses fils.

Dans les opérations qui suivent les arbres parfaits sont plus efficacement représentés par des tableaux.

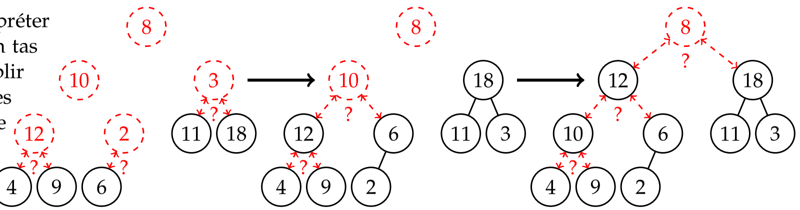
Insertion : ajouter l'élément à la fin du tas, l'échanger avec son père tant qu'il lui est inférieur (en remontant vers la racine). $T_{\text{insert}} = \mathrm{O}(\log n)$



Suppression de la racine : la remplacer par le dernier nœud ; l'échanger avec son plus grand fils tant que celui-ci est plus grand. $T_{\text{rem}} = \mathrm{O}(\log n)$



Construction : interpréter le tableau comme un tas (incorrect) puis rétablir l'ordre en partant des feuilles (vues comme des tas corrects). $T_{\text{build}} = \Theta(n)$

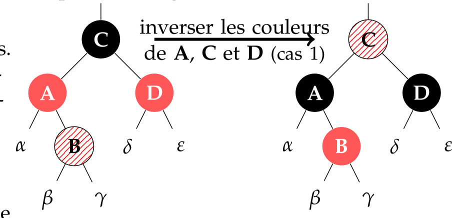


Arbres Rouge et Noir

Les ARN sont des arbres binaires de recherche dans lesquels : (1) un nœud est rouge ou noir, (2) racine et feuilles (NIL) sont noires, (3) les fils d'un nœud rouge sont noirs, et (4) tous les chemins reliant un nœud à une feuille (de ses descendants) contiennent le même nombre de nœuds noirs (= la *hauteur noire*). Ces propriétés interdisent un trop fort déséquilibre de l'arbre, sa hauteur reste en $\Theta(\log n)$.

Insertion d'une valeur : insérer le nœud avec la couleur rouge à la position qu'il aurait dans un arbre binaire de recherche classique, puis, si le père est rouge, considérer les trois cas suivants dans l'ordre.

Cas 1 : Le père et l'oncle du nœud considéré sont tous les deux rouges. **Répéter** cette transformation à partir du grand-père si l'arrière grand-père est aussi rouge.



(Dans tous ces cas, les lettres grecques représentent des sous-arbres avec la même hauteur noire.)

Cas 2 : Si le père est rouge, l'oncle noir, et que le nœud courant n'est pas dans l'axe père–grand-père, une rotation permet d'aligner fils, père, et grand-père.

Cas 3 : Si le père est rouge, l'oncle noir, et que le nœud courant est dans l'axe père–grand-père, un rotation et une inversion de couleurs rétablissent les propriétés des ARN.